secevent
Internet-Draft
Intended status: Informational
Expires: December 31, 2017

M. Scurtescu
Google
A. Backman
Amazon

June 29, 2017

Management API for SET Event Streams
draft-scurtescu-secevent-simple-control-plane-00

Abstract

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

Status of This Memo

Copyright Notice

Table of Contents

```
   1.  Introduction  . . . . . . . . . . . . . . . . . . . . . . . .   2
   2.  Notational Conventions  . . . . . . . . . . . . . . . . . . .   3
   3.  Definitions . . . . . . . . . . . . . . . . . . . . . . . . .   3
   4.  Event Stream Management . . . . . . . . . . . . . . . . . . .   4
     4.1.  Stream Configuration  . . . . . . . . . . . . . . . . . .   4
       4.1.1.  Reading a Stream’s Configuration  . . . . . . . . . .   4
     4.2.  Subjects  . . . . . . . . . . . . . . . . . . . . . . . .   5
       4.2.1.  Adding a Subject to a Stream  . . . . . . . . . . . .   5
       4.2.2.  Removing a Subject  . . . . . . . . . . . . . . . . .   7
     4.3.  Verification  . . . . . . . . . . . . . . . . . . . . . .   8
       4.3.1.  Triggering a Verification Event.  . . . . . . . . . .   8
   5.  Normative References  . . . . . . . . . . . . . . . . . . . .  10
   Authors’ Addresses  . . . . . . . . . . . . . . . . . . . . . . .  10
```

1. Introduction

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

```
   +------------+                +------------+
   |            | Stream Status  |            |
   | Event      <----------------+ Event      |
   | Stream     |                | Receiver   |
   | Management | Add Subject    |            |
   | API        <----------------+            |
   |            |                |            |
   |            | Remove Subject |            |
   |            <----------------+            |
   |            |                |            |
   |            | Verification   |            |
   |            <----------------+            |
   |            |                |            |
   +------------+                +------------+
```

Figure 1: Event Stream Management API

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

2. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

3. Definitions

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

Event Stream
: An Event Stream is a configured relationship between a single Event Transmitter and a single Event Receiver, describing one or more methods by which the Event Transmitter may deliver SETs to the Event Receiver. Event Streams are unidirectional, with only one Event Transmitter and one Event Receiver. Event Transmitters support only one Event Streams for a single Event Receiver.

Event Stream Management Endpoint
: A URL hosted by the transmitter; it serves as the stream management API for a stream. An Event Transmitter MAY use a single Management Endpoint for multiple streams, provided that the transmitter has some mechanism through which they can identify the applicable stream for any given request, e.g. from authentication credentials. The definition of such mechanisms is outside the scope of this specification.

Add Subject Endpoint
: A URL hosted by the transmitter used to add subjects to an Event Stream.

Remove Subject Endpoint
: A URL hosted by the transmitter used to remove subjects from an Event Stream.

Verification Endpoint
: A URL hosted by the transmitter used to trigger a Verification Event to be sent to the receiver.

Event Stream Management API
: The API collectively made up by the four endpoints defined above.

Subject Identifier Object
: A JSON object containing a set of one or more claims about a subject that when taken together uniquely identify that subject. This set of claims SHOULD be declared as an acceptable way to

identify subjects of SETs by one or more specifications that profile [SET].

Verification Event
: A special event type for testing Event Streams. Receivers can request such an event through the Verification Endpoint. Transmitters can periodically send these events to ensure the connection is alive.

4. Event Stream Management

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

4.1. Stream Configuration

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

aud
: A string containing an audience claim as defined in JSON Web Token (JWT) [RFC7519] that identifies the Event Receiver for the Event Stream.

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

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

4.1.1. Reading a Stream’s Configuration

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

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

```
GET /set/stream HTTP/1.1
Host: transmitter.example.com
Authorization: Bearer eyJ0b2tlbiI6ImV4YW1wbGUifQo=
```

Figure 2: Example: Stream Status Request

The following is a non-normative example response:

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

{
  "aud": "http://www.example.com",
  "delivery": {
    "delivery_method": "https://schemas.example.com/set/http-push",
    "url": "https://receiver.example.com/events"
  },
  "events": [
    "https://schemas.openid.net/risc/event-type/account-at-risk",
    "https://schemas.openid.net/risc/event-type/account-deleted",
    "https://schemas.openid.net/risc/event-type/account-locked",
    "https://schemas.openid.net/risc/event-type/account-unlocked",
    "https://schemas.openid.net/risc/event-type/client-credentials-
        revoked",
    "https://schemas.openid.net/risc/event-type/sessions-revoked",
    "https://schemas.openid.net/risc/event-type/tokens-revoked"
  ]
}
```

Figure 3: Example: Stream Status Response

4.2. Subjects

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

4.2.1. Adding a Subject to a Stream

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

The Event Transmitter MAY choose to silently ignore the request, for example if the subject has previously indicated to the transmitter that they do not want events to be transmitted to the Event Receiver. In this case, the transmitter MUST return an empty 200 OK response, and MUST NOT indicate to the receiver that the request was ignored.

Errors are signaled with HTTP staus codes as follows:

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

Table 1: Add Subject Errors

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

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

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

Figure 4: Example: Add Subject Request

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

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

Figure 5: Example: Add Subject Response

4.2.2. Removing a Subject

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

Errors are signaled with HTTP staus codes as follows:

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

Table 2: Remove Subject Errors

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

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

{
  "phone_number": "123-456-7890"
}
```

Figure 6: Example: Remove Subject Request

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

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

Figure 7: Example: Remove Subject Response

4.3. Verification

In some cases, the frequency of event transmission on an Event Stream will be very low, making it difficult for an Event Receiver to tell the difference between expected behavior and event transmission failure due to a misconfigured stream. Event Receivers can request that a verification event be transmitted over the Event Stream, allowing the receiver to confirm that the stream is configured correctly upon successful receipt of the event.

Verification requests have the following properties:

state
: OPTIONAL. An arbitrary string that the Event Transmitter MUST echo back to the Event Receiver in the verification event’s payload. Event Receivers MAY use the value of this parameter to correlate a verification event with a verification request.

4.3.1. Triggering a Verification Event.

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

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

Errors are signaled with HTTP staus codes as follows:

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

Table 3: Verification Errors

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

```
POST /set/verify HTTP/1.1
Host: transmitter.example.com
Authorization: Bearer eyJ0b2tlbiI6ImV4YW1wbGUifQo=
Content-Type: application/json; charset=UTF-8

{
  "state": "VGhpcyBpcyBhbiBleGFtcGxlIHN0YXRlIHZhbHVlLgo="
}
```

Figure 8: Example: Trigger Verification Request

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

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

Figure 9: Example: Trigger Verification Response

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

```
{
  "jti": "123456",
  "iss": "https://transmitter.example.com",
  "aud": "receiver.example.com",
  "iat": "1493856000",
  "events": [
    "urn:ietf:params:secevent:event-type:core:verify" : {
      "state": "VGhpcyBpcyBhbiBleGFtcGxlIHN0YXRlIHZhbHVlLgo=",
    },
  ],
}
```

Figure 10: Example: Verification SET

5. Normative References

[DELIVERY]
"SET Token Delivery Using HTTP", n.d., <https://github.com/independentid/Identity-Events/blob/master/draft-hunt-secevent-delivery.txt>.

[JSON] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <http://www.rfc-editor.org/info/rfc7519>.

[SET] "Security Event Token (SET)", n.d., <https://tools.ietf.org/html/draft-ietf-secevent-token-01>.

Authors’ Addresses

Marius Scurtescu
Google

Email: mscurtescu@google.com

Annabelle Backman
Amazon

Email: richanna@amazon.com