

Security Events Working Group
Internet-Draft
Intended status: Standards Track
Expires: December 26, 2020

A. Backman, Ed.
Amazon
M. Jones, Ed.
Microsoft
M. Scurtescu
Coinbase
M. Ansari
Cisco
A. Nadalin
Microsoft
June 24, 2020

Poll-Based Security Event Token (SET) Delivery Using HTTP
draft-ietf-secevent-http-poll-12

Abstract

This specification defines how a series of Security Event Tokens (SETs) can be delivered to an intended recipient using HTTP POST over TLS initiated as a poll by the recipient. The specification also defines how delivery can be assured, subject to the SET Recipient's need for assurance.

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 December 26, 2020.

Copyright Notice

Copyright (c) 2020 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

1. Introduction and Overview	2
1.1. Notational Conventions	3
1.2. Definitions	3
2. SET Delivery	3
2.1. Polling Delivery using HTTP	4
2.2. Polling HTTP Request	5
2.3. Polling HTTP Response	6
2.4. Poll Request	6
2.4.1. Poll-Only Request	7
2.4.2. Acknowledge-Only Request	8
2.4.3. Poll with Acknowledgement	9
2.4.4. Poll with Acknowledgement and Errors	10
2.5. Poll Response	10
2.5.1. Poll Error Response	12
2.6. Error Response Handling	12
3. Authentication and Authorization	13
4. Security Considerations	13
4.1. Authentication Using Signed SETs	14
4.2. HTTP Considerations	14
4.3. Confidentiality of SETs	14
4.4. Access Token Considerations	14
4.4.1. Bearer Token Considerations	14
5. Privacy Considerations	15
6. IANA Considerations	15
7. References	15
7.1. Normative References	15
7.2. Informative References	17
Appendix A. Unencrypted Transport Considerations	18
Appendix B. Other Streaming Specifications	18
Appendix C. Acknowledgments	18
Appendix D. Change Log	19
Authors' Addresses	21

1. Introduction and Overview

This specification defines how a stream of Security Event Tokens (SETs) [RFC8417] can be transmitted to an intended SET Recipient using HTTP [RFC7231] over TLS. The specification defines a method to

poll for SETs using HTTP POST. This is an alternative SET delivery method to the one defined in [I-D.ietf-secevent-http-push].

Poll-based SET delivery is intended for scenarios where all of the following apply:

- o The recipient of the SET is capable of making outbound HTTP requests.
- o The transmitter is capable of hosting a TLS-enabled HTTP endpoint that is accessible to the recipient.
- o The transmitter and recipient are willing to exchange data with one another.

In some scenarios, either push-based or poll-based delivery could be used, and in others, only one of them would be applicable.

A mechanism for exchanging configuration metadata such as endpoint URLs, cryptographic keys, and possible implementation constraints such as buffer size limitations between the transmitter and recipient is out of scope for this specification. How SETs are defined and the process by which security events are identified for SET Recipients are specified in [RFC8417].

1.1. 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.

Throughout this document, all figures may contain spaces and extra line wrapping for readability and due to space limitations.

1.2. Definitions

This specification utilizes terminology defined in [RFC8417] and [I-D.ietf-secevent-http-push].

2. SET Delivery

When a SET is available for a SET Recipient, the SET Transmitter queues the SET in a buffer so that a SET Recipient can poll for SETs using HTTP POST.

In poll-based SET delivery using HTTP over TLS, zero or more SETs are delivered in a JSON [RFC8259] document to a SET Recipient in response to an HTTP POST request to the SET Transmitter. Then in a following request, the SET Recipient acknowledges received SETs and can poll for more. All requests and responses are JSON documents and use a "Content-Type" of "application/json", as described in Section 2.1.

After successful (acknowledged) SET delivery, SET Transmitters are not required to retain or record SETs for retransmission. Once a SET is acknowledged, the SET Recipient SHALL be responsible for retention, if needed. Transmitters may also discard undelivered SETs under deployment-specific conditions, such as if they have not been polled for over too long a period of time or if an excessive amount of storage is needed to retain them.

Upon receiving a SET, the SET Recipient reads the SET and validates it in the manner described in Section 2 of [I-D.ietf-secevent-http-push]. The SET Recipient MUST acknowledge receipt to the SET Transmitter, and SHOULD do so in a timely fashion, as described in Section 2.4. The SET Recipient SHALL NOT use the event acknowledgement mechanism to report event errors other than those relating to the parsing and validation of the SET.

2.1. Polling Delivery using HTTP

This method allows a SET Recipient to use HTTP POST (Section 4.3.3 of [RFC7231]) to acknowledge SETs and to check for and receive zero or more SETs. Requests MAY be made at a periodic interval (short polling) or requests MAY wait, pending availability of new SETs using long polling, per Section 2 of [RFC6202]. Note that short polling will result in retrieving zero or more SETs whereas long polling will typically result in retrieving one or more SETs unless a timeout occurs.

The delivery of SETs in this method is facilitated by HTTP POST requests initiated by the SET Recipient in which:

- o The SET Recipient makes a request for available SETs using an HTTP POST to a pre-arranged endpoint provided by the SET Transmitter or,
- o after validating previously received SETs, the SET Recipient initiates another poll request using HTTP POST that includes acknowledgement of previous SETs and requests the next batch of SETs.

The purpose of the acknowledgement is to inform the SET Transmitter that delivery has succeeded and redelivery is no longer required.

Before acknowledgement, SET Recipients validate the received SETs and retain them in a manner appropriate to the recipient's requirements. The level and method of retention of SETs by SET Recipients is out of scope of this specification.

2.2. Polling HTTP Request

When initiating a poll request, the SET Recipient constructs a JSON document that consists of polling request parameters and SET acknowledgement parameters in the form of JSON objects.

When making a request, the HTTP "Content-Type" header field is set to "application/json".

The following JSON object members are used in a polling request:

Request Processing Parameters

maxEvents

An OPTIONAL integer value indicating the maximum number of unacknowledged SETs to be returned. The SET Transmitter SHOULD NOT send more SETs than the specified maximum. If more than the maximum number of SETs are available, the SET Transmitter determines which to return first; the oldest SETs available MAY be returned first, or another selection algorithm MAY be used, such as prioritizing SETs in some manner that makes sense for the use case. first. A value of "0" MAY be used by SET Recipients that would like to perform an acknowledge-only request. This enables the Recipient to use separate HTTP requests for acknowledgement and reception of SETs. If this parameter is omitted, no limit is placed on the number of SETs to be returned.

returnImmediately

An OPTIONAL JSON boolean value that indicates the SET Transmitter SHOULD return an immediate response even if no results are available (short polling). The default value is "false", which indicates the request is to be treated as an HTTP Long Poll, per Section 2 of [RFC6202]. The timeout for the request is part of the configuration between the participants, which is out of scope of this specification.

SET Acknowledgment Parameters

ack

A JSON array of strings whose values are the "jti" [RFC7519] values of successfully received SETs that are being acknowledged. If there are no outstanding SETs to acknowledge,

this member is omitted or contains an empty array. Once a SET has been acknowledged, the SET Transmitter is released from any obligation to retain the SET.

setErrs

A JSON object with one or more members whose keys are the "jti" values of invalid SETs received. The values of these objects are themselves JSON objects that describe the errors detected using the "err" and "description" values specified in Section 2.6. If there are no outstanding SETs with errors to report, this member is omitted or contains an empty JSON object.

2.3. Polling HTTP Response

In response to a poll request, the SET Transmitter checks for available SETs and responds with a JSON document containing the following JSON object members:

sets

A JSON object containing zero or more SETs being returned. Each member name is the "jti" of a SET to be delivered and its value is a JSON string representing the corresponding SET. If there are no outstanding SETs to be transmitted, the JSON object SHALL be empty. Note that both SETs being transmitted for the first time and SETs that are being re-transmitted after not having been acknowledged are communicated here.

moreAvailable

A JSON boolean value that indicates if more unacknowledged SETs are available to be returned. This member MAY be omitted, with the meaning being the same as including it with the boolean value "false".

When making a response, the HTTP "Content-Type" header field is set to "application/json".

2.4. Poll Request

The SET Recipient performs an HTTP POST (see Section 4.3.4 of [RFC7231]) to a pre-arranged polling endpoint URI to check for SETs that are available. Because the SET Recipient has no prior SETs to acknowledge, the "ack" and "setErrs" request parameters are omitted.

After a period of time configured in an out-of-band manner between the SET Transmitter and Recipient, a SET Transmitter MAY redeliver SETs it has previously delivered. The SET Recipient SHOULD accept repeat SETs and acknowledge the SETs regardless of whether the

Recipient believes it has already acknowledged the SETs previously. A SET Transmitter MAY limit the number of times it attempts to deliver a SET.

If the SET Recipient has received SETs from the SET Transmitter, the SET Recipient parses and validates that received SETs meet its own requirements and SHOULD acknowledge receipt in a timely fashion (e.g., seconds or minutes) so that the SET Transmitter can mark the SETs as received. SET Recipients SHOULD acknowledge receipt before taking any local actions based on the SETs to avoid unnecessary delay in acknowledgement, where possible.

Poll requests have three variations:

Poll-Only

In which a SET Recipient asks for the next set of events where no previous SET deliveries are acknowledged (such as in the initial poll request).

Acknowledge-Only

In which a SET Recipient sets the "maxEvents" value to "0" along with "ack" and "setErrs" members indicating the SET Recipient is acknowledging previously received SETs and does not want to receive any new SETs in response to the request.

Combined Acknowledge and Poll

In which a SET Recipient is both acknowledging previously received SETs using the "ack" and "setErrs" members and will wait for the next group of SETs in the SET Transmitters response.

2.4.1. Poll-Only Request

In the case where no SETs were received in a previous poll (see Figure 7), the SET Recipient simply polls without acknowledgement parameters ("ack" and "setErrs").

The following is a non-normative example request made by a SET Recipient that has no outstanding SETs to acknowledge and is polling for available SETs at the endpoint "https://notify.idp.example.com/Events":

```
POST /Events HTTP/1.1
Host: notify.idp.example.com
Content-Type: application/json

{
  "returnImmediately": true
}
```

Figure 1: Example Initial Poll Request

A SET Recipient can poll using default parameter values by passing an empty JSON object.

The following is a non-normative example default poll request to the endpoint "https://notify.idp.example.com/Events":

```
POST /Events HTTP/1.1
Host: notify.idp.example.com
Content-Type: application/json

{}
```

Figure 2: Example Default Poll Request

2.4.2. Acknowledge-Only Request

In this variation, the SET Recipient acknowledges previously received SETs and indicates it does not want to receive SETs in response by setting the "maxEvents" value to "0". This variation might be used, for instance, when a SET Recipient needs to acknowledge received SETs independently (e.g., on separate threads) from the process of receiving SETs.

If the poll needs to return immediately, then "returnImmediately" MUST also be present with the value "true". If it is "false", then a long poll will still occur until an event is ready to be returned, even though no events will be returned.

The following is a non-normative example poll request with acknowledgement of SETs received (for example as shown in Figure 6):

```
POST /Events HTTP/1.1
Host: notify.idp.example.com
Content-Type: application/json

{
  "ack": [
    "4d3559ec67504aaba65d40b0363faad8",
    "3d0c3cf797584bd193bd0fb1bd4e7d30"
  ],
  "maxEvents": 0,
  "returnImmediately": true
}
```

Figure 3: Example Acknowledge-Only Request

2.4.3. Poll with Acknowledgement

This variation allows a recipient thread to simultaneously acknowledge previously received SETs and wait for the next group of SETs in a single request.

The following is a non-normative example poll with acknowledgement of the SETs received in Figure 6:

```
POST /Events HTTP/1.1
Host: notify.idp.example.com
Content-Type: application/json

{
  "ack": [
    "4d3559ec67504aaba65d40b0363faad8",
    "3d0c3cf797584bd193bd0fb1bd4e7d30"
  ],
  "returnImmediately": false
}
```

Figure 4: Example Poll with Acknowledgement and No Errors

In the above acknowledgement, the SET Recipient has acknowledged receipt of two SETs and has indicated it wants to wait until the next SET is available.

2.4.4. Poll with Acknowledgement and Errors

In the case where errors were detected in previously delivered SETs, the SET Recipient MAY use the "setErrs" member to communicate the errors in the following poll request.

The following is a non-normative example of a response acknowledging one successfully received SET and one SET with an error from the two SETs received in Figure 6:

```
POST /Events HTTP/1.1
Host: notify.idp.example.com
Content-Language: en-US
Content-Type: application/json

{
  "ack": ["3d0c3cf797584bd193bd0fb1bd4e7d30"],
  "setErrs": {
    "4d3559ec67504aaba65d40b0363faad8": {
      "err": "authentication_failed",
      "description": "The SET could not be authenticated"
    }
  },
  "returnImmediately": true
}
```

Figure 5: Example Poll Acknowledgement with Error

2.5. Poll Response

In response to a valid poll request, the service provider MAY respond immediately if SETs are available to be delivered. If no SETs are available at the time of the request, the SET Transmitter SHALL delay responding until a SET is available or the timeout interval has elapsed unless the poll request parameter "returnImmediately" is present with the value "true".

As described in Section 2.3, a JSON document is returned containing members including "sets", which SHALL contain zero or more SETs.

The following is a non-normative example response to the request shown in Section 2.4. This example shows two SETs being returned:

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

```
{
  "sets": {
    "4d3559ec67504aaba65d40b0363faad8":
      "eyJhbGciOiJub25lIn0.eyJqdGkiOiI0ZDM1NTllYzY3NTA0YWFiYTUyMTQwYjAzNjNmYWVkbGciOiImbDQ5NjQwNCwiaXNzIjoiaHR0cHM6Ly9zY2ltLmV4YW1wbGUuY29tIiwiaXYkIjpjbImh0dHBzOi8vc2NpbS5leGFtcGxlLmNvbS9GZWVkcyc85OGQ1MjQ2MWZhNWJiYzg3OTU5M2I3NzU0IiwiaHR0cHM6Ly9zY2ltLmV4YW1wbGUuY29tL0ZlZWRzLzVknzYwNDUxNmIxZDA4NjQxZDc2NzZlZTciXSwicXNzZXZlbnRzIjp7InVybjppZXRmOnBhcmtfczpzY2ltOmV2ZW50OmNyZWF0ZSI6eyJyZWYiOiJodHRwciovL3NjaW0uZXhhbXBsZS5jb20vVXNlcniMvNDRMnJEMmRmOTZiZDZhYjYxZTclMjFkOSIsImF0dHJpYnV0ZXMiOlsiaWQiLCJuYWI1IiwidXNlck5hbWUiLCJwYXNzd29yZCIsImVtYWlscyJdfX19.",
    "3d0c3cf797584bd193bd0fb1bd4e7d30":
      "eyJhbGciOiJub25lIn0.eyJqdGkiOiIzZDBjM2NmNzk3NTg0YmQxOTNiZDBmYjFiZDRlN2QzMCI6ImbDQ5NjAyNSwiaXNzIjoiaHR0cHM6Ly9zY2ltLmV4YW1wbGUuY29tIiwiaXYkIjpjbImh0dHBzOi8vamh1Yi5leGFtcGxlLmNvbS9GZWVkcyc85OGQ1MjQ2MWZhNWJiYzg3OTU5M2I3NzU0IiwiaHR0cHM6Ly9qaHVhbmV4YW1wbGUuY29tL0ZlZWRzLzVknzYwNDUxNmIxZDA4NjQxZDc2NzZlZTciXSwicXNzZXZlbnRzIjoiaHR0cHM6Ly9zY2ltLmV4YW1wbGUuY29tL1VzZXJzLzQ0ZjYxNDJkZjk2YmQ2YWI2MWU3NTIxZDkiLCJldmVudHMiOmsidXJuOmllZGY6cGFyYWIzOnNjaW06ZXZlbnQ6cGFzc3dvcmRSZXNldCI6eyJpZCI6IjQ0ZjYxNDJkZjk2YmQ2YWI2MWU3NTIxZDkiSWiaHR0cHM6Ly9leGFtcGxlLmNvbS9zY2ltL2V2ZW50L3Bhc3N3b3JkUmVzZXRFbHQiOmsicmVzZXRBdHRlbXB0cyI6NX19fQ."
  }
}
```

Figure 6: Example Poll Response

In the above example, two SETs whose "jti" values are "4d3559ec67504aaba65d40b0363faad8" and "3d0c3cf797584bd193bd0fblbd4e7d30" are delivered.

The following is a non-normative example response to the request shown in Section 2.4.1, which indicates that no new SETs or unacknowledged SETs are available:

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

{
  "sets": {}
}
```

Figure 7: Example No SETs Poll Response

Upon receiving the JSON document (e.g., as shown in Figure 6), the SET Recipient parses and verifies the received SETs and notifies the SET Transmitter of successfully received SETs and SETs with errors via the next poll request to the SET Transmitter, as described in Section 2.4.3 or Section 2.4.4.

2.5.1. Poll Error Response

In the event of a general HTTP error condition in the context of processing a poll request, the service provider responds with the applicable HTTP Response Status Code, as defined in Section 6 of [RFC7231].

Service providers MAY respond to any invalid poll request with an HTTP Response Status Code of 400 (Bad Request) even when a more specific code might apply, for example if the service provider deemed that a more specific code presented an information disclosure risk. When no more specific code might apply, the service provider SHALL respond to an invalid poll request with an HTTP Status Code of 400.

The response body for responses to invalid poll requests is left undefined, and its contents SHOULD be ignored.

The following is a non-normative example of a response to an invalid poll request:

```
HTTP/1.1 400 Bad Request
```

Example Poll Error Response

2.6. Error Response Handling

If a SET is invalid, error codes from the IANA "Security Event Token Delivery Error Codes" registry established by [I-D.ietf-secevent-http-push] are used in error responses. As

described in Section 2.3 of [I-D.ietf-secevent-http-push], an error response is a JSON object providing details about the error that includes the following name/value pairs:

err

A value from the IANA "Security Event Token Delivery Error Codes" registry that identifies the error.

description

A human-readable string that provides additional diagnostic information.

When included as part of a batch of SETs, the above JSON is included as part of the "setErrs" member, as defined in Section 2.2 and Section 2.4.4.

When the SET Recipient includes one or more error responses in a request to the SET Transmitter, it must also include in the request a "Content-Language" header field whose value indicates the language of the error descriptions included in the request. The method of language selection in the case when the SET Recipient can provide error messages in multiple languages is out of scope for this specification.

3. Authentication and Authorization

The SET delivery method described in this specification is based upon HTTP over TLS [RFC2818] and standard HTTP authentication and authorization schemes, as per [RFC7235]. The TLS server certificate MUST be validated using DNS-ID [RFC6125] and/or DANE [RFC6698]. As per Section 4.1 of [RFC7235], a SET delivery endpoint SHALL indicate supported HTTP authentication schemes via the "WWW-Authenticate" header field when using HTTP authentication.

Authorization for the eligibility to provide actionable SETs can be determined by using the identity of the SET Issuer, validating the identity of the SET Transmitter, or via other employed authentication methods. Likewise, the SET Transmitter may choose to validate the identity of the SET Recipient, perhaps using mutual TLS. Because SETs are not commands, SET Recipients are free to ignore SETs that are not of interest after acknowledging their receipt.

4. Security Considerations

4.1. Authentication Using Signed SETs

JWS signed SETs can be used (see [RFC7515] and Section 5 of [RFC8417]) to enable the SET Recipient to validate that the SET Issuer is authorized to provide actionable SETs.

4.2. HTTP Considerations

SET delivery depends on the use of Hypertext Transfer Protocol and is thus subject to the security considerations of HTTP Section 9 of [RFC7230] and its related specifications.

4.3. Confidentiality of SETs

SETs may contain sensitive information, including Personally Identifiable Information (PII), or be distributed through third parties. In such cases, SET Transmitters and SET Recipients MUST protect the confidentiality of the SET contents. In some use cases, using TLS to secure the transmitted SETs will be sufficient. In other use cases, encrypting the SET as described in JWE [RFC7516] will also be required. The Event delivery endpoint MUST support at least TLS version 1.2 [RFC5246] and SHOULD support the newest version of TLS that meets its security requirements, which as of the time of this publication is TLS 1.3 [RFC8446]. The client MUST perform a TLS/SSL server certificate check using DNS-ID [RFC6125] and/or DANE [RFC6698]. How a SET Recipient determines the expected service identity to match the SET Transmitter's server certificate against is out of scope for this document. The implementation security considerations for TLS in "Recommendations for Secure Use of TLS and DTLS" [RFC7525] MUST be followed.

4.4. Access Token Considerations

If HTTP Authentication is performed using OAuth access tokens [RFC6749], implementers MUST take into account the threats and countermeasures documented in Section 8 of [RFC7521].

4.4.1. Bearer Token Considerations

Transmitting Bearer tokens [RFC6750] using TLS helps prevent their interception.

Bearer tokens SHOULD have a limited lifetime that can be determined directly or indirectly (e.g., by checking with a validation service) by the service provider. By expiring tokens, clients are forced to obtain a new token (which usually involves re-authentication) for continued authorized access. For example, in OAuth 2.0, a client MAY use an OAuth refresh token to obtain a new bearer token after

authenticating to an authorization server, per Section 6 of [RFC6749].

Implementations supporting OAuth bearer tokens need to factor in security considerations of this authorization method [RFC7521]. Since security is only as good as the weakest link, implementers also need to consider authentication choices coupled with OAuth bearer tokens. The security considerations of the default authentication method for OAuth bearer tokens, HTTP Basic, are well documented in [RFC7617], therefore implementers are encouraged to prefer stronger authentication methods.

5. Privacy Considerations

SET Transmitters should attempt to deliver SETs that are targeted to the specific business and protocol needs of subscribers.

When sharing personally identifiable information or information that is otherwise considered confidential to affected users, SET Transmitters and Recipients MUST have the appropriate legal agreements and user consent or terms of service in place. Furthermore, data that needs confidentiality protection MUST be encrypted, at least with TLS and sometimes also using JSON Web Encryption (JWE) [RFC7516].

In some cases, subject identifiers themselves may be considered sensitive information, such that their inclusion within a SET may be considered a violation of privacy. SET Issuers and SET Transmitters should consider the ramifications of sharing a particular subject identifier with a SET Recipient (e.g., whether doing so could enable correlation and/or de-anonymization of data) and choose appropriate subject identifiers for their use cases.

6. IANA Considerations

This specification requires no IANA actions.

7. References

7.1. Normative References

[I-D.ietf-secevent-http-push]

Backman, A., Jones, M., Scurtescu, M., Ansari, M., and A. Nadalin, "Push-Based Security Event Token (SET) Delivery Using HTTP", draft-ietf-secevent-http-push-12 (work in progress), June 2020.

- [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>>.
- [RFC2818] Rescorla, E., "HTTP Over TLS", RFC 2818, DOI 10.17487/RFC2818, May 2000, <<https://www.rfc-editor.org/info/rfc2818>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <<https://www.rfc-editor.org/info/rfc6125>>.
- [RFC6698] Hoffman, P. and J. Schlyter, "The DNS-Based Authentication of Named Entities (DANE) Transport Layer Security (TLS) Protocol: TLSA", RFC 6698, DOI 10.17487/RFC6698, August 2012, <<https://www.rfc-editor.org/info/rfc6698>>.
- [RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <<https://www.rfc-editor.org/info/rfc7231>>.
- [RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <<https://www.rfc-editor.org/info/rfc7515>>.
- [RFC7516] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, DOI 10.17487/RFC7516, May 2015, <<https://www.rfc-editor.org/info/rfc7516>>.
- [RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <<https://www.rfc-editor.org/info/rfc7519>>.
- [RFC7521] Campbell, B., Mortimore, C., Jones, M., and Y. Goland, "Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants", RFC 7521, DOI 10.17487/RFC7521, May 2015, <<https://www.rfc-editor.org/info/rfc7521>>.

- [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <<https://www.rfc-editor.org/info/rfc7525>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [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/info/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/info/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/info/rfc8446>>.

7.2. Informative References

- [RFC6202] Loreto, S., Saint-Andre, P., Salsano, S., and G. Wilkins, "Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP", RFC 6202, DOI 10.17487/RFC6202, April 2011, <<https://www.rfc-editor.org/info/rfc6202>>.
- [RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <<https://www.rfc-editor.org/info/rfc6749>>.
- [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>>.
- [RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <<https://www.rfc-editor.org/info/rfc7230>>.

- [RFC7235] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, DOI 10.17487/RFC7235, June 2014, <<https://www.rfc-editor.org/info/rfc7235>>.
- [RFC7617] Reschke, J., "The 'Basic' HTTP Authentication Scheme", RFC 7617, DOI 10.17487/RFC7617, September 2015, <<https://www.rfc-editor.org/info/rfc7617>>.

Appendix A. Unencrypted Transport Considerations

Earlier versions of this specification made the use of TLS optional and described security and privacy considerations resulting from use of unencrypted HTTP as the underlying transport. When the working group decided to mandate usage HTTP over TLS, it also decided to preserve the description of these considerations in a non-normative manner.

The considerations for using unencrypted HTTP with this protocol are the same as those described in Appendix A of [I-D.ietf-secevent-http-push], and are therefore not repeated here.

Appendix B. Other Streaming Specifications

[[NOTE TO THE RFC EDITOR: This section to be removed prior to publication]]

A number of pub/sub, queuing, and streaming systems were reviewed as possible solutions or as input to the current draft. These are listed in Appendix B of [I-D.ietf-secevent-http-push], and are therefore not repeated here.

Appendix C. Acknowledgments

The editors would like to thank the members of the SCIM working group, which began discussions of provisioning events starting with draft-hunt-scim-notify-00 in 2015. We would like to thank Phil Hunt and the other the authors of draft-ietf-secevent-delivery-02, upon which this specification is based. We would like to thank the participants in the SecEvents working group for their contributions to this specification.

Additionally, we would like to thank the following individuals for their reviews of the specification: Roman Danyliw, Martin Duke, Benjamin Kaduk, Erik Kline, Murray Kucherawy, Warren Kumari, Barry Leiba, Mark Nottingham, Alvaro Retana, Yaron Sheffer, Valery Smyslov, Robert Sparks, Eric Vyncke, and Robert Wilton.

Appendix D. Change Log

[[to be removed by the RFC Editor before publication as an RFC]]

Draft 00 - AB - Based on draft-ietf-secevent-delivery-02 with the following additions:

- o Renamed to "Poll-Based SET Token Delivery Using HTTP"
- o Removed references to the HTTP Push delivery method.

Draft 01 - mbj:

- o Addressed problems identified in my 18-Jul-18 review message titled "Issues for both the Push and Poll Specs".
- o Changes to align terminology with RFC 8417, for instance, by using the already defined term SET Recipient rather than SET Receiver.
- o Applied editorial and minor normative corrections.
- o Updated Marius' contact information.
- o Begun eliminating redundancies between this specification and "Push-Based Security Event Token (SET) Delivery Using HTTP" [I-D.ietf-secevent-http-push], referencing, rather than duplicating common normative text.

Draft 02 - mbj:

- o Removed vestigial language remaining from when the push and poll delivery methods were defined in a common specification.
- o Replaced remaining uses of the terms Event Transmitter and Event Recipient with the correct terms SET Transmitter and SET Recipient.
- o Removed uses of the unnecessary term "Event Stream".
- o Removed dependencies between the semantics of "maxEvents" and "returnImmediately".
- o Said that PII in SETs is to be encrypted with TLS, JWE, or both.
- o Corrected grammar and spelling errors.

Draft 03 - mbj:

- o Corrected uses of "attribute" to "member" when describing JSON objects.
- o Further alignment with the push draft.

Draft 04 - AB + mbj

- o Referenced SET Transmitter definition in http-push.
- o Removed incorrect normative text regarding SET construction.
- o Consolidated general out-of-scope items under Introduction.
- o Removed unnecessary HTTP headers in examples and added Content-Type.
- o Added Content-Language requirement for error descriptions, aligning with http-push.
- o Stated that bearer tokens SHOULD have a limited lifetime.
- o Minor editorial fixes.

Draft 05 - AB + mbj

- o Added normative text defining how to respond to invalid poll requests.
- o Addressed shepherd comments by Yaron Sheffer.

Draft 06 - mbj

- o Addressed nits identified by the idnits tool.

Draft 07 - mbj

- o Addressed area director review comments by Benjamin Kaduk.

Draft 08 - mbj + AB

- o Corrected editorial nits.

Draft 09 - AB

- o Addressed area director review comments by Benjamin Kaduk:
 - * Added text clarifying that determining the SET Recipient's service identity is out of scope.

- * Removed unelaborated reference to use of authentication to prevent DoS attacks.

Draft 10 - mbj

- o Addressed SecDir review comments by Valery Smyslov on draft-ietf-secevent-http-push-10 that also applied here.
- o Addressed IETF last call comments by Mark Nottingham.
- o Addressed GenArt review comments by Robert Sparks.

Draft 11 - mbj

- o Revised to unambiguously require the use of TLS, while preserving descriptions of precautions needed for non-TLS use in an appendix.

Draft 12 - mbj

- o Addressed IESG comments.

Authors' Addresses

Annabelle Backman (editor)
Amazon

Email: richanna@amazon.com

Michael B. Jones (editor)
Microsoft

Email: mbj@microsoft.com
URI: <https://self-issued.info/>

Marius Scurtescu
Coinbase

Email: marius.scurtescu@coinbase.com

Morteza Ansari
Cisco

Email: morteza.ansari@cisco.com

Anthony Nadalin
Microsoft

Email: tonynad@microsoft.com

Security Events Working Group
Internet-Draft
Intended status: Standards Track
Expires: 23 October 2022

A. Backman, Ed.
Amazon
M. Scurtescu
Coinbase
P. Jain
Fastly
21 April 2022

Subject Identifiers for Security Event Tokens
draft-ietf-secevent-subject-identifiers-11

Abstract

Security events communicated within Security Event Tokens may support a variety of identifiers to identify subjects related to the event. This specification formalizes the notion of subject identifiers as structured information that describe a subject, and named formats that define the syntax and semantics for encoding subject identifiers as JSON objects. It also defines a registry for defining and allocating names for such formats, as well as the sub_id JSON Web Token (JWT) claim.

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 23 October 2022.

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	3
2. Notational Conventions	5
2.1. Definitions	5
3. Subject Identifiers	5
3.1. Identifier Formats versus Principal Types	6
3.2. Identifier Format Definitions	6
3.2.1. Account Identifier Format	6
3.2.2. Email Identifier Format	7
3.2.3. Issuer and Subject Identifier Format	7
3.2.4. Opaque Identifier Format	8
3.2.5. Phone Number Identifier Format	8
3.2.6. Aliases Identifier Format	9
4. Subject Identifiers in JWTs	10
4.1. sub_id Claim	10
4.2. sub_id and iss_sub Subject Identifiers	11
5. Considerations for Specifications that Define Identifier Formats	13
6. Privacy Considerations	13
6.1. Identifier Correlation	13
7. Security Considerations	13
7.1. Confidentiality and Integrity	13
8. IANA Considerations	14
8.1. Security Event Identifier Formats Registry	14
8.1.1. Registry Location	14
8.1.2. Registration Template	14
8.1.3. Initial Registry Contents	15
8.1.4. Guidance for Expert Reviewers	16
8.2. JSON Web Token Claims Registration	17
8.2.1. Registry Contents	17
9. References	17
9.1. Normative References	17
9.2. Informative References	18
Acknowledgements	19
Change Log	19
Authors' Addresses	22

1. Introduction

As described in Section 1.2 of SET [RFC8417], subjects related to security events may take a variety of forms, including but not limited to a JWT [RFC7519] principal, an IP address, a URL, etc. Different types of subjects may need to be identified in different ways (e.g., a host might be identified by an IP or MAC address, while a user might be identified by an email address). Furthermore, even in the case where the type of the subject is known, there may be multiple ways by which a given subject may be identified. For example, an account may be identified by an opaque identifier, an email address, a phone number, a JWT iss claim and sub claim, etc., depending on the nature and needs of the transmitter and receiver. Even within the context of a given transmitter and receiver relationship, it may be appropriate to identify different accounts in different ways, for example if some accounts only have email addresses associated with them while others only have phone numbers. Therefore it can be necessary to indicate within a SET the mechanism by which a subject is being identified.

To address this problem, this specification defines Subject Identifiers - JSON [RFC7159] objects containing information identifying a subject - and Identifier Formats - named sets of rules describing how to encode different kinds of subject identifying information (e.g., an email address, or an issuer and subject pair) as a Subject Identifier.

Below is a non-normative example of a Subject Identifier that identifies a subject by email address, using the Email Identifier Format.

```
{
  "format": "email",
  "email": "user@example.com"
}
```

Figure 1: Example: Subject Identifier using the Email Identifier Format

Subject Identifiers are intended to be a general-purpose mechanism for identifying subjects within JSON objects and their usage need not be limited to SETs. Below is a non-normative example of a JWT that uses a Subject Identifier in the sub_id claim (defined in this specification) to identify the JWT Subject.

```
{
  "iss": "issuer.example.com",
  "sub_id": {
    "format": "phone_number",
    "phone_number": "+12065550100"
  }
}
```

Figure 2: Example: JWT using a Subject Identifier with the "sub_id" claim

Usage of Subject Identifiers also need not be limited to identifying JWT Subjects. They are intended as a general-purpose means of expressing identifying information in an unambiguous manner. Below is a non-normative example of a SET containing a hypothetical security event describing the interception of a message, using Subject Identifiers to identify the sender, intended recipient, and interceptor.

```
{
  "iss": "issuer.example.com",
  "iat": 1508184845,
  "aud": "aud.example.com",
  "events": {
    "https://secevent.example.com/events/message-interception": {
      "from": {
        "format": "email",
        "email": "alice@example.com"
      },
      "to": {
        "format": "email",
        "email": "bob@example.com"
      },
      "interceptor": {
        "format": "email",
        "email": "eve@example.com"
      }
    }
  }
}
```

Figure 3: Example: SET with an event payload containing multiple Subject Identifiers

2. Notational Conventions

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].

2.1. Definitions

This specification utilizes terminology defined in [RFC7159] and [RFC8417].

Within this specification, the terms "Subject" and "subject" refer generically to anything being identified via one or more pieces of information. The term "JWT Subject" refers specifically to the to the subject of a JWT. (i.e., the subject that the JWT asserts claims about)

3. Subject Identifiers

A Subject Identifier is a JSON [RFC7159] object whose contents may be used to identify a subject within some context. An Identifier Format is a named definition of a set of information that may be used to identify a subject, and the rules for encoding that information as a Subject Identifier; they define the syntax and semantics of Subject Identifiers. A Subject Identifier MUST conform to a specific Identifier Format, and MUST contain a format member whose value is the name of that Identifier Format.

Every Identifier Format MUST have a unique name registered in the IANA "Security Event Identifier Formats" registry established by Section 8.1, or a Collision-Resistant Name as defined in [RFC7519]. Identifier Formats that are expected to be used broadly by a variety of parties SHOULD be registered in the "Security Event Identifier Formats" registry.

An Identifier Format MAY describe more members than are strictly necessary to identify a subject, and MAY describe conditions under which those members are required, optional, or prohibited. The format member is reserved for use as described in this specification; Identifier Formats MUST NOT declare any rules regarding the format member.

Every member within a Subject Identifier MUST match the rules specified for that member by this specification or by Subject Identifier's Identifier Format. A Subject Identifier MUST NOT contain any members prohibited or not described by its Identifier Format, and MUST contain all members required by its Identifier Format.

3.1. Identifier Formats versus Principal Types

Identifier Formats define how to encode identifying information for a subject. They do not define the type or nature of the subject itself. E.g., While the email Identifier Format declares that the value of the email member is an email address, a subject in a Security Event that is identified by an email Subject Identifier could be an end user who controls that email address, the mailbox itself, or anything else that the transmitter and receiver both understand to be associated with that email address. Consequently Subject Identifiers remove ambiguity around how a subject is being identified, and how to parse an identifying structure, but do not remove ambiguity around how to resolve that identifier to a subject. For example, consider a directory management API that allows callers to identify users and groups through both opaque unique identifiers and email addresses. Such an API could use Subject Identifiers to disambiguate between which of these two types of identifiers is in use. However, the API would have to determine whether the subject is a user or group via some other means, such as by querying a database, interpreting other parameters in the request, or inferring the type from the API contract.

3.2. Identifier Format Definitions

The following Identifier Formats are registered in the IANA "Security Event Identifier Formats" registry established by Section 8.1.

3.2.1. Account Identifier Format

The Account Identifier Format identifies a subject using an account at a service provider, identified with an acct URI as defined in [RFC7565]. Subject Identifiers in this format MUST contain a uri member whose value is the acct URI for the subject. The uri member is REQUIRED and MUST NOT be null or empty. The Account Identifier Format is identified by the name account.

Below is a non-normative example Subject Identifier for the Account Identifier Format:

```
{
  "format": "account",
  "uri": "acct:example.user@service.example.com"
}
```

Figure 4: Example: Subject Identifier for the Account Identifier Format

3.2.2. Email Identifier Format

The Email Identifier Format identifies a subject using an email address. Subject Identifiers in this format MUST contain an email member whose value is a string containing the email address of the subject, formatted as an addr-spec as defined in Section 3.4.1 of [RFC5322]. The email member is REQUIRED and MUST NOT be null or empty. The value of the email member SHOULD identify a mailbox to which email may be delivered, in accordance with [RFC5321]. The Email Identifier Format is identified by the name email.

Below is a non-normative example Subject Identifier in the Email Identifier Format:

```
{
  "format": "email",
  "email": "user@example.com"
}
```

Figure 5: Example: Subject Identifier in the Email Identifier Format

3.2.2.1. Email Canonicalization

Many email providers will treat multiple email addresses as equivalent. While the domain portion of an [RFC5322] email address is consistently treated as case-insensitive per [RFC1034], some providers treat the local part of the email address as case-insensitive as well, and consider "user@example.com", "User@example.com", and "USER@example.com" as the same email address. This has led users to view these strings as equivalent, driving service providers to implement proprietary email canonicalization algorithms to ensure that email addresses entered by users resolve to the same canonical string. When receiving an Email Subject Identifier, the recipient SHOULD use their implementation's canonicalization algorithm to resolve the email address to the same string used in their system.

3.2.3. Issuer and Subject Identifier Format

The Issuer and Subject Identifier Format identifies a subject using a pair of iss and sub members, analagous to how subjects are identified using the iss and sub claims in OpenID Connect [OpenID.Core] ID Tokens. These members MUST follow the formats of the iss member and sub member defined by [RFC7519], respectively. Both the iss member and the sub member are REQUIRED and MUST NOT be null or empty. The Issuer and Subject Identifier Format is identified by the name iss_sub.

Below is a non-normative example Subject Identifier in the Issuer and Subject Identifier Format:

```
{
  "format": "iss_sub",
  "iss": "http://issuer.example.com/",
  "sub": "145234573"
}
```

Figure 6: Example: Subject Identifier in the Issuer and Subject Identifier Format

3.2.4. Opaque Identifier Format

The Opaque Identifier Format describes a subject that is identified with a string with no semantics asserted beyond its usage as an identifier for the subject, such as a UUID or hash used as a surrogate identifier for a record in a database. Subject Identifiers in this format MUST contain an `id` member whose value is a JSON string containing the opaque string identifier for the subject. The `id` member is REQUIRED and MUST NOT be null or empty. The Opaque Identifier Format is identified by the name `opaque`.

Below is a non-normative example Subject Identifier in the Opaque Identifier Format:

```
{
  "format": "opaque",
  "id": "11112222333344445555"
}
```

Figure 7: Example: Subject Identifier in the Opaque Identifier Format

3.2.5. Phone Number Identifier Format

The Phone Number Identifier Format identifies a subject using a telephone number. Subject Identifiers in this format MUST contain a `phone_number` member whose value is a string containing the full telephone number of the subject, including international dialing prefix, formatted according to E.164 [E164]. The `phone_number` member is REQUIRED and MUST NOT be null or empty. The Phone Number Identifier Format is identified by the name `phone_number`.

Below is a non-normative example Subject Identifier in the Email Identifier Format:

```
{
  "format": "phone_number",
  "phone_number": "+12065550100"
}
```

Figure 8: Example: Subject Identifier in the Phone Number Identifier Format

3.2.6. Aliases Identifier Format

The Aliases Identifier Format describes a subject that is identified with a list of different Subject Identifiers. It is intended for use when a variety of identifiers have been shared with the party that will be interpreting the Subject Identifier, and it is unknown which of those identifiers they will recognize or support. Subject Identifiers in this format MUST contain an `identifiers` member whose value is a JSON array containing one or more Subject Identifiers. Each Subject Identifier in the array MUST identify the same entity. The `identifiers` member is REQUIRED and MUST NOT be null or empty. It MAY contain multiple instances of the same Identifier Format (e.g., multiple Email Subject Identifiers), but SHOULD NOT contain exact duplicates. This format is identified by the name `aliases`.

`aliases` Subject Identifiers MUST NOT be nested; i.e., the `identifiers` member of an `aliases` Subject Identifier MUST NOT contain a Subject Identifier in the `aliases` format.

Below is a non-normative example Subject Identifier in the Aliases Identifier Format:

```
{
  "format": "aliases",
  "identifiers": [
    {
      "format": "email",
      "email": "user@example.com"
    },
    {
      "format": "phone_number",
      "phone_number": "+12065550100"
    },
    {
      "format": "email",
      "email": "user+qualifier@example.com"
    }
  ]
}
```

Figure 9: Example: Subject Identifier in the Aliases Identifier Format

4. Subject Identifiers in JWTs

4.1. sub_id Claim

The sub JWT Claim is defined in Section 4.1.2 of [RFC7519] as containing a string value, and therefore cannot contain a Subject Identifier (which is a JSON object) as its value. This document defines the sub_id JWT Claim, in accordance with Section 4.2 of [RFC7519], as a common claim that identifies the JWT Subject using a Subject Identifier. When present, the value of this claim MUST be a Subject Identifier that identifies the subject of the JWT. The sub_id claim MAY be included in a JWT, whether or not the sub claim is present. When both the sub and sub_id claims are present in a JWT, they MUST identify the same subject, as a JWT has one and only one JWT Subject.

When processing a JWT with both sub and sub_id claims, implementations MUST NOT rely on both claims to determine the JWT Subject. An implementation MAY attempt to determine the JWT Subject from one claim and fall back to using the other if it determines it does not understand the format of the first claim. For example, an implementation may attempt to use sub_id, and fall back to using sub upon finding that sub_id contains a Subject Identifier whose format is not recognized by the implementation.

Below are non-normative examples of JWTs containing the sub_id claim:

```
{
  "iss": "issuer.example.com",
  "sub_id": {
    "format": "email",
    "email": "user@example.com"
  }
}
```

Figure 10: Example: JWT containing a "sub_id" claim and no "sub" claim


```
{
  "iss": "issuer.example.com",
  "sub": "user@example.com",
  "sub_id": {
    "format": "email",
    "email": "user@example.com"
  }
}
```

Figure 11: Example: JWT where both the "sub" and "sub_id" claims identify the JWT Subject using the same identifier

```
{
  "iss": "issuer.example.com",
  "sub": "liz@example.com",
  "sub_id": {
    "format": "email",
    "email": "elizabeth@example.com"
  }
}
```

Figure 12: Example: JWT where both the "sub" and "sub_id" claims identify the JWT Subject using different values of the same identifier type

```
{
  "iss": "issuer.example.com",
  "sub": "user@example.com",
  "sub_id": {
    "format": "account",
    "uri": "acct:example.user@service.example.com"
  }
}
```

Figure 13: Example: JWT where the "sub" and "sub_id" claims identify the JWT Subject via different types of identifiers

4.2. sub_id and iss_sub Subject Identifiers

The sub_id claim MAY contain an iss_sub Subject Identifier. In this case, the JWT's iss claim and the Subject Identifier's iss member MAY be different. For example, in OpenID Connect [OpenID.Core] client may construct such a JWT when sending JWTs back to its OpenID Connect Identity Provider, in order to identify the JWT Subject using an identifier known to be understood by both parties. Similarly, the JWT's sub claim and the Subject Identifier's sub member MAY be different. For example, this may be used by an OpenID Connect client to communicate the JWT Subject's local identifier at the client back

to its Identity Provider.

Below are non-normative examples of a JWT where the iss claim and iss member within the sub_id claim are the same, and a JWT where they are different.

```
{
  "iss": "issuer.example.com",
  "sub_id": {
    "format": "iss_sub",
    "iss": "issuer.example.com",
    "sub": "example_user"
  }
}
```

Figure 14: Example: JWT with an "iss_sub" Subject Identifier
where JWT issuer and JWT Subject issuer are the same

```
{
  "iss": "client.example.com",
  "sub_id": {
    "format": "iss_sub",
    "iss": "issuer.example.com",
    "sub": "example_user"
  }
}
```

Figure 15: Example: JWT with an "iss_sub" Subject Identifier
where the JWT issuer and JWT Subject issuer are different

```
{
  "iss": "client.example.com",
  "sub": "client_user",
  "sub_id": {
    "format": "iss_sub",
    "iss": "issuer.example.com",
    "sub": "example_user"
  }
}
```

Figure 16: Example: JWT with an "iss_sub" Subject Identifier
where the JWT "iss" and "sub" claims differ from the JWT
Subject's "iss" and "sub" members

5. Considerations for Specifications that Define Identifier Formats

Identifier Format definitions MUST NOT make assertions or declarations regarding the subject being identified by the Subject Identifier (e.g., an Identifier Format cannot be defined as specifically identifying human end users), as such statements are outside the scope of Identifier Formats and Subject Identifiers, and expanding that scope for some Identifier Formats but not others would harm interoperability, as applications that depend on this expanded scope to disambiguate the subject type would be unable to use Identifier Formats that do not provide such rules.

6. Privacy Considerations

6.1. Identifier Correlation

The act of presenting two or more identifiers for a single subject together (e.g., within an aliases Subject Identifier, or via the sub and sub_id JWT claims) may communicate more information about the subject than was intended. For example, the entity to which the identifiers are presented now knows that both identifiers relate to the same subject, and may be able to correlate additional data based on that. When transmitting Subject Identifiers, the transmitter SHOULD take care that they are only transmitting multiple identifiers together when it is known that the recipient already knows that the identifiers are related (e.g., because they were previously sent to the recipient as claims in an OpenID Connect ID Token), or when correlation is essential to the use case. Implementers must consider such risks, and specs that use subject identifiers must provide appropriate privacy considerations of their own.

The considerations described in Section 6 of [RFC8417] also apply when Subject Identifiers are used within SETs. The considerations described in Section 12 of [RFC7519] also apply when Subject Identifiers are used within JWTs.

7. Security Considerations

7.1. Confidentiality and Integrity

This specification does not define any mechanism for ensuring the confidentiality or integrity of a Subject Identifier. Where such properties are required, implementations MUST use mechanisms provided by the containing format (e.g., integrity protecting SETs or JWTs using JWS [RFC7515]), or at the transport layer or other layer in the application stack (e.g., using TLS [RFC8446]).

Further considerations regarding confidentiality and integrity of SETs can be found in Section 5.1 of [RFC8417].

8. IANA Considerations

8.1. Security Event Identifier Formats Registry

This document defines Identifier Formats, for which IANA is asked to create and maintain a new registry titled "Security Event Identifier Formats". Initial values for the Security Event Identifier Formats registry are given in Section 3. Future assignments are to be made through the Expert Review registration policy [BCP26] and shall follow the template presented in Section 8.1.2.

It is suggested that multiple Designated Experts be appointed who are able to represent the perspectives of different applications using this specification, in order to enable broadly informed review of registration decisions. In cases where a registration decision could be perceived as creating a conflict of interest for a particular Expert, that Expert should defer to the judgment of the other Experts.

8.1.1. Registry Location

(This section to be removed by the RFC Editor before publication as an RFC.)

The authors recommend that the Identifier Formats registry be located at <https://www.iana.org/assignments/secevent/>.

8.1.2. Registration Template

Format Name

The name of the Identifier Format, as described in Section 3. The name MUST be an ASCII string consisting only of lower-case characters ("a" - "z"), digits ("0" - "9"), underscores ("_"), and hyphens ("-"), and SHOULD NOT exceed 20 characters in length.

Format Description

A brief description of the Identifier Format.

Change Controller

For formats defined in documents published by the IETF or its working groups, list "IETF". For all other formats, list the name of the party responsible for the registration. Contact information such as mailing address, email address, or phone number may also be provided.

Defining Document(s)

A reference to the document or documents that define the Identifier Format. The definition MUST specify the name, format, and meaning of each member that may occur within a Subject Identifier of the defined format, as well as whether each member is optional, required, prohibited, or the circumstances under which the member may be optional, required, or prohibited. URIs that can be used to retrieve copies of each document SHOULD be included.

8.1.3. Initial Registry Contents

8.1.3.1. Account Identifier Format

- * Format Name: "account"
- * Format Description: Subject identifier based on acct URI.
- * Change Controller: IETF
- * Defining Document(s): Section 3 of this document.

8.1.3.2. Decentralized Identifier Format

- * Format Name: "did"
- * Format Description: Subject identifier based on a Decentralized Identifier (DID) URL.
- * Change Controller: IETF
- * Defining Document(s): Section 3 of this document.

8.1.3.3. Email Identifier Format

- * Format Name: email
- * Format Description: Subject identifier based on email address.
- * Change Controller: IETF
- * Defining Document(s): Section 3 of this document.

8.1.3.4. Issuer and Subject Identifier Format

- * Format Name: "iss_sub"

- * Format Description: Subject identifier based on an issuer and subject.
- * Change Controller: IETF
- * Defining Document(s): Section 3 of this document.

8.1.3.5. Opaque Identifier Format

- * Format Name: "opaque"
- * Format Description: Subject identifier based on an opaque string.
- * Change Controller: IETF
- * Defining Document(s): Section 3 of this document.

8.1.3.6. Phone Number Identifier Format

- * Format Name: "phone_number"
- * Format Description: Subject identifier based on an phone number.
- * Change Controller: IETF
- * Defining Document(s): Section 3 of this document.

8.1.3.7. Aliases Identifier Format

- * Format Name: "aliases"
- * Format Description: Subject identifier that groups together multiple different subject identifiers for the same subject.
- * Change Controller: IETF
- * Defining Document(s): Section 3 of this document.

8.1.4. Guidance for Expert Reviewers

The Expert Reviewer is expected to review the documentation referenced in a registration request to verify its completeness. The Expert Reviewer must base their decision to accept or reject the request on a fair and impartial assessment of the request. If the Expert Reviewer has a conflict of interest, such as being an author of a defining document referenced by the request, they must recuse themselves from the approval process for that request. In the case where a request is rejected, the Expert Reviewer should provide the

requesting party with a written statement expressing the reason for rejection, and be prepared to cite any sources of information that went into that decision.

Identifier Formats need not be generally applicable and may be highly specific to a particular domain; it is expected that formats may be registered for niche or industry-specific use cases. The Expert Reviewer should focus on whether the format is thoroughly documented, and whether its registration will promote or harm interoperability. In most cases, the Expert Reviewer should not approve a request if the registration would contribute to confusion, or amount to a synonym for an existing format.

8.2. JSON Web Token Claims Registration

This document defines the sub_id JWT Claim, which IANA is asked to register in the "JSON Web Token Claims" registry IANA JSON Web Token Claims Registry [IANA.JWT.Claims] established by [RFC7519].

8.2.1. Registry Contents

- * Claim Name: "sub_id"
- * Claim Description: Subject Identifier
- * Change Controller: IESG
- * Specification Document(s): Section 4.1 of this document.

9. References

9.1. Normative References

- [BCP26] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <<https://www.rfc-editor.org/info/rfc8126>>.
- [DID] World Wide Web Consortium (W3C), "Decentralized Identifiers (DIDs) v1.0", 2021, <<https://www.w3.org/TR/did-core/>>.
- [E164] International Telecommunication Union, "The international public telecommunication numbering plan", 2010, <<http://www.itu.int/rec/T-REC-E.164-201011-I/en>>.

- [IANA.JWT.Claims]
IANA, "JSON Web Token Claims", n.d.,
<<http://www.iana.org/assignments/jwt>>.
- [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>>.
- [RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, DOI 10.17487/RFC5321, October 2008, <<https://www.rfc-editor.org/info/rfc5321>>.
- [RFC5322] Resnick, P., Ed., "Internet Message Format", RFC 5322, DOI 10.17487/RFC5322, October 2008, <<https://www.rfc-editor.org/info/rfc5322>>.
- [RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <<https://www.rfc-editor.org/info/rfc7159>>.
- [RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <<https://www.rfc-editor.org/info/rfc7519>>.
- [RFC7565] Saint-Andre, P., "The 'acct' URI Scheme", RFC 7565, DOI 10.17487/RFC7565, May 2015, <<https://www.rfc-editor.org/info/rfc7565>>.
- [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/info/rfc8417>>.

9.2. Informative References

- [OpenID.Core]
Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and C. Mortimore, "OpenID Connect Core 1.0", November 2014, <http://openid.net/specs/openid-connect-core-1_0.html>.
- [RFC1034] Mockapetris, P., "Domain names - concepts and facilities", STD 13, RFC 1034, DOI 10.17487/RFC1034, November 1987, <<https://www.rfc-editor.org/info/rfc1034>>.
- [RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <<https://www.rfc-editor.org/info/rfc7515>>.

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

Acknowledgements

The authors would like to thank the members of the IETF Security Events working group, as well as those of the OpenID Shared Signals and Events Working Group, whose work provided the original basis for this document. We would also like to acknowledge Aaron Parecki, Denis Pinkas, Justin Richer, Mike Jones and other members of the working group for reviewing this document.

Change Log

(This section to be removed by the RFC Editor before publication as an RFC.)

Draft 00 - AB - First draft

Draft 01 - AB:

- * Added reference to RFC 5322 for format of email claim.
- * Renamed iss_sub type to iss-sub.
- * Renamed id_token_claims type to id-token-claims.
- * Added text specifying the nature of the subjects described by each type.

Draft 02 - AB:

- * Corrected format of phone numbers in examples.
- * Updated author info.

Draft 03 - AB:

- * Added account type for acct URIs.
- * Replaced id-token-claims type with aliases type.
- * Added email canonicalization guidance.
- * Updated semantics for email, phone, and iss-sub types.

Draft 04 - AB:

- * Added sub_id JWT Claim definition, guidance, examples.
- * Added text prohibiting aliases nesting.
- * Added privacy considerations for identifier correlation.

Draft 05 - AB:

- * Renamed the phone type to phone-number and its phone claim to phone_number.

Draft 06 - AB:

- * Replaced usage of the word "claim" to describe members of a Subject Identifier with the word "member", in accordance with terminology in RFC7159.
- * Renamed the phone-number type to phone_number and iss-sub to iss_sub.
- * Added normative requirements limiting the use of both sub and sub_id claims together when processing a JWT.
- * Clarified that identifier correlation may be acceptable when it is a core part of the use case.
- * Replaced references to OIDF with IETF in IANA Considerations.
- * Recommended the appointment of multiple Designated Experts, and a location for the Subject Identifier Types registry.
- * Added "_" to list of allowed characters in the Type Name for Subject Identifier Types.
- * Clarified that Subject Identifiers don't provide confidentiality or integrity protection.
- * Added references to SET, JWT privacy and security considerations.
- * Added section describing the difference between subject identifier type and principal type that hopefully clarifies things and doesn't just muddy the water further.

Draft 07 - AB:

- * Emphasized that the spec is about identifiers, not the things they identify:

- Renamed "Subject Identifier Type" to "Identifier Format".
- Renamed subject_type to format.
- Renamed "Security Event Subject Identifier Type Registry" to "Security Event Identifier Format Registry".
- Added new section with guidance for specs defining Identifier Formats, with normative prohibition on formats that describe the subject itself, rather than the identifier.
- * Clarified the meaning of "subject":
 - Defined "subject" as applying generically and "JWT Subject" as applying specifically to the subject of a JWT.
 - Replaced most instances of the word "principal" with "subject".
- * Added opaque Identifier Format

Draft 08 - JR, AB:

- * Added did Identifier Format
- * Alphabetized identifier format definitions
- * Replaced "type" with "format" in places that had been missed in the -07 change. (mostly IANA Considerations)
- * Miscellaneous editorial fixes

Draft 09 - AB:

- * Miscellaneous editorial fixes

Draft 10 - PJ:

- * Added author
- * Editorial nits

Draft 11 - PJ:

- * Miscellaneous editorial fixes
- * Moved aliases to the last in identifier format definitions
- * Acknowledged individual reviewers

Authors' Addresses

Annabelle Backman (editor)
Amazon
Email: richanna@amazon.com

Marius Scurtescu
Coinbase
Email: marius.scurtescu@coinbase.com

Prachi Jain
Fastly
Email: prachi.jain1288@gmail.com