

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: October 18, 2018

A. Backman, Ed.  
Amazon  
M. Jones, Ed.  
Microsoft  
P. Hunt, Ed.  
Oracle  
M. Scurtescu  
Google  
M. Ansari  
Cisco  
A. Nadalin  
Microsoft  
April 16, 2018

**Poll-Based SET Token Delivery Using HTTP**  
**draft-ietf-secevent-http-poll-00**

Abstract

This specification defines how a series of security event tokens (SETs) may be delivered to a previously registered receiver using HTTP POST over TLS initiated as a poll by the receiver. The specification also defines how delivery can be assured subject to the SET Token Receiver'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 October 18, 2018.

Copyright Notice

Copyright (c) 2018 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

<a href="#">1.</a>	<a href="#">Introduction and Overview</a>	<a href="#">2</a>
<a href="#">1.1.</a>	<a href="#">Notational Conventions</a>	<a href="#">3</a>
<a href="#">1.2.</a>	<a href="#">Definitions</a>	<a href="#">3</a>
<a href="#">2.</a>	<a href="#">SET Event Stream Protocol</a>	<a href="#">4</a>
<a href="#">2.1.</a>	<a href="#">Event Delivery Process</a>	<a href="#">4</a>
<a href="#">2.2.</a>	<a href="#">Polling Delivery using HTTP</a>	<a href="#">5</a>
<a href="#">2.2.1.</a>	<a href="#">Polling HTTP Request Attributes</a>	<a href="#">6</a>
<a href="#">2.2.2.</a>	<a href="#">Polling HTTP Response Attributes</a>	<a href="#">7</a>
<a href="#">2.2.3.</a>	<a href="#">Poll Request</a>	<a href="#">7</a>
<a href="#">2.2.4.</a>	<a href="#">Poll Response</a>	<a href="#">11</a>
<a href="#">2.3.</a>	<a href="#">Error Response Handling</a>	<a href="#">13</a>
<a href="#">3.</a>	<a href="#">Authentication and Authorization</a>	<a href="#">14</a>
<a href="#">3.1.</a>	<a href="#">Use of Tokens as Authorizations</a>	<a href="#">15</a>
<a href="#">4.</a>	<a href="#">Security Considerations</a>	<a href="#">15</a>
<a href="#">4.1.</a>	<a href="#">Authentication Using Signed SETs</a>	<a href="#">15</a>
<a href="#">4.2.</a>	<a href="#">HTTP Considerations</a>	<a href="#">15</a>
<a href="#">4.3.</a>	<a href="#">TLS Support Considerations</a>	<a href="#">16</a>
<a href="#">4.4.</a>	<a href="#">Authorization Token Considerations</a>	<a href="#">16</a>
<a href="#">4.4.1.</a>	<a href="#">Bearer Token Considerations</a>	<a href="#">16</a>
<a href="#">5.</a>	<a href="#">Privacy Considerations</a>	<a href="#">17</a>
<a href="#">6.</a>	<a href="#">IANA Considerations</a>	<a href="#">17</a>
<a href="#">7.</a>	<a href="#">References</a>	<a href="#">17</a>
<a href="#">7.1.</a>	<a href="#">Normative References</a>	<a href="#">17</a>
<a href="#">7.2.</a>	<a href="#">Informative References</a>	<a href="#">18</a>
<a href="#">Appendix A.</a>	<a href="#">Other Streaming Specifications</a>	<a href="#">20</a>
<a href="#">Appendix B.</a>	<a href="#">Acknowledgments</a>	<a href="#">21</a>
<a href="#">Appendix C.</a>	<a href="#">Change Log</a>	<a href="#">21</a>
	<a href="#">Authors' Addresses</a>	<a href="#">22</a>

## **1. Introduction and Overview**

This specification defines how a stream of SETs (see [[I-D.ietf-secevent-token](#)]) can be transmitted to a previously registered Event Receiver using HTTP [[RFC7231](#)] over TLS. The specification defines a method to poll for SETs using HTTP POST.



This specification defines a method of SET delivery in what is known as Event Streams.

This specification does not define the method by which Event Streams are defined, provisioned, managed, monitored, and configured and is out of scope of this specification.

[[This work is TBD by the SECEVENTS WG]]

## **[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.

## **[1.2.](#) Definitions**

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

The following definitions are defined for Security Event distribution:

### **Event Transmitter**

A service provider that delivers SETs to other providers known as Event Receivers. An Event Transmitter is responsible for offering a service that allows the Event Receiver to check the Event Stream configuration and status known as the "Control Plane".

### **Event Receiver**

A service provider that registers to receive SETs from an Event Transmitter and provides an endpoint to receive SETs via HTTP POST. Event Receivers can check current Event Stream configuration and status by accessing the Event Transmitters "Control Plane".

### **Event Stream**



An Event Stream is a defined location, distribution method and whereby an Event Transmitter and Event Receiver exchange a pre-defined family of SETs. A Stream is assumed to have configuration data such as HTTP endpoints, timeouts, public key sets for signing and encryption, and Event Families.

#### Subject

The security subject around which a security event has occurred. For example, a security subject might be a user, a person, an email address, a service provider entity, an IP address, an OAuth Client, a mobile device, or any identifiable thing referenced in security and authorization systems.

#### 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.

## **[2. SET Event Stream Protocol](#)**

An Event Stream represents the communication channel over which a series of SETs are delivered to a configured Event Receiver.

### **[2.1. Event Delivery Process](#)**

When an Event occurs, the Event Transmitter constructs a SET token [[I-D.ietf-secevent-token](#)] that describes the Event. The Event Transmitter determines the Event Streams over which the SET should be distributed to.

How SETs are defined and the process by which Events are identified for Event Receivers is out-of-scope of this specification.

When a SET is available for an Event Receiver, the Event Transmitter attempts to deliver the SET based on the Event Receiver's registered delivery mechanism:

- o The Event Transmitter queues up the SET in a buffer so that an Event Receiver MAY poll for SETs using HTTP/1.1 POST.



- o Or, the Event Transmitter delivers the Event through a different method not defined by this specification.

In Poll-Based SET Token Delivery Using HTTP, multiple SETs are delivered in a JSON document [[RFC7159](#)] to an Event Receiver in response to an HTTP POST request to the Event Transmitter. Then in a following request, the Event Receiver acknowledges received SETs and MAY poll for more. All requests and responses are JSON documents and use a "Content-Type" of "application/json" as described in [Section 2.2](#).

After successful (acknowledged) SET delivery, Event Transmitters SHOULD NOT be required to maintain or record SETs for recovery. Once a SET is acknowledged, the Event Receiver SHALL be responsible for retention and recovery.

Transmitted SETs SHOULD be self-validating (e.g. signed) if there is a requirement to verify they were issued by the Event 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 Event Receiver reads the SET and validates it. The Event Receiver MUST acknowledge receipt to the Event Transmitter, using the defined acknowledgement or error method depending on the method of transfer.

The Event Receiver SHALL NOT use the Event acknowledgement mechanism to report Event errors other than relating to the parsing and validation of the SET.

## **[2.2](#). Polling Delivery using HTTP**

This method allows an Event Receiver to use HTTP POST ([Section 4.3.3 \[RFC7231\]](#)) 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 (see [Section 2 \[RFC6202\]](#)).

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

- o The Event Receiver makes a request for available SETs using an HTTP POST to a pre-arranged endpoint provided by the Event Transmitter. Or,
- o After validating previously received SETs, the Event Receiver 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 Event Transmitter that has successfully been delivered and attempts to re-deliver are no longer required. Before acknowledgement, Event Receivers SHOULD ensure received SETs have been validated and retained in a manner appropriate to the receiver's retention requirements. The level and method of retention of SETs by Event Receivers is out-of-scope of this specification.

### **2.2.1. Polling HTTP Request Attributes**

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

The request payloads are delivered in one of two forms as described in [Section 2.2.3](#) and [Section 2.2.4](#)

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

The following JSON Attributes 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 Event Receivers that would like to perform an acknowledge only request. This enables the Receiver to use separate HTTP requests for acknowledgement and reception of SETs. When zero returned events is requested, the value of the attribute "returnImmediately" SHALL be ignored as an immediate response is expected.

##### **returnImmediately**

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



## SET Acknowledgment Parameters

### ack

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

### 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](#).

### **[2.2.2.](#) Polling HTTP Response Attributes**

In response to a poll request, the Event Transmitter checks for available SET events and responds with a JSON document containing the following JSON attributes:

#### sets

A JSON object that contains zero 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 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".

### **[2.2.3.](#) Poll Request**

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

If after a period of time, negotiated between the Event Transmitter and Receiver, an Event Transmitter MAY re-issue SETs it has previously delivered. The Event Receiver SHOULD accept repeat SETs and acknowledge the SETs regardless of whether the Receiver believes



it has already acknowledged the SETs previously. An Event Transmitter MAY limit the number of times it attempts to deliver a SET. Upon abandoning delivery of a SET, the Event Transmitter SHOULD have a method to notify the Event Receiver of the loss such as through a status service (not defined by this specification).

If the Event Receiver has received SETs from the Event Transmitter, the Event Receiver SHOULD parse and validate received SETs to meet its own requirements and SHOULD acknowledge receipt in a timely (e.g. minutes) fashion so that the Event Transmitter may mark the SETs as received. Event Receivers 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 an Event Receiver 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 an Event Receiver sets the "maxEvents" attribute to "0" along with "ack" and "err" attributes indicating the Event Receiver 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 an Event Receiver is both acknowledging previously received SETs using the "ack" and "err" attributes and will wait for the next group of SETs in the Event Transmitters response.

#### **2.2.3.1. Poll Only Request**

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

The following is an example request made by an Event Receiver that has no outstanding SETs to acknowledge and is polling for available SETs.



The following is a non-normative example 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
```

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

Figure 1: Example Initial Poll Request

An Event Receiver MAY poll with no parameters at all 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

#### **2.2.3.2. Acknowledge Only Request**

In this variation, the Event Receiver acknowledges previously received SETs and indicates it does not want to receive SETs in response by setting the "maxEvents" attribute to "0".

This variation is typically used when an Event Receiver 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 with acknowledgement of SETs received (for example as shown in Figure 6).

```
POST /Events HTTP/1.1

Host: notify.exampleidp.com
Authorization: Bearer h480djs93hd8
Content-Type: application/json
Authorization: Bearer h480djs93hd8

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

Figure 3: Example Acknowledge Only equest

#### **2.2.3.3. Poll with Acknowledgement**

This variation allows a receiver 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 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":[
    "4d3559ec67504aaba65d40b0363faad8",
    "3d0c3cf797584bd193bd0fb1bd4e7d30"
  ],
  "returnImmediately":false
}
```

Figure 4: Example Poll With Acknowledgement and No Errors



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

#### **2.2.3.4. Poll with Acknowledgement and Errors**

In the case where errors were detected in previously delivered SETs, the Event Receiver MAY use the "setErrs" attribute to indicate errors in the following poll request.

The following is a non-normative example of a response acknowledging 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/json  
Authorization: Bearer h480djs93hd8

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

Figure 5: Example Poll Acknowledgement With Error

#### **2.2.4. 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 Event Transmitter SHALL delay responding until a SET is available unless the poll request parameter "returnImmediately" is "true".

As described in [Section 2.2.2](#) a JSON document is returned containing a number of attributes including "sets" which SHALL contain zero or more SETs.



The following is a non-normative example response to the request shown [Section 2.2.3](#). This example shows two SETs are returned.

HTTP/1.1 200 OK

Content-Type: application/json

Location: <https://notify.exampleidp/Events>

```
{
  "sets": {
    "4d3559ec67504aaba65d40b0363faad8":
      "eyJhbGciOiJub25lIn0.eyJqdGkiOiI0ZDM1NTllYzY3NTA0YWFiYTY1ZDQwYjAzNjNmYWFKOCIsIm1hdCI6MTQ1ODQ5NjQwNCwiaXNzIjoiaHR0cHM6Ly9zY2ltLmV4YW1wbGUuY29tIiwiaXVkiIjpbImh0dHBzOi8vc2NpbS5leGFtcGx1LmNvbS9GZWVkcyc850GQ1MjQ2MWZhNWJiYzg3OTU5M2I3NzU0IiwiaHR0cHM6Ly9zY2ltLmV4YW1wbGUuY29tL0ZlZWRzLzVknzYwNDUxNmIxZDA4NjQxZDc2NzZlZTciXSwiZXZlbnRzIjp7InVyb2ppZXRmOnBhcmFtczpzY2ltOmV2ZW50OmNyZWFOZSI6eyJyZWYiOiJodHRwczovL3Njaw0uZXhhbXBsZS5jb20vVXNlcnMvNDRmNjE0MmRmOTZiZDZhYjYxZTc1MjFkOSIsImF0dHJpYnV0ZXMiOiI0lsiaWQiLCJ1YyW1lIiwidXNlck5hbWUiLCJwYXNzd29yZCI6ImVtYWlscyJdfX19.",
    "3d0c3cf797584bd193bd0fb1bd4e7d30":
      "eyJhbGciOiJub25lIn0.eyJqdGkiOiIzZDBjM2NmNzk3NTg0YmQxOTNiZDBmYjFiZDRlN2QzMCI6MTQ1ODQ5NjAyNSwiaXNzIjoiaHR0cHM6Ly9zY2ltLmV4YW1wbGUuY29tIiwiaXVkiIjpbImh0dHBzOi8vamh1Yi5leGFtcGx1LmNvbS9GZWVkcyc850GQ1MjQ2MWZhNWJiYzg3OTU5M2I3NzU0IiwiaHR0cHM6Ly9qaHViLmV4YW1wbGUuY29tL0ZlZWRzLzVknzYwNDUxNmIxZDA4NjQxZDc2NzZlZTciXSwic3ViIjoiaHR0cHM6Ly9zY2ltLmV4YW1wbGUuY29tL1VzZXJzLzQ0ZjYxNDJkZjk2YmQ2YWI2MWU3NTIxZDkiLCJldmVudHMiOiI0nsidXJuOmllZGY6cGFyYyW1zOnNjaw06ZXZlbnQ6cGFzc3dvcmRSZXNldCI6eyJpZCI6IjQ0ZjYxNDJkZjk2YmQ2YWI2MWU3NTIxZDkiSwiaHR0cHM6Ly9leGFtcGx1LmNvbS9zY2ltL2V2ZW50L3Bhc3N3b3JkUmVzZXRFbHQiOiI0nsicmVzZXRBdHRlbXB0cyI6NX19fQ."
  }
}
```

Figure 6: Example Poll Response

In the above example, a two SETs whose "jti" are "4d3559ec67504aaba65d40b0363faad8" and "3d0c3cf797584bd193bd0fb1bd4e7d30" are delivered.



The following is a non-normative example response to the request shown [Section 2.2.3](#) showing 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 Event Receiver parses and verifies the received SETs and notifies the Event Transmitter via the next poll request to the Event Transmitter as described in [Section 2.2.3.3](#) or [Section 2.2.3.4](#).

### [2.3.](#) Error Response Handling

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

Err Value	Description
json	Invalid JSON object.
jwtParse	Invalid or unparsable JWT or JSON structure.
jwtHdr	In invalid JWT header was detected.
jwtCrypto	Unable to parse due to unsupported algorithm.
jws	Signature was not validated.
jwe	Unable to decrypt JWE encoded data.
jwtAud	Invalid audience value.
jwtIss	Issuer not recognized.
setType	An unexpected Event type was received.
setParse	Invalid structure was encountered such as an inability to parse or an incomplete set of Event claims.
setData	SET event claims incomplete or invalid.
dup	A duplicate SET was received and has been ignored.

Table 1: SET Errors

An error response SHALL include a JSON object which provides details about the error. The JSON object 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 a batch of SETs, the above JSON is included as part of the "setErrs" attribute as defined in [Section 2.2.2](#) and [Section 2.2.3.4](#)

### **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. 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).

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:

- o Clients using an assertion to authenticate and/or act on behalf of itself;
- o Clients acting on behalf of a user; and,
- o 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 the Event Receiver 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 Transmitters and Event 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 specialize Event Streams so that the content is 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, 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]  
Hunt, P., Denniss, W., Ansari, M., and M. Jones, "Security Event Token (SET)", [draft-ietf-secevent-token-00](#) (work in progress), January 2017.
- [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>>.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, [RFC 3986](#), DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/info/rfc3986>>.
- [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>>.





- [RFC5988] Nottingham, M., "Web Linking", [RFC 5988](#), DOI 10.17487/RFC5988, October 2010, <<https://www.rfc-editor.org/info/rfc5988>>.
- [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>>.
- [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>>.
- [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>>.
- [RFC7517] Jones, M., "JSON Web Key (JWK)", [RFC 7517](#), DOI 10.17487/RFC7517, May 2015, <<https://www.rfc-editor.org/info/rfc7517>>.
- [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>>.
- [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>>.

## **7.2. Informative References**

- [openid-connect-core] NRI, "OpenID Connect Core 1.0", Nov 2014.
- [POSIX.1] Institute of Electrical and Electronics Engineers, "The Open Group Base Specifications Issue 7", IEEE Std 1003.1, 2013 Edition, 2013.



- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", [RFC 3339](#), DOI 10.17487/RFC3339, July 2002, <<https://www.rfc-editor.org/info/rfc3339>>.
- [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>>.
- [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>>.
- [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>>.
- [RFC7617] Reschke, J., "The 'Basic' HTTP Authentication Scheme", [RFC 7617](#), DOI 10.17487/RFC7617, September 2015, <<https://www.rfc-editor.org/info/rfc7617>>.



[saml-core-2.0]

Internet2, "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0", March 2005.

## **[Appendix A](#). Other Streaming Specifications**

[[EDITORS NOTE: This section to be removed prior to publication]]

The following pub/sub, queuing, streaming systems were reviewed as possible solutions or as input to the current draft:

### XMPP Events

The WG considered the XMPP events and its ability to provide a single messaging solution without the need for both polling and push modes. The feeling was the size and methodology of XMPP was too far apart from the current capabilities of the SECEVENTs community which focuses in on HTTP based service delivery and authorization.

### Amazon Simple Notification Service

Simple Notification Service, is a pub/sub messaging product from AWS. SNS supports a variety of subscriber types: HTTP/HTTPS endpoints, AWS Lambda functions, email addresses (as JSON or plain text), phone numbers (via SMS), and AWS SQS standard queues. It doesn't directly support pull, but subscribers can get the pull model by creating an SQS queue and subscribing it to the topic. Note that this puts the cost of pull support back onto the subscriber, just as it is in the push model. It is not clear that one way is strictly better than the other; larger, sophisticated developers may be happy to own message persistence so they can have their own internal delivery guarantees. The long tail of OIDC clients may not care about that, or may fail to get it right. Regardless, I think we can learn something from the Delivery Policies supported by SNS, as well as the delivery controls that SQS offers (e.g. Visibility Timeout, Dead-Letter Queues). I'm not suggesting that we need all of these things in the spec, but they give an idea of what features people have found useful.

Other information:

- o API Reference:  
<http://docs.aws.amazon.com/AWSSimpleQueueService/latest/APIReference/Welcome.html>
- o Visibility Timeouts:  
<http://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/sqs-visibility-timeout.html>



## Apache Kafka

Apache Kafka is an Apache open source project based upon TCP for distributed streaming. It prescribes some interesting general purpose features that seem to extend far beyond the simpler streaming model SECEVENTs is after. A comment from MS has been that Kafka does an acknowledge with poll combination event which seems to be a performance advantage. See: <https://kafka.apache.org/intro>

## Google Pub/Sub

Google Pub Sub system favours a model whereby polling and acknowledgement of events is done as separate endpoints as separate functions.

Information:

- o Cloud Overview - <https://cloud.google.com/pubsub/>
- o Subscriber Overview - <https://cloud.google.com/pubsub/docs/subscriber>
- o Subscriber Pull(poll) - <https://cloud.google.com/pubsub/docs/pull>

## **Appendix B.** Acknowledgments

The editors would like to thanks the members of the SCIM WG which began discussions of provisioning events starting with: [draft-hunt-scim-notify-00](#) in 2015.

The editors would like to thank the authors of [draft-ietf-secevent-delivery-02](#), on which this draft is based.

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

## **Appendix C.** Change Log

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.





Authors' Addresses

Annabelle Backman (editor)  
Amazon

Email: [richanna@amazon.com](mailto:richanna@amazon.com)

Michael B. Jones (editor)  
Microsoft

Email: [mbj@microsoft.com](mailto:mbj@microsoft.com)

URI: <http://self-issued.info/>

Phil Hunt (editor)  
Oracle Corporation

Email: [phil.hunt@yahoo.com](mailto:phil.hunt@yahoo.com)

Marius Scurtescu  
Google

Email: [mscurtescu@google.com](mailto:mscurtescu@google.com)

Morteza Ansari  
Cisco

Email: [morteza.ansari@cisco.com](mailto:morteza.ansari@cisco.com)

Anthony Nadalin  
Microsoft

Email: [tonynad@microsoft.com](mailto:tonynad@microsoft.com)

