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.

Table of Contents

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 os 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 intenting 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 an 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 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].

PHIONE 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 extented 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:

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

- 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.

- 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".

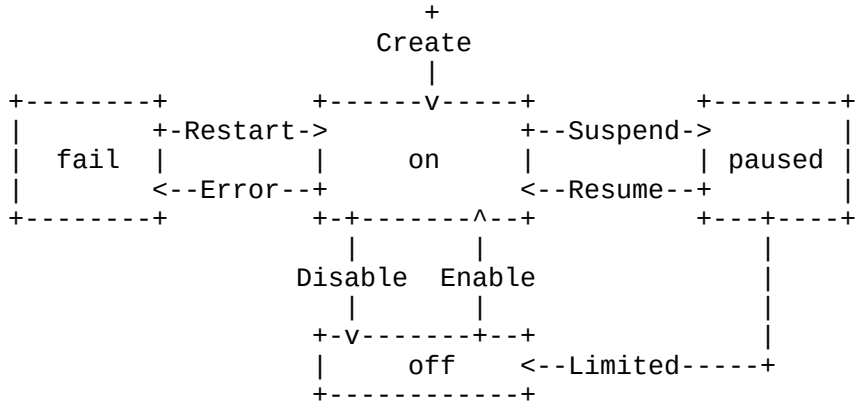

Figure 5: Event Stream States at Event Transmitter

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 Event Stream 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 "eventUris" and "eventUris_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",
  "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 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:

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

- 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:

- As an attribute of an Event Stream configuration (see Section 4.2);
- As a member of SCIM Group (see Section 4.3); and,
- 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",
  "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":"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 specfied 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" : fase,
      "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 stre
ams.",
  "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" : "eventUris_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