Poll-Based Security Event Token (SET) Delivery Using HTTP

draft-ietf-secevent-http-poll-03

Abstract

This specification defines how a series of Security Event Tokens (SETs) may 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 January 9, 2020.

Copyright Notice

Copyright (c) 2019 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
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.
A mechanism for exchanging configuration metadata such as endpoint
URLs and cryptographic key parameters between the transmitter and
recipient is out of scope for this specification.

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], as well
as the terms defined below:

SET Transmitter
An entity that delivers SETs in its possession to one or more SET
Recipients.

2. SET Delivery

When an event occurs, the SET Transmitter constructs a SET [RFC8417]
that describes the event. The SET Transmitter determines the SET
Recipients that the SET should be distributed to.

How SETs are defined and the process by which events are identified
for SET Recipients is out of scope of this specification.

When a SET is available for a SET Recipient, the SET Transmitter
attempts to deliver the SET by queuing the SET in a buffer so that a
SET Recipient can poll for SETs using HTTP/1.1 POST.

In Poll-Based SET Delivery Using HTTP, 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 be required to retain or record SETs for retransmission. Once a
SET is acknowledged, the SET Recipient SHALL be responsible for
retention, if needed.
Transmitted SETs SHOULD be self-validating (signed) if there is a requirement to verify they were issued by the SET Transmitter at a later date when de-coupled from the original delivery where authenticity could be checked via the HTTP or TLS mutual authentication.

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. The SET Recipient SHALL NOT use the event acknowledgement mechanism to report event errors other than 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].

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

- The SET Recipient makes a request for available SETs using an HTTP POST to a pre-arranged endpoint provided by the SET Transmitter or,

- after validating previously received SETs, the SET Recipient initiates another poll request using HTTP POST that includes acknowledgement of previous SETs and waits for 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 SHOULD ensure that received SETs have been validated and retained 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. The request payloads are delivered in a JSON document, as described in Section 2.4 and Section 2.5.
When making a request, the HTTP header "Content-Type" is set to "application/json".

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

Request Processing Parameters

maxEvents
An OPTIONAL JSON integer value indicating the maximum number of unacknowledged SETs that SHOULD be returned. If more than the maximum number of SETs are available, the oldest SETs available SHOULD be returned 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
An array of strings that each corresponds to the "jti" of a successfully received SET. If there are no outstanding SETs to acknowledge, the member MAY be omitted. When acknowledging a SET, the SET Transmitter is released from any obligation to retain the SET.

setErrs
A JSON Object that contains one or more nested JSON object members that correspond to the "jti" of each invalid SET received. The value of each is a JSON object whose contents is an "err" member and "description" member, whose values correspond to the errors described in Section 2.6.

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 that contains zero or more nested JSON objects. Each nested JSON object corresponds to the "jti" of a SET to be delivered and whose value is a JSON string containing the value of the encoded corresponding SET. If there are no outstanding SETs to be transmitted, the JSON object SHALL be empty.

moreAvailable

A JSON boolean value that indicates if more unacknowledged SETs are available to be returned.

When making a response, the HTTP header "Content-Type" 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 "errs" request parameters are omitted.

If after a period of time, negotiated 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 SHOULD parse and validate received SETs to 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 "err" 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 "err" 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 ("sets" and "setErrs").

The following is an example request made by a SET Recipient that has
no outstanding SETs to acknowledge and is polling for available SETs
at the endpoint "https://nofity.exampleidp.com/Events":

POST /Events HTTP/1.1
Host: notify.exampleidp.com
Authorization: Bearer h480djs93hd8
Accept: 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://nofity.exampleidp.com/Events":

POST /Events HTTP/1.1
Host: notify.exampleidp.com
Authorization: Bearer h480djs93hd8
Accept: 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.

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

```plaintext
POST /Events HTTP/1.1
Host: notify.exampleidp.com
Authorization: Bearer h480djs93hd8
Content-Type: application/json
Authorization: Bearer h480djs93hd8

{
    "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.exampleidp.com
Authorization: Bearer h480djs93hd8
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.exampleidp.com
Authorization: Bearer h480djs93hd8
Content-Type: application/json
Authorization: Bearer h480djs93hd8

{
    "ack": ["3d0c3cf797584bd193bd0fb1bd4e7d30"],
    "setErrs": {
        "4d3559ec67504aaba65d40b0363faad8": {
            "err": "jwtAud",
            "description": "The audience value was invalid."
        }
    },
    "returnImmediately": true
}
```

Figure 5: Example Poll Acknowledgement with Error

2.5. Poll Response

In response to a 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 "true".

As described in Section 2.3, a JSON document is returned containing a number of 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
Location: https://notify.exampleidp/Events

```json
{
  "sets": {
    "4d3559ec67504aaba65d40b0363faad8": "eyJhbGciOiJub25lIn0.
  "3d0c3cf797584bd193bd0fb1bd4e7d30": "eyJhbGciOiJub25lIn0.
```

Figure 6: Example Poll Response

In the above example, two SETs whose "jti" values are "4d3559ec67504aaba65d40b0363faad8" and "3d0c3cf797584bd193bd0fb1bd4e7d30" are delivered.
The following is a non-normative example response to the request shown in Section 2.4, which indicates that no new SETs or unacknowledged SETs are available:

HTTP/1.1 200 OK
Content-Type: application/json
Location: https://notify.exampleidp/Events

{
  "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 via the next poll request to the SET Transmitter, as described in Section 2.4.3 or Section 2.4.4.

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 2.3 and Section 2.4.4.

3. Authentication and Authorization

The SET delivery method described in this specification is based upon HTTP and depends on the use of TLS and/or standard HTTP authentication and authorization schemes, as per [RFC7235]. For example, the following methodologies could be used among others:

TLS Client Authentication
Event delivery endpoints MAY request TLS mutual client authentication, per Section 7.3 of [RFC5246].

Bearer Tokens
Bearer tokens [RFC6750] MAY be used when combined with TLS and a token framework such as OAuth 2.0 [RFC6749]. For security considerations regarding the use of bearer tokens in SET delivery, see Section 4.4.1.

Basic Authentication
Use of HTTP BASIC authentication should be avoided due to its use of a single factor that is based upon a relatively static, symmetric secret. When used, implementers SHOULD combine the use of basic authentication with other factors. The security considerations of HTTP BASIC are well documented in [RFC7617] and SHOULD be considered along with using signed SETs, as described in Section 4.1.

As per Section 4.1 of [RFC7235], a SET delivery endpoint SHALL indicate supported HTTP authentication schemes via the "WWW-Authenticate" header.

Authorization for the ability to pick-up or deliver SETs can be determined by using the identity of the SET issuer, or via an authentication method above. This specification considers authentication as a feature to prevent denial-of-service attacks. Because SETs are not commands, SET Recipients are free to ignore SETs that are not of interest after acknowledging their receipt.

For illustrative purposes only, SET delivery examples show an OAuth 2.0 bearer token value [RFC6750] in the authorization header. This is not intended to imply that bearer tokens are preferred. However, the use of bearer tokens in the specification does reflect common practice.

3.1. Use of Tokens as Authorizations

When using bearer tokens or proof-of-possession tokens that represent an authorization grant such as issued by OAuth (see [RFC6749]), implementers SHOULD consider the type of authorization granted, any authorized scopes (see Section 3.3 of [RFC6749]), and the security subject(s) that SHOULD be mapped from the authorization when considering local access control rules. Section 6 of the OAuth Assertion Framework specification [RFC7521] documents common scenarios for authorization including:

- Clients using an assertion to authenticate and/or act on behalf of itself;
Clients acting on behalf of a user; and,

A Client acting on behalf of an anonymous user.

When using OAuth access tokens, implementers MUST take into account the threats and countermeasures documented in the security considerations for the use of client authorizations (see Section 8 of [RFC7521]). When using other token formats or frameworks, implementers MUST take into account similar threats and countermeasures, especially those documented by the relevant specifications.

4. Security Considerations

4.1. Authentication Using Signed SETs

In scenarios where HTTP authorization or TLS mutual authentication are not used or are considered weak, JWS signed SETs SHOULD be used (see [RFC7515] and Section 5 of [RFC8417]). This enables the SET Recipient to validate that the SET issuer is authorized to deliver the SET.

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.

As stated in Section 2.7.1 of [RFC7230], an HTTP requestor MUST NOT generate the "userinfo" (i.e., username and password) component (and its "@" delimiter) when an "http" URI reference is generated with a message, as they are now disallowed in HTTP.

4.3. Confidentiality of SETs

SETs may contain sensitive information that is considered Personally Identifiable Information (PII). In such cases, SET Transmitters and SET Recipients MUST protect the confidentiality of the SET contents by encrypting the SET as described in JWE [RFC7516], using a transport-layer security mechanism such as TLS, or both. If an Event delivery endpoint supports TLS, it MUST support at least TLS version 1.2 [RFC5246] and SHOULD support the newest version of TLS that meets its security requirements. When using TLS, the client MUST perform a TLS/SSL server certificate check, per [RFC6125]. Implementation security considerations for TLS can be found in "Recommendations for Secure Use of TLS and DTLS" [RFC7525].
4.4. Access Token Considerations

When using access tokens, such as those issued by OAuth 2.0 [RFC6749], implementers MUST take into account threats and countermeasures documented in Section 8 of [RFC7521].

4.4.1. Bearer Token Considerations

Due to the possibility of interception, Bearer tokens MUST be exchanged using TLS.

Bearer tokens MUST 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. Designating the specific methods of authentication and authorization are out of scope for the delivery of SETs, however this information is provided as a resource to implementers.

5. Privacy Considerations

If a SET needs to be retained for audit purposes, a JWS signature MAY be used to provide verification of its authenticity.

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.

The propagation of subject identifiers can be perceived as personally identifiable information. Where possible, SET Transmitters and Recipients SHOULD devise approaches that prevent propagation, for
example, the passing of a hash value that requires the subscriber to already know the subject.

6. IANA Considerations

This specification requires no IANA actions.

7. References

7.1. Normative References

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


7.2. Informative References


Appendix A. 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.

The editors would like to thank Phil Hunt and the other the authors of draft-ietf-secevent-delivery-02, on which this specification is based.

The editors would like to thank the participants in the SecEvents working group for their contributions to this specification.

Appendix B. 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:

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

Draft 01 - mbj:

- Addressed problems identified in my 18-Jul-18 review message titled "Issues for both the Push and Poll Specs".

Changes to align terminology with RFC 8417, for instance, by using the already defined term SET Recipient rather than SET Receiver.

Applied editorial and minor normative corrections.

Updated Marius’ contact information.


Draft 02 - mbj:

- Removed vestigial language remaining from when the push and poll delivery methods were defined in a common specification.

- Replaced remaining uses of the terms Event Transmitter and Event Recipient with the correct terms SET Transmitter and SET Recipient.

- Removed uses of the unnecessary term "Event Stream".

- Removed dependencies between the semantics of "maxEvents" and "returnImmediately".

- Said that PII in SETs is to be encrypted with TLS, JWE, or both.

- Corrected grammar and spelling errors.

Draft 03 - mbj:

- Corrected uses of "attribute" to "member" when describing JSON objects.

- Further alignment with the push draft.

Authors’ Addresses

Annabelle Backman (editor)
Amazon

Email: richanna@amazon.com
Michael B. Jones (editor)
Microsoft
Email: mbj@microsoft.com
URI: http://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
Subject Identifiers for Security Event Tokens

draft-ietf-secevent-subject-identifiers-05

Abstract

Security events communicated within Security Event Tokens may support a variety of identifiers to identify the subject and/or other principals related to the event. This specification formalizes the notion of subject identifiers as named sets of well-defined claims describing the subject, a mechanism for representing subject identifiers within a [JSON] object such as a JSON Web Token [JWT] or Security Event Token [SET], and a registry for defining and allocating names for these claim sets.

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 January 25, 2020.

Copyright Notice

Copyright (c) 2019 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
1. Introduction

As described in section 1.2 of [SET], the subject of a security event may take a variety of forms, including but not limited to a JWT principal, an IP address, a URL, etc. Furthermore, even in the case where the subject of an event is more narrowly scoped, 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 the subject of the security event is being identified.

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].

3. Subject Identifiers

A Subject Identifier Type is a light-weight schema that describes a set of claims that identifies a subject. Every Subject Identifier Type MUST have a unique name registered in the IANA "Security Event Subject Identifier Types" registry established by Section 7.1. A Subject Identifier Type MAY describe more claims than are strictly necessary to identify a subject, and MAY describe conditions under which those claims are required, optional, or prohibited.

A Subject Identifier is a [JSON] object containing a "subject_type" claim whose value is the name of a Subject Identifier Type, and a set of additional "payload claims" which are to be interpreted according to the rules defined by that Subject Identifier Type. Payload claim values MUST match the format specified for the claim by the Subject Identifier Type. A Subject Identifier MUST NOT contain any payload claims prohibited or not described by its Subject Identifier Type, and MUST contain all payload claims required by its Subject Identifier Type.

The following Subject Identifier Types are registered in the IANA "Security Event Subject Identifier Types" registry established by Section 7.1.

3.1. Account Subject Identifier Type

The Account Subject Identifier Type describes a user account at a service provider, identified with an "acct" URI as defined in [RFC7565]. Subject Identifiers of this type MUST contain a "uri" claim whose value is the "acct" URI for the subject. The "uri" claim is REQUIRED and MUST NOT be null or empty. The Account Subject Identifier Type is identified by the name "account".

Below is a non-normative example Subject Identifier for the Account Subject Identifier Type:
3.2. Email Subject Identifier Type

The Email Subject Identifier Type describes a principal identified with an email address. Subject Identifiers of this type MUST contain an "email" claim 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" claim is REQUIRED and MUST NOT be null or empty. The value of the "email" claim SHOULD identify a mailbox to which email may be delivered, in accordance with [RFC5321]. The Email Subject Identifier Type is identified by the name "email".

Below is a non-normative example Subject Identifier for the Email Subject Identifier Type:

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

Figure 2: Example: Subject Identifier for the Email Subject Identifier Type.

3.2.1. Email Canonicalization

Many email providers will treat multiple email addresses as equivalent. For example, some providers treat email addresses as case-insensitive, 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 subject identifier string used in their system.
3.3. Phone Number Subject Identifier Type

The Phone Number Subject Identifier Type describes a principal identified with a telephone number. Subject Identifiers of this type MUST contain a "phone_number" claim 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" claim is REQUIRED and MUST NOT be null or empty. The Phone Number Subject Identifier Type is identified by the name "phone-number".

Below is a non-normative example Subject Identifier for the Email Subject Identifier Type:

```json
{
    "subject_type": "phone-number",
    "phone_number": "+12065550100",
}
```

Figure 3: Example: Subject Identifier for the Phone Number Subject Identifier Type.

3.4. Issuer and Subject Subject Identifier Type

The Issuer and Subject Subject Identifier Type describes a principal identified with a pair of "iss" and "sub" claims, as defined by [JWT]. These claims MUST follow the formats of the "iss" claim and "sub" claim defined by [JWT], respectively. Both the "iss" claim and the "sub" claim are REQUIRED and MUST NOT be null or empty. The Issuer and Subject Subject Identifier Type is identified by the name "iss-sub".

Below is a non-normative example Subject Identifier for the Issuer and Subject Subject Identifier Type:

```json
{
    "subject_type": "iss-sub",
    "iss": "http://issuer.example.com/",
    "sub": "145234573",
}
```

Figure 4: Example: Subject Identifier for the Issuer and Subject Subject Identifier Type.
3.5. Aliases Subject Identifier Type

The Aliases Subject Identifier Type 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 of this type MUST contain an "identifiers" claim 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" claim is REQUIRED and MUST NOT be null or empty. It MAY contain multiple instances of the same Subject Identifier Type (e.g., multiple Email Subject Identifiers), but SHOULD NOT contain exact duplicates. This type is identified by the name "aliases".

"alias" Subject Identifiers MUST NOT be nested; i.e., the "identifiers" claim of an "alias" Subject Identifier MUST NOT contain a Subject Identifier of type "aliases".

Below is a non-normative example Subject Identifier for the Aliases Subject Identifier Type:

```json
{
  "subject_type": "aliases",
  "identifiers": [
    {
      "subject_type": "email",
      "email": "user@example.com",
    },
    {
      "subject_type": "phone-number",
      "phone_number": "+12065550100",
    },
    {
      "subject_type": "email",
      "email": "user+qualifier@example.com",
    }
  ]
}
```

Figure 5: Example: Subject Identifier for the Aliases Subject Identifier Type.
4. Subject Identifiers in JWTs

4.1. "sub_id" Claim

This document defines the "sub_id" JWT Claim, in accordance with Section 4.2 of [RFC7519]. When present, the value of this claim MUST be a Subject Identifier that identifies the principal that is 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 principal.

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

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

Figure 6: Example: JWT containing a 'sub_id' claim and no 'sub' claim.

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

Figure 7: Example: JWT where both the 'sub' and 'sub_id' claims identify the subject using the same identifier.
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" claim MAY be different. For example, an OpenID Connect [OIDC] client may construct such a JWT when issuing a JWT back to its OpenID Connect Identity Provider, in order to communicate information about the services’ shared subject principal using an identifier the Identity Provider is known to understand. Similarly, the JWT’s "sub" claim and the Subject Identifier’s "sub" claim MAY be different. For example, this may be used by an OpenID Connect client to communicate the subject principal’s local identifier at the client back to its Identity Provider.

Below are non-normative examples of a JWT where the "iss" claims are the same, and a JWT where they are different.
Figure 10: Example: JWT with a 'iss-sub' Subject Identifier where JWT issuer and subject issuer are the same.

Figure 11: Example: JWT with an 'iss-sub' Subject Identifier where the JWT issuer and subject issuer are different.

Figure 12: Example: JWT with an 'iss-sub' Subject Identifier where the JWT 'iss' and 'sub' claims differ from the Subject Identifier's 'iss' and 'sub' claims.

5. Privacy Considerations

5.1. Identifier Correlation

The act of presenting two or more identifiers for a single principal together (e.g., within an "aliases" Subject Identifier, or via the "sub" and "sub_id" JWT claims) may communicate more information about the principal than was intended. For example, the entity to which the identifiers are presented, now knows that both identifiers relate to the same principal, 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).

6. Security Considerations

There are no security considerations.

7. IANA Considerations

7.1. Security Event Subject Identifier Types Registry

This document defines Subject Identifier Types, for which IANA is asked to create and maintain a new registry titled "Security Event Subject Identifier Types". Initial values for the Security Event Subject Identifier Types 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 7.1.1.

7.1.1. Registration Template

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

Type Description
A brief description of the Subject Identifier Type.

Change Controller
For types defined in documents published by the OpenID Foundation or its working groups, list "OpenID Foundation RISC Working Group". For all other types, 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 Subject Identifier Type. The definition MUST specify the name, format, and meaning of each claim that may occur within a Subject Identifier of the defined type, as well as whether each claim is optional or required, or the circumstances under which the claim
7.1.2. Initial Registry Contents

7.1.2.1. Account Subject Identifier Type

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

7.1.2.2. Email Subject Identifier Type

- Type Name: "email"
- Type Description: Subject identifier based on email address.
- Change Controller: IETF secevent Working Group
- Defining Document(s): Section 3 of this document.

7.1.2.3. Issuer and Subject Subject Identifier Type

- Type Name: "iss-sub"
- Type Description: Subject identifier based on an issuer and subject.
- Change Controller: IETF secevent Working Group
- Defining Document(s): Section 3 of this document.

7.1.2.4. Phone Number Subject Identifier Type

- Type Name: "phone-number"
- Type Description: Subject identifier based on an phone number.
- Change Controller: IETF secevent Working Group
- Defining Document(s): Section 3 of this document.
7.1.2.5. Aliases Subject Identifier Type

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

7.1.3. 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.

Subject Identifier Types need not be generally applicable and may be highly specific to a particular domain; it is expected that types may be registered for niche or industry-specific use cases. The Expert Reviewer should focus on whether the type 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 type.

7.2. 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 [SET].

7.2.1. Registry Contents

- **Claim Name**: "sub_id"
- **Claim Description**: Subject Identifier
- **Change Controller**: IESG
8. References

8.1. Normative References


8.2. Informative References


Acknowledgements

This document is based on work developed within the OpenID RISC Working Group. The authors would like to thank the members of this group for their hard work and contributions.

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".

Authors’ Addresses

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

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