This specification defines how security event tokens (SETs) may be exchanged between a client and service provider using HTTP POST over TLS using a symmetric format. The specification supports three modes of operation: "push", "pull", and "push-pull" bi-directional SET exchange. The specification also defines a simple acknowledge mechanism allowing parties to confirm delivery.
the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction ................................................. 2
   1.1. Notational Conventions ................................. 4
   1.2. Definitions ............................................ 4
2. Simple SET Transfer Protocol ................................. 5
   2.1. SSTP Message Format (Content-Type: application/ssspt+json) 5
   2.2. Examples ............................................... 7
      2.2.1. Example Push Transfer .............................. 8
      2.2.2. Example Pull Transfer .............................. 9
      2.2.3. Example Push-Pull Bi-directional Transfer ....... 13
   2.3. Error Response Handling .............................. 15
3. Authentication and Authorization ............................ 16
   3.1. Use of Tokens as Authorizations ........................ 17
4. Security Considerations ..................................... 18
   4.1. Authentication Using Signed SETs ......................... 18
   4.2. HTTP Considerations ..................................... 18
   4.3. TLS Support Considerations ............................. 18
   4.4. Authorization Token Considerations .................... 19
      4.4.1. Bearer Token Considerations ....................... 19
5. Privacy Considerations ...................................... 19
6. IANA Considerations ......................................... 20
7. References .................................................. 20
   7.1. Normative References ................................... 20
   7.2. Informative References ................................ 21
Appendix A. Acknowledgments .................................. 22
Appendix B. Change Log ......................................... 23
Authors’ Addresses ............................................ 23

1. Introduction

[EDITORS NOTE: This specification is based upon draft-ietf-secevent-delivery and attempts to provide a unified single MTI protocol solution satisfying all use cases for SECEVENTS.]

This specification defines how SETs (see [I-D.ietf-secevent-token]) can be exchanged using HTTP [RFC7231] over TLS using a symmetric request/response format. The specification supports three modes of operation: "push", "pull", and "push-pull" bi-directional SET exchange. The specification also defines a simple acknowledge mechanism allowing parties to confirm delivery or attempt re-delivery.

This specification makes several simplifying assumptions:
SSTP is a symmetric protocol meaning it uses the same HTTP content-type and JSON structure to send requests and process responses.

- Recovery is provided for unacknowledged SETs for short term or missed deliveries. Issuers are able to re-transmit SETs in situations where a SET has been unacknowledged (e.g. over more than one request/response cycles).

- The protocol can be used for unidirectional, or bi-directional communication avoiding the need to implement multiple delivery methods.

- The use of HTTP Long Polling MAY be used in cases where pull or bi-directional communication is needed in real time.

- Only one party in an exchange needs to have an addressable fixed URI endpoint and can act as an SSTP protocol service provider to a mobile or otherwise unreachable client.

- SET message exchanges are secured through the use of TLS and some form of HTTP authorization (e.g. RFC6750, RFC7519), and MAY in turn be signed and encrypted.

This specification supports the following use-cases:

**Broadcaster**

Where a large entity (e.g. an Identity Provider) needs to issue SETs to a large number of relying parties.

**Protected Client**

Where a client party is behind a firewall or otherwise network restricted location and cannot act as a SSTP service provider. Examples include Enterprise on-premise security systems, IoT devices which may be shielded in restricted network environments.

**Mobile**

Where a client is mobile and thus would be unable to maintain a permanent HTTP endpoint.

This specification does not define how endpoints are configured, nor does it define the specifics of which SET event types are exchanged over any particular delivery relationship.
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.

For purposes of readability examples are not URL encoded. Implementers MUST percent encode URLs as described in Section 2.1 of [RFC3986].

Throughout this documents all figures MAY contain spaces and extra line-wrapping for readability and space limitations. Similarly, some URI’s contained within examples, have been shortened for space and readability reasons. All examples are non-normative.

1.2. Definitions

This specification assumes terminology defined in the Security Event Token specification [I-D.ietf-secevent-token].

This specification defines the following terms:

**Client**  An entity which acts as an HTTP client as defined by [RFC7231] which is communicating with an SSTP server.

**Server**  An entity which has a permanent endpoint reachable by a client which acts as an HTTP server as defined by [RFC7231] and supports this specification.

**Event**  An Event is defined to be an event as represented by a security event token (SET). See [I-D.ietf-secevent-token].

**NumericDate**  A JSON numeric value representing the number of seconds from 1970-01-01T00:00:00Z UTC until the specified UTC date/time, ignoring leap seconds. This is equivalent to the IEEE Std 1003.1, 2013 Edition [POSIX.1] definition "Seconds Since the Epoch", in which each day is accounted for by exactly 86400 seconds, other than that non-integer values can be represented. See [RFC3339] for details regarding date/times in general and UTC in particular.
Internet-Draft
draft-hunt-secevent-sstp
March 2018

2. Simple SET Transfer Protocol

SSTP is a symmetric protocol. As such an SSTP client uses HTTP POST (Section 4.3.3 [RFC7231]) with a body with "Content-Type" of "application/sstp+json" to deliver 0 or more SETs and/or to acknowledge previously received SETs from an SSTP server. In response, the SSTP server returns an HTTP body with the same document-type which may also return 0 or more SETs, acknowledgments, and errors.

Requests MAY be spontaneous (in the case of push mode), scheduled over a a periodic interval (in the case of pull), or requests to pull MAY await new SETS using HTTP long polling (see Section 2 [RFC6202]). An SSTP server choosing NOT to support HTTP long polling MAY do so by returning HTTP Status of 403 "Forbidden" (see Section 6.5.3 [RFC7231]) if a particular client is not authorized, or HTTP status 501 "Not implemented" (see Section 6.6.2 [RFC7231]) if the server does not support long polling.

SSTP provides an acknowledgement capability for the purpose of informing communications partners about which SETs have been successfully delivered. Upon receipt of a SET and before acknowledgement, receivers SHOULD ensure received SETs are valid and have been retained in a manner appropriate to the receiver’s retention needs. The level and method of retention of SETs by receivers is out-of-scope of this specification.

If after a period of time, negotiated between the client and server, unacknowledged SETs MAY be re-transmitted. The receiver SHOULD accept repeat SETs and acknowledge the SETs regardless of whether the receiver believes it has already acknowledged the SETs previously. A SET issuer MAY limit the number of attempts to deliver a SET.

A receiving party (client or server) of SETs SHOULD parse and validate each SET to meet its own requirements and SHOULD acknowledge receipt in a timely (e.g. minutes) fashion so that the issuer may mark the SETs as received. Receivers SHOULD acknowledge receipt before taking any local actions based on the SETs to avoid unnecessary delay in acknowledgement to avoid unnecessary re-transmission.

2.1. SSTP Message Format (Content-Type: application/sstp+json)

The body (or message) of an SSTP request or response is a "Content-Type" of "application/sstp+json" which is a JSON document consisting of the following optional JSON attributes:

Request Handling:
returnEvents
   A JSON boolean parameter which indicates whether the receiver
   SHOULD return SETs in its upcoming response. When part of an HTTP
   Request, it indicates the SSTP server SHOULD return with SETs in
   its response (and optionally wait). When asserted as "false" by
   an SSTP server in its response, it indicates that the SSTP server
   is not requesting SETs in the next client request. If omitted,
   this attribute SHALL have a default value of "true".

returnImmediately
   An OPTIONAL JSON boolean value which when "true" has the effect of
   declining HTTP long polling. A value of "false" indicates the
   request is to be treated as an HTTP Long Poll (see Section 2
   [RFC6202]). When asserted in an HTTP response, the value
   indicates the SSTP has more information and the client SHOULD NOT
   wait before initiating its next request. When omitted, the
   default value of "false" SHALL be assumed.

Delivery:

sets
   A JSON object that contains one or more nested JSON attributes.
   Each nested attribute 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 SETs to be
   transmitted, the attribute MAY be omitted.

Acknowledgment:

ack
   An array of Strings that each correspond to the "jti" of a
   successfully received SET by the client. If there are no
   outstanding SETs to acknowledge, the attribute MAY be omitted.
   When acknowledging a SET, the issuer is released from any
   obligation to retain the SET (e.g. for a future re-try).

setErrs
   A JSON Object that contains one or more nested JSON attributes
   that correspond to the "jti" of each invalid SET received. The
   value of each is a JSON object whose contents is an "err"
   attribute and "description" attribute whose value correspond to
   the errors described in Section 2.3. If there are no errors to
   acknowledge, the attribute MAY be omitted.
The following is an example SSTP message that could be exchanged either in a HTPP POST request or response between a client and an SSTP server.

```
{
  "returnImmediately":true,
  "returnEvents":true,
  "sets":{
    "4d3559ec67504aaba65d40b0363faad8":
      "eyJhbGciOiJub25lIn0.
      "eyJqdGkiOiI0ZDM1NTllYzY3NTA0YWFiYTM1YzIwM2EwNTMwZDQwYjAzNjNmYWFkOCIIsImhidCI6MTQ1ODQ5NjQwNCwiXzIiOCwiHRcHMDLYzY2IiLCJmYiI6IiIsImlhdCI6MTQ1ODQ5NjQwNSwiYXVkIjpbIiIiLCJlZmF1bWJpbGl0eSI6WyIubnVwdWJsZSIsInByb3BzZXRlIiwiY2F0aW9uIiwiY2F0aW9uIiwidXNlck5hbWUiXSwiaWQiOiJodHRwczovL3NjaW0uZXhhbXBsZS5cIiwic3lzZXRlIjoiY2F1c2ksZSIsInNyZWF0b3JhbWUiXSwicmVnaW5nIjoiZmFsc2VzcyIsImlkIjoiZ3JhcGwibG9naW4iXSwiYXNzd29yZCI6e19.
      "3d0c3cf797584bd193bd0fb1bd4e7d30":
      "eyJhbGciOiJub25lIn0.
      "eyJqdGkiOiIzZDBmMzYyNTk5MTQwMTM5NjA1ZjZjM2IyY2UyZDE1MzciLCJlZmF1bWJpbGl0eSI6WyIuOTciIiwic3lzZXRlIjoiY2F1c2ksZSIsInNyZWF0b3JhbWUiXSwicmVnaW5nIjoiZmFsc2VzcyIsImlkIjoiZ3JhcGwibG9naW4iXSwiYXNzd29yZCI6e19.
    }
}
```

Figure 1: Example Request or Response Document

2.2. Examples

Three examples are provided where:

- A client pushes SETs to a server using the "sets" JSON attribute in its request and receives "ack" values in response from the SSTP server.

- A polling client that requests SETs from a server and uses the "ack" parameter in its request to acknowledge SETs from a previous HTTP request, and receives new SETs in the response.
o A client and SSTP server bi-directionally exchange SETs using both the "sets" and "ack" JSON attributes in both the request and response messages.

2.2.1.  Example Push Transfer

In this example, a client posts SETs to an SSTP server which in turn acknowledges the transferred SETs in its response.

In the figure, two SETs are transferred to the SSTP server. The parameter "returnEvents" indicates the client is not interested in receiving SETs.

POST /Events HTTP/1.1
Host: notify.exampleidp.com
Authorization: Bearer h480djs93hd8
Content-Type: application/sstp+json
Accept: application/sstp+json

{"returnImmediately":true,
"returnEvents":false,
"sets":{
"4d3559ec67504aaba65d40b0363faad8":
"eyJhbGciOiJub25lIn0.
eyJqdGkiOiI0ZDM1NTllYzY3NTA0YWFiYTY1ZDQwYjAzNjNmYWFkOjIsmImZldmVydGggb25lcy5jcmwqCnVub29nIj0h
"3d0c3cf797584bd0f0bb4e7d30":
"eyJhbGciOiJub25lIn0.
eyJqdGkiOiIzZDBmM2MnNzN3NTg0YmQxOTNjZDgwMnZ2czpOZDu5MCIsmImZldmVydGggb25lcy5jcmwqCnVub29nIj0h
}}

Figure 2: Example Push Request
The figure below shows the response returned from the SSTP server in response to the request in Figure 2.

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

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

Figure 3: Example Push Response

2.2.2. Example Pull Transfer

The figure below is an example pull request to the SSTP server whose location is: "https://notify.exampleidp.com/Events". In this example, the client is requesting an immediate response whether or not new SETs are available.

POST /Events HTTP/1.1  
Host: notify.exampleidp.com  
Authorization: Bearer h480djs93hd8  
Content-Type: application/sstp+json  
Accept: application/sstp+json

```{  
"returnImmediately":true
}
```

Figure 4: Example Initial Pull Request

The following is an example default pull request to the SSTP server whose location is: "https://notify.exampleidp.com/Events". The default mode to return events and to wait if necessary is assumed.

POST /Events HTTP/1.1  
Host: notify.exampleidp.com  
Authorization: Bearer h480djs93hd8  
Accept: application/sstp+json

```{  
}
```

Figure 5: Example Default Pull Request
In response to the above two requests, the SSTP server responds immediately if "respondImmediately" is true or SETs are available. If no SETs are available at the time of the request and "respondImmediately" is false, the SSTP server delays response until a SET is available.

As described in Section 2.1 a JSON document is returned containing the JSON attribute "sets".

The following is an example response to the requests shown Figure 4 and Figure 5. This example shows two SETs are returned.

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

```json
{
  "sets":{
    "4d3559ec67504aaa65d40b0363faad8":
      "eyJhbGciOiJub25lIn0.
  eyJqdGkiOiI0ZDM1NT11Yz3NTA0YWFiYTY1ZDQwYjAzNjNmYWY2O1CiImlhdCI6MTQ1ODQ5NjQwNCwiaXNzIjoib3JlZC4zdWN0cyIsImdvb3BcIjpbM1NT
  "3d0c3cf797584bd193bd0fb1bd4e7d30":
      "eyJhbGciOiJub25lIn0.
  eyJqdGkiOiIzZDBjM2NmNtN3NTA0YWFiYTY1ZDQwYjAzNjNmYWY2O1CiImlhdCI6MTQ1ODQ5NjQwNCwiaXNzIjoib3JlZC4zdWN0cyIsImdvb3BcIjpbM1NT

Figure 6: Example Pull Response

In the above example, two SETs whose "jti" are "4d3559ec67504aaa65d40b0363faad8" and "3d0c3cf797584bd193bd0fbb1bd4e7d30" are delivered.
The following is an example response to the request shown Figure 4 where "returnImmediately" was set showing no new SETs are available.

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

{
   "sets":{}
}

Figure 7: Example No SETs Pull Response

2.2.2.1. Acknowledge Only Request

This variation is typically used when a client needs to acknowledge received SETs on a separate thread from one receiving SETs.

An SSTP client acknowledges previously received SETs but indicates it does not want to receive SETs in the current request/response by setting the "returnEvents" attribute to "false".

The following example is an acknowledgement of SETs previously received (e.g. from the response shown in Figure 6).

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

{
   "ack":[
      "4d3559ec67504aaba65d40b363fad8",
      "3d0c3cf797584bd193bd0fb1bd4e7d30"
   ],
   "returnEvents":false
}

Figure 8: Example Acknowledge Only Request

2.2.2.2. Poll with Acknowledgement

This variation allows a client to simultaneously acknowledge previously received SETs and wait for the next group of SETs in a single HTTP request.
The following SSTP request contains an acknowledgement of SETs received from Figure 6.

POST /Events HTTP/1.1
Host: notify.exampleidp.com
Authorization: Bearer h480djs93hd8
Content-Type: application/sstp+json
Accept: application/sstp+json
Authorization: Bearer h480djs93hd8

{
    "ack": [
        "4d3559ec67504aaba65d40b0363faad8",
        "3d0c3cf797584bd193bd0f81bd4e7d30"
    ],
    "returnImmediately":false
}

Figure 9: Example Pull With Acknowledgement and No Errors

2.2.2.3. Pull with Acknowledgement and Errors

In the case where errors are detected in previously received SETs, the client (or server) uses the "setErrs" attribute to indicate errors in its request.
The following is an example of an SSTP response acknowledges 1 error and 1 receipt of two SETs received in Figure 6.

POST /Events HTTP/1.1
Host: notify.exampleidp.com
Authorization: Bearer h480djs93hd8
Content-Type: application/sstp+json
Accept: application/sstp+json
Authorization: Bearer h480djs93hd8

{}<br>

Figure 10: Example Pull Acknowledgement With Error

2.2.3. Example Push-Pull Bi-directional Transfer

In push-pull mode, JSON attributes "sets", "ack" and "setErrs" are used in both HTTP request and response messages between client and SSTP server.
In the following example, two SETs are transferred to the SSTP server and events are requested in return.

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

{ "returnEvents": true,
  "ack": ["b453bf9972b84e0492d5f7f56f86bfa"],
  "sets": {
    "4d3559ec67504aaba65d40b0363fa6d": "eyJhbGciOiJub25lIn0.
    h480djs93hd8",
    "3d0c3cf797584bd193bd0fb1bd4e7d30": "eyJhbGciOiJub25lIn0.
    h480djs93hd8"
  }
}
```

Figure 11: Example Push-Pull Request
The following is an example response to the request shown Figure 11.

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

{
"ack": [
"4d3559ec67504aaba65d40b0363faad8",
"3d0c3cf797584bd193bd0fb1bd4e7d30"
],
"sets":{
"6f332aefc730400a9f645d36a12ba4ab":
"eyJhbGciOiJub25lIn0.
eyJqdGkiOiI0ZDM1NTllYzY3NTA0YWFiYTY1ZDQwYjAzNjNmYWFkOCIsImh0dHA6Ly9z
b3JkLmNvbS5ucGhwLmNvbS5zY2ltcy9zaWQvMjAyMjAxNjI0MjEyOS5jb20iLCJvbmx5
IiwiYXNzd29yZCI6NjQwMzI0NjQwNSwiYXV0aD0wMDAwMywiaWRvYl90b2JsZGVyIjoi
MjAyMjAxNjI0MjEyOSIsImNyb3NzIiwiMS4wIiwiaWF0Y29uIjoyMjAyMjAxNjI0MjEyOS
Iiwic2VuY25vdFwiOntcImlzQ29yZCI6NDQuNjIwIiwic2l0ZSB0b2JsZGVyIjoiMDAx
MjIwOS0wMDAwMyIiLCJ0b2JsZGVyIjoiMDAxMjIwOS0wMDAwMyJ9.
"],
}

Figure 12: Example Push-Pull Response

Following the response from the SSTP server, the client would subsequently repeats the request-response cycle by acknowledging the SET identified by a "jti" value of "6f332aefc730400a9f645d36a12ba4ab".

2.3. Error Response Handling

If a SET is invalid, the following error codes are defined:
Table 1: SET Errors

An error response has a "Content-Type" of "application/sstp+json" which is a JSON document that provides details about the error. The JSON document includes the JSON attributes:

- **err**: A value which is a keyword that describes the error (see Table 1).
- **description**: A human-readable text that provides additional diagnostic information.

When included as part of an HTTP Status 400 response, the above JSON is the HTTP response body in the JSON attribute "setErrs" (see Section 2.1).

3. Authentication and Authorization

The SET delivery methods described in this specification are based upon HTTP and depend 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**: SSTP server MAY negotiate TLS mutual client authentication. See Section 7.3 [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

Usage of basic authentication should be avoided due to its use of a single factor that is based upon a relatively static, symmetric secret. 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 (see SET Payload Authentication below).

SET Payload Authentication

In scenarios where SETs are signed and the delivery method is HTTP POST, SSTP clients MAY elect to use Basic Authentication or not to use HTTP or TLS based authentication at all. See Section 4.1 for considerations.

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

Because SET Delivery describes a simple function, authorization for the ability to pick-up or deliver SETs can be derived by considering 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 (see ), event receivers are free to ignore SETs that are not of interest.

For illustrative purposes only, SET delivery examples show an OAuth2 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
Assertions draft [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 (e.g., see next section).

When using OAuth authorization 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 Security Considerations [I-D.ietf-secevent-token]). This enables event receivers to validate that the SET issuer is authorized to deliver SETs.

4.2. HTTP Considerations

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

As stated in Section 2.7.1 [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. TLS Support Considerations

SETs contain sensitive information that is considered PII (e.g., subject claims). Therefore, event issuers and receivers MUST require the use of a transport-layer security mechanism. Event delivery endpoints MUST support TLS 1.2 [RFC5246] and MAY support additional transport-layer mechanisms meeting 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. Authorization Token Considerations

When using authorization 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 OAuth2, a client MAY use OAuth token refresh to obtain a new bearer token after authenticating to an authorization server. See 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 SET tokens, however this information is provided as a resource to implementers.

5. Privacy Considerations

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

Event transmitters SHOULD attempt to filter SETs issued so that the content is targeted to the specific business and protocol needs of receivers.

When sharing personally identifiable information or information that is otherwise considered confidential to affected users, event
transmitters and receivers 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, event transmitters and receivers 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

There are no IANA considerations.

7. References

7.1. Normative References

[I-D.ietf-secevent-token]


7.2. Informative References

[openid-connect-core]
NRI, "OpenID Connect Core 1.0", Nov 2014.


Appendix A. Acknowledgments

The editor would like to thank the participants in the the SECEVENTS working group for their support of this specification.

This specification is based on and ideally replaces draft-ietf-secevent-delivery, and we thank its contributors Annabelle Backman, Marius Scurtescu, and Morteza Ansari.
Appendix B. Change Log

Draft 00 - PH - Original

Authors’ Addresses

Phil Hunt (editor)
Oracle Corporation

Email: phil.hunt@yahoo.com

Anthony Nadalin
Microsoft

Email: tonynad@microsoft.com