Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 9, 2017

P. Hunt, Ed.
Oracle
M. Scurtescu
Google

March 8, 2017

SET Token Delivery Using HTTP
draft-hunt-secevent-distribution-01

Abstract

This specification defines how a series of security event tokens (SETs) may be delivered to a previously registered receiver using HTTP over TLS. The specification defines the metadata the an Event Transmitter uses to describe the Event Receiver's HTTP endpoint and the SET token delivery configuration. The specification defines how the Event Receiver may check the current configuration metadata and delivery status using HTTP GET over TLS. 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 will expire on September 9, 2017.

Copyright Notice

Table of Contents

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 POST [RFC7231] over TLS. The specification defines the metadata the Event Transmitter uses to describe the Event Receiver's HTTP endpoint and the SET token delivery configuration. The specification defines how the Event Receiver may check the current configuration metadata and delivery status using HTTP GET over TLS. The specification also defines how delivery can be assured subject to the SET Token Receiver's need for assurance.

The following diagram shows a typical SET Event Stream. A stream consists of a pair of HTTP endpoints, one for the event stream transmitter and one for the receiver. The receiver endpoint is used by the transmitter to deliver SET events via HTTPS POST and is known as the "Data Plane". The transmitter's HTTP endpoint is used by the receiver to perform HTTPS GET requests to check the stream status and is known as the "Control Plane". In the diagram, the arrow heads point to the service provider (the direction of an HTTP request):

```
          +-----------+ Data Plane            +----------+
          |Transmitter+------HTTP POST--------> Receiver |
          |           <------HTTP GET---------+          |
          +-----------+         Control Plane +----------+
```

Figure 1: SET Event Stream

In some service provider relationships, for example between Identity Providers and Relying Parties, there may be a need to have bi-directional SET event exchange. This involves establishing a second event stream that works with transmitter and receiver roles reversed.

```
           Identity                             Relying
           Provider                             Party

                          IDP to RP Stream

         +-----------+ Data Plane            +------------+
         |Transmitter+-----------------------> Receiver   |
         |           <-----------------------+            |
         +-----------+         Control Plane +------------+


                          RP to IDP Stream

         +-----------+ Data Plane            +------------+
         | Receiver  <-----------------------+ Transmitter|
         |           +----------------------->            |
         +-----------+         Control Plane +------------+
```

Figure 2: Duplexed Streams

This specification contains two major sections:

Control Plane The service through which Event Receivers can review and optionally managed Event Streams. It defines the metadata associated with Event Streams along with stream status reporting.

Data Plane Through which SET Events are delivered by an Event Transmitter to an Event Receiver using a defined Event Stream. The Data Plane includes a verification process which tests and validates Event Stream configuration. The Data plan defines processing and error signaling used in the delivery of SETs.

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] .

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:

Identity Provider
An Identity Provider is a service provider that issues authentication assertions that may be used by Relying Party service providers to establish login sessions with users. Examples of Identity Providers are defined in: OpenID Connect [openid-connect-core] and SAML2 [saml-core-2.0]. For the purpose of this specification an Identity Provider also includes any provider of services where the compromise of an account may open up relying parties to attack. For example for the purposes of security events, an email service provider could be considered an "implicit" Identity Provider.

Relying Party

A Relying Party is a service provider that accepts assertions from Identity Providers to establish sessions. Examples of Relying Parties are defined in: OpenID Connect [openid-connect-core] and SAML2 [saml-core-2.0]

Event Transmitter

A service provider that delivers SETs to other providers known as Event Receivers. Some examples of Event Transmitters are Identity Providers and Relying Parties. 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 (known as the "Data Plane"). Some examples of Event Receivers are Identity Providers and Relying Parties. Event Receivers can check current Event Stream configuration and status by accessing the Event Transmitters "Control Plane".

Event Stream

An Event Stream establishes Event Receiver communication endpoints, security configuration and feed content that is used by an Event Transmitter to send a series of SET Events to an Event Receiver. An Event Stream defines a "Data Plane" and "Control Plane" service relationship between an Event Transmitter and and Event Receiver.

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.

Event Family

An Event Family is a URI that describes the set of events types be issued in an Event Stream.

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.

2. Control Plane - Monitoring

The Control Plane is provided by the Event Transmitter and enables Event Receivers to check the Event Stream configuration and check for transmission errors. This section describes mandatory to implement functionality to enable Event Receivers to detect SET delivery problems that may occur when an Event Transmitter fails to deliver SETs.

Implementers MAY optionally implement and support full Event Stream provisioning and management as described in Section 4. This functionality also allows Event Receivers to "pause", "disable", or re-enable Event Streams in scenario where the operational needs of the receiver need to be co-ordinated with Event Transmitters (see Section 2.2 and Section 4.3).

SCIM defines flexible mechanisms to ease adaptability to different underlying data systems while maximizing inter-operabilty. Section 2 [RFC7643] SHALL provide the processing rule that enable Control Plane providers and clients negotiate specific attributes (metadata) including differing provider definitions of attribute types, mutability, cardinality, or returnability that MAY differ. For HTTP method handling and error signaling, the processing rules in [RFC7644] SHALL apply.

2.1. Event Stream Configuration

An Event Stream represents an agreement to deliver SETs from a specified Feed URI from an Event Transmitter to an Event Receiver. The method of delivery and the parameters for delivery are specified a set of parameters called Event Stream metadata (see Section 2.1).

An Event Stream is defined by the following metadata:

feedUri

An OPTIONAL JSON String value containing the URI for a feed supported by the feed provider. It describes the content of the feed and MAY also be a resolvable URI where the feed meta data may be returned as a JSON object. REQUIRED.

methodUri

A REQUIRED JSON String value which is a URI with a prefix of "urn:ietf:params:set:method". This specification defines HTTP

POST delivery method:
"urn:ietf:params:set:method:HTTP:webCallback"
in which the Feed Provider delivers events using HTTP POST to a specified callback URI.

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".

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.

feedJwk

An OPTIONAL public JSON Web Key (see [RFC7517]) from the Event Transmitter that will be used by the Event Receiver to verify the authenticity of issued SETs.

confidentialJwk

An OPTIONAL public JSON Web Key (see [RFC7517]) for the Event Receiver that MAY be used by the Feed Provider to encrypt SET tokens for the specified Event Receiver.

subStatus

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 2.2. Valid keywords are:

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

"verify" - indicates the Event Stream is pending verification. While in "verify", SETs, except for the verify SET (see Section 3.4) are not delivered to the Event Receiver. Once verified, the status returns to "on".

"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 metadata 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 "subStatus" 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 "subStatus" 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.

[[Editors note: other conditions?]]

txErrDesc

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

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

2.2. Event Stream State Model

The Event Stream configuration attribute "subStatus" tracks the state of any particular Event Stream with regards to whether SETs are ready or able to be delivered. The impact on delivery processing is described in Table 1.

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 "verify".

```
                                  +
                                  |
                                Create
                                  v
       +------+              +----------+
       | fail +->Restart---->|  verify  |
       +------+              +----+-----+
           ^                      |
           |<----Confirm Fail<----+
           |                   Confirm
           |                      v
           |                 +----------+               +--------+
           |                 |          +--->Suspend--->|        |
           +------Timeout<---+    on    |               | paused |
                             |          |<--Resume<-----+        |
                             +-+--------+               +----+---+
                               |      ^                      |
                            Disable Enable                   |
                               v      |                      |
                             +--------+-+                    |
                             |   off    |<----Limited<-------+
                             +----------+
```

Figure 3: Event Stream States at Event Transmitter

In the above diagram, the following actions impact the state of an Event Stream. "subStatus" 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 using SCIM as described in Section 4.2. The initial state is "verify".

Confirm

The Event Transmitter sends a verification SET to the Event Receiver which confirms with the correct response as described in Section 3.4. If it succeeds to deliver, the Event Transmitter SHALL set state to "on".

Confirm Fail

If the confirmation fails, the Event Transmitter sets the state to "fail" requiring administrative action to correct the issue and "Restart".

Timeout

A Event Transmitter who 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". In general, the intention is to indicate the maximum number of retries or time a Event Transmitter is able to wait until SET event loss begins to occur resulting in the failed state.

Limited

A paused Event Stream has reached a limit and the Event Transmitter can no longer retain SETs. The Event Transmitter changes the state to "off".

Restart

An administrator having corrected the failed delivery condition modifies the Event Stream state to "verify" (e.g. see Section 4.3).

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 4.3. While suspended, the Event Transmitter MAY retain undelivered SETs for a period of time. If the Event Transmitter is no longer able to retain SETs, the Event Stream state SHOULD be set to "off" to indicate SETs are being lost.

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 4.3. While the Event Stream is disabled, all SETs that occur at the Event Transmitter are lost.

2.3. Checking Stream Configuration and Stream State

An Event Receiver MAY check the current status of a Stream with the Event Transmitter, by performing an HTTP GET using the provided URI from the Transmitter.

The format of the response is defined by Section TBD [RFC7644].

In addition to the attributes defined in Section 2.1, the response SHALL include an additional JSON attribute "schemas" with at least a single value of "urn:ietf:params:scim:schemas:event:2.0:EventStream". This static attribute is provided to enable optional SCIM client compatibility and informs the client of the type of JSON object being returned. Service providers may offer additional attributes by adding additional schema values as per [RFC7644].

The response below shows an example response to an HTTP GET, in this case to "https://example.com/v2/EventStreams/767aad7853d240debc8e3c962051c1c0".

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

{
  "schemas":["urn:ietf:params:scim:schemas:event:2.0:EventStream"],
  "id":"767aad7853d240debc8e3c962051c1c0",
  "feedName":"OIDCLogoutFeed",
  "feedUri":
    "https://example.com/v2/Feeds/88bc00de776d49d5b535ede882d98f74",
  "methodUri":"urn:ietf:params:set:method:HTTP:webCallback",
  "deliveryUri":"https://notify.examplerp.com/Events",
  "aud":"https://sets.myexamplerp.com",
  "subStatus":"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 Stream GET Response

In the above figure, the Event Stream is showing a failed status due to a TCP connection error. The Event Receiver is able to discover that its endpoint was unavailable and has been marked failed by the Event Transmitter. It is expected that the appropriate operations staff would be alerted and some corrective action would be taken.

The frequency with which Event Receivers should poll the Event Stream status depends on the following factors:

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

- A frequency appropriate to 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.

In most cases Event Stream status polling can 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.

3. Data Plane

The data plane represent the HTTP request channel by which the Event Transmitter delivers SET Events to an Event Receiver.

3.1. Event Delivery Process

When a Security Event occurs, the Feed Provider constructs a SET token [I-D.ietf-secevent-token] that describes the event. The feed provider determines the feeds that the event should be distributed to, and determines which Event Receivers need to be notified.

How SET Events 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 a Event Receiver, the Feed Transmitter attempts to deliver the SET based on the Event Receiver's registered delivery mechanism:

- The Event Transmitter uses an HTTP/1.1 POST to the Event Receiver endpoint to deliver the SET;
- Or, the Feed Transmitter delivers the event through a different method not defined by this specification.

Feed Transmitters SHALL NOT be required to main or record SETs. As such, transmitted SETs SHOULD be self-validating (e.g. signed).

If delivery to any particular Event Receiver has been delayed for an extended period of time, the Feed Transmitter MAY suspend the affected Event Stream and even stop maintaining outstanding SETs for the Event Receiver at its discretion and available resources. See Event Stream "subState" in Section 2.1.

Upon receiving a SET, the Event Receiver reads the SET and validates it. Based upon the content of the token, the Event Receiver decides what, if any, action needs to be taken in response to the received SET. For example, in response to a SCIM provisioning event [idevent-scim] indicating a changed resource, the Event Receiver might perform a SCIM GET request (see Section 3.4 [RFC7644]) to the affected resource URI in order to confidentially obtain the current state of the transmitter's affected SCIM resource in order to reconcile local corresponding state changes.

The action a Event Receiver takes in response to a SET MAY be substantially different than merely copying the action of the SET issuer. A single SET can trigger one or more receiver actions or it can be ignored. For example, upon receiving notification that a user resource has been added to a group, the Event Receiver may first determine that the user does not exist in the Event Receiver's domain. The Event Receiver translates the event into two actions:

1. Retrieve the user (e.g. using SCIM GET) and then provisions the user locally. After enabling the user,

2. The Event Receiver then enables the user for the application associated with membership in the issuer's group.

3.2. Event Stream State

As mentioned in Section 2.1, the attribute "subStatus" defines the current state of an Event Stream. Figure 3 shows a state diagram for Event Streams. The following describes that actions taken by the Event Transmitter based upon "subStatus".

Status	Action
on	Delivery SHALL be attempted based on the method defined in the Event Stream attribute "methodUri". If the SET fails to deliver it MAY be retained for a retry delivery in a minimum of "minDeliveryInterval" seconds. If new SETs arrive before the interval, the SETs MUST be held for delivery in order of reception. If this is a repeat attempt to deliver, the Event Transmitter MAY discard the SET if "maxRetries" or "maxDeliveryTime" is exceeded. If a SET is discarded, the Event Transmitter MAY set "subStatus" to "failed".
verify	If the SET is not a Verify SET, the SET MAY be retained for a retry at the Event Transmitter's discretion. If a Verify SET fails to deliver, the Event Transmitter SHALL set "subStatus" to "failed". The Event Transmitter MAY opt to make multiple attempts to complete a verification during which status remains as "verify".
paused	The SET is held for delivery in a queue. The Event Transmitter MAY at its own discretion set the Event Stream state to "failed" if "subStatus" is not returned to "on" in what the Event Transmitter determines to be a reasonable amount of time.
off	The SET is ignored.
fail	The SET is ignored due to a previous unrecoverable error.

Table 1: Delivery Processing By Status

3.3. HTTP POST Delivery

This method allows a feed provider to use HTTP POST (Section 4.3.3 [RFC7231]) to deliver SETs to the registered web callback URI identified in the Event Stream configuration. The Event Stream "methodUri" value for this method is "urn:ietf:params:set:method:HTTP:webCallback".

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

The Event Stream's "deliveryUri" attribute indicates the location of a Event Receiver provided endpoint which accepts HTTP POST requests (e.g. "https://notify.examplerp.com/Events").

The content-type for the HTTP POST is "application/jwt" and SHALL consists of a single SET token (see [I-D.ietf-secevent-token]).

```
eyJhbGciOiJub25lIn0
.
eyJwdWJsaXNoZXJVcmkiOiJodHRwczovL3NjaW0uZXhhbXBsZS5jb20iLCJmZWV
kVXJpcyI6WyJodHRwczovL2podWIuZXhhbXBsZS5jb20vRmVlZHMvOThkNTI0Nj
FmYTViYmM4Nzk1OTNiNzc1NCIsImh0dHBzOi8vamh1Yi5leGFtcGxlLmNvbS9GZ
WVkcy81ZDc2MDQ1MTZiMWQwODY0MWQ3Njc2ZWU3Il0sInJlc291cmNlVXJpcyI6
WyJodHRwczovL3NjaW0uZXhhbXBsZS5jb20vVXNlcnMvNDRmNjE0MmRmOTZiZDZ
hYjYxZTc1MjFkOSJdLCJldmVudFR5cGVzIjpbIkNSRUFURSJdLCJhdHRyaWJ1dG
VzIjpbImlkIiwibmFtZSIsInVzZXJOYW1lIiwicGFzc3dvcmQiLCJlbWFpbHMiX
SwidmFsdWVzIjp7ImVtYWlscyI6W3sidHlwZSI6IndvcmsiLCJ2YWx1ZSI6Impk
b2VAZXhhbXBsZS5jb20ifV0sInBhc3N3b3JkIjoibm90NHUybm8iLCJ1c2VyTmF
tZSI6Impkb2UiLCJpZCI6IjQ0ZjYxNDJkZjk2YmQ2YWI2MWU3NTIxZDkiLCJuYW
1lIjp7ImdpdmVuTmFtZSI6IkpvaG4iLCJmYW1pbHlOYW1lIjoiRG9lIn19fQ
.
```

Figure 5: Encoded SET To Be Transmitted

To deliver an event, the Event Transmitter generates an event delivery message and uses HTTP POST to the EventStream configured endpoint. The content-type of the message is "application/jwt" and the expected response type (accept) is "application/json".

```
POST /Events  HTTP/1.1

Host: notify.examplerp.com
Accept: application/json
Content-Type: application/jwt
"eyJhbGciOiJub25lIn0
.
eyJwdWJsaXNoZXJVcmkiOiJodHRwczovL3NjaW0uZXhhbXBsZS5jb20iLCJmZWV
kVXJpcyI6WyJodHRwczovL2podWIuZXhhbXBsZS5jb20vRmVlZHMvOThkNTI0Nj
FmYTViYmM4Nzk1OTNiNzc1NCIsImh0dHBzOi8vamh1Yi5leGFtcGxlLmNvbS9GZ
WVkcy81ZDc2MDQ1MTZiMWQwODY0MWQ3Njc2ZWU3Il0sInJlc291cmNlVXJpcyI6
WyJodHRwczovL3NjaW0uZXhhbXBsZS5jb20vVXNlcnMvNDRmNjE0MmRmOTZiZDZ
hYjYxZTc1MjFkOSJdLCJldmVudFR5cGVzIjpbIkNSRUFURSJdLCJhdHRyaWJ1dG
VzIjpbImlkIiwibmFtZSIsInVzZXJOYW1lIiwicGFzc3dvcmQiLCJlbWFpbHMiX
SwidmFsdWVzIjp7ImVtYWlscyI6W3sidHlwZSI6IndvcmsiLCJ2YWx1ZSI6Impk
b2VAZXhhbXBsZS5jb20ifV0sInBhc3N3b3JkIjoibm90NHUybm8iLCJ1c2VyTmF
tZSI6Impkb2UiLCJpZCI6IjQ0ZjYxNDJkZjk2YmQ2YWI2MWU3NTIxZDkiLCJuYW
1lIjp7ImdpdmVuTmFtZSI6IkpvaG4iLCJmYW1pbHlOYW1lIjoiRG9lIn19fQ
.
```

Figure 6: Example Web Callback 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 indicate successful submission by responding with HTTP Status 202 as "Accepted" (see Section 6.3.3 [RFC7231]).

If SET or JWT is invalid, or there is an HTTP error, the Event Receiver SHALL respond with the appropriate HTTP error or an HTTP Status 400 Bad Request error as follows:

Err Value	Description
jwtParse	Invalid or unparsable JWT or JSON structure.
jwtHdr	In invalid JWT header was detected.
jwtCypto	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 inability to parse SET event payload.
setData	SET event claims incomplete or invalid.
dup	A duplicate SET was received and has been ignored.

Table 2: HTTP Status 400 Errors

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

```
HTTP/1.1 202 Accepted
```

Figure 7: Example Successful Delivery Response

An HTTP Status 400 Bad Request response includes 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 2).

description
   A human-readable text that provides additional diagnostic information.

The following is an example non-normative Bad Request error.

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

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

}
```

Figure 8: Example Bad Request Response

3.4. Event Stream Verification

To confirm an Event Stream configuration, the Event Transmitter SHALL send a verification SET to the Event Receiver using the registered "methodUri" mechanism which in this case is "urn:ietf:params:set:method:HTTP:webCallback".

The Verify SET contains the following attributes:

events Set with a value of "[[this RFC URL]]#verify".

iss Set to the URI defined in the Event Stream metadata (see Section 2.1).

aud MUST be set to a value that matches the EventStream "aud" value (see Section 2.1).

exp A value that indicates the time the verification request will expire. Once expired, the server will set the Event Stream state to "fail".

If the Event Stream "confidentialJWK" value was supplied, then the SET SHOULD be encrypted with the provided key. Successful parsing of the message confirms that provides confirmation of correct configuration and possession of keys.

A payload attribute "confirmChallenge" is provided with a JSON String value that the Event Receiver SHALL echo back in its response. The intent is to confirm that the Event Receiver has successfully parsed the SET and is not just echoing back HTTP success.

A non-normative JSON representation of an event to be sent to a Event Receiver as a Event Stream confirmation. Note the event is not yet encoded as a JWT token:

```
{
  "jti": "4d3559ec67504aaba65d40b0363faad8",
  "events":["[[this RFC URL]]#verify"],
  "iat": 1458496404,
  "iss": "https://scim.example.com",
  "exp": 1458497000,
  "aud":[
   "https://scim.example.com/Feeds/98d52461fa5bbc879593b7754",
   "https://scim.example.com/Feeds/5d7604516b1d08641d7676ee7"
  ],
  "[[this RFC URL]]#verify":{
    "confirmChallenge":"ca2179f4-8936-479a-a76d-5486e2baacd7"
  }
}
```

Figure 9: Example Verification SET with Challenge

The above SET is encoded as a JWT and transmitted to the Event Receiver as shown in Figure 6.

Upon receiving a verify SET, the Event Receiver SHALL respond with a JSON object that includes a "challengeResponse" attribute and the value that was provided in "confirmChallenge". The content type header is set to "application/json".

The following is a non-normative example response to a Verify SET received via HTTP/1.1 POST and includes a JSON object containing the confirmation attribute and value.

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

{
  "challengeResponse":"ca2179f4-8936-479a-a76d-5486e2baacd7"
}
```

Figure 10: Example Response to Verify SET with Challenge

If the Event Receiver returns a non-matching value or an HTTP status other than a 200 series response, the Event Stream "state" SHALL be set to "fail". A declining Event Receiver MAY simply respond with any 400 series HTTP error (e.g. 404).

4. Control Plane - Management and Provisioning

This section describes how SCIM [RFC7644] and [RFC7643] MAY be used to add create, read, update, delete capability to the Control Plane to enable provisioning and operational management of Event Streams. In addition to provisioning of Event Streams, it can also be used by Event Receivers to change or reset the operational state of Event Streams such as pausing, stopping, or re-enabling after a failure.

SCIM is a protocol used by many security systems for provisioning and co-ordinating identities and other security subjects in cross-domain scenarios. SCIM is a RESTful profile of HTTP that is intended to be implemented by applications that need provisioning and management of security subjects and is ideal to the task of provisioning related security event signal systems. Examples of provisioning endpoints (SCIM service providers) include both Identity Providers and Relying Party applications (e.g. business and consumer web applications) as well as security and authorization infrastructure components.

[[Editors Note: At the time of writing, some groups feel a CRUD API is not required and participants would prefer to manage streams using an out-of-band workflow approach.]]

4.1. Event Stream Resource Type Definition

To extend SCIM to support Event Streams, requires defining an "EventStream" SCIM resource type, and implementing the corresponding RESTful HTTP operations to create, update, retrieve EventStream Resources. For SCIM service provider capability and schema discovery (see Sections 3 and 4 [RFC7644]).

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": []
}
```

The resource type above is discoverable in the "/ResourceTypes" 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]).

Figure 11: SCIM EventStream Resource Type Definition

To retrieve information about one or more Event Streams, authorized clients MAY query the "/EventStreams" endpoint as defined in Section 3.4 [RFC7644].

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 12: Example SCIM EventStream HTTP GET Request

The response below shows an example Feed resource that describes an available feed.

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

{
  "schemas":["urn:ietf:params:scim:schemas:event:2.0:EventStream"],
  "id":"767aad7853d240debc8e3c962051c1c0",
  "feedName":"OIDCLogoutFeed",
  "feedUri":
    "https://example.com/v2/Feeds/88bc00de776d49d5b535ede882d98f74",
  "methodUri":"urn:ietf:params:set:method:HTTP:webCallback",
  "deliveryUri":"https://notify.examplerp.com/Events",
  "aud":"https://sets.myexamplerp.com",
  "subStatus":"verify",
  "maxDeliveryTime":3600,
  "minDeliveryInterval":0,
  "description":"Logout events from oidc.example.com",
  "meta":{
     ... SCIM meta attributes ...
  }
}
```

Figure 13: Example EventStream HTTP GET Response

In the above example (Figure 13) the EventStream is for the the Feed "https://example.com/v2/Feeds/88bc00de776d49d5b535ede882d98f74". The current Event Stream state is "verify" which suggest the Event Stream Verification (see Section 3.4) process has not yet completed. Since there is no value for "feedJwk",) or "confidentialJwk", SETs will be sent without signing or encryption (plain text).

4.2. Creating A New Event Stream

To subscribe to a feed, the Event Receiver first obtains an authorization credential authorizing to to make the request (this process is out of scope of the specification but is often completed through OAuth). Upon obtaining authorization, the Event Receiver organization uses the SCIM Create operation (HTTP POST) as defined in Section 3.3 [RFC7644]. Event Transmitter's Control Plane service MAY have additional schema requirements for Event Stream creation which MAY be discovered using SCIM service configuration and schema discovery, see Section 4 [RFC7644].

In the following non-normative example, a new EventStream is created. Note that the Event Transmitter's control-plane automatically assigns the "id" attribute.

```
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",
  "feedUri":
    "https://example.com/v2/Feeds/88bc00de776d49d5b535ede882d98f74",
  "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 14: Example Create Event Stream Request

In following non-normative response, the Event service provider has automatically assigned a resource location as well as an "id". Usually upon creation, the initial value of "subStatus" is "pending" indicating that the Stream Verification process (see Section 3.4) has not been completed.

```
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",
  "feedUri":
    "https://example.com/v2/Feeds/88bc00de776d49d5b535ede882d98f74",
  "methodUri":"urn:ietf:params:set:method:HTTP:webCallback",
  "deliveryUri":"https://notify.examplerp.com/Events",
  "aud":"https://sets.myexamplerp.com",
  "subStatus":"verify",
  "maxDeliveryTime":3600,
  "minDeliveryInterval":0,
  "description":"Logout events from oidc.example.com",
  "meta":{
     ... SCIM meta attributes ...
  }
}
```

Figure 15: Example Response to Create EventStream Request

4.3. Updating An Event Stream

Periodically, Event Receivers MAY have need to update an Event Stream 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.

To modify an EventStream, an Event Receiver or authorized management client MAY use the HTTP PUT operation (see Section 3.5.1 [RFC7644])

or MAY use the HTTP PATCH operation (see Section 3.5.2 [RFC7644]) if supported by the Event Transmitter's control plane service. Note that HTTP PATCH enables more specific changes. This is particularly useful when updating multi-valued attributes that may contain large numbers of values. An example of this would be an EventStream that uses a "members" attribute to define the subjects of the Event Stream.

In the following non-normative example, the client is requesting that "subStatus" 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":"subStatus",
    "value":"paused"
  }]
}
```

Upon receiving the request, the Event Transmitter would stop sending Events to the Receiver. Note that while the request MAY seem complex it avoids the need for the requestor to have all of the current EventStream values in order to make a PUT request. In other words, an HTTP PATCH can be typically done in a single request response whereas an HTTP POST usually is preceded by an HTTP GET.

Figure 16: Example EventStream PATCH Request

5. Security Considerations

[TO BE COMPLETED]

6. IANA Considerations

6.1. 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

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, <http://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, <http://www.rfc-editor.org/info/rfc3986>.

[RFC5988] Nottingham, M., "Web Linking", RFC 5988, DOI 10.17487/RFC5988, October 2010, <http://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, <http://www.rfc-editor.org/info/rfc7231>.

[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>.

7.2. Informative References

[idevent-scim]
Oracle Corporation, "SCIM Event Extensions (work in progress)".

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

[RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <http://www.rfc-editor.org/info/rfc7515>.

[RFC7516] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, DOI 10.17487/RFC7516, May 2015, <http://www.rfc-editor.org/info/rfc7516>.

[RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <http://www.rfc-editor.org/info/rfc7517>.

[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, <http://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, <http://www.rfc-editor.org/info/rfc7644>.

[saml-core-2.0]
Internet2, "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0", March 2005.

Appendix A. 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 B. Change Log

Draft 00 - PH - First Draft based on reduced version of draft-hunt-idevent-distribution

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)

Authors' Addresses

Phil Hunt (editor)
Oracle Corporation

Email: phil.hunt@yahoo.com

Marius Scurtescu
Google

Email: mscurtescu@google.com