

Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: May 2, 2018

P. Hunt, Ed.  
Oracle  
October 29, 2017

SET Security Event Stream Management and Provisioning  
draft-hunt-secevent-stream-mgmt-00

Abstract

This specification defines a "control plane" service which enables a client (e.g. an Event Receiver) to establish, monitor, and manage a Security Event Token Stream.

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 May 2, 2018.

Copyright Notice

Copyright (c) 2017 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction and Overview . . . . .	2
1.1. Notational Conventions . . . . .	3
1.2. Definitions . . . . .	4
2. Stream Monitoring and Configuration Retrieval . . . . .	4
2.1. Event Stream Configuration Attributes . . . . .	5
2.2. Checking Stream Configuration and Stream State . . . . .	8
2.3. Event Stream State Model . . . . .	12
3. Stream Management and Provisioning . . . . .	14
3.1. Creating An Event Stream . . . . .	14
3.2. Updating An Event Stream . . . . .	16
3.2.1. Update using HTTP PUT . . . . .	17
3.2.2. Update using HTTP PATCH . . . . .	19
4. Models for Managing Stream Subjects . . . . .	21
4.1. General Considerations for Managing Subjects . . . . .	22
4.2. Subjects as Part of Stream Configuration . . . . .	22
4.2.1. Checking Subject Membership . . . . .	22
4.2.2. Adding and Removing Subjects to a Stream . . . . .	25
4.3. Subjects as Members of a Group . . . . .	27
4.3.1. Checking Membership . . . . .	28
4.3.2. Adding and Removing SCIM Users to a Group . . . . .	29
4.4. Subjects as a Resource (aka POST Profile) . . . . .	31
4.4.1. Adding A Subject to a Stream . . . . .	33
4.4.2. Querying for Subject in Event Streams . . . . .	34
4.4.3. Removing a Subject from an Event Stream . . . . .	35
5. Event Stream Verification . . . . .	35
6. Privacy Considerations . . . . .	37
6.1. Subject Management . . . . .	37
7. Security Considerations . . . . .	37
7.1. Multi-Party Access to Streams . . . . .	37
8. IANA Considerations . . . . .	38
8.1. Registration of Verify Event URI . . . . .	38
8.2. SCIM Schema Registration . . . . .	38
9. References . . . . .	39
9.1. Normative References . . . . .	39
9.2. Informative References . . . . .	39
Appendix A. Event Stream Resource Type and Schema Definitions . . . . .	40
Appendix B. Acknowledgments . . . . .	47
Appendix C. Change Log . . . . .	47
Author's Address . . . . .	47

## 1. Introduction and Overview

This specification defines a "Control Plane" service that defines how an Event Receiver or its agent may provision, monitor, and manage the configuration of an Event Stream that delivers Security Event Tokens (see [I-D.ietf-secevent-token]) using delivery methods such as

specified in the SET Delivery Using HTTP Specification (see [I-D.ietf-secevent-delivery]).

The specification defines the common metadata Event Transmitters and Receivers use to describe HTTP service endpoints, methods, optional signing and encryption modes, as well as the type and content of SETs delivered over a Stream. The specification defines how the Event Receiver parties may review and update the current configuration and confirm operational delivery status using HTTP over TLS.

The mandatory part of this specification (see Section 2) uses a profile of SCIM (see [RFC7643] and [RFC7644]) to implement Event Stream configuration, monitoring and retrieval using HTTP GET Section 4.3.1 [RFC7231]. Additionally, SCIM MAY be used to manage and update Event Stream configuration and operational state.

The choice of SCIM has been recommended as it is intended as a general purpose layer that can be applied to many underlying systems. SCIM's extensibility mechanisms to define data types (resource types) enable it to be flexibly used by specifications intending to profile SET Tokens and Delivery for use in many ways.

For the purposes of the Control Plane, SCIM Section 2 [RFC7643] provides the JSON data definitions that enable the Control Plane to allow service providers and clients to negotiate attributes and resource types used in different SET Profiles. This includes declarations and discovery of attribute types, mutability, cardinality, and returnability that MAY differ between deployments and SET Event type profiles. For HTTP protocol handling and error signaling, the processing rules in [RFC7644] SHALL be applied.

### 1.1. 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]. These keywords are capitalized when used to unambiguously specify requirements of the protocol or application features and behavior that affect the inter-operability and security of implementations. When these words are not capitalized, they are meant in their natural-language sense.

For purposes of readability examples are not URL encoded. Implementers MUST percent encode URLs as described in Section 2.1 of [RFC3986]. Many examples show only partial response and may use "... " to indicate omitted data.

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] and SET Token Delivery specification [I-D.ietf-secevent-delivery].

The following definitions are defined for Security Event distribution:

### Control Plane

A Control Plane represents a service offered by an Event Transmitter that lets an Event Receiver query the current operational and/or error status of an Event Stream. The Control Plane MAY also be used to retrieve Event Stream and SET configuration data.

### Data Plane

The Data Plane represents the HTTP service offered by an Event Receiver that allows the Event Transmitter to deliver multiple SETs via HTTP POST as part of an Event Stream.

**Client** A Client is any actor, typically represented by an authorization credential, authorized to make changes to an Event Stream. Verify often this is an actor belonging to the Event Receiver organization. Actors can be servers, monitoring services, and administrators.

## 2. Stream Monitoring and Configuration Retrieval

The Control Plane is an HTTP service associated with an Event Transmitter that enables the provisioning and monitoring of Event Streams by entities such as Event Receivers, administrators, and monitoring services. This section describes required functionality to enable Event Receivers to retrieve configuration attributes and to detect SET delivery problems that may occur when an Event Transmitter fails to deliver SETs.

This specification also defines optional Control Plane services to create and update streams in sections Section 3 and Section 4.

## 2.1. Event Stream Configuration Attributes

An Event Stream is defined by a set of attributes which together define an Event Stream's operational configuration:

### eventUris

A read only array of JSON String values which are the URIs of events configured for the Event Stream. This attribute is assigned by the Control Plane provider in response to receiving an Event Stream creation or update request. See "eventUris\_req".

### eventUris\_req

An array of JSON String values which are the URIs of events requested by the Event Receiver for the Stream. This attribute is modifiable. An Event Stream provider MAY use this attribute to request requested Event URIs over time that may not be initially offered.

### eventUris\_avail

A read only array of JSON String values which are the URIs of events that the Event Transmitter is able to support. This attribute MAY be used by Control Plane clients to discover new events that may become available over time.

### methodUri

A REQUIRED JSON String value which represents the method used to transfer SETs to the Event Receiver. See [I-D.ietf-secevent-delivery].

### deliveryUri

A JSON String value containing a URI that describes the location where SETs are received (e.g. via HTTP POST). Its format and usage requirements are defined by the associated "methodUri".

### iss

The URI for the publisher of the SETs that will be issued for the Event Stream. See Section 2.1 [I-D.ietf-secevent-token].

### aud

An OPTIONAL JSON Array of JSON String values which are URIs representing the audience(s) of the Event Stream. The value SHALL be the value of SET "aud" claim sent to the Event Receiver.

### iss\_jwksUri

An OPTIONAL String that contains the URL of the SET issuers public JSON Web Key Set [RFC7517]. This contains the signing key(s) the Event Receiver uses to validate SET signatures from the Event

Transmitter that will be used by the Event Receiver to verify the authenticity of issued SETs.

**aud\_jwksUri**

An OPTIONAL JSON Web Key Set [RFC7517] that contains the Event Receiver's encryption keys that MAY be used by the Event Transmitter to encrypt SET tokens for the specified Event Receiver.

**status**

An OPTIONAL JSON String keyword that indicates the current state of an Event Stream. More information on the Event Stream state can be found in Section 2.3. Valid keywords are:

"on" - indicates the Event Stream has been verified and that the Feed Provider MAY pass SETs to the Event Receiver.

"paused" - indicates the Event Stream is temporarily suspended. While "paused", SETs SHOULD be retained and delivered when state returns to "on". If delivery is paused for an extended period defined by the Event Transmitter, the Event Transmitter MAY change the state to "off" indicating SETs are no longer retained.

"off" - indicates that the Event Stream is no longer passing SETs. While in off mode, the Event Stream configuration is maintained, but new events are ignored, not delivered or retained. Before returning to "on", a verification MUST be performed.

"fail" - indicates that the Event Stream was unable to deliver SETs to the Event Receiver due an unrecoverable error or for an extended period of time. Unlike paused status, a failed Event Stream does not retain existing or new SETs that are issued. Before returning to "on", a verification MUST be performed.

**maxRetries**

An OPTIONAL JSON number indicating the maximum number of attempts to deliver a SET. A value of '0' indicates there is no maximum. Upon reaching the maximum, the Event Stream "status" attribute is set to "failed".

**maxDeliveryTime**

An OPTIONAL number indicating the maximum amount of time in seconds a SET MAY take for successful delivery per request or cumulatively across multiple retries. Upon reaching the maximum, the Event Stream "status" is set to "failed". If undefined, there is no maximum time.

**minDeliveryInterval**

An OPTIONAL JSON integer that represents the minimum interval in seconds between deliveries. A value of '0' indicates delivery should happen immediately. When delivery is a polling method (e.g. HTTP GET), it is the expected time between Event Receiver attempts. When in push mode (e.g. HTTP POST), it is the interval the server will wait before sending a new event or events.

**txErr**

An OPTIONAL JSON String keyword value. When the Event Stream has "subState" set to "fail", one of the following error keywords is set:

"connection" indicates an error occurred attempting to open a TCP connection with the assigned endpoint.

"tls" indicates an error occurred establishing a TLS connection with the assigned endpoint.

"dnsname" indicates an error occurred establishing a TLS connection where the dnsname was not validated.

"receiver" indicates an error occurred whereby the Event Receiver has indicated an error for which the Event Transmitter is unable to correct.

**txErrDesc**

An OPTIONAL String value that is usually human readable that provides further diagnostic detail by the indicated "txErr" error code.

**verifyNonce** A String value that when changed or set by a Control Plane client will cause the Event Transmitter to issue a single Verify Event based on the nonce value provided (see Section 5). The intent of the value is to allow the Event Receiver to confirm the Verify Event received matches the value set in the configuration. While this value MAY be updated (see Section 5), its value is usually not returned as part of an Event Stream configuration.

**subjects**

An OPTIONAL complex attribute containing sub objects whose sub-attributes define subjects against which SETs may be issued. The following sub-attributes are defined:

**value** A String which uniquely identifies a subject (or set of subjects) to be included in the Stream. The format and type of value is defined by the 'type' sub-attribute.

**iss** A String which contains the URI of the issuer of the subject identified in the "value" attribute. When not supplied the issuer is assumed to be the Event Stream issuer.

**type** A case-insensitive canonical String value which defines the contents of the attribute 'value'. Valid type values are:

**OIDC** Is a String value corresponding to an OpenID Connect subject. The corresponding "iss" attribute is set with the OpenId Connect iss value.

**SAML** A String value that is a URI that represents the subject of a SAML Identity Provider.

**EMAIL** A String Value that is the Email addresses for a subject. The value SHOULD be specified according to [RFC5321].

**PHONE** Phone numbers for the user. The value SHOULD be specified according to the format defined in [RFC3966], e.g., 'tel:+1-201-555-0123'.

**User** A SCIM User where value is the 'id' of a User resource in the local SCIM service provider.

**Group** A SCIM Group where the value is the 'id' of a Group resource in the local SCIM service provider.

**URI** A miscellaneous subject that can be identified by a URI.

Additional Event Stream configuration (attributes) MAY be defined as extensions. The method for adding new attributes is defined in Section 3.3 [RFC7643].

## 2.2. Checking Stream Configuration and Stream State

An Event Receiver MAY check the current status of a Stream the Event Transmitter's Control Plane service by performing an HTTP GET using the provided URI from the Event Transmitter either through an administrative process or via the optional Stream creation response defined in Section 3.1.

The format of the Stream GET request and response is defined by Section 3.4 [RFC7644].

In addition to the basic attributes defined in Section 2 [RFC7643] common to all resource types, an "EventStream" resource types uses the attributes defined in Section 2.1. As with any SCIM resource, an



"EventStream" resource MUST include the JSON attributes "schemas" and "id" as defined in [RFC7643]:

schemas

Is an array of Strings with at least a single value of "urn:ietf:params:scim:schemas:event:2.0:EventStream". Configuration MAY be extended through the addition of other schema URI values such as in the case where a new delivery method or SET profile needs to define additional attributes.

id

Is a String which is a permanent unique identifier for "EventStream" resources. The value which is also used to define a permanent Event Stream Resource URI.

The example below retrieves a specific "EventStream" resource whose "id" is "548b7c3f77c8bab33a4fef40".

```
GET /EventStreams/767aad7853d240debc8e3c962051c1c0
Host: example.com
Accept: application/json
Authorization: Bearer h480djs93hd8
```

Figure 1: Example EventStream HTTP GET Request

Below is an example response to the "EventStream" retrieval made in Figure 1.

```
HTTP/1.1 200 OK
Content-Type: application/scim+json
Location:
  https://example.com/EventStreams/767aad7853d240debc8e3c962051c1c0

{
  "schemas":["urn:ietf:params:scim:schemas:event:2.0:EventStream"],
  "id":"767aad7853d240debc8e3c962051c1c0",
  "eventUris_req":[
    "http://schemas.openid.net/event/backchannel-logout"
  ],
  "eventUris":[
    "http://schemas.openid.net/event/backchannel-logout"
  ],
  "methodUri":"urn:ietf:params:set:method:HTTP:webCallback",
  "deliveryUri":"https://notify.examplerp.com/Events",
  "aud":"https://sets.myexamplerp.com",
  "status":"fail",
  "txErr":"connection",
  "txErrDesc":"TCP connect error to notify.examplerp.com.",
  "maxDeliveryTime":3600,
  "minDeliveryInterval":0,
  "description":"Logout events from oidc.example.com",
  "meta":{
    ... SCIM meta attributes ...
  }
}
```

Figure 2: Example Stream GET Response

In the above figure, the "EventStream" shows a "status" of "fail" due to a TCP connection error. In this case, the Event Receiver is able to discover that its endpoint was unavailable and has been marked failed by the Event Transmitter (possibly explaining a lack of received SETs). Typically, with this type of error, appropriate operations staff would be alerted and some corrective action would be taken to check for a configuration error or service failure.

The frequency with which Event Receivers poll the Event Stream status depends on factors such as:

- o The level of technical fault tolerance and availability of the receiving endpoint.

- o The amount of risk that can be tolerated for lost events. For example, if Security Events are considered informational, then infrequent (hourly or daily) may be sufficient.
- o The amount of buffer recovery offered by an Event Transmitter which MAY be minutes depending on SET frequency and buffer size.

In many cases Event Stream status monitoring may be triggered on a timeout basis. Event Receivers would typically poll if they have not received a SET for some period during which SETs would be expected based on past experience.

Receivers MAY use the endpoint "/EventStreams" to query and retrieve available Event Streams based on the provided "Authorization" header.

The example below retrieves any "EventStream" resources based solely on the requestor's authorization header.

```
GET /EventStreams/  
Host: example.com  
Accept: application/json  
Authorization: Bearer h480djs93hd8
```

Figure 3: Example Stream HTTP GET Request From Common Endpoint

```
HTTP/1.1 200 OK
Content-Type: application/scim+json
Location:
  https://example.com/EventStreams/767aad7853d240debc8e3c962051c1c0

{
  "schemas": [ "urn:ietf:params:scim:api:messages:2.0:ListResponse" ],
  "totalResults": 1,
  "itemsPerPage": 10,
  "startIndex": 1,
  "Resources": [
    {
      "schemas": [ "urn:ietf:params:scim:schemas:event:2.0:EventStream" ],
      "id": "767aad7853d240debc8e3c962051c1c0",
      "feedName": "OIDCLogoutFeed",
      "eventUris_req": [
        "http://schemas.openid.net/event/backchannel-logout"
      ],
      "eventUris": [
        "http://schemas.openid.net/event/backchannel-logout"
      ],
      "eventUris_avail": [
        "http://schemas.openid.net/event/backchannel-logout"
      ],
      "methodUri": "urn:ietf:params:set:method:HTTP:webCallback",
      "deliveryUri": "https://notify.examplerp.com/Events",
      "aud": "https://sets.myexamplerp.com",
      "status": "fail",
      "txErr": "connection",
      "txErrDesc": "TCP connect error to notify.examplerp.com.",
      "maxDeliveryTime": 3600,
      "minDeliveryInterval": 0,
      "description": "Logout events from oidc.example.com",
      "meta": {
        ... SCIM meta attributes ...
      }
    }
  ]
}
```

Figure 4: Example Event Stream List/Query Response Form

### 2.3. Event Stream State Model

The Event Stream configuration attribute "status" reports the current state of an Event Stream with regards to whether the stream is operational or is in a suspended or failed state. Additionally, the "status" attribute can be used to pause or stop streams using the stream configuration update functions described in Section 3.

The following is the state machine representation of a Event Stream on a Event Transmitter. Note that a Event Stream cannot be made active until a verification process has been completed. As such, a newly created Event Stream begins with state "on".

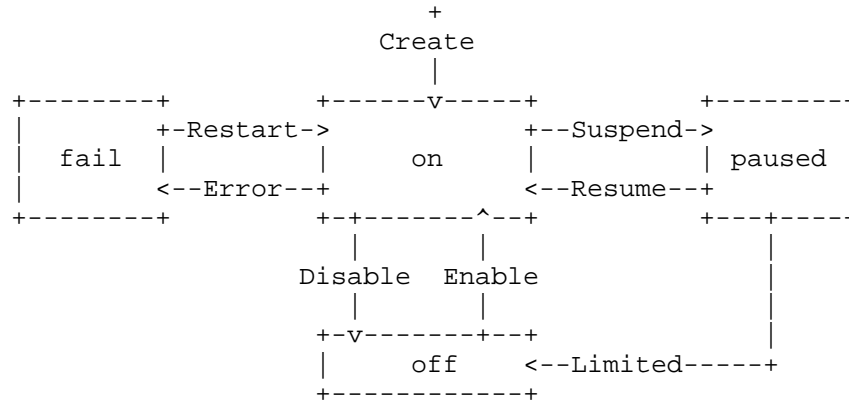


Figure 5: Event Stream States at Event Transmitter

In Figure 5, the following actions impact the operational state of an Event Stream. "status" values are shown in the boxes, and change based on the following actions:

#### Create

A Event Receiver or an administrator creates a new Event Stream as described in Section 3.1. The initial state is "on".

#### Error

An Event Transmitter that has not been able to deliver a SET over one or more retries which has reached a limit of attempts ("maxRetries") or time ("maxDeliveryTime") MAY set the Event Stream state to "fail". What stream status is set to "failed", the Event Transmitter is indicating that SETs are being lost and may not be recoverable.

#### Limited

A paused Event Stream has reached the transmitters ability to retain SETs for delivery. The Event Transmitter changes the state to "off" indicating SET loss is potentially occurring.

#### Restart

An administrator having corrected the failed delivery condition modifies the Event Stream state to "on" (e.g. see Section 3.2).

#### Suspend and Resume

An Event Stream MAY be suspended and resumed by updating the Event Stream state to "paused" or "on". For example, see see Section 3.2. While suspended, the Event Transmitter retains undelivered SETs for a period of time and resources specified by the Event Transmitter (see "Limited").

#### Enable and Disable

A Event Stream MAY be disabled and enabled by updating the Event Stream "state" to "off" or "on". For example, see see Section 3.2. While the Event Stream is disabled, all SETs that occur at the Event Transmitter are lost.

### 3. Stream Management and Provisioning

This section describes optional Stream management provisioning features that allow receivers or provisioning systems to create streams and update configuration to perform actions such as rotation, and operational state (e.g. suspend, stop, or resume) management.

The operations specified in this section are based on [RFC7644]. SCIM schema declarations for the "EventStream" resources are defined in Appendix A. HTTP Protocol usage and processing rules are provided by [RFC7644].

#### 3.1. Creating An Event Stream

To define an Event Stream, the Event Receiver or its administrator (known as the client) first obtains an authorization credential allowing the ability to define a new Stream. Note: the process for registering to obtain credentials and permission to register is out-of-scope of this specification.

Upon obtaining authorization, the client issues an HTTP POST request as defined in Section 3.3 [RFC7644]. To complete the request, the administrative entity provides the required Stream configuration attributes as specified in Section 2.1, the delivery method [I-D.ietf-secevent-delivery] and any additional configuration specified by the SET Event Specifications that are being used.

The client MAY discover the Event Transmitter's Control Plane service for the schema requirements for "EventStream" resource type and any other extensions using SCIM schema discovery in Section 4 [RFC7644].

The process to create an Event Stream is as follows:

1. The client initiates an HTTP POST to the Control Plane endpoint and provides a JSON document defining an EventStream which

contains information about the Event Receivers endpoints, settings, and keys.

2. Upon validating the request, the Event Transmitters control plane provisions the stream and updates the EventStream configuration with the corresponding Event Transmitter information.
3. The Control Plane responds to the request from step 1 and returns the final representation of the Event Stream configuration along with a pointer to the created EventStream resource that the client MAY use to monitor status and update configuration.
4. Upon receiving the response, the client completes the client side configuration and provisioning based upon the returned EventStream configuration.

In the following non-normative example, a request to create a new "EventStream" is submitted.

```
POST /EventStreams
Host: example.com
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Bearer h480djs93hd8

{
  "schemas":["urn:ietf:params:scim:schemas:event:2.0:EventStream"],
  "feedName":"OIDCLogoutFeed",
  "eventUris_req":[
    "http://schemas.openid.net/event/backchannel-logout"
  ],
  "methodUri":"urn:ietf:params:set:method:HTTP:webCallback",
  "deliveryUri":"https://notify.examplerp.com/Events",
  "aud":"https://sets.myexamplerp.com",
  "maxDeliveryTime":3600,
  "minDeliveryInterval":0,
  "description":"Logout events from oidc.example.com"
}
```

Figure 6: Example Create Event Stream Request

In following non-normative response, the Control Plane provider has automatically assigned an HTTP addressable location for the EventStream resource as well as an "id". Additionally, the Control Plane response below includes additional configuration data for "iss" and "iss\_jwksUri".

```
HTTP/1.1 201 Created
Content-Type: application/scim+json
Location:
  https://example.com/v2/EventStreams/767aad7853d240debc8e3c962051c1c0

{
  "schemas":["urn:ietf:params:scim:schemas:event:2.0:EventStream"],
  "id":"767aad7853d240debc8e3c962051c1c0",
  "feedName":"OIDCLogoutFeed",
  "eventUris_req":[
    "http://schemas.openid.net/event/backchannel-logout"
  ],
  "eventUris":[
    "http://schemas.openid.net/event/backchannel-logout"
  ],
  "eventUris_avail":[
    "http://schemas.openid.net/event/backchannel-logout"
  ],
  "methodUri":"urn:ietf:params:set:method:HTTP:webCallback",
  "deliveryUri":"https://notify.examplerp.com/Events",
  "aud":"https://sets.myexamplerp.com",
  "status":"on",
  "maxDeliveryTime":3600,
  "minDeliveryInterval":0,
  "iss":"oidc.example.com"
  "iss_jwksUri":"https://example.com/keys/oidc-example-com.jwks"
  "description":"Logout events from oidc.example.com",
  "meta":{
    ... SCIM meta attributes ...
  }
}
```

Figure 7: Example Response to Create EventStream Request

### 3.2. Updating An Event Stream

Two HTTP methods are available to update an Event Stream configuration.

The HTTP PUT operation accepts a JSON Document representing an existing EventStream configuration and replaces it.



An optional HTTP PATCH operation uses a JSON Patch [RFC6902] style request format to allow manipulation of specific EventStream configuration such as (but not limited to) "status", and "subjects".

### 3.2.1. Update using HTTP PUT

The HTTP PUT method allows a client having previously received the EventStream JSON document to modify the document and replace the Control Plane provider's copy. In using this method, the client is not required to remove data normally asserted or defined by the EventStream Control Plane provider (e.g. attributes that are read only). The processing rules of [RFC7644] enable the client to "put back" what was previously received allowing the Control Plane provider to figure out what attributes need updating and which attributes are ignored. For example, while "id" is immutable, the Control Plane provider will simply ignore attempts to replace its value. When processing is complete the final accepted state is represented in the HTTP Response.

In the following non-normative example, a request to replace the existing EventStream "EventStream" is submitted. In this example, the change shown is the status is now set to "off". Note that the client does not have to remove read-only attributes such as "eventUri" and "eventUri\_avail" as these values are ignored as per Section 3.5.1 [RFC7644].

```
PUT /EventStreams/767aad7853d240debc8e3c962051c1c0
Host: example.com
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Bearer h480djs93hd8

{
  "schemas":["urn:ietf:params:scim:schemas:event:2.0:EventStream"],
  "id":"767aad7853d240debc8e3c962051c1c0",
  "feedName":"OIDCLogoutFeed",
  "eventUri_req":[
    "http://schemas.openid.net/event/backchannel-logout"
  ],
  "eventUri":[
    "http://schemas.openid.net/event/backchannel-logout"
  ],
  "eventUri_avail":[
    "http://schemas.openid.net/event/backchannel-logout"
  ],
  "methodUri":"urn:ietf:params:set:method:HTTP:webCallback",
  "deliveryUri":"https://notify.examplerp.com/Events",
  "aud":"https://sets.myexamplerp.com",
  "status":"off",
  "maxDeliveryTime":3600,
  "minDeliveryInterval":0,
  "iss":"oidc.example.com"
  "iss_jwksUri":"https://example.com/keys/oidc-example-com.jwks"
  "description":"Logout events from oidc.example.com",
  "meta":{
    ... SCIM meta attributes ...
  }
}
```

Figure 8: Example Replace Event Stream Request

In following non-normative response, the Control Plane provider responds with the processed final state of the submitted EventStream.

```
HTTP/1.1 200 OK
Content-Type: application/scim+json
Location:
  https://example.com/v2/EventStreams/767aad7853d240debc8e3c962051c1c0

{
  "schemas":["urn:ietf:params:scim:schemas:event:2.0:EventStream"],
  "id":"767aad7853d240debc8e3c962051c1c0",
  "feedName":"OIDCLogoutFeed",
  "eventUris_req":[
    "http://schemas.openid.net/event/backchannel-logout"
  ],
  "eventUris":[
    "http://schemas.openid.net/event/backchannel-logout"
  ],
  "eventUris_avail":[
    "http://schemas.openid.net/event/backchannel-logout"
  ],
  "methodUri":"urn:ietf:params:set:method:HTTP:webCallback",
  "deliveryUri":"https://notify.examplerp.com/Events",
  "aud":"https://sets.myexamplerp.com",
  "status":"off",
  "maxDeliveryTime":3600,
  "minDeliveryInterval":0,
  "iss":"oidc.example.com",
  "iss_jwksUri":"https://example.com/keys/oidc-example-com.jwks",
  "description":"Logout events from oidc.example.com",
  "meta":{
    ... SCIM meta attributes ...
  }
}
```

Figure 9: Example Response to PUT EventStream Request

### 3.2.2. Update using HTTP PATCH

Periodically, Event Receiver parties MAY have need to update an EventStream configuration for the purpose of:

- o Rotating access credentials or keys
- o Updating endpoint configuration
- o Making operational changes such as pausing, resetting, or disabling an Event Stream.

- o Other operations (e.g. such as adding or removing subjects) as defined by profiling Event specifications.

As documented in Section 3.5.2 [RFC7644], one or more PATCH operations (which are based on [RFC6902]) can be made against a single EventStream resource. The update is expressed as a JSON document. The JSON document contains an attribute "Operations" which contains an array of JSON objects each of which each have the following attributes:

op

A JSON attribute whose value is one of "add", "remove", or "replace".

path

A JSON attribute whose value is a document attribute path (see Section 3.5.2 [RFC7644]) describing the attribute or sub-attribute or value to be updated in the case of multi-valued complex attributes such as "subjects".

value

The value to be assigned to the JSON document attribute defined in "path".

In the following non-normative example, the client requests that the "status" configuration attribute be changed to "paused" for the EventStream whose path is identified by the request URI path.

```
PATCH /EventStreams/767aad7853d240debc8e3c962051c1c0
Host: example.com
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Bearer h480djs93hd8

{
  "schemas":
    [ "urn:ietf:params:scim:api:messages:2.0:PatchOp" ],
  "Operations": [{
    "op": "replace",
    "path": "status",
    "value": "paused"
  }]
}
```

Figure 10: Example EventStream PATCH Request

In the above figure, upon receiving the request, the Event Transmitter would stop sending Events to the Receiver based on the requested value of "status" being set to "paused".

In the following non-normative example, the client requests the addition and removal of two subjects from an existing EventStream. This operation is discussed further in Section 4.2.

```
PATCH /EventStreams/767aad7853d240debc8e3c962051c1c0
Host: example.com
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Bearer h480djs93hd8

{
  "schemas":
    [ "urn:ietf:params:scim:api:messages:2.0:PatchOp" ],
  "Operations": [{
    "op": "add",
    "path": "subjects",
    "value": {
      "type": "EMAIL",
      "value": "alice@example.com"
    }
  },
  {
    "op": "remove",
    "path": "subjects[value eq \"bob@example.com\"]"
  }
]
```

Figure 11: Example Changing the Members of EventStream

In the above request, the second operation, the remove operation, uses a "path" attribute to specify a matching filter the correct array element of "subjects" by matching the appropriate sub-attributes which are denoted by square brackets (see Figure 1 and 2 [RFC7644] for other examples and ABNF for filters). In this case the composite filter of the "subjects" sub-attributes "type" and "value" are used to remove the correct JSON array element. Upon receiving the request, the EventStream subjects attribute would be updated to reflect the changes.

#### 4. Models for Managing Stream Subjects

The extensibility of SCIM enables many ways to model subjects that are part of an Event Stream. This section explores a few alternatives that profiling specifications could use to manage the

contents of an Event Stream. These examples include managing subjects:

- o As an attribute of an Event Stream configuration (see Section 4.2);
- o As a member of SCIM Group (see Section 4.3); and,
- o As a specific Subject resource (see Section 4.4).

#### 4.1. General Considerations for Managing Subjects

As a privacy and scalability consideration, profiling specifications SHOULD consider that most deployments SHOULD not allow the subjects that are part of an Event Stream to be enumerated in a single request. For example, in Section 4.2, the Event Stream configuration attribute "subjects" is typically not returned when querying Event Stream configurations (see Section 2.2). This is because the number of values may be too large (e.g. great than 100k values or even in the billions or more). Further, depending on the Security Event types being exchanged, Event Receivers MAY confirm that a subject is part of a stream for privacy reasons.

The ability to return attributes such as "subjects" is indicated by Control Plane service providers in schema discovery (see Section 4 [RFC7644]) as the schema attribute "returned". For "subjects" this attribute SHOULD be set to "request" or "never". In "request" mode, the client must specifically request the attribute "subjects" to have it enumerated. If the mode is `_never_`, the attribute SHALL NOT be returned to clients. In all cases however, a client MAY execute a query to verify the presence of a subject:

#### 4.2. Subjects as Part of Stream Configuration

In this section, examples are given using the "subjects" attribute of Event Stream configuration described in Section 2.1

The following sections assume subject membership within streams is defined by the "subjects" attribute of the Event Stream configuration. As defined, subjects can support a number of value types including: OIDC Connect Subjects, SCIM Users and Groups, e-mail and telephone number identifiers, and URI referencable entities.

##### 4.2.1. Checking Subject Membership

Checking subject membership is a matter of performing a query using a filter to achieve a match based on a value of the "subjects" attribute.

#### 4.2.1.1. Email Based Subjects

In this section, values have been added to the "subjects" attribute that are email addresses which clients to the Control Plane would like to verify are present or not. The "subjects" attribute has sub-attribute "type" set to "EMAIL" and the sub-attribute "value" contains an email address.

In the following non-normative example, a client queries the Control Plane to see if "alice@example.com" is part of any defined stream configuration. In the request, only the attribute "id" of the Event Stream is requested as the client does not need to see the rest of the Event Stream attributes. Note, for readability, the URL is not encoded.

```
GET /EventStreams?filter=(subjects.value eq "alice@example.com")
    &attributes=id
Host: example.com
Accept: application/scim+json
Authorization: Bearer h480djs93hd8
```

Figure 12: Determining if an EMail Subject is in an Event Stream

In this non-normative example response, the subject is confirmed as not part of any Event Streams associated with the requester. In this case an empty list is returned with no values and "totalResults" is "0".

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

```
{
  "schemas":["urn:ietf:params:scim:api:messages:2.0:ListResponse"],
  "totalResults":0,
  "Resources":[]
}
```

Figure 13: Example Response With No Subject Match

In the response below, a match for subject "alice@example.com" is found and the "id" of the Event Stream configuration that contains the subject is returned is "767aad7853d240debc8e3c962051c1c0".

HTTP/1.1 200 OK

Content-Type: application/scim+json

```
{
  "schemas":["urn:ietf:params:scim:api:messages:2.0:ListResponse"],
  "totalResults":1,
  "Resources":[
    {
      "id":"767aad7853d240debc8e3c962051c1c0",
      ...other meta attributes...
    }
  ]
}
```

Figure 14: Example Response With Single Match

#### 4.2.1.2. OIDC Based Subjects

In this section, values in the "subjects" attribute are OIDC users which clients would like to verify are present. The attribute "subjects" has the sub-attribute "type" set to "OIDC" and the sub-attribute "value" contains an OIDC "sub" value and the "iss" sub-attribute contains the corresponding OIDC Provider "iss" value. For this example, the OIDC "iss" is "op.example.com" and the "sub" is "123456".

In the following non-normative example, a client queries the Control Plane to see if the above OIDC user is part of any defined stream configuration. In the request, only the attribute "id" is requested. Note, for readability, the URL is not encoded.

```
GET /EventStreams?filter=(subjects[value eq "123456" and
  iss eq "op.example.com"])&attributes=id
Host: example.com
Accept: application/scim+json
Authorization: Bearer h480djs93hd8
```

Figure 15: Determining if an OIDC Subject is in an Event Stream

In the above request note that "iss" and "value" are enclosed within square brackets. This is done, per Figure 1 [RFC7644] to ensure the matching condition within "[" and "]" is matched against the same JSON array record. If the filter was expressed as "(subjects.value eq "123456" and subjects.iss eq "op.example.com")" Then an improper



match may occur because a composite value of "subjects" may have "123456" while another has "op.example.com".

For examples of responses to Figure 15, see Figure 13 and Figure 14

#### 4.2.2. Adding and Removing Subjects to a Stream

Adding and removing subjects to an Event Stream is performed using the HTTP PATCH method described in Section 3.2.2. The following provides examples of adding and removing subjects based on EMAIL and OIDC subects.

In the following non-normative example, the client requests the addition of a subject identified by an EMAIL address to an existing EventStream. The composite value of subjects has the sub-attributes "type" and "value" which are assigned.

```
PATCH /EventStreams/767aad7853d240debc8e3c962051c1c0
Host: example.com
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Bearer h480djs93hd8

{
  "schemas":
    [ "urn:ietf:params:scim:api:messages:2.0:PatchOp" ],
  "Operations": [{
    "op": "add",
    "path": "subjects",
    "value": {
      "type": "EMAIL",
      "value": "alice@example.com"
    }
  }]
}
```

Figure 16: Adding an EMAIL Subject to a Stream

In the following non-normative example, the client requests the addition of an OIDC subject to an existing EventStream. The composite value of "subjects" has the sub-attributes "type", "iss", and "value" which are assigned.

```
PATCH /EventStreams/767aad7853d240debc8e3c962051c1c0
Host: example.com
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Bearer h480djs93hd8

{
  "schemas":
    ["urn:ietf:params:scim:api:messages:2.0:PatchOp"],
  "Operations": [{
    "op": "add",
    "path": "subjects",
    "value": {
      "type": "OIDC",
      "value": "123456",
      "iss": "op.example.com"
    }
  }]
}
```

Figure 17: Adding an OIDC Provider Subject to a Stream

In the following non-normative example, the client requests the removal of a subject selected by using a filter against the "subjects" attribute.

```
PATCH /EventStreams/767aad7853d240debc8e3c962051c1c0
Host: example.com
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Bearer h480djs93hd8

{
  "schemas":
    ["urn:ietf:params:scim:api:messages:2.0:PatchOp"],
  "Operations": [{
    "op": "remove",
    "path": "subjects[value eq \"123456\" and iss eq \"op.example.com\"]",
  }]
}
```

Figure 18: Removing an OIDC Connect Subject from a Stream

#### 4.3. Subjects as Members of a Group

SCIM defines a resource type called a "Group" which can be used as a container to manage one or more objects in a collection (See Section 4.2 of [RFC7643]). Groups work in similar fashion to the operations described in Section 4.2 except instead of operations against the "subjects", the "members" attribute is used. Typically the value of the members attribute is the "id" of a local User or Group.

In order to use this method, the Stream configuration indicates the SCIM Group being used by adding Group as a member of the "subjects" attribute of the Stream configuration and indicating a "type" of "Group".

The following is an example Stream configuration that has a Group as a member of the subjects. In the example below, "e18c2dfb5d588" is the identifier of a SCIM Group containing a list of member resources.

```
{
  "schemas":["urn:ietf:params:scim:schemas:event:2.0:EventStream"],
  "id":"767aad7853d240debc8e3c962051c1c0",
  "feedName":"OIDCLogoutFeed",
  "eventUri":[
    "http://schemas.openid.net/event/backchannel-logout"
  ],
  "methodUri":"urn:ietf:params:set:method:HTTP:webCallback",
  "deliveryUri":"https://notify.examplerp.com/Events",
  "aud":"https://sets.myexamplerp.com",
  "status":"off",
  "maxDeliveryTime":3600,
  "minDeliveryInterval":0,
  "iss":"oidc.example.com"
  "subjects":[
    {
      "type":"Group"
      "value":"e18c2dfb5d588"
    }
  ]
  "iss_jwksUri":"https://example.com/keys/oidc-example-com.jwks"
  "description":"Logout events from oidc.example.com",
  "meta":{
    ... SCIM meta attributes ...
  }
}
```

Figure 19: Event Stream Configured to Use SCIM Group

#### 4.3.1. Checking Membership

In this section, values have been added to the "members" attribute are "id" values of known SCIM resources. These values can be queried against the Group to see if the "id" is a member.

If the requester does not know the "id" of the resource for which they would like to check membership, a query can be performed as described in Section 3.4 [RFC7644].

In the following non-normative example, a client queries the Control Plane to see if a User with "id" value of "413861904646" is a part of any Group. .

```
GET /Groups?filter=members.value eq "413861904646"
  &attributes=id
Host: scim.example.com
Accept: application/scim+json
Authorization: Bearer h480djs93hd8
```

Figure 20: Determining if a SCIM Resource is in a Event Stream Example

In this non-normative response, the "id" is confirmed as not part of the Group queried by the requester. In this case an empty list is returned with no values and "totalResults" is "0".

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

{
  "schemas":["urn:ietf:params:scim:api:messages:2.0:ListResponse"],
  "totalResults":0,
  "Resources":[]
}
```

Figure 21: Example Response With No Match

In the response below, a match for resource with an "id" value of "413861904646" is found and the "id" of any matched Groups that contain the "id" is returned. In this case a "Group" with an "id" value of \_e18c2dfb5d588\_ is returned.

HTTP/1.1 200 OK

Content-Type: application/scim+json

```
{
  "schemas":["urn:ietf:params:scim:api:messages:2.0:ListResponse"],
  "totalResults":1,
  "Resources":[
    {
      "id":"e18c2dfb5d588",
    }
  ]
}
```

Figure 22: Example Response With Single Match

#### 4.3.2. Adding and Removing SCIM Users to a Group

Adding and removing Users to an Event Stream Group is performed using the HTTP PATCH method described in Section 3.2.2. The following provides examples of adding and removing "Users" to a "Group"

In the following non-normative example, the client requests the addition of a User identified by an "id" to an existing "Group" which is used by an Event Stream to denote member subjects.

```
PATCH /Groups/e18c2dfb5d588
Host: example.com
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Bearer h480djs93hd8

{
  "schemas":
    [ "urn:ietf:params:scim:api:messages:2.0:PatchOp" ],
  "Operations": [ {
    "op": "add",
    "path": "members",
    "value": {
      "type": "User",
      "value": "8c16-01f8e146b87a"
    }
  } ]
}
```

Figure 23: Example Adding a User to a Group

In the following non-normative example, the client requests the removal of a User identified by "id" with value "8c16-01f8e146b87a". The item to be removed is selected by using a filter against the "members" attribute.

```
PATCH /Groups/e18c2dfb5d588
Host: example.com
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Bearer h480djs93hd8

{
  "schemas":
    [ "urn:ietf:params:scim:api:messages:2.0:PatchOp" ],
  "Operations": [ {
    "op": "remove",
    "path": "members.value eq \"8c16-01f8e146b87a\"",
  } ]
}
```

Figure 24: Example Removing a User from a Group

#### 4.4. Subjects as a Resource (aka POST Profile)

This section demonstrates how to model subject participation in an Event Stream by treating subjects as a resource (a "Subject"). In this model, a subject is added, removed, and queried from an Event Stream by through HTTP POST, DELETE, and GET methods.

In this model, a new SCIM resource type is defined that describes the "Subject" resource and its associated schema.

The following is a non-normative example of a Resource Type definition that is configured in a SCIM service provider. The Resource Type defines the "Subjects" endpoint as well as a URI for the schema definition. The Resource Type configuration can be discovered by querying the SCIM service provider's "/ResourceTypes" endpoint (see Section 4 [RFC7644]).

```
{
  "schemas": ["urn:ietf:params:scim:schemas:core:2.0:ResourceType"],
  "id": "Subject",
  "name": "Subject",
  "endpoint": "/Subjects",
  "description": "Endpoint managing SECEVENT Subjects.",
  "schema": "urn:ietf:params:scim:schemas:event:2.0:Subject",
  "schemaExtensions": []
}
```

Figure 25: Example Resource Type Definition for Subjects

The following is an example schema definition for a Subject resource. This configuration can be retrieved from the SCIM Service Provider using the "/Schemas" endpoint (see Section 4 [RFC7644]).

```
{
  "id" : "urn:ietf:params:scim:schemas:event:2.0:Subject",
  "name" : "Subject",
  "description" : "Subject stream configuration",
  "attributes" : [
    {
      "name" : "email",
      "type" : "string",
      "multiValued" : false,
      "description" : "The email of the subject being added to Event Streams. The value SHOULD be specified according to [RFC5321].",
      "required" : true,
      "caseExact" : false,
      "mutability" : "readWrite",
      "returned" : "default",
      "uniqueness" : "none"
    },
    {
      "name" : "displayName",
      "type" : "string",
      "multiValued" : false,
      "description" : "A simple representation of a Subject's name that can be used for display purposes.",
      "required" : false,
      "caseExact" : false,
      "mutability" : "readWrite",
      "returned" : "default",
      "uniqueness" : "none"
    },
    {
      "name" : "streamId",
      "type" : "string",
      "multiValued" : false,
      "description" : "An Event Stream a Subject is part of as identified by EventStream id",
      "required" : false,
      "mutability" : "readWrite",
      "returned" : "default"
    }
  ]
}
```

Figure 26: Example Schema for Subject Resources



#### 4.4.1. Adding A Subject to a Stream

To add a Subject to a Stream, an HTTP POST is used to create a new Subject resource which contains attributes identifying the subject and the associated Event Stream "id".

The following non-normative example demonstrates adding a subject identified by "example.user@example.com" to an "EventStream" identified by "id" with value "767aad7853d240debc8e3c962051c1c0".

```
POST /Subjects/ HTTP/1.1
Host: transmitter.example.com
Authorization: Bearer eyJ0b2tlbiI6ImV4YW1wbGUifQo=
{
  "schemas":["urn:ietf:params:scim:schemas:event:2.0:Subject"],
  "email": "example.user@example.com",
  "streamIds": "767aad7853d240debc8e3c962051c1c0",
  "schemas":["urn:ietf:params:scim:schemas:event:2.0:Subject"]
}
```

Figure 27: Adding a Subject to a Stream Using POST

In response the server indicates the record is created and returns a permanent URI of the entry. This URI can later be used to remove the subject from the identified EventStream.

```
HTTP/1.1 201 Created
Content-Type: application/scim+json
Location:
  https://example.com/v2/Subjects/e3c962051c1c0
{
  "schemas":["urn:ietf:params:scim:schemas:event:2.0:Subject"],
  "id": "e3c962051c1c0",
  "email": "example.user@example.com",
  "streamIds": "767aad7853d240debc8e3c962051c1c0",
  "meta":{
    ...SCIM meta data...
  }
}
```

Figure 28: Response to Adding a Subject to a Stream Using POST

Should the record be a duplicate Subject, the Control Plane implementation MAY choose to return the original resource registration and location with HTTP Status 200 (OK).

#### 4.4.2. Querying for Subject in Event Streams

To query Subjects, SCIM filters can be used to return matching resources as per Section 3.4.2 [RFC7643]

Assuming the "id" of the EventStream is known, a query can be made against that identifier and the subjects identifier - in this case, an email address.

```
GET /Subjects?filter=(streamId eq "e3c962051c1c0" and
  email eq "example.user@example.com")
```

Figure 29: Querying if a Subject is in an EventStream

If a match to the request in Figure 29 is made, the following is a non-normative example response showing the matched Subject resource.

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

{
  "schemas":["urn:ietf:params:scim:api:messages:2.0:ListResponse"],
  "totalResults":1,
  "Resources":[
    {
      "id":"e3c962051c1c0",
      "email": "example.user@example.com"
      "streamIds": "e3c962051c1c0",
      "schemas":["urn:ietf:params:scim:schemas:event:2.0:Subject"],
      ...additional meta data...
    }
  ]
}
```

Figure 30: Response to Query

To see if an email address is present in multiple EventStreams, the following query MAY be used.

```
GET /Subjects?filter=(email eq "example.user@example.com")
```

Figure 31: Querying All Event Streams for Subject

Assuming only one match is found, than a response similar to Figure 30 is returned. If not matches occur, a response is returned with "totalResults" equal to 0. If more than one match is returned, the additional matches are returned in the "Resources" array.

#### 4.4.3. Removing a Subject from an Event Stream

Assuming the "id" of the Subject resource is known, HTTP DELETE MAY be used. If it is not known, the "id" MAY be queried as per Section 4.4.2.

To remove a Subject resource perform an HTTP DELETE using the resource's URI (see Section 3.6 [RFC7644]).

```
DELETE /Subjects/e3c962051c1c0
```

Figure 32: Removing a Subject from an Event Stream

### 5. Event Stream Verification

In the verify process, the Event Receiver organization initiates a request to the Event Transmitter to verify the Stream is working correctly. This can be used to both test for configuration errors (e.g. incorrect keys for signing and/or encryption, endpoints) and to verify operational state by using a Verify Event as an occasional 'ping' test.

To initiate a Verify Event, the Event Receiver organization using the Control Plane to set a nonce value for the Stream Configuration attribute "verifyConfirm". Once set, the Event Transmitter SHALL issue a Verify SET the includes the client specified nonce value.

In the following non-normative example, the client requests a Verify Event by setting the attribute "verifyNonce" as part of the Event Stream configuration.

```
PATCH /EventStreams/767aad7853d240debc8e3c962051c1c0
Host: example.com
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Bearer h480djs93hd8

{
  "schemas":
    [ "urn:ietf:params:scim:api:messages:2.0:PatchOp" ],
  "Operations": [ {
    "op": "replace",
    "path": "verifyNonce",
    "value": "VGhpcyBpcyBhbi"
  } ]
}
```

Figure 33: Requesting a Verify using PATCH

Upon the changing of the Event Stream configuration attribute "verifyNonce", the Event Transmitter sends a Verify Event SET to the Event Receiver using the registered "methodUri" mechanism.

The Verify SET contains the following attributes:

events

Set with an event attribute of "urn:ietf:params:secevent:verification" and contains the sub-attribute "nonce" which contains the value of "verifyNonce" .

iss

Set to the URI defined in the Event Stream configuration.

aud

MUST be set to a value that matches the EventStream "aud" value agreed to.

If the Event Stream is configured to encrypt SETs for the Event Receiver, then the SET MUST be encrypted with the provided key. Successful parsing of the message confirms that provides confirmation of correct configuration and possession of keys.

The following is a non-normative JSON representation of a Verify Event issued to an Event Receiver. Included in the SET is an example nonce value "VGhpcyBpcyBhbi".

```
{
  "jti": "123456",
  "iss": "https://transmitter.example.com",
  "aud": "receiver.example.com",
  "iat": "1493856000",
  "events": [
    "urn:ietf:params:secevent:verification" : {
      "nonce": "VGhpcyBpcyBhbi",
    },
  ],
}
```

Figure 34: Example Verification SET

The above SET is encoded as a JWT and transmitted to the Event Receiver using the configured delivery method.

Upon receiving a verify SET, the Event Receiver SHALL parse the SET and verify its claims. In particular, the Event Receiver SHALL confirm that the values for "nonce" match the value assigned to "verifyNonce" in the Event Stream Configuration via the Control

Plane. If the values do not match, administrative action should be taken to address the mis-configuration. Similarly if the SET is not received or is unparseable, the Event Receiver organization can check Event Stream configuration and check for errors by reviewing the Stream configuration attributes "status" and "txErr".

## 6. Privacy Considerations

See Section 7.5 [RFC7644] for protocol specific privacy considerations.

The Privacy Considerations of SET Token Specification [I-D.ietf-secevent-token] and the SET Token Delivery specification [I-D.ietf-secevent-delivery] SHALL apply.

### 6.1. Subject Management

The exact set of subject entities upon which SETs can be issued SHOULD NOT be made available to any single party. This is because a subject's relationship with an Event Transmitter MAY change over time and may not be known to the Event Receiver. A design consideration is that an Event Receiver MUST already know personal identifiers before asking an Event Transmitter if there is an existing relationship by asking if that personal identifier is part of a stream. Accordingly the "subjects" attribute of an Event Stream can not normally be returned. Instead, a Control Plane provider MAY confirm a subject is part of a stream. See Section 4.1 and Section 4.2.1.

When receiving a request from a Control Plane client to add a subject, the provider SHOULD consider if the subject is appropriate to the purpose of the Event Stream being managed. For example, for an OpenID Connect Provider, was consent obtained to share security data with the Relying Party. Such authorization may have been previously authorized by a user via the OpenID consent process. Having obtained consent, the Control Plane provider SHOULD consider if the SET Events being requested to be streamed are appropriate.

## 7. Security Considerations

This specification depends on the Security Considerations of [RFC7644].

### 7.1. Multi-Party Access to Streams

Implementations SHOULD support access roles which enable different types of access to Event Streams via the Control Plane service. A minimal suggested set of roles includes:

**Monitor** For clients to retrieve Event Stream configuration and obtain current status. Access is limited to read-only operations.

**Control** Adds the ability to modify the "status" attribute to control the operational state of the Event Stream in addition to the rights granted by "Monitor".

**Manage** Provides the ability to list, create and manage Event Streams including updating and verifying subjects.

Typically these roles are rights or scopes associated with the security credential presented in the HTTP Authorization header of requests (see Section 7 [RFC7644]). The method by which these roles are implemented is out of scope of this specification.

## 8. IANA Considerations

### 8.1. Registration of Verify Event URI

IANA is requested to add an entry to the 'IETF URN Sub-namespace for Registered Protocol Parameter Identifiers' registry and create a sub-namespace for the Registered Parameter Identifier as per [RFC3553]: "urn:ietf:params:secevent:verification".

The identifier is used to indicate a Verify Event as defined in Section 5 for use in the "events" attribute defined in [I-D.ietf-secevent-token].

### 8.2. SCIM Schema Registration

As per the "SCIM Schema URIs for Data Resources" registry established by Section 10.3 [RFC7643], the following defines and registers the following SCIM URIs and Resource Types for Feeds and Event Streams.

Schema URI	Name	ResourceType	Reference
urn:ietf:params:scim:schemas:event:2.0:EventStream	SET Event Stream	EventStream	Section 2.1

Attributes for SET Event Streams are defined in Section 2.1

SCIM Schema and ResourceType definitions are defined in Appendix A

## 9. References

### 9.1. Normative References

- [I-D.ietf-secevent-delivery]  
Hunt, P., Scurtescu, M., Ansari, M., Nadalin, A., and A. Backman, "SET Token Delivery Using HTTP", draft-ietf-secevent-delivery-00 (work in progress), July 2017.
- [I-D.ietf-secevent-token]  
Hunt, P., Denniss, W., Ansari, M., and M. Jones, "Security Event Token (SET)", draft-ietf-secevent-token-02 (work in progress), June 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>>.
- [RFC5988] Nottingham, M., "Web Linking", RFC 5988, DOI 10.17487/RFC5988, October 2010, <<https://www.rfc-editor.org/info/rfc5988>>.
- [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>>.

### 9.2. Informative References

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

- [RFC3553] Mealling, M., Masinter, L., Hardie, T., and G. Klyne, "An IETF URN Sub-namespace for Registered Protocol Parameters", BCP 73, RFC 3553, DOI 10.17487/RFC3553, June 2003, <<https://www.rfc-editor.org/info/rfc3553>>.
- [RFC3966] Schulzrinne, H., "The tel URI for Telephone Numbers", RFC 3966, DOI 10.17487/RFC3966, December 2004, <<https://www.rfc-editor.org/info/rfc3966>>.
- [RFC5321] Klensin, J., "Simple Mail Transfer Protocol", RFC 5321, DOI 10.17487/RFC5321, October 2008, <<https://www.rfc-editor.org/info/rfc5321>>.
- [RFC6902] Bryan, P., Ed. and M. Nottingham, Ed., "JavaScript Object Notation (JSON) Patch", RFC 6902, DOI 10.17487/RFC6902, April 2013, <<https://www.rfc-editor.org/info/rfc6902>>.
- [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>>.
- [RFC7643] Hunt, P., Ed., Grizzle, K., Wahlstroem, E., and C. Mortimore, "System for Cross-domain Identity Management: Core Schema", RFC 7643, DOI 10.17487/RFC7643, September 2015, <<https://www.rfc-editor.org/info/rfc7643>>.
- [RFC7644] Hunt, P., Ed., Grizzle, K., Ansari, M., Wahlstroem, E., and C. Mortimore, "System for Cross-domain Identity Management: Protocol", RFC 7644, DOI 10.17487/RFC7644, September 2015, <<https://www.rfc-editor.org/info/rfc7644>>.

#### Appendix A. Event Stream Resource Type and Schema Definitions

The "EventStream" resource type definition is defined as follows:



```
{
  "schemas": ["urn:ietf:params:scim:schemas:core:2.0:ResourceType"],
  "id": "EventStream",
  "name": "EventStream",
  "endpoint": "/EventStreams",
  "description": "Endpoint and event configuration and status for SEC EVENT streams.",
  "schema": "urn:ietf:params:scim:schemas:event:2.0:EventStream",
  "schemaExtensions": []
}
```

Figure 35: SCIM EventStream Resource Type Definition

The resource type above is discoverable in the `/ResourceTypes` endpoint of a SCIM service provider and informs SCIM clients about the endpoint location of EventStream resources and the SCIM schema used to define the resource. The corresponding schema for the EventStream resource MAY be retrieved from the SCIM `/Schemas` endpoint (see Section 3.2 [RFC7644]).

The attributes for the EventStream resource type are defined in Section 2.1.

```
{
  "id" : "urn:ietf:params:scim:schemas:event:2.0:EventStream",
  "name" : "EventStream",
  "description" : "Event Stream Configuration",
  "attributes" : [
    {
      "name" : "eventUris",
      "type" : "string",
      "multiValued" : true,
      "description" : "An array of String value containing a logical unique URI for Events that may be issued in the Stream",
      "required" : false,
      "caseExact" : false,
      "mutability" : "readOnly",
      "returned" : "default",
      "uniqueness" : "none"
    },
    {
      "name" : "eventUris_req",
      "type" : "string",
      "multiValued" : true,
      "description" : "An array of String value containing a logical unique URI for Events that an Event Receiver is requesting.",
      "required" : true,
      "caseExact" : false,
      "mutability" : "readWrite",
    }
  ]
}
```

```
    "returned" : "default",
    "uniqueness" : "none"
  },
  {
    "name" : "eventUri_avail",
    "type" : "string",
    "multiValued" : true,
    "description" : "An array of String value containing a logical
unique URI for Events that are supported by the Transmitter",
    "required" : false,
    "caseExact" : false,
    "mutability" : "readOnly",
    "returned" : "default",
    "uniqueness" : "none"
  },
  {
    "name" : "methodUri",
    "type" : "string",
    "multiValued" : false,
    "description" : "A String value containing the URI for the
method used to deliver SET events. The method used indicates
the required configuration parameters for an
operational Event Stream configuration.",
    "required" : true,
    "caseExact" : false,
    "mutability" : "readWrite",
    "returned" : "default",
    "uniqueness" : "none"
  },
  {
    "name" : "deliveryUri",
    "type" : "string",
    "multiValued" : false,
    "description" : "A String value containing the URI for a
feed endpoint used to pick up or deliver SET events based on
a configured method.",
    "required" : true,
    "caseExact" : false,
    "mutability" : "readWrite",
    "returned" : "default",
    "uniqueness" : "none"
  },
  {
    "name" : "iss",
    "type" : "string",
    "multiValued" : false,
    "description" : "The URI for the publisher of the SETs that will
be issued for the Event Stream.",
```

```
"required" : true,
"caseExact" : false,
"mutability" : "readWrite",
"returned" : "default",
"uniqueness" : "none"
},
{
  "name" : "aud",
  "type" : "string",
  "multiValued" : true,
  "description" : "An OPTIONAL Array of JSON String values which
are URIs representing the audience(s) of the Event Stream.
Values SHALL be the value of SET \"aud\" claim sent to the Event
Receiver.",
  "required" : true,
  "caseExact" : false,
  "mutability" : "readWrite",
  "returned" : "default",
  "uniqueness" : "none"
},
{
  "name" : "iss_jwksUri",
  "type" : "string",
  "multiValued" : false,
  "description" : "An OPTIONAL
String that contains the URL of the SET issuers public JSON
Web Key Set [RFC7517]. This contains the signing key(s) the
Event Receiver uses to validate SET signatures from the Event
Transmitter that will be used by the Event Receiver to verify
the authenticity of issued SETs.",
  "required" : false,
  "caseExact" : false,
  "mutability" : "readWrite",
  "returned" : "default",
  "uniqueness" : "none"
},
{
  "name" : "aud_jwksUri",
  "type" : "string",
  "multiValued" : false,
  "description" : "An OPTIONAL JSON Web Key Set [RFC7517] that
contains the Event Receiver's encryption keys that MAY be used
by the Event Transmitter to encrypt SET tokens for the specified
Event Receiver.",
  "required" : false,
  "caseExact" : false,
  "mutability" : "readWrite",
  "returned" : "default",
```

```
    "uniqueness" : "none"
  },
  {
    "name" : "status",
    "type" : "string",
    "multiValued" : false,
    "description" : "An OPTIONAL JSON String keyword that indicates
the current state of an Event Stream. More information on the E
vent Stream state can be found in Section 2.3.",
    "required" : false,
    "caseExact" : false,
    "mutability" : "readWrite",
    "returned" : "default",
    "uniqueness" : "none",
    "canonicalValues" : [
      "on",
      "off",
      "verify",
      "paused",
      "fail"
    ]
  },
  {
    "name" : "maxRetries",
    "type" : "integer",
    "multiValued" : false,
    "description" : "An OPTIONAL JSON number indicating the maximum
number of attempts to deliver a SET. A value of '0' indicates
there is no maximum. Upon reaching the maximum, the Event Stream
'status' attribute is set to 'failed'.",
    "required" : false,
    "mutability" : "readWrite",
    "returned" : "default",
    "uniqueness" : "none"
  },
  {
    "name" : "maxDeliveryTime",
    "type" : "integer",
    "multiValued" : false,
    "description" : "An OPTIONAL number indicating the maximum
amount of time in seconds a SET MAY take for successful delivery
per request or cumulatively across multiple retries. Upon
reaching the maximum, the Event Stream 'status' is set to
'failed'. If undefined, there is no maximum time.",
    "required" : false,
    "mutability" : "readWrite",
    "returned" : "default",
    "uniqueness" : "none"
  }
]
```

```
    },
    {
      "name" : "minDeliveryInterval",
      "type" : "integer",
      "multiValued" : false,
      "description" : "An OPTIONAL JSON integer that represents the
        minimum interval in seconds between deliveries. A value of '0'
        indicates delivery should happen immediately. When delivery is
        a polling method (e.g. HTTP GET), it is the expected time
        between Event Receiver attempts. When in push mode (e.g.
        HTTP POST), it is the interval the server will wait before
        sending a new event or events.",
      "required" : false,
      "mutability" : "readWrite",
      "returned" : "default",
      "uniqueness" : "none"
    },
    {
      "name" : "txErr",
      "type" : "string",
      "multiValued" : false,
      "description" : "An OPTIONAL JSON String keyword value. When
        the Event Stream has 'status' set to 'fail', a keyword condition
        is set.",
      "required" : false,
      "caseExact" : false,
      "mutability" : "readWrite",
      "returned" : "default",
      "uniqueness" : "none",
      "canonicalValues" : [
        "connection",
        "tls",
        "dnsname",
        "receiver",
        "other"
      ]
    },
    {
      "name" : "txErrDesc",
      "type" : "string",
      "multiValued" : false,
      "description" : "An OPTIONAL String value that is usually human
        readable that provides further diagnostic detail by the
        indicated 'txErr' error code.",
      "required" : false,
      "caseExact" : false,
      "mutability" : "readWrite",
      "returned" : "default",
```

```
    "uniqueness" : "none"
  },
  {
    "name" : "verifyNonce",
    "type" : "string",
    "multiValued" : false,
    "description" : "An OPTIONAL String value that when changed
or set by a Control Plane client will cause the Event Transmitter
to issue a single Verify Event based on the value provided.",
    "required" : false,
    "caseExact" : false,
    "mutability" : "writeOnly",
    "returned" : "never",
    "uniqueness" : "none"
  },
  {
    "name" : "subjects",
    "type" : "complex",
    "multiValued" : true,
    "description" : "An optional list of subjects that are part of
the Stream.",
    "required" : false,
    "subAttributes" : [
      {
        "name" : "value",
        "type" : "string",
        "multiValued" : false,
        "description" : "Identifier of the member of this Group.
The contents of this parameter are determined by the value
of the sub-attribute 'type'.",
        "required" : false,
        "caseExact" : false,
        "mutability" : "immutable",
        "returned" : "default",
        "uniqueness" : "none"
      },
      {
        "name" : "type",
        "type" : "string",
        "multiValued" : false,
        "description" : "A label indicating the type of resource,
e.g., OIDC Connect Subject, SAML Subject, Email address,
Telephone Number, SCIM User or SCIM Group, or the URI
of some other network addressable subject.",
        "required" : false,
        "caseExact" : false,
        "canonicalValues" : [
          "User",
```

```
        "Group",
        "OIDC",
        "SAML",
        "EMIL",
        "PHONE",
        "URI"
      ],
      "mutability" : "immutable",
      "returned" : "default",
      "uniqueness" : "none"
    }
  ],
  "mutability" : "readWrite",
  "returned" : "request"
},
"meta" : {
  "resourceType" : "Schema",
  "location" :
    "/v2/Schemas/urn:ietf:params:scim:schemas:core:2.0:Group"
},
},
```

## 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.

This draft is a re-work of material from Sections 2 and 4 of draft-hunt-secevent-distribution-01. The editor would like to thank Marius Scurtescu, Tony Nadalin, and Morteza Ansari for their contributions in previous drafts.

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

## Appendix C. Change Log

Draft 00 - PH - First Draft based on control plane portions of draft-hunt-idevent-distribution

## Author's Address

Phil Hunt (editor)  
Oracle Corporation

Email: phil.hunt@yahoo.com



Network Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: September 5, 2018

P. Hunt, Ed.  
Oracle  
M. Scurtescu  
Google  
M. Ansari  
Cisco  
A. Nadalin  
Microsoft  
A. Backman  
Amazon  
March 4, 2018

SET Token Delivery Using HTTP  
draft-ietf-secevent-delivery-02

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 push to the receiver, or 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 September 5, 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

1. Introduction and Overview . . . . .	2
1.1. Notational Conventions . . . . .	3
1.2. Definitions . . . . .	3
2. SET Event Stream Protocol . . . . .	4
2.1. Event Delivery Process . . . . .	4
2.2. Push Delivery using HTTP . . . . .	6
2.3. Polling Delivery using HTTP . . . . .	8
2.3.1. Polling HTTP Request Attributes . . . . .	8
2.3.2. Polling HTTP Response Attributes . . . . .	10
2.3.3. Poll Request . . . . .	10
2.3.4. Poll Response . . . . .	14
2.4. Error Response Handling . . . . .	16
3. Authentication and Authorization . . . . .	17
3.1. Use of Tokens as Authorizations . . . . .	18
4. Security Considerations . . . . .	18
4.1. Authentication Using Signed SETs . . . . .	18
4.2. HTTP Considerations . . . . .	19
4.3. TLS Support Considerations . . . . .	19
4.4. Authorization Token Considerations . . . . .	19
4.4.1. Bearer Token Considerations . . . . .	19
5. Privacy Considerations . . . . .	20
6. IANA Considerations . . . . .	20
7. References . . . . .	20
7.1. Normative References . . . . .	20
7.2. Informative References . . . . .	22
Appendix A. Other Streaming Specifications . . . . .	23
Appendix B. Acknowledgments . . . . .	24
Appendix C. Change Log . . . . .	24
Authors' Addresses . . . . .	26

## 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 push SETs via HTTP POST and another method to poll for SETs using HTTP POST.

This specification defines two methods 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 per 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 uses an HTTP/1.1 POST to the Event Receiver endpoint to deliver the SET;

- 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.

Delivery of SETs MAY be delivered using one of two modes:

#### PUSH

In which SETs are delivered one at a time using HTTP POST requests by an Event Transmitter to an Event Receiver. The HTTP request body is a JSON Web Token [RFC7519] with a "Content-Type" header of "application/secevent+jwt" as defined in Section 2.2 and 6.2 of [I-D.ietf-secevent-token]. Upon receipt, the Event Receiver acknowledges receipt with a response with HTTP Status 202, as described below in Section 2.2.

**POLLING** Where 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. In POLLING mode, all requests and responses are JSON documents and use a "Content-Type" of "application/json" as described in Section 2.3.

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. Push Delivery using HTTP

This method allows an Event Transmitter to use HTTP POST (Section 4.3.3 [RFC7231]) to deliver SETs to a previously registered web callback URI supplied by the Event Receiver as part of an Event Stream configuration process (not defined by this document).

The SET to be delivered MAY be signed and/or encrypted as defined in [I-D.ietf-secevent-token].

The Event Stream configuration defines a URI of an Event Receiver provided endpoint which accepts HTTP POST requests (e.g. "https://rp.example.com/Events").

The HTTP Content-Type (see Section 3.1.1.5 [RFC7231]) for the HTTP POST is "application/secevent+jwt" and SHALL consist of a single SET (see [I-D.ietf-secevent-token]). As per Section 5.3.2 [RFC7231], the expected media type ("Accept" header) response is "application/json".

To deliver an Event, the Event Transmitter generates an event delivery message and uses HTTP POST to the configured endpoint with the appropriate "Accept" and "Content-Type" headers.

POST /Events HTTP/1.1

Host: notify.examplerp.com

Accept: application/json

Authorization: Bearer h480djs93hd8

Content-Type: application/secevent+jwt

eyJhbGciOiJub251In0

.

eyJwdWJsaXNoZXJvcmkioiJodHRwc2ovL3NjaW0uZXhhbXBsZS5jb20iLCJmZWV  
kVXJpcyI6WyJodHRwc2ovL2podWIuZXhhbXBsZS5jb20vRmVlZHMvOThkNTI0Nj  
FmYTViYmM4Nzk1OTNiNzc1NCIsImh0dHBzOi8vamhlYi5leGFtcGxlLmNvbS9GZ  
WVkc291ZDc2MDQ1MTZiMWQwODY0MWQ3Njc2ZWU3Il0sInJlc291cmNlVXJpcyI6  
WyJodHRwc2ovL3NjaW0uZXhhbXBsZS5jb20vVXNlcnMvNDRmNjE0MmRmOTZiZDZ  
hYjYxZTclMjFkOSJdLCJldmVudFR5cGVzIjpbIkNSRUFURSJdLCJhdHRyaWJldG  
VzIjpbImlkIiwibmFtZSI6InVzZXJOYW1lIiwicGFzc3dvcmQiLCJlbWFpbiBMiX  
SwidmFsdWVzIjpbImVtYWlscyI6W3sidHlwZSI6IndvcmsiLCJ2YWx1ZSI6Impk  
b2VAZXhhbXBsZS5jb20ifV0sInBhc3N3b3JkIjoibm90NHUybm8iLCJlc2VyTmF  
tZSI6Impkb2UiLCJpZCI6IjQ0ZjYxNDJkZjk2YmQ2YWI2MWU3NTIxZDkiLCJuYW  
1lIjpbImdpdmVuTmFtZSI6IkpvG4iLCJmYW1pbHl0YW1lIjoiriRG91In19fQ

.

Figure 1: Example HTTP POST Request

Upon receipt of the request, the Event Receiver SHALL validate the JWT structure of the SET as defined in Section 7.2 [RFC7519]. The

Event Receiver SHALL also validate the SET information as described in Section 2 [I-D.ietf-secevent-token].

If the SET is determined to be valid, the Event Receiver SHALL "acknowledge" successful submission by responding with HTTP Status 202 as "Accepted" (see Section 6.3.3 [RFC7231]).

In order to maintain compatibility with other methods of transmission, the Event Receiver SHOULD NOT include an HTTP response body representation of the submitted SET or what the SET's pending status is when acknowledging success. In the case of an error (e.g. HTTP Status 400), the purpose of the HTTP response body is to indicate any SET parsing, validation, or cryptographic errors.

The following is a non-normative example of a successful receipt of a SET.

```
HTTP/1.1 202 Accepted
```

Figure 2: Example Successful Delivery Response

Note that the purpose of the "acknowledgement" response is to let the Event Transmitter know that a SET has been delivered and the information no longer needs to be retained by the Event Transmitter. Before acknowledgement, Event Receivers SHOULD ensure they have validated received SETs and retained them in a manner appropriate to information retention requirements appropriate to the SET event types signaled. The level and method of retention of SETs by Event Receivers is out-of-scope of this specification.

In the Event of a general HTTP error condition, the Event Receiver MAY respond with an appropriate HTTP Status code as defined in Section 6 [RFC7231].

When the Event Receiver detects an error parsing or validating a received SET (as defined by [I-D.ietf-secevent-token]), the Event Receiver SHALL indicate an HTTP Status 400 error with an error code as described in Section 2.4.

The following is an example non-normative error response.

```
HTTP/1.1 400 Bad Request
Content-Type: application/json

{
  "err": "dup",
  "description": "SET already received. Ignored."
}
```

Figure 3: Example HTTP Status 400 Response

### 2.3. 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.

#### 2.3.1. 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 2.3.3 and Section 2.3.4

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 2.4.

### 2.3.2. 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".

### 2.3.3. 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.

#### 2.3.3.1. Poll Only Request

In the case where no SETs were received in a previous poll (see Figure 10), 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://notify.exampleidp.com/Events".

```
POST /Events HTTP/1.1
```

```
Host: notify.exampleidp.com
Authorization: Bearer h480djs93hd8
Accept: application/json
```

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

Figure 4: 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 5: Example Default Poll Request

#### 2.3.3.2. 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 9).

```
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 6: Example Acknowledge Only request

#### 2.3.3.3. 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 9.

```
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 7: 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.

#### 2.3.3.4. 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 9.

```
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 8: Example Poll Acknowledgement With Error

#### 2.3.4. 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 2.3.2 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 2.3.3. This example shows two SETs are returned.

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

```
{
  "sets": {
    "4d3559ec67504aaba65d40b0363faad8":
      "eyJhbGciOiJub25lIn0.eyJqdGkiOiI0ZDMlNTllYzY3NTA0YWFiYTl1ZDQwYjAzNjNmYWFKOCIsImVhdCI6MTQ1ODQ5NjNjQWNCwiaXNzIjoiaHR0cHM6Ly9yZ2ltLmV4YW1wbGUuY29tIiwiaXYkIjpjbImh0dHBzOi8vc2NpbS5leGFtcGxlLmNvbS9GZWVkcy85OGQ1MjQ2MWZhNWJiYzg3OTU5M2I3NzU0IiwiaHR0cHM6Ly9yZ2ltLmV4YW1wbGUuY29tL0ZlZWZrZlZVKnZyYwNDUxNmIxZDA4NjQxZDc2NzZlZTciXSwiZXZlbnRzIjp7InVybjppZXRMOnBhcmtFtczpzY2ltOmV2ZW50cmNyZWFOZSI6eyJpY29yYyIjOiJodHRwc29vLnjaW0uZXhhbXBsZS5jb20vVXNlcniMvNDRMnE0MmRmOTZlZDZhYjYxZTclMjFfKOSISImF0dHJpYnV0ZXMiOlsiaWQiLCJuYWllIiwidXNlck5hbWUiLCJwYXNzd29yZCIsImVtYWIscyJdfX19." ,
    "3d0c3cf797584bd193bd0fb1bd4e7d30":
      "eyJhbGciOiJub25lIn0.eyJqdGkiOiIzZDBjM2NmNzk3NTg0YmQxOTNiZDBmYjFiZDRlN2QzMCI6MTQ1ODQ5NjNjYyNSwiaXNzIjoiaHR0cHM6Ly9yZ2ltLmV4YW1wbGUuY29tIiwiaXYkIjpjbImh0dHBzOi8vamhlYi5leGFtcGxlLmNvbS9GZWVkcy85OGQ1MjQ2MWZhNWJiYzg3OTU5M2I3NzU0IiwiaHR0cHM6Ly9qaHVilMvV4YW1wbGUuY29tL0ZlZWZrZlZVKnZyYwNDUxNmIxZDA4NjQxZDc2NzZlZTciXSwic3ViIjoiaHR0cHM6Ly9yZ2ltLmV4YW1wbGUuY29tLlVzZXJzLzQ0ZjYxNDJkZjk2YmQ2YWI2MWU3NTIxZDkiLCJldmVudHMiOmsidXJuOmlldGY6cGFyYW1zOnNjaW06ZXZlbnQ6cGFzc3dvcmRSZXNldCI6eyJpZCI6IjQ0ZjYxNDJkZjk2YmQ2YWI2MWU3NTIxZDkifSwiaHR0cHM6Ly9leGFtcGxlLmNvbS9yZ2ltL2V2ZW50L3Bhc3N3b3JkUmVzZXRFehQiOmsicmVzZXRBdHRlbXB0cyI6NX19fQ."
  }
}
```

Figure 9: Example Poll Response

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

The following is a non-normative example response to the request shown Section 2.3.3 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 10: Example No SETs Poll Response

Upon receiving the JSON document (e.g. as shown in Figure 9), 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 2.3.3.3 or Section 2.3.3.4.

## 2.4. Error Response Handling

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

Err Value	Description
json	Invalid JSON object.
jwtParse	Invalid or unparseable 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 an HTTP Status 400 response, the above JSON is the HTTP response body (see Figure 3). When included as part of a batch of SETs, the above JSON is included as part of the "setErrs" attribute as defined in Section 2.3.2 and Section 2.3.3.4

### 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

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).

SET Payload Authentication

In scenarios where SETs are signed and the delivery method is HTTP POST (see Section 2.2), Event Receivers 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:

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

#### Appendix C. Change Log

Draft 00 - PH - Based on draft-hunt-secevent.distribution with the following additions:



- o Removed Control Plane from specification
- o Added new HTTP Polling delivery method
- o Added general HTTP security considerations
- o Added authentication and authorization
- o Revised Verify Event to work with both types of delivery

Draft 01 - PH - Removed Verification section per feedback from IETF99.

Draft 02 - MS -

- o Minor editorial improvements
- o Removed Identity Provider / Relying Party Terminology
- o Changed boilerplate language according to RFC8174

This draft was based on draft-hunt-secevent.distribution revision history:

- o Draft 00 - PH - First Draft based on reduced version of draft-hunt-idevent-distribution
- o Draft 01 - PH -
  - \* Reworked terminology to match new WG Transmitter/Receiver terms
  - \* Reworked sections into Data Plane vs. Control Plane
  - \* Removed method transmission registry in order to simplify the specification
  - \* Made Create, Update operations optional for Control Plane (Read is MTI)
- o Draft 02 - PH
  - \* Added iss metadata for Event Stream
  - \* Changed to using JWKS\_uri for issuer and receiver.
  - \* Control Plane sections moved to draft-hunt-secevent-stream-mgmt

- \* Added support for delivering multiple events using HTTP POST polling

Authors' Addresses

Phil Hunt (editor)  
Oracle Corporation

Email: phil.hunt@yahoo.com

Marius Scurtescu  
Google

Email: mscurtescu@google.com

Morteza Ansari  
Cisco

Email: morteza.ansari@cisco.com

Anthony Nadalin  
Microsoft

Email: tonymad@microsoft.com

Annabelle Richard Backman  
Amazon

Email: richanna@amazon.com

Security Events Working Group  
Internet-Draft  
Intended status: Standards Track  
Expires: November 10, 2018

P. Hunt, Ed.  
Oracle  
M. Jones  
Microsoft  
W. Denniss  
Google  
M. Ansari  
Cisco  
May 9, 2018

Security Event Token (SET)  
draft-ietf-secevent-token-13

Abstract

This specification defines the Security Event Token (SET) data structure. A SET describes statements of fact from the perspective of an issuer about a subject. These statements of fact represent an event that occurred directly to or about a security subject, for example, a statement about the issuance or revocation of a token on behalf of a subject. This specification is intended to enable representing security- and identity-related events. A SET is a JSON Web Token (JWT), which can be optionally signed and/or encrypted. SETs can be distributed via protocols such as HTTP.

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 November 10, 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

1. Introduction and Overview . . . . .	3
1.1. Notational Conventions . . . . .	4
1.2. Definitions . . . . .	4
2. The Security Event Token (SET) . . . . .	5
2.1. Illustrative Examples . . . . .	6
2.1.1. SCIM Example . . . . .	6
2.1.2. Logout Example . . . . .	8
2.1.3. Consent Example . . . . .	8
2.1.4. RISC Example . . . . .	9
2.2. Core SET Claims . . . . .	10
2.3. Explicit Typing of SETs . . . . .	12
2.4. Security Event Token Construction . . . . .	13
3. Requirements for SET Profiles . . . . .	14
4. Preventing Confusion between SETs and other JWTs . . . . .	16
4.1. Distinguishing SETs from ID Tokens . . . . .	16
4.2. Distinguishing SETs from Access Tokens . . . . .	16
4.3. Distinguishing SETs from other kinds of JWTs . . . . .	17
5. Security Considerations . . . . .	18
5.1. Confidentiality and Integrity . . . . .	18
5.2. Delivery . . . . .	18
5.3. Sequencing . . . . .	18
5.4. Timing Issues . . . . .	19
5.5. Preventing Confusion . . . . .	19
6. Privacy Considerations . . . . .	19
7. IANA Considerations . . . . .	20
7.1. JSON Web Token Claims Registration . . . . .	20
7.1.1. Registry Contents . . . . .	20
7.2. Structured Syntax Suffix Registration . . . . .	21
7.2.1. Registry Contents . . . . .	21
7.3. Media Type Registration . . . . .	22
7.3.1. Registry Contents . . . . .	22
8. References . . . . .	22
8.1. Normative References . . . . .	22
8.2. Informative References . . . . .	24
Appendix A. Acknowledgments . . . . .	25
Appendix B. Change Log . . . . .	25

Authors' Addresses . . . . .	30
------------------------------	----

## 1. Introduction and Overview

This specification defines an extensible Security Event Token (SET) data structure, which can be exchanged using protocols such as HTTP. The specification builds on the JSON Web Token (JWT) format [RFC7519] in order to provide a self-contained token that can be optionally signed using JSON Web Signature (JWS) [RFC7515] and/or encrypted using JSON Web Encryption (JWE) [RFC7516].

This specification profiles the use of JWT for the purpose of issuing Security Event Tokens (SETs). This specification defines a base format used by profiling specifications to define actual events and their meanings. This specification uses non-normative example events to demonstrate how events can be constructed.

This specification is scoped to security- and identity-related events. While Security Event Tokens may be used for other purposes, the specification only considers security and privacy concerns relevant to identity and personal information.

Security events are not commands issued between parties. A SET describes statements of fact from the perspective of an issuer about a subject (e.g., a web resource, token, IP address, the issuer itself). These statements of fact represent a logical event that occurred directly to or about a security subject, for example, a statement about the issuance or revocation of a token on behalf of a subject. A security subject may be permanent (e.g., a user account) or temporary (e.g., an HTTP session) in nature. A state change could describe a direct change of entity state, an implicit change of state, or other higher-level security statements such as:

- o The creation, modification, removal of a resource.
- o The resetting or suspension of an account.
- o The revocation of a security token prior to its expiry.
- o The logout of a user session. Or,
- o An indication that a user has been given control of an email identifier that was previously controlled by another user.

While subject state changes are often triggered by a user agent or security subsystem, the issuance and transmission of an event may occur asynchronously and in a back channel to the action that caused the change that generated the security event. Subsequently, a SET

recipient, having received a SET, validates and interprets the received SET and takes its own independent actions, if any. For example, having been informed of a personal identifier being associated with a different security subject (e.g., an email address is being used by someone else), the SET recipient may choose to ensure that the new user is not granted access to resources associated with the previous user. Or, the SET recipient may not have any relationship with the subject, and no action is taken.

While SET recipients will often take actions upon receiving SETs, security events cannot be assumed to be commands or requests. The intent of this specification is to define a syntax for statements of fact that SET recipients may interpret for their own purposes.

### 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 document, all figures may contain spaces and extra line-wrapping for readability and space limitations. Similarly, some URIs contained within examples have been shortened for space and readability reasons.

### 1.2. Definitions

The following definitions are used with SETs:

#### Security Event Token (SET)

A SET is a JWT [RFC7519] conforming to this specification.

#### SET Issuer

A service provider that creates SETs to be sent to other service providers known as SET recipients.

#### SET Recipient

A SET recipient is an entity that receives SETs through some distribution method. A SET recipient is the same entity referred as a "recipient" in [RFC7519] or "receiver" in related specifications.

#### Subject

A SET describes an event or state change that has occurred to a subject. A subject might, for instance, be a principal (e.g., Section 4.1.2 of [RFC7519]), a web resource, an entity such as an IP address, or the issuer of the SET.

#### Event Identifier

A member name for an element of the JSON object that is the value of the "events" claim in a SET. This member name **MUST** be a URI.

#### Event Payload

A member value for an element of the JSON object that is the value of the "events" claim in a SET. This member value **MUST** be a JSON object.

#### Profiling Specification

A specification that profiles the SET data structure to define one or more specific event types and their associated claims and processing rules.

## 2. The Security Event Token (SET)

A SET is a JWT [RFC7519] data structure that represents one or more related aspects of a security event that occurred to a subject. The JWT Claims Set in a SET has the following structure:

- o The top-level claims in the JWT Claims Set are called the SET "envelope". Some of these claims are present in every SET; others will be specific to particular SET profiles or profile families. Claims in the envelope **SHOULD** be registered in the "JSON Web Token Claims" registry [IANA.JWT.Claims] or be Public Claims or Private Claims, as defined in [RFC7519].
- o Envelope claims that are profiled and defined in this specification are used to validate the SET and provide information about the event data included in the SET. The claim "events" contains the event identifiers and event-specific data expressed about the security subject. The envelope **MAY** include event-specific or profile-specific data. The "events" claim value **MUST** be a JSON object that contains at least one member.
- o Each member of the "events" JSON object is a name/value pair. The JSON member name is a URI string value, which is the event identifier, and the corresponding value is a JSON object known as the event "payload". The payload JSON object contains claims that pertain to that event identifier and need not be registered as JWT claims. These claims are defined by the profiling specification

that defines the event. An event with no payload claims SHALL be represented as the empty JSON object ("{}").

- o When multiple event identifiers are contained in a SET, they represent multiple aspects of the same state transition that occurred to the security subject. They are not intended to be used to aggregate distinct events about the same subject. Beyond this, the interpretation of SETs containing multiple event identifiers is out of scope for this specification; profiling specifications MAY define their own rules regarding their use of SETs containing multiple event identifiers, as described in Section 3. Possible uses of multiple values include, but are not limited to:
  - \* Values to provide classification information (e.g., threat type or level).
  - \* Additions to existing event representations.
  - \* Values used to link potential series of events.
  - \* Specific-purpose event URIs used between particular SET issuers and SET recipients.

## 2.1. Illustrative Examples

This section illustrates several possible uses of SETs through non-normative examples.

### 2.1.1. SCIM Example

The following example shows the JWT Claims Set for a hypothetical SCIM [RFC7644] password reset SET. Such a SET might be used by a receiver as a trigger to reset active user-agent sessions related to the identified user.



```
{
  "iss": "https://scim.example.com",
  "iat": 1458496025,
  "jti": "3d0c3cf797584bd193bd0fb1bd4e7d30",
  "aud": [
    "https://jhub.example.com/Feeds/98d52461fa5bbc879593b7754",
    "https://jhub.example.com/Feeds/5d7604516b1d08641d7676ee7"
  ],
  "sub": "https://scim.example.com/Users/44f6142df96bd6ab61e7521d9",
  "events": {
    "urn:ietf:params:scim:event:passwordReset":
      { "id": "44f6142df96bd6ab61e7521d9" },
    "https://example.com/scim/event/passwordResetExt":
      { "resetAttempts": 5 }
  }
}
```

Figure 1: Example SCIM Password Reset Event

The JWT Claims Set usage consists of:

- o The "events" claim specifying the hypothetical SCIM URN ("urn:ietf:params:scim:event:passwordReset") for a password reset, and a second value, "https://example.com/scim/event/passwordResetExt", that is used to provide additional event information such as the current count of resets.
- o The "iss" claim, denoting the SET issuer.
- o The "sub" claim, specifying the SCIM resource URI that was affected.
- o The "aud" claim, specifying the intended audiences for the event. (The syntax of the "aud" claim is defined in Section 4.1.3 of [RFC7519].)

The SET contains two event payloads:

- o The "id" claim represents SCIM's unique identifier for a subject.
- o The second payload identified by "https://example.com/scim/event/passwordResetExt") and the payload claim "resetAttempts" conveys the current count of reset attempts. In this example, while the count is a simple factual statement for the issuer, the meaning of the value (a count) is up to the receiver. As an example, such a value might be used by the receiver to infer increasing risk.

In this example, the SCIM event indicates that a password has been updated and the current password reset count is 5. Notice that the value for "resetAttempts" is in the event payload of an event used to convey this information.

#### 2.1.2. Logout Example

Here is another example JWT Claims Set for a security event token, this one for a Logout Token:

```
{
  "iss": "https://server.example.com",
  "sub": "248289761001",
  "aud": "s6BhdRkqt3",
  "iat": 1471566154,
  "jti": "bWJq",
  "sid": "08a5019c-17e1-4977-8f42-65a12843ea02",
  "events": {
    "http://schemas.openid.net/event/backchannel-logout": {}
  }
}
```

Figure 2: Example OpenID Back-Channel Logout Event

Note that the above SET has an empty JSON object and uses the JWT claims "sub" and "sid" to identify the subject that was logged out. At the time of this writing, this example corresponds to the logout token defined in the OpenID Connect Back-Channel Logout 1.0 [OpenID.BackChannel] specification.

#### 2.1.3. Consent Example

In the following example JWT Claims Set, a fictional medical service collects consent for medical actions and notifies other parties. The individual for whom consent is identified was originally authenticated via OpenID Connect. In this case, the issuer of the security event is an application rather than the OpenID provider:

```
{
  "iss": "https://my.med.example.org",
  "iat": 1458496025,
  "jti": "fb4e75b5411e4e19b6c0fe87950f7749",
  "aud": [
    "https://rp.example.com"
  ],
  "events": {
    "https://openid.net/heart/specs/consent.html": {
      "iss": "https://connect.example.com",
      "sub": "248289761001",
      "consentUri": [
        "https://terms.med.example.org/labdisclosure.html#Agree"
      ]
    }
  }
}
```

Figure 3: Example Consent Event

In the above example, the attribute "iss" contained within the payload for the event "https://openid.net/heart/specs/consent.html" refers to the issuer of the security subject ("sub") rather than the SET issuer "https://my.med.example.org". They are distinct from the top-level value of "iss", which always refers to the issuer of the event -- a medical consent service that is a relying party to the OpenID Provider.

#### 2.1.4. RISC Example

The following example JWT Claims Set is for an account disabled event. This example was taken from a working draft of the RISC events specification, where RISC is the OpenID RISC (Risk and Incident Sharing and Coordination) working group [RISC]. The example is subject to change.

```
{
  "iss": "https://idp.example.com/",
  "jti": "756E69717565206964656E746966696572",
  "iat": 1508184845,
  "aud": "636C69656E745F6964",
  "events": {
    "http://schemas.openid.net/secevent/risc/event-type/\
account-disabled": {
      "subject": {
        "subject_type": "iss-sub",
        "iss": "https://idp.example.com/",
        "sub": "7375626A656374"
      },
      "reason": "hijacking",
      "cause-time": 1508012752
    }
  }
}
```

Figure 4: Example RISC Event

Notice that parameters to the event are included in the event payload, in this case, the "reason" and "cause-time" values. The subject of the event is identified using the "subject" payload value, which itself is a JSON object.

## 2.2. Core SET Claims

The following claims from [RFC7519] are profiled for use in SETs:

### "iss" (Issuer) Claim

As defined by Section 4.1.1 of [RFC7519], this claim contains a string identifying the service provider publishing the SET (the issuer). In some cases, the issuer of the SET will not be the issuer associated with the security subject of the SET.

Therefore, implementers cannot assume that the issuers are the same unless the profiling specification specifies that they are for SETs conforming to that profile. This claim is REQUIRED.

### "iat" (Issued At) Claim

As defined by Section 4.1.6 of [RFC7519], this claim contains a value representing when the SET was issued. This claim is REQUIRED.

"jti" (JWT ID) Claim

As defined by Section 4.1.7 of [RFC7519], this claim contains a unique identifier for the SET. The identifier MUST be unique within a particular event feed and MAY be used by clients to track whether a particular SET has already been received. This claim is REQUIRED.

"aud" (Audience) Claim

As defined by Section 4.1.3 of [RFC7519], this claim contains one or more audience identifiers for the SET. This claim is RECOMMENDED.

"sub" (Subject) Claim

As defined by Section 4.1.2 of [RFC7519], this claim contains a StringOrURI value representing the principal that is the subject of the SET. This is usually the entity whose "state" was changed. For example:

- \* an IP Address was added to a black list;
- \* a URI representing a user resource that was modified; or,
- \* a token identifier (e.g. "jti") for a revoked token.

If used, the profiling specification MUST define the content and format semantics for the value. This claim is OPTIONAL, as the principal for any given profile may already be identified without the inclusion of a subject claim. Note that some SET profiles MAY choose to convey event subject information in the event payload (either using the "sub" member name or another name), particularly if the subject information is relative to issuer information that is also conveyed in the event payload, which may be the case for some identity SET profiles.

"exp" (Expiration Time) Claim

As defined by Section 4.1.4 of [RFC7519], this claim is the time after which the JWT MUST NOT be accepted for processing. In the context of a SET however, this notion does not typically apply, since a SET represents something that has already occurred and is historical in nature. Therefore, its use is NOT RECOMMENDED. (Also, see Section 4.1 for additional reasons not to use the "exp" claim in some SET use cases.)

The following new claims are defined by this specification:

#### "events" (Security Events) Claim

This claim contains a set of event statements that each provide information describing a single logical event that has occurred about a security subject (e.g., a state change to the subject). Multiple event identifiers with the same value MUST NOT be used. The "events" claim MUST NOT be used to express multiple independent logical events.

The value of the "events" claim is a JSON object whose members are name/value pairs whose names are URIs identifying the event statements being expressed. Event identifiers SHOULD be stable values (e.g., a permanent URL for an event specification). For each name present, the corresponding value MUST be a JSON object. The JSON object MAY be an empty object ("{}"), or it MAY be a JSON object containing data described by the profiling specification.

#### "txn" (Transaction Identifier) Claim

An OPTIONAL string value that represents a unique transaction identifier. In cases in which multiple related JWTs are issued, the transaction identifier claim can be used to correlate these related JWTs. Note that this claim can be used in JWTs that are SETs and also in JWTs using non-SET profiles.

#### "toe" (Time of Event) Claim

A value that represents the date and time at which the event occurred. This value is a NumericDate (see Section 2 of [RFC7519]). By omitting this claim, the issuer indicates that they are not sharing an event time with the recipient. (Note that in some use cases, the represented time might be approximate; statements about the accuracy of this field MAY be made by profiling specifications.) This claim is OPTIONAL.

### 2.3. Explicit Typing of SETs

This specification registers the "application/secevent+jwt" media type, which can be used to indicate that the content is a SET. SETs MAY include this media type in the "typ" header parameter of the JWT representing the SET to explicitly declare that the JWT is a SET. This MUST be included if the SET could be used in an application context in which it could be confused with other kinds of JWTs.

Per the definition of "typ" in Section 4.1.9 of [RFC7515], it is RECOMMENDED that the "application/" prefix be omitted. Therefore, the "typ" value used SHOULD be "secevent+jwt".

## 2.4. Security Event Token Construction

This section describes how to construct a SET.

The following is an example JWT Claims Set for a hypothetical SCIM SET (which has been formatted for readability):

```
{
  "iss": "https://scim.example.com",
  "iat": 1458496404,
  "jti": "4d3559ec67504aaba65d40b0363faad8",
  "aud": [
    "https://scim.example.com/Feeds/98d52461fa5bbc879593b7754",
    "https://scim.example.com/Feeds/5d7604516b1d08641d7676ee7"
  ],
  "events": {
    "urn:ietf:params:scim:event:create": {
      "ref":
        "https://scim.example.com/Users/44f6142df96bd6ab61e7521d9",
      "attributes": ["id", "name", "userName", "password", "emails"]
    }
  }
}
```

Figure 5: Example Event Claims

The JSON Claims Set is encoded per [RFC7519].

In this example, the SCIM SET claims are encoded in an unsecured JWT. The JOSE Header for this example is:

```
{"typ": "secevent+jwt", "alg": "none"}
```

Base64url encoding (see Section 2 of [RFC7515]) of the octets of the UTF-8 [RFC3629] representation of the JOSE Header yields:

```
eyJ0eXAiOiJzZW5ldmVudCtqd3QiLCJhbGciOiJub25lIn0
```

The above example JWT Claims Set is encoded as follows:

```
eyJqdGkiOiI0ZDMlNTllYzY3NTA0YWFhYTY1ZDQwYjAzNjNmYWVhOCIsImVudCI6MTQ1ODQ5NjQwN2NpbS5leGFtcGxlLmNvbS9GZWVkcj85OGQ1MjQ2MWZhNWJiYzgzOTU5M2I3NzU0IiwiaHR0cHM6Ly9zY2ltLmV4YW1wbGUuY29tL0ZlZWRzLzVkNzYwNDUxNmIxZDA4NjQxZDc2NzZlZTciXSwiZXZlbnRzIjp7InVybjppZXRmOnBhcmFtczpzY2ltOmV2ZW50OmNyZWF0ZSI6eyJyZWYiOiJodHRwczovL3NjaW0uZXhhbXBsZS5jb20vVXNlcnMvNDRmNjE0MmRmOTZiZDZhYjYxZTclMjFkOSIsImF0dHJpYnV0ZXMiOlsiaWQiLCJuYVllIiwidXNlc5hbWUlcjwYXNzd29yZCIsImVtYWlscyJdfX19
```

The encoded JWS signature is the empty string. Concatenating the parts yields this complete SET:

```
eyJ0eXAiOiJzZW5ldmVudCtqd3QiLCJhbGciOiJub251In0.eyJqdGkiOiI0ZDMlNTllYzY3NTA0YWFhYTY1ZDQwYjAzNjNmYWVhOCIsImVudCI6MTQ1ODQ5NjQwN2NpbS5leGFtcGxlLmNvbS9GZWVkcj85OGQ1MjQ2MWZhNWJiYzgzOTU5M2I3NzU0IiwiaHR0cHM6Ly9zY2ltLmV4YW1wbGUuY29tL0ZlZWRzLzVkNzYwNDUxNmIxZDA4NjQxZDc2NzZlZTciXSwiZXZlbnRzIjp7InVybjppZXRmOnBhcmFtczpzY2ltOmV2ZW50OmNyZWF0ZSI6eyJyZWYiOiJodHRwczovL3NjaW0uZXhhbXBsZS5jb20vVXNlcnMvNDRmNjE0MmRmOTZiZDZhYjYxZTclMjFkOSIsImF0dHJpYnV0ZXMiOlsiaWQiLCJuYVllIiwidXNlc5hbWUlcjwYXNzd29yZCIsImVtYWlscyJdfX19.
```

Figure 6: Example Unsecured Security Event Token

For the purpose of having a simpler example in Figure 6, an unsecured token is shown. When SETs are not signed or encrypted, other mechanisms such as TLS MUST be employed to provide integrity protection, confidentiality, and issuer authenticity, as needed by the application.

When validation (i.e., auditing), or additional transmission security is required, JWS signing and/or JWE encryption MAY be used. To create and/or validate a signed and/or encrypted SET, follow the instructions in Section 7 of [RFC7519].

### 3. Requirements for SET Profiles

Profiling specifications of this specification define actual SETs to be used in particular use cases. These profiling specifications define the syntax and semantics of SETs conforming to that SET profile and rules for validating those SETs. Profiling specifications SHOULD define syntax, semantics, subject identification, and validation.

#### Syntax

The syntax of the SETs defined, including:



#### Top-Level Claims

Claims and values placed at the JWT Claims Set. Examples are claims defined by the JWT specification (see [RFC7519]), the SET specification, and by the profiling specification.

#### Event Payload

The JSON data structure contents and format, containing event-specific information, if any (see Section 1.2).

#### Semantics

Defining the semantics of the SET contents for SETs utilizing the profile is equally important. Possibly most important is defining the procedures used to validate the SET issuer and to obtain the keys controlled by the issuer that were used for cryptographic operations used in the JWT representing the SET. For instance, some profiles may define an algorithm for retrieving the SET issuer's keys that uses the "iss" claim value as its input. Likewise, if the profile allows (or requires) that the JWT be unsecured, the means by which the integrity of the JWT is ensured MUST be specified.

#### Subject Identification

Profiling specifications MUST define how the event subject is identified in the SET, as well as how to differentiate between the event subject's issuer and the SET issuer, if applicable. It is NOT RECOMMENDED for profiling specifications to use the "sub" claim in cases in which the subject is not globally unique and has a different issuer from the SET itself.

#### Validation

Profiling specifications MUST clearly specify the steps that a recipient of a SET utilizing that profile MUST perform to validate that the SET is both syntactically and semantically valid.

Among the syntax and semantics of SETs that a profiling specification may define is whether the value of the "events" claim may contain multiple members, and what processing instructions are employed in the single- and multiple-valued cases for SETs conforming to that profile. Many valid choices are possible. For instance, some profiles might allow multiple event identifiers to be present and specify that any that are not understood by recipients be ignored, thus enabling extensibility. Other profiles might allow multiple event identifiers to be present but require that all be understood if the SET is to be accepted. Some profiles might require that only a single value be present. All such choices are within the scope of profiling specifications to define.

#### 4. Preventing Confusion between SETs and other JWTs

Because [RFC7519] states that "all claims that are not understood by implementations MUST be ignored", there is a consideration that a SET might be confused with another kind of JWT from the same issuer. Unless this confusion is prevented, this might enable an attacker who possesses a SET to use it in a context in which another kind of JWT is expected, or vice-versa. This section presents concrete techniques for preventing confusion between SETs and several other specific kinds of JWTs, as well as generic techniques for preventing possible confusion between SETs and other kinds of JWTs.

##### 4.1. Distinguishing SETs from ID Tokens

A SET might be confused with ID Token [OpenID.Core] if a SET is mistakenly or maliciously used in a context requiring an ID Token. If a SET could otherwise be interpreted as a valid ID Token (because it includes the required claims for an ID Token and valid issuer and audience claim values for an ID Token) then that SET profile MUST require that the "exp" claim not be present in the SET. Because "exp" is a required claim in ID Tokens, valid ID Token implementations will reject such a SET if presented as if it were an ID Token.

Excluding "exp" from SETs that could otherwise be confused with ID Tokens is actually defense in depth. In any OpenID Connect contexts in which an attacker could attempt to substitute a SET for an ID Token, the SET would actually already be rejected as an ID Token because it would not contain the correct "nonce" claim value for the ID Token to be accepted in contexts for which substitution is possible.

Note that the use of explicit typing, as described in Section 2.3, will not achieve disambiguation between ID Tokens and SETs, as the ID Token validation rules do not use the "typ" header parameter value.

##### 4.2. Distinguishing SETs from Access Tokens

OAuth 2.0 [RFC6749] defines access tokens as being opaque. Nonetheless, some implementations implement access tokens as JWTs. Because the structure of these JWTs is implementation-specific, ensuring that a SET cannot be confused with such an access token is therefore likewise, in general, implementation specific. Nonetheless, it is recommended that SET profiles employ the following strategies to prevent possible substitutions of SETs for access tokens in contexts in which that might be possible:

- o Prohibit use of the "exp" claim, as is done to prevent ID Token confusion.
- o Where possible, use a separate "aud" claim value to distinguish between the SET recipient and the protected resource that is the audience of an access token.
- o Modify access token validation systems to check for the presence of the "events" claim as a means to detect security event tokens. This is particularly useful if the same endpoint may receive both types of tokens.
- o Employ explicit typing, as described in Section 2.3, and modify access token validation systems to use the "typ" header parameter value.

#### 4.3. Distinguishing SETs from other kinds of JWTs

JWTs are now being used in application areas beyond the identity applications in which they first appeared. For instance, the "Session Initiation Protocol (SIP) Via Header Field Parameter to Indicate Received Realm" [RFC8055] and "Personal Assertion Token (PASSport)" [RFC8225] specifications both define JWT profiles that use mostly or completely different sets of claims than are used by ID Tokens. If it would otherwise be possible for an attacker to substitute a SET for one of these (or other) kinds of JWTs, then the SET profile must be defined in such a way that any substituted SET will result in its rejection when validated as the intended kind of JWT.

The most direct way to prevent confusion is to employ explicit typing, as described in Section 2.3, and modify applicable token validation systems to use the "typ" header parameter value. This approach can be employed for new systems but may not be applicable to existing systems.

Another way to ensure that a SET is not confused with another kind of JWT is to have the JWT validation logic reject JWTs containing an "events" claim unless the JWT is intended to be a SET. This approach can be employed for new systems but may not be applicable to existing systems. Validating that the JWT has an "events" claim will be effective in preventing attackers from passing other kinds of JWTs off as SETs.

For many use cases, the simplest way to prevent substitution is requiring that the SET not include claims that are required for the kind of JWT that might be the target of an attack. For example, for

[RFC8055], the "sip\_callid" claim could be omitted and for [RFC8225], the "orig" claim could be omitted.

In many contexts, simple measures such as these will accomplish the task, should confusion otherwise even be possible. Note that this topic is being explored in a more general fashion in JSON Web Token Best Current Practices [I-D.ietf-oauth-jwt-bcp]. The proposed best practices in that draft may also be applicable for particular SET profiles and use cases.

## 5. Security Considerations

### 5.1. Confidentiality and Integrity

SETs may contain sensitive information. Therefore, methods for distribution of events SHOULD require the use of a transport-layer security mechanism when distributing events. Parties MUST support TLS 1.2 [RFC5246] or a higher version and MAY support additional transport-layer mechanisms meeting its security requirements. When using TLS, the client MUST perform a TLS server certificate check, per [RFC6125]. Implementation security considerations for TLS can be found in "Recommendations for Secure Use of TLS and DTLS" [RFC7525].

Security events distributed through third parties or that carry personally identifiable information MUST be encrypted using JWE [RFC7516] or secured for confidentiality by other means.

Unless integrity of the JWT is ensured by other means, it MUST be signed using JWS [RFC7515] by an issuer that is trusted to do so for the use case so that the SET can be authenticated and validated by the SET recipient.

### 5.2. Delivery

This specification does not define a delivery mechanism for SETs. In addition to confidentiality and integrity (discussed above), implementers and profiling specifications must consider the consequences of delivery mechanisms that are not secure and/or not assured. For example, while a SET may be end-to-end secured using JWE encrypted SETs, without (mutual) TLS, there is no assurance that the correct endpoint received the SET and that it could be successfully processed.

### 5.3. Sequencing

This specification defines no means of ordering multiple SETs in a sequence. Depending on the type and nature of the events represented by SETs, order may or may not matter. For example, in provisioning,

event order is critical -- an object cannot be modified before it is created. In other SET types, such as a token revocation, the order of SETs for revoked tokens does not matter. If, however, the event conveys a logged in or logged out status for a user subject, then order becomes important.

Profiling specifications and implementers SHOULD take caution when using timestamps such as "iat" to define order. Distributed systems will have some amount of clock skew. Thus, time by itself will not guarantee order.

Specifications profiling SET SHOULD define a mechanism for detecting order or sequence of events when the order matters. For example, the "txn" claim could contain an ordered value (e.g., a counter) that the issuer includes, although just as for timestamps, ensuring such ordering can be difficult in distributed systems.

#### 5.4. Timing Issues

When SETs are delivered asynchronously and/or out-of-band with respect to the original action that incurred the security event, it is important to consider that a SET might be delivered to a SET recipient in advance of or behind the process that caused the event. For example, a user having been required to log out and then log back in again, may cause a "token revoked" SET to be issued, typically causing the receiver to reset all active sessions at the receiver that are related to that user. If revocation SET arrives at the same time as the user agent re-logs in, timing could cause problems by erroneously treating the new user session as logged out. Profiling specifications SHOULD be careful to consider both SET expression and timing issues. For example, it might be more appropriate to revoke a specific session or identity token rather than a general logout statement about a "user". Alternatively, profiling specifications could use timestamps that allow new sessions to be started immediately after a stated logout event time.

#### 5.5. Preventing Confusion

Also, see Section 4 above for both additional security considerations and normative text on preventing SETs from being confused with other kinds of JWTs.

#### 6. Privacy Considerations

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

SET issuers SHOULD attempt to specialize SETs so that their content is targeted to the specific business and protocol needs of the intended SET recipients.

When sharing personally identifiable information or information that is otherwise considered confidential to affected users, SET issuers and recipients should have the appropriate legal agreements and user consent and/or terms of service in place.

The propagation of subject identifiers can be perceived as personally identifiable information. Where possible, SET issuers and recipients SHOULD devise approaches that prevent propagation -- for example, the passing of a salted hash value that requires the SET recipient to know the subject.

In some cases, it may be possible for a SET recipient to correlate different events and thereby gain information about a subject that the SET issuer did not intend to share. For example, a SET recipient might be able to use "iat" values or highly precise "toe" values to determine that two otherwise un-relatable events actually relate to the same real-world event. The union of information from both events could allow a SET recipient to de-anonymize data or recognize that unrelated identifiers relate to the same individual. SET issuers SHOULD take steps to minimize the chance of event correlation, when such correlation would constitute a privacy violation. For instance, they could use approximate values for the "toe" claim or arbitrarily delay SET issuance, where such delay can be tolerated.

## 7. IANA Considerations

### 7.1. JSON Web Token Claims Registration

This specification registers the "events", "toe", and "txn" claims in the IANA "JSON Web Token Claims" registry [IANA.JWT.Claims] established by [RFC7519].

#### 7.1.1. Registry Contents

- o Claim Name: "events"
- o Claim Description: Security Events
- o Change Controller: IESG
- o Specification Document(s): Section 2.2 of [[ this specification ]]
  
- o Claim Name: "toe"
- o Claim Description: Time of Event
- o Change Controller: IESG
- o Specification Document(s): Section 2.2 of [[ this specification ]]

- o Claim Name: "txn"
- o Claim Description: Transaction Identifier
- o Change Controller: IESG
- o Specification Document(s): Section 2.2 of [[ this specification ]]

## 7.2. Structured Syntax Suffix Registration

This section registers the "+jwt" structured syntax suffix [RFC6838] in the "Structured Syntax Suffix" registry [IANA.StructuredSuffix] in the manner described in [RFC6838], which can be used to indicate that the media type is encoded as a JWT.

### 7.2.1. Registry Contents

- o Name: JSON Web Token (JWT)
- o +suffix: +jwt
- o References: Section 3 of [RFC7519]
- o Encoding considerations: binary; JWT values are encoded as a series of base64url-encoded values (some of which may be the empty string) separated by period ('.') characters.
- o Interoperability considerations: n/a
- o Fragment identifier considerations:  
The syntax and semantics of fragment identifiers specified for +jwt SHOULD be as specified for "application/jwt". (At publication of this document, there is no fragment identification syntax defined for "application/jwt".)

The syntax and semantics for fragment identifiers for a specific "xxx/yyy+jwt" SHOULD be processed as follows:

For cases defined in +jwt, where the fragment identifier resolves per the +jwt rules, then process as specified in +jwt.

For cases defined in +jwt, where the fragment identifier does not resolve per the +jwt rules, then process as specified in "xxx/yyy+jwt".

For cases not defined in +jwt, then process as specified in "xxx/yyy+jwt".

- o Security considerations: See Section 11 of [RFC7519].
- o Contact:  
Michael B. Jones, mbj@microsoft.com
- o Author/Change controller:  
Security Events Working Group.  
The IESG has change control over this registration.

### 7.3. Media Type Registration

#### 7.3.1. Registry Contents

This section registers the "application/secevent+jwt" media type [RFC2046] in the "Media Types" registry [IANA.MediaType] in the manner described in [RFC6838], which can be used to indicate that the content is a SET.

- o Type name: application
- o Subtype name: secevent+jwt
- o Required parameters: n/a
- o Optional parameters: n/a
- o Encoding considerations: binary; A SET is a JWT; JWT values are encoded as a series of base64url-encoded values (some of which may be the empty string) separated by period ('.') characters.
- o Security considerations: See Section 5 of [[ this specification ]]
- o Interoperability considerations: n/a
- o Published specification: Section 2.3 of [[ this specification ]]
- o Applications that use this media type: Applications that exchange SETs
- o Fragment identifier considerations: n/a
- o Additional information:
  - Magic number(s): n/a
  - File extension(s): n/a
  - Macintosh file type code(s): n/a
- o Person & email address to contact for further information: Michael B. Jones, mbj@microsoft.com
- o Intended usage: COMMON
- o Restrictions on usage: none
- o Author: Michael B. Jones, mbj@microsoft.com
- o Change controller: IESG
- o Provisional registration? No

### 8. References

#### 8.1. Normative References

- [IANA.JWT.Claims]
  - IANA, "JSON Web Token Claims",
  - <<http://www.iana.org/assignments/jwt>>.
- [IANA.MediaType]
  - IANA, "Media Types",
  - <<http://www.iana.org/assignments/media-types>>.



- [IANA.StructuredSuffix]  
IANA, "Structured Syntax Suffix",  
<[https://www.iana.org/assignments/  
media-type-structured-suffix/](https://www.iana.org/assignments/media-type-structured-suffix/)>.
- [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>>.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <<https://www.rfc-editor.org/info/rfc3629>>.
- [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>>.
- [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>>.
- [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>>.
- [RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <<https://www.rfc-editor.org/info/rfc7515>>.
- [RFC7516] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, DOI 10.17487/RFC7516, May 2015, <<https://www.rfc-editor.org/info/rfc7516>>.
- [RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <<https://www.rfc-editor.org/info/rfc7519>>.

- [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>>.

## 8.2. Informative References

- [I-D.ietf-oauth-jwt-bcp] Sheffer, Y., Hardt, D., and M. Jones, "JSON Web Token Best Current Practices", draft-ietf-oauth-jwt-bcp-03 (work in progress), May 2018.
- [OpenID.BackChannel] Jones, M. and J. Bradley, "OpenID Connect Back-Channel Logout 1.0", January 2017, <[http://openid.net/specs/openid-connect-backchannel-1\\_0.html](http://openid.net/specs/openid-connect-backchannel-1_0.html)>.
- [OpenID.Core] Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and C. Mortimore, "OpenID Connect Core 1.0", November 2014, <[http://openid.net/specs/openid-connect-core-1\\_0.html](http://openid.net/specs/openid-connect-core-1_0.html)>.
- [RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <<https://www.rfc-editor.org/info/rfc2046>>.
- [RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <<https://www.rfc-editor.org/info/rfc6838>>.
- [RFC7644] Hunt, P., Ed., Grizzle, K., Ansari, M., Wahlstroem, E., and C. Mortimore, "System for Cross-domain Identity Management: Protocol", RFC 7644, DOI 10.17487/RFC7644, September 2015, <<https://www.rfc-editor.org/info/rfc7644>>.
- [RFC8055] Holmberg, C. and Y. Jiang, "Session Initiation Protocol (SIP) Via Header Field Parameter to Indicate Received Realm", RFC 8055, DOI 10.17487/RFC8055, January 2017, <<https://www.rfc-editor.org/info/rfc8055>>.

[RFC8225] Wendt, C. and J. Peterson, "PASSporT: Personal Assertion Token", RFC 8225, DOI 10.17487/RFC8225, February 2018, <<https://www.rfc-editor.org/info/rfc8225>>.

[RISC] OpenID Foundation, "OpenID Risk and Incident Sharing and Coordination (RISC) Working Group", <<http://openid.net/wg/risc/>>.

#### Appendix A. Acknowledgments

The editors would like to thank the members of the IETF SCIM working group, which began discussions of provisioning events starting with draft-hunt-scim-notify-00 in 2015. The editors would like to thank the participants in the IETF id-event mailing list, the Security Events working group, and related working groups for their contributions to this specification. The specification incorporates suggestions made by many people, including Annabelle Backman, John Bradley, Alissa Cooper, Ned Freed, Dick Hardt, Russ Housley, Benjamin Kaduk, Mirja Kuehlewind, Mark Lizar, Alexey Melnikov, Andrew Nash, Eric Rescorla, Adam Roach, Justin Richer, Nat Sakimura, Marius Scurtescu, Yaron Sheffer, and Martin Vigoureux.

#### Appendix B. Change Log

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

From the original draft-hunt-idevent-token:

Draft 01 - PH - Renamed eventUris to events

Draft 00 - PH - First Draft

Draft 01 - PH - Fixed some alignment issues with JWT. Remove event type attribute.

Draft 02 - PH - Renamed to Security Events, removed questions, clarified examples and intro text, and added security and privacy section.

Draft 03 - PH

General edit corrections from Sarah Squire

Changed "event" term to "SET"

Corrected author organization for William Denniss to Google

Changed definition of SET to be 2 parts, an envelope and 1 or more payloads.

Clarified that the intent is to express a single event with optional extensions only.

- mbj - Registered "events" claim, and proof-reading corrections.

Draft 04 - PH -

- o Re-added the "sub" claim with clarifications that any SET type may use it.
- o Added additional clarification on the use of envelope vs. payload attributes
- o Added security consideration for event timing.
- o Switched use of "attribute" to "claim" for consistency.
- o Revised examples to put "sub" claim back in the top level.
- o Added clarification that SETs typically do not use "exp".
- o Added security consideration for distinguishing Access Tokens and SETs.

Draft 05 - PH - Fixed find/replace error that resulted in claim being spelled claimc

Draft 06 - PH -

- o Corrected typos
- o New txn claim
- o New security considerations Sequencing and Timing Issues

Draft 07 -

- o PH - Moved payload objects to be values of event URI attributes, per discussion.
- o mbj - Applied terminology consistency and grammar cleanups.

Draft 08 - PH -

- o Added clarification to status of examples

- o Changed from primary vs. extension to state that multiple events may be expressed, some of which may or may not be considered extensions of others (which is for the subscriber or profiling specifications to determine).
- o Other editorial changes suggested by Yaron  
From draft-ietf-secevent-token:

Draft 00 - PH - First WG Draft based on draft-hunt-idevent-token

Draft 01 - PH - Changes as follows:

- o Changed terminology away from pub-sub to transmitter/receiver based on WG feedback
- o Cleaned up/removed some text about extensions (now only used as example)
- o Clarify purpose of spec vs. future profiling specs that define actual events

Draft 02 - Changes are as follows:

- o mbj - Added the Requirements for SET Profiles section.
- o mbj - Expanded the Security Considerations section to describe how to prevent confusion of SETs with ID Tokens, access tokens, and other kinds of JWTs.
- o mbj - Registered the "application/secevent+jwt" media type and defined how to use it for explicit typing of SETs.
- o mbj - Clarified the misleading statement that used to say that a SET conveys a single security event.
- o mbj - Added a note explicitly acknowledging that some SET profiles may choose to convey event subject information in the event payload.
- o PH - Corrected encoded claim example on page 10.
- o mbj - Applied grammar corrections.

Draft 03 - Changes are as follows:

- o pjh - Corrected old "subscriber" to "Event Receiver". Added clarification in definition that Event Receiver is the same as JWT recipient.

- o pjh - Added definition for "toe" (and IANA registration).
- o pjh - Removed "nbf" claim.
- o pjh - Figure 3, moved "sub" to the events payload next to "iss".
- o pjh - Clarified the use of "nonce" in contexts where substitution is possible.
- o mbj - Addressed WGLC comments by Nat Sakimura.
- o mbj - Addressed WGLC comments by Annabelle Backman.
- o mbj - Addressed WGLC comments by Marius Scurtescu.

Draft 04 - mbj - Changes were as follows:

- o Clarified that all "events" values must represent aspects of the same state change that occurred to the subject -- not an aggregation of unrelated events about the subject.
- o Removed ambiguities about the roles of multiple "events" values and the responsibilities of profiling specifications for defining how and when they are used.
- o Corrected places where the term JWT was used when what was actually being discussed was the JWT Claims Set.
- o Addressed terminology inconsistencies. In particular, standardized on using the term "issuer" to align with JWT terminology and the "iss" claim. Previously the term "transmitter" was sometimes used and "issuer" was sometimes used. Likewise, standardized on using the term "recipient" instead of "receiver" for the same reasons.
- o Added a RISC event example, courtesy of Marius Scurtescu.
- o Applied wording clarifications suggested by Annabelle Backman and Yaron Sheffer.
- o Applied numerous grammar, syntax, and formatting corrections.

Draft 05 - mbj - Changes were as follows:

- o Simplified the definitions of the "iat" and "toe" claims in ways suggested by Annabelle Backman.
- o Added privacy considerations text suggested by Annabelle Backman.

- o Updated the RISC event example, courtesy of Marius Scurtescu.
- o Reordered the claim definitions to place the required claims first.
- o Changed to using the RFC 8174 boilerplate instead of the RFC 2119 boilerplate.

Draft 06 - mbj - Changes were as follows:

- o Changed "when the event was issued" to "when the SET was issued" in the "iat" description, as suggested by Annabelle Backman.
- o Applied editorial improvements that improve the consistency of the specification that were suggested by Annabelle Backman, Marius Scurtescu, and Yaron Sheffer.

Draft 07 - PH - Text refinement to Section 3 proposed by Annabelle Backman post WGLC

Draft 08 - mbj - Changes were as follows:

- o Incorporated wording improvements resulting from Russ Housley's SecDir comments.
- o Acknowledged individuals who made significant contributions.

Draft 09 - pjh/mbj - Changes addressing AD review comments by Benjamin Kaduk

Draft 10 - pjh/mbj - Changes were as follows:

- o Incorporated wording improvements resulting from Russ Housley's additional SecDir comments.
- o Registered +jwt structured syntax suffix.

Draft 11 - pjh/mbj - Incorporated feedback from Security Area Director Eric Rescorla and IANA Designated Expert Ned Freed.

- o Clarified "iss" claim language about the SET issuer versus the security subject issuer.
- o Changed a "SHOULD" to a "MUST" in the "sub" claim description to be consistent with the Requirements for SET Profiles section.
- o Described the use of the "events" claim to prevent attackers from passing off other kinds of JWTs as SETs.

- o Stated that SETs are to be signed by an issuer that is trusted to do so for the use case.
- o Added quotes in the phrase '"token revoked" SET to be issued' in the Timing Issues section.
- o Added section number references to the media type and media type suffix registrations.
- o Changed the encodings of the media type and media type suffix registrations to binary (since no line breaks are allowed).
- o Replaced a "TBD" in the media type registration with descriptive text.
- o Acknowledged Eric Rescorla and Ned Freed.

Draft 12 - pjh/mbj - Incorporated feedback from Adam Roach, Alexey Melnikov, and Alissa Cooper.

- o Removed unused references to RFC 7009 and RFC 7517.
- o Corrected name of RFC 8055 in Section 4.3 to "Session Initiation Protocol (SIP) Via Header Field Parameter to Indicate Received Realm".
- o Added normative references for base64url and UTF-8.
- o Section 5.1 - Changed SHOULD to MUST in "personally identifiable information MUST be encrypted using JWE [RFC7516] or ...".
- o Section 5.2 - Changed "MUST consider" to "must consider".

Draft 13 - ph - Added edit from Martin Vigoureaux regarding a non-normative "MAY" in Section 1.1. Updated acknowledgements.

#### Authors' Addresses

Phil Hunt (editor)  
Oracle Corporation

Email: phil.hunt@yahoo.com



Michael B. Jones  
Microsoft

Email: [mbj@microsoft.com](mailto:mbj@microsoft.com)  
URI: <http://self-issued.info/>

William Denniss  
Google

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

Morteza Ansari  
Cisco

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

secevent  
Internet-Draft  
Intended status: Informational  
Expires: February 11, 2018

M. Scurtescu  
Google  
A. Backman  
Amazon  
August 10, 2017

Management API for SET Event Streams  
draft-scurtescu-secevent-event-stream-mgmt-api-00

Abstract

Security Event Token (SET) delivery requires event receivers to indicate to event transmitters the subjects about which they wish to receive events, and how they wish to receive them. This specification defines an HTTP API for a basic control plane that event transmitters can implement and event receivers may use to manage the flow of events from one to the other.

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 <http://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 February 11, 2018.

Copyright Notice

Copyright (c) 2017 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 (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Notational Conventions . . . . .	3
3. Definitions . . . . .	3
4. Event Stream Management . . . . .	3
4.1. Stream Configuration . . . . .	4
4.1.1. Checking a Stream's Status . . . . .	5
4.1.2. Reading a Stream's Configuration . . . . .	6
4.1.3. Updating a Stream's Configuration . . . . .	7
4.2. Subjects . . . . .	9
4.2.1. Adding a Subject to a Stream . . . . .	10
4.2.2. Removing a Subject . . . . .	11
4.3. Verification . . . . .	13
4.3.1. Verification Event . . . . .	13
4.3.2. Triggering a Verification Event. . . . .	13
5. Security Considerations . . . . .	15
5.1. Subject Probing . . . . .	15
5.2. Information Harvesting . . . . .	16
5.3. Malicious Subject Removal . . . . .	16
6. Normative References . . . . .	16
Authors' Addresses . . . . .	17

## 1. Introduction

This specification defines an HTTP API to be implemented by Event Transmitters and that can be used by Event Receivers to query the Event Stream status, to add and remove subjects and to trigger verification.

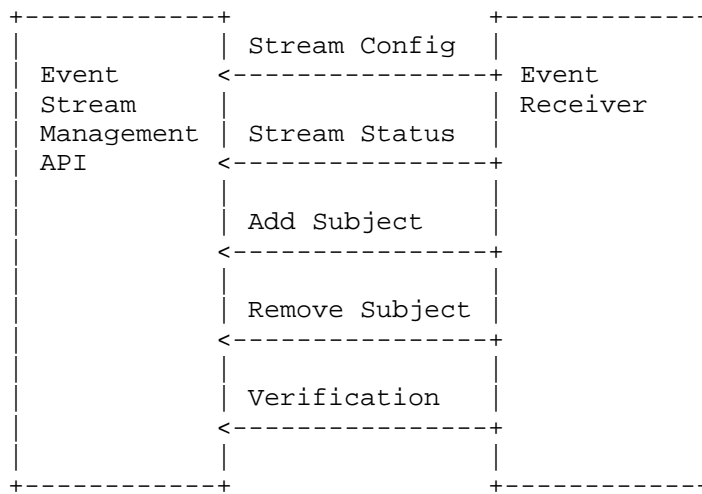


Figure 1: Event Stream Management API

How events are delivered and the structure of events are not in scope for this specification.

## 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. Definitions

In addition to terms defined in [SET], this specification uses the following terms:

### Subject Identifier Object

A JSON object containing a set of one or more claims about a subject that when taken together uniquely identify that subject. This set of claims SHOULD be declared as an acceptable way to identify subjects of SETs by one or more specifications that profile [SET].

## 4. Event Stream Management

Event Receivers manage how they receive events, and the subjects about which they want to receive events over an Event Stream by making HTTP requests to endpoints in the Event Stream Management API.

The Event Stream Management API is implemented by the Event Transmitter and consists of the following endpoints:

Configuration Endpoint

An endpoint used to read the Event Stream's current configuration.

Status Endpoint

An endpoint used to read the Event Stream's current status.

Add Subject Endpoint

An endpoint used to add subjects to an Event Stream.

Remove Subject Endpoint

An endpoint used to remove subjects from an Event Stream.

Verification Endpoint

An endpoint used to request the Event Transmitter transmit a Verification Event over the Event Stream.

An Event Transmitter MAY use the same URLs as endpoints for multiple streams, provided that the Event Transmitter has some mechanism through which they can identify the applicable Event Stream for any given request, e.g. from authentication credentials. The definition of such mechanisms is outside the scope of this specification.

#### 4.1. Stream Configuration

An Event Stream's configuration is represented as a JSON object with the following properties:

aud

A string containing an audience claim as defined in JSON Web Token (JWT) [RFC7519] that identifies the Event Receiver for the Event Stream. This property cannot be updated.

events

OPTIONAL. An array of URIs identifying the set of events which MAY be delivered over the Event Stream. If omitted, Event Transmitters SHOULD make this set available to the Event Receiver via some other means (e.g. publishing it in online documentation).

delivery

A JSON object containing a set of name/value pairs specifying configuration parameters for the SET delivery method. The actual delivery method is identified by the special key "delivery\_method" with the value being a URI as defined in [DELIVERY].

**min\_verification\_interval**

An integer indicating the minimum amount of time in seconds that must pass in between verification requests. If an Event Receiver submits verification requests more frequently than this, the Event Transmitter MAY respond with a 429 status code. An Event Transmitter SHOULD NOT respond with a 429 status code if an Event Receiver is not exceeding this frequency.

**status**

A string indicating the current status of the event stream. It MUST have one of the following values:

**enabled**

The transmitter will transmit events over the stream, according to the stream's configured delivery method.

**paused**

The transmitter will not transmit events over the stream. The transmitter will hold any events it would have transmitted while paused, and will transmit them when the stream's status becomes "enabled".

**disabled**

The transmitter will not transmit events over the stream, and will not hold any events for later transmission.

**4.1.1.1. Checking a Stream's Status**

An Event Receiver checks the current status of an event stream by making an HTTP GET request to the stream's Status Endpoint. On receiving a valid request the Event Transmitter responds with a 200 OK response containing a [JSON] object with a single attribute "status", whose string value is the value of the stream's status.

The following is a non-normative example request to check an event stream's status:

```
GET /set/stream/status HTTP/1.1
Host: transmitter.example.com
Authorization: Bearer eyJ0b2t1biI6ImV4YW1wbGUifQo=
```

Figure 2: Example: Check Stream Status Request

The following is a non-normative example response:

```
HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
  "status": "enabled"
}
```

Figure 3: Example: Check Stream Status Response

#### 4.1.2. Reading a Stream's Configuration

An Event Receiver gets the current configuration of a stream by making an HTTP GET request to the Configuration Endpoint. On receiving a valid request the Event Transmitter responds with a 200 OK response containing a [JSON] representation of the stream's configuration in the body.

The following is a non-normative example request to read an Event Stream's configuration:

```
GET /set/stream HTTP/1.1
Host: transmitter.example.com
Authorization: Bearer eyJ0b2tlbiI6ImV4YWlwbGUifQo=
```

Figure 4: Example: Read Stream Configuration Request

The following is a non-normative example response:

```

HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
  "aud": "http://www.example.com",
  "delivery": {
    "delivery_method": "urn:example:secevent:delivery:http_post",
    "url": "https://receiver.example.com/events"
  },
  "status": "enabled",
  "events": [
    "urn:example:secevent:events:type_1",
    "urn:example:secevent:events:type_2",
    "urn:example:secevent:events:type_3"
  ],
  "min_verification_interval": 60,
}

```

Figure 5: Example: Read Stream Configuration Response

Errors are signaled with HTTP status codes as follows:

Code	Description
401	if authorization failed or it is missing
403	if the Event Receiver is not allowed to read the stream configuration
404	if there is no Event Stream configured for this Event Receiver

Table 1: Read Stream Configuration Errors

#### 4.1.3. Updating a Stream's Configuration

An Event Receiver updates the current configuration of a stream by making an HTTP POST request to the Configuration Endpoint. The POST body contains a `{!JSON}` representation of the updated configuration. On receiving a valid request the Event Transmitter responds with a 200 OK response containing a `[JSON]` representation of the updated stream configuration in the body.



The full set of editable properties must be present in the POST body, not only the ones that are specifically intended to be changed. Missing properties SHOULD be interpreted as requested to be deleted. Event Receivers should read the configuration first, modify the [JSON] representation, then make an update request.

Properties that cannot be updated MAY be present, but they MUST match the expected value.

The following is a non-normative example request to read an Event Stream's configuration:

```
POST /set/stream HTTP/1.1
Host: transmitter.example.com
Authorization: Bearer eyJ0b2tldiI6ImV4YW1wbGUifQo=

{
  "aud": "http://www.example.com",
  "delivery": {
    "delivery_method": "urn:example:secevent:delivery:http_post",
    "url": "https://receiver.example.com/events"
  },
  "status": "paused",
  "events": [
    "urn:example:secevent:events:type_1",
    "urn:example:secevent:events:type_2",
    "urn:example:secevent:events:type_3"
  ]
}
```

Figure 6: Example: Update Stream Configuration Request

The following is a non-normative example response:

```

HTTP/1.1 200 OK
Content-Type: application/json; charset=UTF-8
Cache-Control: no-store
Pragma: no-cache

{
  "aud": "http://www.example.com",
  "delivery": {
    "delivery_method": "urn:example:secevent:delivery:http_post",
    "url": "https://receiver.example.com/events"
  },
  "status": "paused",
  "events": [
    "urn:example:secevent:events:type_1",
    "urn:example:secevent:events:type_2",
    "urn:example:secevent:events:type_3"
  ]
}

```

Figure 7: Example: Update Stream Configuration Response

Errors are signaled with HTTP status codes as follows:

Code	Description
400	if the request body cannot be parsed or if the request is otherwise invalid
401	if authorization failed or it is missing
403	if the Event Receiver is not allowed to update the stream configuration

Table 2: Update Stream Configuration Errors

#### 4.2. Subjects

An Event Receiver can indicate to an Event Transmitter whether or not the receiver wants to receive events about a particular subject by "adding" or "removing" that subject to the Event Stream, respectively.

#### 4.2.1. Adding a Subject to a Stream

To add a subject to an Event Stream, the Event Receiver makes an HTTP POST request to the Add Subject Endpoint, containing in the body a Subject Identifier Object identifying the subject to be added. On a successful response, the Event Transmitter responds with an empty 200 OK response.

The Event Transmitter MAY choose to silently ignore the request, for example if the subject has previously indicated to the transmitter that they do not want events to be transmitted to the Event Receiver. In this case, the transmitter MAY return an empty 200 OK response or an appropriate error code (See Security Considerations (Section 5)).

Errors are signaled with HTTP status codes as follows:

Code	Description
400	if the request body cannot be parsed or if the request is otherwise invalid
401	if authorization failed or it is missing
403	if the Event Receiver is not allowed to add this particular subject
404	if the subject is not recognized by the Event Transmitter, the Event Transmitter may choose to stay silent in this case and respond with 200
429	if the Event Receiver is sending too many requests in a given amount of time

Table 3: Add Subject Errors

The following is a non-normative example request to add a subject to a stream, where the subject is identified by an OpenID Connect email claim:

```
POST /set/subjects:add HTTP/1.1
Host: transmitter.example.com
Authorization: Bearer eyJ0b2tlbiI6ImV4YW1wbGUifQo=

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

Figure 8: Example: Add Subject Request

The following is a non-normative example response to a successful request:

```
HTTP/1.1 200 OK
Server: transmitter.example.com
Cache-Control: no-store
Pragma: no-cache
```

Figure 9: Example: Add Subject Response

#### 4.2.2. Removing a Subject

To remove a subject from an Event Stream, the Event Receiver makes an HTTP POST request to the Remove Subject Endpoint, containing in the body a Subject Identifier Object identifying the subject to be removed. On a successful response, the Event Transmitter responds with a 204 No Content response.

Errors are signaled with HTTP status codes as follows:

Code	Description
400	if the request body cannot be parsed or if the request is otherwise invalid
401	if authorization failed or it is missing
403	if the Event Receiver is not allowed to remove this particular subject
404	if the subject is not recognized by the Event Transmitter, the Event Transmitter may chose to stay silent in this case and respond with 204
429	if the Event Receiver is sending too many requests in a gvien amount of time

Table 4: Remove Subject Errors

The following is a non-normative example request where the subject is identified by a phone\_number claim:

```
POST /set/subjects:remove HTTP/1.1
Host: transmitter.example.com
Authorization: Bearer eyJ0b2tldiI6ImV4YWlwbGUifQo=

{
  "phone_number": "+1 206 555 0123"
}
```

Figure 10: Example: Remove Subject Request

The following is a non-normative example response to a successful request:

```
HTTP/1.1 204 No Content
Server: transmitter.example.com
Cache-Control: no-store
Pragma: no-cache
```

Figure 11: Example: Remove Subject Response

#### 4.3. Verification

In some cases, the frequency of event transmission on an Event Stream will be very low, making it difficult for an Event Receiver to tell the difference between expected behavior and event transmission failure due to a misconfigured stream. Event Receivers can request that a verification event be transmitted over the Event Stream, allowing the receiver to confirm that the stream is configured correctly upon successful receipt of the event. The acknowledgment of a Verification Event also confirms to the Event Transmitter that end-to-end delivery is working, including signature verification and encryption.

An Event Transmitter MAY send a Verification Event at any time, even if one was not requested by the Event Receiver.

##### 4.3.1. Verification Event

The Verification Event is a standard SET with the following attributes:

event type

The Event Type URI is: "urn:ietf:params:secevent:event-type:core:verification".

state

OPTIONAL An opaque value provided by the Event Receiver when the event is triggered. This is a nested attribute in the event payload.

Upon receiving a Verification Event, the Event Receiver SHALL parse the SET and validate its claims. In particular, the Event Receiver SHALL confirm that the value for "state" is as expected. If the value of "state" does not match, an error response of "setData" SHOULD be returned (see Section 2.4 of [DELIVERY]).

In many cases, Event Transmitters MAY disable or suspend an Event Stream that fails to successfully verify based on the acknowledgement or lack of acknowledgement by the Event Receiver.

##### 4.3.2. Triggering a Verification Event.

To request that a verification event be sent over an Event Stream, the Event Receiver makes an HTTP POST request to the Verification Endpoint, with a JSON object containing the parameters of the verification request, if any. On a successful request, the event transmitter responds with an empty 204 No Content response.

Verification requests have the following properties:

state

OPTIONAL. An arbitrary string that the Event Transmitter MUST echo back to the Event Receiver in the verification event's payload. Event Receivers MAY use the value of this parameter to correlate a verification event with a verification request. If the verification event is initiated by the transmitter then this parameter MUST not be set.

A successful response from a POST to the Verification Endpoint does not indicate that the verification event was transmitted successfully, only that the Event Transmitter has transmitted the event or will do so at some point in the future. Event Transmitters MAY transmit the event via an asynchronous process, and SHOULD publish an SLA for verification event transmission times. Event Receivers MUST NOT depend on the verification event being transmitted synchronously or in any particular order relative to the current queue of events.

Errors are signaled with HTTP status codes as follows:

Code	Description
400	if the request body cannot be parsed or if the request is otherwise invalid
401	if authorization failed or it is missing
429	if the Event Receiver is sending too many requests in a given amount of time

Table 5: Verification Errors

The following is a non-normative example request to trigger a verification event:

```
POST /set/verify HTTP/1.1
Host: transmitter.example.com
Authorization: Bearer eyJ0b2t1biI6ImV4YW1wbGUifQo=
Content-Type: application/json; charset=UTF-8

{
  "state": "VGhpcyBpcyBhbiBleGFtcGxlIHNOYXRlIHZhbHVlLgo="
}
```

Figure 12: Example: Trigger Verification Request

The following is a non-normative example response to a successful request:

```
HTTP/1.1 204 No Content
Server: transmitter.example.com
Cache-Control: no-store
Pragma: no-cache
```

Figure 13: Example: Trigger Verification Response

And the following is a non-normative example of a verification event sent to the Event Receiver as a result of the above request:

```
{
  "jti": "123456",
  "iss": "https://transmitter.example.com",
  "aud": "receiver.example.com",
  "iat": "1493856000",
  "events": [
    "urn:ietf:params:secevent:event-type:core:verification" : {
      "state": "VGhpcyBpcyBhbiBleGFtcGxlIHNOYXRlIHZhbHVlLgo=",
    },
  ],
}
```

Figure 14: Example: Verification SET

## 5. Security Considerations

### 5.1. Subject Probing

It may be possible for an Event Transmitter to leak information about subjects through their responses to add subject requests. A 404 response may indicate to the Event Receiver that the subject does not exist, which may inadvertently reveal information about the subject (e.g. that a particular individual does or does not use the Event Transmitter's service).



Event Transmitters SHOULD carefully evaluate the conditions under which they will return error responses to add subject requests. Event Transmitters MAY return a 204 response even if they will not actually send any events related to the subject, and Event Receivers MUST NOT assume that a 204 response means that they will receive events related to the subject.

## 5.2. Information Harvesting

SETs may contain personally identifiable information (PII) or other non-public information about the event transmitter, the subject (of an event in the SET), or the relationship between the two. It is important for Event Transmitters to understand what information they are revealing to Event Receivers when transmitting events to them, lest the event stream become a vector for unauthorized access to private information.

Event Transmitters SHOULD interpret add subject requests as statements of interest in a subject by an Event Receiver, and ARE NOT obligated to transmit events related to every subject an Event Receiver adds to the stream. Event Transmitters MAY choose to transmit some, all, or no events related to any given subject and SHOULD validate that they are permitted to share the information contained within an event with the Event Receiver before transmitting the event. The mechanisms by which such validation is performed are outside the scope of this specification.

## 5.3. Malicious Subject Removal

A malicious party may find it advantageous to remove a particular subject from a stream, in order to reduce the Event Receiver's ability to detect malicious activity related to the subject, inconvenience the subject, or for other reasons. Consequently it may be in the best interests of the subject for the Event Transmitter to continue to send events related to the subject for some time after the subject has been removed from a stream.

Event Transmitters MAY continue sending events related to a subject for some amount of time after that subject has been removed from the stream. Event Receivers MUST tolerate receiving events for subjects that have been removed from the stream, and MUST NOT report these events as errors to the Event Transmitter.

## 6. Normative References

- [DELIVERY] "SET Token Delivery Using HTTP", n.d., <<https://github.com/independentid/Identity-Events/blob/master/draft-hunt-secevent-delivery.txt>>.
- [JSON] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <<http://www.rfc-editor.org/info/rfc7159>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<http://www.rfc-editor.org/info/rfc2119>>.
- [RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <<http://www.rfc-editor.org/info/rfc7519>>.
- [SET] "Security Event Token (SET)", n.d., <<https://tools.ietf.org/html/draft-ietf-secevent-token-01>>.

#### Authors' Addresses

Marius Scurtescu  
Google

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

Annabelle Backman  
Amazon

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