

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 1, 2018

P. Hunt, Ed.
Oracle
M. Scurtescu
Google
M. Ansari
Cisco
A. Nadalin
Microsoft
A. Backman
Amazon
June 30, 2017

SET Token Delivery Using HTTP
draft-hunt-secevent-delivery-00

Abstract

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

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 1, 2018.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents

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

Table of Contents

1. Introduction and Overview	2
1.1. Notational Conventions	3
1.2. Definitions	3
2. SET Event Stream Protocol	5
2.1. Event Delivery Process	5
2.2. Push Delivery using HTTP	6
2.3. Polling Delivery using HTTP	8
2.3.1. Polling HTTP Request Attributes	9
2.3.2. Polling HTTP Response Attributes	10
2.3.3. Poll Request	10
2.3.4. Poll Response	14
2.4. Error Response Handling	16
2.5. Event Stream Verification	17
3. Authentication and Authorization	19
3.1. Use of Tokens as Authorizations	20
4. Security Considerations	20
4.1. Authentication Using Signed SETs	20
4.2. HTTP Considerations	20
4.3. TLS Support Considerations	21
4.4. Authorization Token Considerations	21
4.4.1. Bearer Token Considerations	21
5. Privacy Considerations	22
6. IANA Considerations	22
7. References	22
7.1. Normative References	22
7.2. Informative References	23
Appendix A. Other Streaming Specifications	25
Appendix B. Acknowledgments	26
Appendix C. Change Log	26
Authors' Addresses	27

1. Introduction and Overview

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

This specification defines to methods of SET delivery in what is known as Event Streams. The specification includes a verification process which tests and validates Event Stream configuration.

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

1.1. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "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

Relying Parties come in multiple forms generally classified as "Explicit" or "Implicit". An Explicit Relying Party is a service provider that accepts a standard security assertion (e.g. a JWT access tokens [RFC7519]) from an Identity Provider to establish a session or authorization. An Implicit Relying Party (implicit) uses a personal identifier such as an email address or telephone number from another provider to establish a Subject's identity. Examples of Explicit Relying Parties are defined in: OpenID Connect [openid-connect-core] and SAML2 [saml-core-2.0]. Implicit relying parties are verified by a common channel associated with the identifier. For example, an email or a text message is sent with a unique link to establish ownership of the identifier by the Subject.

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

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.

Event

An Event is defined to be an event as represented by a security event token (SET). See [I-D.ietf-secevent-token].

NumericDate

A JSON numeric value representing the number of seconds from 1970-01-01T00:00:00Z UTC until the specified UTC date/time, ignoring leap seconds. This is equivalent to the IEEE Std 1003.1, 2013 Edition [POSIX.1] definition "Seconds Since the Epoch", in which each day is accounted for by exactly 86400 seconds, other than that non-integer values can be represented. See [RFC3339] for details regarding date/times in general and UTC in particular.

2. SET Event Stream Protocol

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

2.1. Event Delivery Process

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

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:

- o The Event Transmitter uses an HTTP/1.1 POST to the Event Receiver endpoint to deliver the SET;
- o The Event Transmitter queues up the SET in a buffer so that an Event Receiver MAY poll for SETs using HTTP/1.1 POST.
- o Or, the Feed Transmitter delivers the Event through a different method not defined by this specification.

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

PUSH

In which SETs are delivered one at a time using HTTP POST requests by an Event Transmitter to an Event Receiver. The HTTP request body is a JSON Web Token [RFC7519] with a "Content-Type" header of "application/secevent+jwt" as defined in Section 2.2 and 6.2 of

[I-D.ietf-secevent-token]. Upon receipt, the Event Receiver acknowledges receipt with an HTTP response which is a JSON document with a "Content-Type" header of "application/json" (see Section 11 of [RFC7159]) as described below in Section 2.2.

POLLING Where multiple SETs are delivered in a JSON document [RFC7159] to an Event Receiver in response to an HTTP POST request to the Event Transmitter. Then in a following request, the Event Receiver acknowledges received SETs and MAY poll for more. In POLLING mode, all requests and responses are JSON documents and use a "Content-Type" of "application/json" as described in Section 2.3.

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

Transmitted SETs SHOULD be self-validating (e.g. signed) if there is a requirement to verify they were issued by the Event Transmitter at a later date when de-coupled from the original delivery where authenticity could be checked via the HTTP or TLS mutual authentication.

Upon receiving a SET, the Event Receiver reads the SET and validates it. The receiver MUST acknowledge receipt to the Event transmitter, using the defined acknowledgement or error method depending on the method of transfer.

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

2.2. Push Delivery using HTTP

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

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

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

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

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

POST /Events HTTP/1.1

Host: notify.examplerp.com

Accept: application/json

Authorization: Bearer h480djs93hd8

Content-Type: application/secevent+jwt
eyJhbGciOiJIub251In0

.

eyJwdWJsaXNoZXJvcmk0IjodHRwczovL3NjaW0uZXhhbXBsZS5jb20iLCJmZWV
kVXJpcyI6WyJodHRwczovL2podWIuZXhhbXBsZS5jb20vRmVlZHMvOThkNTI0Nj
FmYTViYmM4NzklOTNiNzclNCIsImh0dHBzOi8vamhlYi5leGFtcGxlLmNvbS9GZ
WVkey81ZDc2MDQ1MTZiMWQwODY0MWQ3Njc2ZWU3Il0sInJlc291cmNlVXJpcyI6
WyJodHRwczovL3NjaW0uZXhhbXBsZS5jb20vVXNlcnMvNDRmNjE0MmRmOTZiZDZ
hYjYxZTclMjFkOSJdLCJldmVudFR5cGVzIjpbIkNSRUFURSJdLCJhdHRYaWJldG
VzIjpbImk0IiwibmFtZSI6ImVzZXJOYWllIiwicGFzc3dvcmQiLCJlbWFPbHMiX
SwidmFsdWVzIjpbImVtYWlscyI6W3sidHlwZSI6IndvcmsiLCJ2YWx1ZSI6Impk
b2VAZXhhbXBsZS5jb20ifV0sInBhc3N3b3JkIjoibm90NHUybm8iLCJlc2VyTmF
tZSI6Impkb2UiLCJpZCI6IjQ0ZjYxNDJkZjk2YmQ2YWI2MWU3NTIxZDkiLCJuYW
llIjpbImdpdmVuTmFtZSI6IkpvaG4iLCJmYWlpbHl0YWllIjoirG91In19fQ

.

Figure 1: Example HTTP POST Request

Upon receipt of the request, the Event Receiver SHALL validate the JWT structure of the SET as defined in Section 7.2 [RFC7519]. The Event Receiver SHALL also validate the SET information as described in Section 2 [I-D.ietf-secevent-token].

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

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

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

HTTP/1.1 202 Accepted

Figure 2: Example Successful Delivery Response

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

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

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

The following is an example non-normative error response.

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

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

Figure 3: Example HTTP Status 400 Response

2.3. Polling Delivery using HTTP

This method allows an Event Receiver to use HTTP POST (Section 4.3.3 [RFC7231]) to acknowledge SETs and to check for and receive zero or more SETs. Requests MAY be made at a periodic interval (short polling) or requests MAY wait pending availability of new SETs using long polling (see Section 2 [RFC6202]).

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

- o The Event Receiver makes an request for available SETs using an HTTP POST to a pre-arranged endpoint provided by the Event Transmitter. Or,
- o After validating previously received SETs, the Event Receiver initiates another poll request using HTTP POST that includes acknowledgement of previous SETs, and waits for the next batch of SETs.

The purpose of the "acknowledgement" is to inform the Event Transmitter that has successfully been delivered and attempts to re-deliver are no longer required. Before acknowledgement, Event Receivers SHOULD ensure received SETs have been validated and retained in a manner appropriate to the receiver's retention requirements. The level and method of retention of SETs by Event Receivers is out-of-scope of this specification.

2.3.1. Polling HTTP Request Attributes

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

The request payloads are delivered in one of two forms as described in Section 2.3.3 and Section 2.3.4

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

The following JSON Attributes are used in a polling request:

Request Processing Parameters

maxEvents

an OPTIONAL JSON integer value indicating the maximum number of unacknowledged SETs that SHOULD be returned. If more than the maximum number of SETs are available, the oldest SETs available SHOULD be returned first. A value of "0" MAY be used by Event Receivers that would like to perform an acknowledge only request. This enables the Receiver to use separate HTTP requests for acknowledgement and reception of SETs. When zero returned events is requested, the value of the attribute "returnImmediately" SHALL be ignored as an immediate response is expected.

returnImmediately

An OPTIONAL JSON boolean value that indicates the Event Transmitter SHOULD return an immediate response even if no

results are available (short polling). The default value is "false" indicates the request is to be treated as an HTTP Long Poll (see Section 2 [RFC6202]). The time out for the request is part of the Stream configuration which is out of scope of this specification.

SET Acknowledgment Parameters

ack

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

setErrs

A JSON Object that contains one or more nested JSON attributes that correspond to the "jti" of each invalid SET received. The value of each is a JSON object whose contents is an "err" attribute and "description" attribute whose value correspond to the errors described in Section 2.4.

2.3.2. Polling HTTP Response Attributes

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

sets

A JSON object that contains zero or more nested JSON attributes. Each nested attribute corresponds to the "jti" of a SET to be delivered and whose value is a JSON String containing the value of the encoded corresponding SET. If there are no outstanding SETs to be transmitted, the JSON object SHALL be empty.

moreAvailable

A JSON boolean value that indicates if more unacknowledged SETs are available to be returned.

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

2.3.3. Poll Request

The Event Receiver performs an HTTP POST (see Section 4.3.4 [RFC7231]) to a pre-arranged polling endpoint URI to check for SETs

that are available. Because the Event Receiver has no prior SETs to acknowledge, the "ack" and "errs" request parameters are omitted.

If after a period of time, negotiated between the Event Transmitter and Receiver, an Event Transmitter MAY re-issue SETs it has previously delivered. The Event Receiver SHOULD accept repeat SETs and acknowledge the SETs regardless of whether the Receiver believes it has already acknowledged the SETs previously. An Event Transmitter MAY limit the number of times it attempts to deliver a SET. Upon abandoning delivery of a SET, the Event Transmitter SHOULD have a method to notify the Event Receiver of the loss such as through a status service (not defined by this specification).

If the Event Receiver has received SETs from the Event Transmitter, the Event Receiver SHOULD parse and validate received SETs to meet its own requirements and SHOULD acknowledge receipt in a timely (e.g. minutes) fashion so that the Event Transmitter may mark the SETs as received. Event Receivers SHOULD acknowledge receipt before taking any local actions based on the SETs to avoid unnecessary delay in acknowledgement where possible.

Poll requests have three variations:

Poll Only

In which an Event Receiver asks for the next set of Events where no previous SET deliveries are acknowledged (such as in the initial poll request).

Acknowledge Only

In which an Event Receiver sets the "maxEvents" attribute to "0" along with "ack" and "err" attributes indicating the Event Receiver is acknowledging previously received SETs and does not want to receive any new SETs in response to the request.

Combined Acknowledge and Poll

In which an Event Receiver is both acknowledging previously received SETs using the "ack" and "err" attributes and will wait for the next group of SETs in the Event Transmitters response.

2.3.3.1. Poll Only Request

In the case where no SETs were received in a previous poll (see Figure 10), the Event Receiver simply polls without acknowledgement parameters ("sets" and "setErrs").

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

The following is a non-normative example poll request to the endpoint: "https://nofity.exampleidp.com/Events".

```
POST /Events HTTP/1.1

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

{
  "returnImmediately":true
}
```

Figure 4: Example Initial Poll Request

An Event Receiver MAY poll with no parameters at all by passing an empty JSON object.

The following is a non-normative example default poll request to the endpoint: "https://nofity.exampleidp.com/Events".

```
POST /Events HTTP/1.1

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

{}
```

Figure 5: Example Default Poll Request

2.3.3.2. Acknowledge Only Request

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

This variation is typically used when an Event Receiver needs to acknowledge received SETs independently (e.g. on separate threads) from the process of receiving SETs.

The following is a non-normative example poll with acknowledgement of SETs received (for example as shown in Figure 9).

```
POST /Events HTTP/1.1

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

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

Figure 6: Example Acknowledge Only equest

2.3.3.3. Poll with Acknowledgement

This variation allows a receiver thread to simultaneously acknowledge previously received SETs and wait for the next group of SETs in a single request.

The following is a non-normative example poll with acknowledgement of SETs received in Figure 9.

```
POST /Events HTTP/1.1

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

{
  "ack":[
    "4d3559ec67504aaba65d40b0363faad8",
    "3d0c3cf797584bd193bd0fb1bd4e7d30"
  ],
  "returnImmediately":false
}
```

Figure 7: Example Poll With Acknowledgement and No Errors

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

2.3.3.4. Poll with Acknowledgement and Errors

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

The following is a non-normative example of a response acknowledging 1 error and 1 receipt of two SETs received in Figure 9.

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

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

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

Figure 8: Example Poll Acknowledgement With Error

2.3.4. Poll Response

In response to a poll request, the service provider MAY respond immediately if SETs are available to be delivered. If no SETs are available at the time of the request, the Event Transmitter SHALL delay responding until a SET is available unless the poll request parameter "returnImmediately" is "true".

As described in Section 2.3.2 a JSON document is returned containing a number of attributes including "sets" which SHALL contain zero or more SETs.

The following is a non-normative example response to the request shown Section 2.3.3. This example shows two SETs are returned.

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

[illegible]

Figure 9: Example Poll Response

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

The following is a non-normative example response to the request shown Section 2.3.3 showing no new SETs or unacknowledged SETs are available.

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

{
  "sets":{ }
}
```

Figure 10: Example No SETs Poll Response

Upon receiving the JSON document (e.g. as shown in Figure 9), the Event Receiver parses and verifies the received SETs and notifies the Event Transmitter via the next poll request to the Event Transmitter as described in Section 2.3.3.3 or Section 2.3.3.4.

2.4. Error Response Handling

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

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

Table 1: SET Errors

An error response SHALL include a JSON object which provides details about the error. The JSON object includes the JSON attributes:

```
err
```


A value which is a keyword that describes the error (see Table 1).

description

A human-readable text that provides additional diagnostic information.

When included as part of an HTTP Status 400 response, the above JSON is the HTTP response body (see Figure 3). When included as part of a batch of SETs, the above JSON is included as part of the "setErrs" attribute as defined in Section 2.3.2 and Section 2.3.3.4

2.5. Event Stream Verification

In the verify process, the Event Receiver organization initiates a request to the Event Transmitter to verify the Stream. The Event Receiver provides a "confirm" value and a "nonce" value that the Event Transmitter is expected to return in the body of a Verify Event so that the Event Receiver can confirm end-to-end configuration of SET delivery including proper signing and encryption depending on the configuration of the Stream. For example, can the Event Transmitter send a encrypted SET that the Receiver can decode? The method by which this is initiated is out-of-scope of this specification and MAY be provided by a profiling specification, or by administrative interfaces offered by the Event Transmitter.

To confirm an Event Stream configuration, the Event Transmitter SHALL send a Verify SET to the Event Receiver using the registered "methodUri" mechanism.

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 configuration.

aud

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

exp

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

confirm

The value given by the Event Receiver to the Event Transmitter to return in the Verify Event.

nonce

A value given by the Event Receiver or otherwise agreed up to return which SHOULD be unique to the Stream and SHOULD change with each test in order to distinguish tests uniquely.

If the Event Stream is configured to encrypt SETs for the Event Receiver, 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 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://event.example.com/Feeds/98d52461fa5bbc879593b7754"
  ],
  "[[this RFC URL]]#verify": {
    "confirm": "ca2179f4-8936-479a-a76d-5486e2baacd7",
    "nonce": "1668c993e95849869e4b3506cccdf9bf"
  }
}
```

Figure 11: Example Verification SET with Challenge

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 "confirm" and "nonce" are as expected. If they do not match, an error response of "setData" SHOULD be returned (see Section 2.4).

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

3. Authentication and Authorization

The SET delivery methods described in this specification are based upon HTTP and depend on the use of TLS and/or standard HTTP authentication and authorization schemes as per [RFC7235]. For example, the following methodologies could be used among others:

TLS Client Authentication

Event delivery endpoints MAY request TLS mutual client authentication. See Section 7.3 [RFC5246].

Bearer Tokens

Bearer tokens [RFC6750] MAY be used when combined with TLS and a token framework such as OAuth 2.0 [RFC6749]. For security considerations regarding the use of bearer tokens in SET delivery see Section 4.4.1.

Basic Authentication

Usage of basic authentication should be avoided due to its use of a single factor that is based upon a relatively static, symmetric secret. Implementers SHOULD combine the use of basic authentication with other factors. The security considerations of HTTP BASIC, are well documented in [RFC7617] and SHOULD be considered along with using signed SETs (see SET Payload Authentication below).

SET Payload Authentication

In scenarios where SETs are signed and the delivery method is HTTP POST (see Section 2.2), Event Receivers MAY elect to use Basic Authentication or not to use HTTP or TLS based authentication at all. See Section 4.1 for considerations.

As per Section 4.1 of [RFC7235], a SET delivery endpoint SHALL indicate supported HTTP authentication schemes via the "WWW-Authenticate" header.

Because SET Delivery describes a simple function, authorization for the ability to pick-up or deliver SETs can be derived by considering the identity of the SET issuer, or via an authentication method above. This specification considers authentication as a feature to prevent denial-of-service attacks. Because SETs are not commands (see), Event Receivers are free to ignore SETs that are not of interest.

For illustrative purposes only, SET delivery examples show an OAuth2 bearer token value [RFC6750] in the authorization header. This is not intended to imply that bearer tokens are preferred. However, the

use of bearer tokens in the specification does reflect common practice.

3.1. Use of Tokens as Authorizations

When using bearer tokens or proof-of-possession tokens that represent an authorization grant such as issued by OAuth (see [RFC6749]), implementers SHOULD consider the type of authorization granted, any authorized scopes (see Section 3.3 of [RFC6749]), and the security subject(s) that SHOULD be mapped from the authorization when considering local access control rules. Section 6 of the OAuth Assertions draft [RFC7521], documents common scenarios for authorization including:

- o Clients using an assertion to authenticate and/or act on behalf of itself;
- o Clients acting on behalf of a user; and,
- o A Client acting on behalf of an anonymous user (e.g., see next section).

When using OAuth authorization tokens, implementers MUST take into account the threats and countermeasures documented in the security considerations for the use of client authorizations (see Section 8 of [RFC7521]). When using other token formats or frameworks, implementers MUST take into account similar threats and countermeasures, especially those documented by the relevant specifications.

4. Security Considerations

4.1. Authentication Using Signed SETs

In scenarios where HTTP authorization or TLS mutual authentication are not used or are considered weak, JWS signed SETs SHOULD be used (see [RFC7515] and Security Considerations [I-D.ietf-secevent-token]). This enables the Event Receiver to validate that the SET issuer is authorized to deliver SETs.

4.2. HTTP Considerations

SET delivery depends on the use of Hypertext Transfer Protocol and thus subject to the security considerations of HTTP Section 9 [RFC7230] and its related specifications.

As stated in Section 2.7.1 [RFC7230], an HTTP requestor MUST NOT generate the "userinfo" (i.e., username and password) component (and

its "@" delimiter) when an "http" URI reference is generated with a message as they are now disallowed in HTTP.

4.3. TLS Support Considerations

SETs contain sensitive information that is considered PII (e.g. subject claims). Therefore, Event Transmitters and Event Receivers MUST require the use of a transport-layer security mechanism. Event delivery endpoints MUST support TLS 1.2 [RFC5246] and MAY support additional transport-layer mechanisms meeting its security requirements. When using TLS, the client MUST perform a TLS/SSL server certificate check, per [RFC6125]. Implementation security considerations for TLS can be found in "Recommendations for Secure Use of TLS and DTLS" [RFC7525].

4.4. Authorization Token Considerations

When using authorization tokens such as those issued by OAuth 2.0 [RFC6749], implementers MUST take into account threats and countermeasures documented in Section 8 of [RFC7521].

4.4.1. Bearer Token Considerations

Due to the possibility of interception, Bearer tokens MUST be exchanged using TLS.

Bearer tokens MUST have a limited lifetime that can be determined directly or indirectly (e.g., by checking with a validation service) by the service provider. By expiring tokens, clients are forced to obtain a new token (which usually involves re-authentication) for continued authorized access. For example, in OAuth2, a client MAY use OAuth token refresh to obtain a new bearer token after authenticating to an authorization server. See Section 6 of [RFC6749].

Implementations supporting OAuth bearer tokens need to factor in security considerations of this authorization method [RFC7521]. Since security is only as good as the weakest link, implementers also need to consider authentication choices coupled with OAuth bearer tokens. The security considerations of the default authentication method for OAuth bearer tokens, HTTP BASIC, are well documented in [RFC7617], therefore implementers are encouraged to prefer stronger authentication methods. Designating the specific methods of authentication and authorization are out-of-scope for the delivery of SET tokens, however this information is provided as a resource to implementers.

5. Privacy Considerations

If a SET needs to be retained for audit purposes, JWS MAY be used to provide verification of its authenticity.

Event Transmitters SHOULD attempt to specialize Event Streams so that the content is targeted to the specific business and protocol needs of subscribers.

When sharing personally identifiable information or information that is otherwise considered confidential to affected users, Event Transmitters and Receivers MUST have the appropriate legal agreements and user consent or terms of service in place.

The propagation of subject identifiers can be perceived as personally identifiable information. Where possible, Event Transmitters and Receivers SHOULD devise approaches that prevent propagation -- for example, the passing of a hash value that requires the subscriber to already know the subject.

6. IANA Considerations

There are no IANA considerations.

7. References

7.1. Normative References

- [I-D.ietf-secevent-token]
Hunt, P., Denniss, W., Ansari, M., and M. Jones, "Security Event Token (SET)", draft-ietf-secevent-token-00 (work in progress), January 2017.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<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>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<http://www.rfc-editor.org/info/rfc5246>>.

- [RFC5988] Nottingham, M., "Web Linking", RFC 5988, DOI 10.17487/RFC5988, October 2010, <<http://www.rfc-editor.org/info/rfc5988>>.
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <<http://www.rfc-editor.org/info/rfc6125>>.
- [RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <<http://www.rfc-editor.org/info/rfc7159>>.
- [RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <<http://www.rfc-editor.org/info/rfc7231>>.
- [RFC7517] Jones, M., "JSON Web Key (JWK)", RFC 7517, DOI 10.17487/RFC7517, May 2015, <<http://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, <<http://www.rfc-editor.org/info/rfc7519>>.
- [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <<http://www.rfc-editor.org/info/rfc7525>>.

7.2. Informative References

- [openid-connect-core] NRI, "OpenID Connect Core 1.0", Nov 2014.
- [POSIX.1] Institute of Electrical and Electronics Engineers, "The Open Group Base Specifications Issue 7", IEEE Std 1003.1, 2013 Edition, 2013.
- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <<http://www.rfc-editor.org/info/rfc3339>>.

- [RFC6202] Loreto, S., Saint-Andre, P., Salsano, S., and G. Wilkins, "Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP", RFC 6202, DOI 10.17487/RFC6202, April 2011, <<http://www.rfc-editor.org/info/rfc6202>>.
- [RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <<http://www.rfc-editor.org/info/rfc6749>>.
- [RFC6750] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, DOI 10.17487/RFC6750, October 2012, <<http://www.rfc-editor.org/info/rfc6750>>.
- [RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <<http://www.rfc-editor.org/info/rfc7230>>.
- [RFC7235] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Authentication", RFC 7235, DOI 10.17487/RFC7235, June 2014, <<http://www.rfc-editor.org/info/rfc7235>>.
- [RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <<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>>.
- [RFC7521] Campbell, B., Mortimore, C., Jones, M., and Y. Goland, "Assertion Framework for OAuth 2.0 Client Authentication and Authorization Grants", RFC 7521, DOI 10.17487/RFC7521, May 2015, <<http://www.rfc-editor.org/info/rfc7521>>.
- [RFC7617] Reschke, J., "The 'Basic' HTTP Authentication Scheme", RFC 7617, DOI 10.17487/RFC7617, September 2015, <<http://www.rfc-editor.org/info/rfc7617>>.
- [saml-core-2.0] Internet2, "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0", March 2005.

Appendix A. Other Streaming Specifications

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

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

XMPP Events

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

Amazon Simple Notification Service

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

Other information:

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

Apache Kafka

Apache Kafka is an Apache open source project based upon TCP for distributed streaming. It prescribes some interesting general

purpose features that seem to extend far beyond the simpler streaming model SECEVENTs is after. A comment from MS has been that Kafka does an acknowledge with poll combination event which seems to be a performance advantage. See: <https://kafka.apache.org/intro>

Google Pub/Sub

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

Information:

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

Appendix B. Acknowledgments

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

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

Appendix C. Change Log

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

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

draft-hunt-secevent.distribution revision history:

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

- o Draft 01 - PH -
 - * Reworked terminology to match new WG Transmitter/Receiver terms
 - * Reworked sections into Data Plane vs. Control Plane
 - * Removed method transmission registry in order to simplify the specification
 - * Made Create, Update operations optional for Control Plane (Read is MTI)
- o Draft 02 - PH
 - * Added iss metadata for Event Stream
 - * Changed to using JWKS_uri for issuer and receiver.
 - * Control Plane sections moved to draft-hunt-secevent-stream-mgmt
 - * Added support for delivering multiple events using HTTP POST polling

Authors' Addresses

Phil Hunt (editor)
Oracle Corporation

Email: phil.hunt@yahoo.com

Marius Scurtescu
Google

Email: mscurtescu@google.com

Morteza Ansari
Cisco

Email: morteza.ansari@cisco.com

Anthony Nadalin
Microsoft

Email: tonymad@microsoft.com

Annabelle Richard Backman
Amazon

Email: richanna@amazon.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 1, 2018

P. Hunt, Ed.
Oracle
M. Ansari
Cisco
June 30, 2017

SCIM Use Cases for SECEVENTS
draft-hunt-secevent-usecases-00

Abstract

This specification defines the SCIM use cases for the SECEVENTS working group.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 1, 2018.

Copyright Notice

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

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

Table of Contents

1. Introduction and Overview	2
1.1. Notational Conventions	3
1.2. Definitions	3
2. SCIM Background	3
3. High-Level Requirements	5
3.1. SCIM Event Trigger Requirements	5
3.2. SCIM Security Model Considerations	5
3.3. Control Plane Assumptions	6
3.4. Network and Protocol Operational Considerations	8
3.5. Dynamic Filtering Considerations	8
3.6. Directory and Application Provisioning	8
4. Use Cases	9
4.1. Scenario 1[P0]: Cloud-to-Enterprise PUSH and Cloud-to-Cloud PUSH	9
4.2. Scenario 2[P0]: Cloud-to-Enterprise POLLING	12
4.3. Scenario 3[P2]: Cloud-to-Mobile Application PUSH	16
5. Security Considerations	16
6. Privacy Considerations	17
7. IANA Considerations	17
8. References	17
8.1. Normative References	17
8.2. Informative References	17
Appendix A. Acknowledgments	18
Appendix B. Change Log	18
Authors' Addresses	19

1. Introduction and Overview

SCIM is a system intended for provisioning identities (such as enterprise users or consumers) and other objects across security domains to a cloud based service providers. SCIM defines an extensible JSON [RFC7643] document format and profiles HTTP protocol [RFC7644]. In practice, SCIM service providers are applications supporting pre-provisioning support, or may be a service provider directory upon which applications are integrated.

This document defines the operational requirements SCIM deployers have for the use of triggers, as defined in the SCIM Use Cases specification [RFC7642], and used in the form of security events and the requirements for management based on SCIM architectural assumptions.

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] .

This specification assumes terminology defined in the SCIM specifications, specifically [RFC7643] and [RFC7644]

This specification defines the following terms:

Directory

Defined as any centralized repository of security objects shared by multiple applications. A SCIM Directory, though not formally defined is simply a directory that supports SCIM protocol.

2. SCIM Background

The SCIM Core Schema specification [RFC7643] is a profile of JSON [RFC7159] that defines attribute types, mutability, data formats, composites, and multi-value attributes as well as SCIM Service Provider feature and schema discovery metadata. As core schema defines standard resource types: Users and Groups which are common to most service providers. Each resource type establishes a common set of attribute definitions that can be mapped to SAML [saml-core-2.0] and to OpenID Connect [openid-connect-core] as well as application specific attributes. The core schema specification provides an extension mechanism which has been popular in:

- o Being extended to describe many security objects such as OAuth Clients, Applications, IoT objects, among others.
- o Enabling localized extensions to standard resource types (e.g. Users) without compromising inter-operability of existing implementations.

The SCIM Protocol specification [RFC7644] describes a RESTful profile of HTTP [RFC7231] that defines create, read, update and delete life-cycle for resources. The processing rules follow Jon Postel's "Robustness Principle" (see Section 2.10 [RFC761]) which help avoid many of the failings of previous XML based approaches. In particular the use of robust RESTful JSON helped ensure client and server ability to deal with inter-domain differences in schema, data, and implementation avoiding a lot of per implementation/deployment custom connector approaches.

SCIM clients use HTTP requests to SCIM service providers as follows to:

- o Query for resources (users and groups) based on filters using HTTP GET or confidentially using HTTP POST.
- o Retrieve specific resources using HTTP GET.
- o Create new resources using HTTP POST.
- o Replace a resource using HTTP PUT.
- o Update a resource using HTTP PATCH. And,
- o Delete a resource using HTTP DELETE.

The SCIM Protocol defines capabilities for:

- o Complex or composite attributes that contain multiple values and the need to select and update specific values. This includes how to express sub-attributes and values in filters and the ability to change them as part of a resource. An example of a composite attribute in SCIM is: addresses (e.g. street name, city, country). Note: In SECEVENTs a corresponding example complex/composite attribute is an OpenID Connect user which is identified by both 'sub' and 'iss'.
- o How to handle attributes that are immutable or read-only in the context of operations like PUT. How to handle attributes that are hashed or write-only and cannot be retrieved.

- o Flexibility for web applications to take what they want without having traditional schema enforcement as with XML Schema.
- o How to handle identifiers between clients and service providers and across domains.
- o Referential stability of resources over time.

Some other relevant information:

- o SCIM Polling Draft from Craig McMurtry [I-D.mcmurtry-scim-polling]
- o Early SCIM Events proposal [I-D.hunt-idevent-scim]

3. High-Level Requirements

3.1. SCIM Event Trigger Requirements

SCIM's need for Security Events arises from a requirement for triggers identified in the SCIM Use Case specification [RFC7642]. Clients and service providers that operate across security domains have independent resource management that causes co-ordination and governance challenges between domains. The use of triggers is intended to alert clients (e.g. enterprises) of state changes within service providers that may be of interest to SCIM clients that may need to be co-ordinated or reconciled across domains.

As a general example, a change to a resource that occurs within a cloud software as a service (SaaS) provider generates an Event to be sent to a registered recipient via an Event Stream. Upon receipt of the event, the receiver performs a SCIM GET to obtain additional information and then decide if a local update or other action is required.

3.2. SCIM Security Model Considerations

Authentication and Authorization

SCIM follows normal authentication and authorization practices for HTTP (See Sections 2 and 7 [RFC7644]). In typical deployed cases, access to SCIM endpoints is managed by OAuth authorization in both cross-domain provisioning, delegated administration, and self-service applications. Many integrators also support basic authentication, and TLS mutual authentication. SCIM is often accessed in a couple of ways:

- * End-user servers (e.g. as facilitated via a /Me endpoint) via a self-service web application or Javascript client.

- * Administrative - where an administrator identity has access to groups of objects they are entitled to administer.
- * Server-to-server, where identity provisioning systems implementing management workflows initiate commands across domains using OAuth enabled authorization.

PII Confidentiality

Querying using personally identifiable information (PII) causes privacy concerns when using HTTP GET. In typical HTTP usage, since HTTP [RFC7231] does not allow for query payloads on an HTTP GET, query parameters and filters are typically passed as part of the URL. When queries contain PII (most will in the case of RISC), there are security issues (e.g. leakage via audit logs and browser histories) relating to passing filter terms that contain PII in URLs. See [RFC7642] Security Considerations, section 7.5.2. From the perspective of SECEVENTs, the SCIM community has the same PII requirement that the management of SECEVENT streams and delivery not pass PII in request URIs.

Scale, PII, and Multi-Valued Data

One of the concerns the SCIM working group had when developing SCIM was the challenge that Groups (e.g. a group of users) will tend to get very big at Internet scale. The bigger a Group gets, the more expensive it is to enumerate. With a high change rate it quickly become impractical to do a simple PUT to replace an entire Group object due to the likely number of independent update conflicts that would occur. To avoid this, implementers often:

- * Severely restrict when clients are actually authorized to return large objects (million member groups).
- * Set access policy to allow search filters that confirm membership but avoid returning the members attribute (to avoid enumeration of all values).
- * Use HTTP PATCH (a derivative of JSON Patch) to remove or add specific subjects without having to know the entire contents (e.g. the group).

3.3. Control Plane Assumptions

In the original SCIM identity event proposals, "Control Plane" functionality was accomplished by SCIM. SCIM protocol was proposed to configure and provision "streams" that deliver events via other protocols or profiles. The SCIM proposal allowed Event Receivers to check for delivery problems by retrieving Stream "resources" (which contain the stream configuration attributes) of which "status" is an

attribute that could be used to report operational state of a stream. Updates to Stream resource enable Event Recipients to do things like rotate credentials, or suspend streams. To initiate a verification to test a stream is functional, the Receiver or an authorized administrator can modify the Stream resource to "request" a verify by changing the value of "status" to "verify". In SCIM the subjects in a stream can be identified by a number of methods:

- o Members of a Group
- o The addition of a "streams" attribute to Users and other objects that may be part of a stream.
- o An attribute or filter condition. E.g. the members of a Stream are defined by those Users with entitlements or roles containing a specific value (e.g. "entitlements" eq "CRM").

The SCIM WG in re-using SCIM as the control plane had assumed the following is already defined (and any alternative proposal would have to support):

- o Defined processing of attributes based on type, mutability, etc for each HTTP method. For example, the handling of omitted attributes in a PUT or POST operation. Is a value intended to be defaulted or set to null?
- o Handling of extensibility semantics as defined in the SCIM specifications such as the definition of new resource types (objects) and addition of new attributes by other profiling specifications.
- o The ability of a service provider to override or modify client provider asserted values.
- o Identifier and resource URI stability and referential integrity.
- o Querying of subjects using various standard identifiers such as "id", "emails", "telephoneNumbers", etc. The ability to express composite queries such as "sub" and "iss" in a query.
- o Ability to add and remove subjects from a group while keeping enumeration of that group from the client. Ability to confirm membership in a group without enumeration (facilitated through support for write-only/compare-only schema or access control).
- o Standardized error control, handling and processing rules. See Section 3.12 [RFC7644] and [RFC7231].

3.4. Network and Protocol Operational Considerations

The SCIM WG discussed that transmission (now called data-plane or stream) can have much simpler semantics and error conditions and thus did not need to profile JSON beyond simple SET transfer (no need for attribute types, filters, etc). The SCIM WG also anticipated some varied requirements for delivery that include:

- o PUSH delivery via HTTP POST (the generally preferred ideal solution).
- o POLLING (to enable delivery across firewalls) using HTTP GET.
- o PUSH delivery via messaging systems like APNS, GMS, SMS, etc - many of these had to do with provisioning and entitlement signals for mobile applications (e.g. WebEx). For example user contacts synchronization where after a change to a user's contact list, an application can receive an Event notification through the mobile platform's messaging solution as a trigger to fetch changes.

3.5. Dynamic Filtering Considerations

When defining filtered Streams, SCIM has to consider some special cases when the contents of a Stream is based upon a filter (query) to define which affected resources are included. For example, if the contents of a Stream is defined as Events related to resources where "emails.value sw "A"" and a resource is deleted, then the deleted resource won't match the filter anymore but notification may still need to be sent.

3.6. Directory and Application Provisioning

Network relationships for connections are typically:

- o Enterprise Directory to Cloud Directory.
- o Cloud Directory to Cloud Directory.
- o Enterprise or Cloud to Cloud Application (applications used by many users).
- o Enterprise or Cloud to Mobile Application (applications running on a device controlled by a single user).

An enterprise directory is typically (but not always) legacy-LDAP. In the cloud, a directory is simply any shared centralized profile store (e.g. Google Dir, Azure Directory/OpenGraph, SCIM Directory, etc). Important: While for many organizations LDAP remains the

center of administrative control, it is important to note that cloud directories and applications hold significantly more PII than enterprise directories. This creates a challenge for enterprise organizations to ensure proper governance and management of data given that a lot of cloud data is independently managed and updated.

As with an enterprise directory, a cloud directory is often shared by multiple applications. Cloud directories not only contain entitlement information but now also contain CRM data, contact, credentials, personalization and localization data, social network data, etc (the list goes on). While some cloud providers centralize others are tenancy structured with different directory endpoints per tenancy (e.g. Oracle).

As described above, because data, particularly PII, is being independently managed across multiple domains, there is a need to generate change signals (events) from cloud based directories and applications back to the enterprise. This was originally identified in the SCIM Use Cases (see Section 2.2.1 [RFC7642]).

4. Use Cases

The following use cases are expressed in terms of the direction of flow of events. In typical SCIM cases, there is only 1-way event exchange. Typical usage of events is to act as a "trigger" (see [RFC7642]) to let a receiver know that an event has occurred in the transmitter's domain that may require action on the part of the receiver. Events can be simple resource changed events, to higher level account status and change events (e.g. account or password reset). While many events are similar to OpenID RISC proposed events, a major distinction is that SCIM events are often triggered by user, administrative, or workflow provisioning action rather than a risk analytical engine (e.g. that might detect suspicious activity).

4.1. Scenario 1[P0]: Cloud-to-Enterprise PUSH and Cloud-to-Cloud PUSH

Pre-conditions:

The Event Receiver already has SCIM access to the Event Transmitter service provider. This includes HTTP credentials and endpoint.

Event Receivers and Transmitters can agree out-of-band on SET/JWT security requirements including use of signing and/or encryption to be documented in a Stream Configuration.

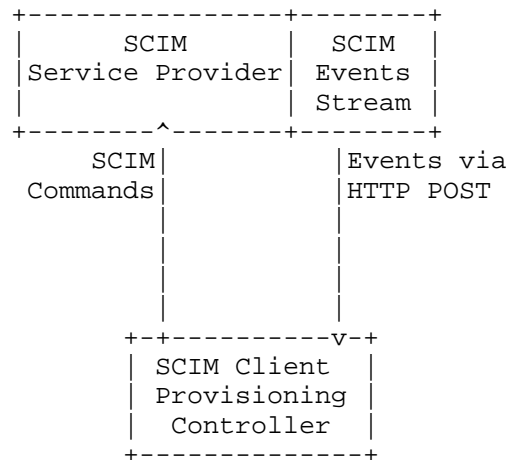


Figure 1: SCIM Provisioning with PUSH Triggers

In Figure 1, the SCIM client initiates RESTful SCIM commands to a SCIM service provider. In addition to provisioning security objects such as Users and Groups, the client also uses SCIM to provision Event Streams in order to receive Events to an endpoint the provisioning controller requests. The service provider MUST be able to POST to the client's domain. Usually this means the client is able to have a public HTTP endpoint available to receive SET events.

Stream Creation Flow:

To create a Stream, the Event Receiver (or an administrator) uses their SCIM access credential to access the SCIM endpoint and creates a Stream resource configuration:

```

POST /Streams
Host: scim.bighost.com
Authorization: Bearer h480djs93hd8
{ "receiverId": "<client-id>",
  "method": "webCallBack",
  "receiverUri": "https://set.example.com/events/",
  "aud": "<client-id>",
  "type": "SCIM",
  "receiverJwkUri": "<receiver's public key url>",
  "authorization": "<btoken|BasicAuth>"
}
  
```

Figure 2: Stream Creation Operation

Note: If the Transmitter does not have an HTTP credential to send events, the receiver should include one in its registration POST request or negotiate one out-of-band.

In the stream configuration there is likely a definition as to what types of events (event families) and which subjects constitute the feed. In SCIM this will likely be a group of objects, or filter condition such as "roles" eq "CRM_Users". This is likely based on the relationship between parties that determines which entities are provisioned between domains.

Upon successful creation of the Stream, the SCIM Event Transmitter Responds with:

```
HTTP/1.1 201 Created
Location: https://events.bighost.com/Streams/2819c223-7f76-453a
{ "receiverId": "<client-id>",
  "method": "webCallBack",
  "receiverUri": "https://set.example.com/events/",
  "aud": "<client-id>",
  "type": "SCIM",
  "receiverJwkUri": "<receiver's public key url>",
  "authorization": "<btoken|BasicAuth>",
  "status": "on"
}
```

Note that in the above figure, the Location URI is the fixed reference to the Stream for as long as it exists. Administrative users and Event Receiver entities MAY use the location to check status or update configuration as needed.

Figure 3: Stream Creation Response

[[TBD, the event receiver, needs to issue the event transmitter a credential in order for it to issue HTTP POSTs to the Event Receivers callback endpoint. In some cases there may be an existing OpenID Connect relationship but in most cases this not expected - especially in directory-to-directory synchronization scenarios.]]

Stream Verification:

During the initial stream creation request and at any point the transmitter deems appropriate (e.g. as a ping), the transmitter verifies configuration by sending a verification event to the receiver that demonstrates the receiver:

- o is willing accept the event, and

- o is able to parse the event - especially if encrypted.

Conversely an Event Receiver should be able to initiate a verification request and may provide a confirmation challenge and nonce to verify the relationship from the Event Receiver's perspective.

Delivery:

Delivery is accomplished by doing a simple HTTP POST to the registered endpoint of the receiver. The payload of the POST is application/jwt and contains a single JWT (which is actually a SET).

Before responding with a 2xx success message, the receiver should ensure it was able to read and validate the SET. If the transmitter receives a 2xx response, the transmitter may assume the event was successfully delivered.

A set of Status 400 error conditions are defined which the receiver can use to indicate various JWT validation conditions.

4.2. Scenario 2[P0]: Cloud-to-Enterprise POLLING

Pre-conditions:

The Event Receiver already has SCIM access to the Event Transmitter service provider. This includes HTTP credentials and endpoint.

Event Receivers and Transmitters can agree out-of-band on SET/JWT security requirements including use of signing and/or encryption to be documented in a Stream Configuration.

The Event Receiver is unable to open an endpoint to receive SETs inside the firewall.

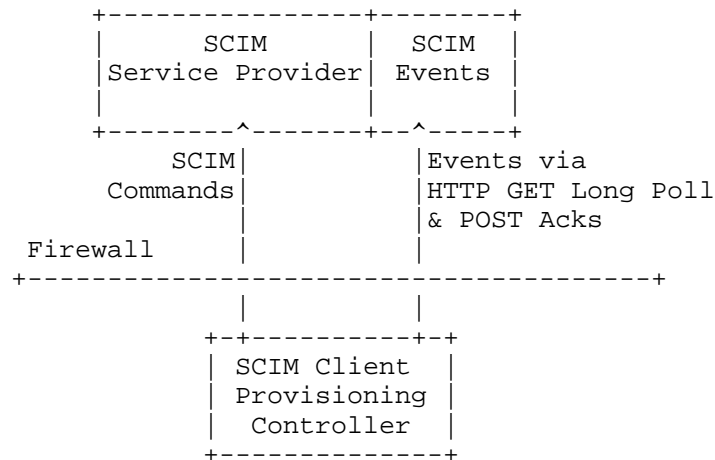


Figure 4: Event Delivery with Firewall

In Figure 4, the SCIM client initiates RESTful SCIM commands to a SCIM service provider. In addition to provisioning security objects such as Users and Groups, the client also uses SCIM to provision Event Streams in order to receive Events to an endpoint the provisioning controller requests. In this case, the SCIM Client "polls" for events using HTTP GET. The client MAY request immediate response based on a timed schedule, or the client MAY use HTTP Long Polling to wait for SETs as they become available.

Stream Creation Flow:

The Event Receiver uses their SCIM credential to access the SCIM service provider endpoint to create a Stream resource by performing a POST

```

POST /Streams
Host: scim.bighost.com
Authorization: Bearer h480djs93hd8
{ "receiverId": "<client-id>",
  "method": "POLLING",
  "aud": "<client-id>",
  "type": "SCIM",
  "receiverJwkUri": "<receiver's public key url>"
}
  
```

Figure 5: Create Polling Stream

It is assumed, but may not always be true. that the POLLING receiver can simply use their SCIM credential to perform HTTP GETs to the polling endpoint. Additional parameters will likely need to be defined to control polling rate, number of events in a message, etc.

Note, in the stream configuration there is likely a definition as to what types of events (event families) and which subjects constitute the feed. In SCIM this will likely be a group of objects, or filter condition such as "roles" eq "CRM_Users". This is likely based on the relationship between parties that determines which entities are provisioned between domains.

Upon successful creation of the stream, the transmitter responds to the receiver with:

```
HTTP/1.1 201 Created
Location: https://events.bighost.com/Streams/2819c223-7f76-453a
{ "receiverId":"<client-id>",
  "method":"POLLING",
  "receiverUri":"https://set.bighost.com/Events/2819c223-7f76-453a",
  "aud":"<client-id>",
  "type":"SCIM",
  "receiverJwkUri":"<receiver's public key url>",
  "status":"on"
}
```

Figure 6: Polling Stream Creation Response

In the above response, the transmitter indicates to the receiver where to poll for events by setting a value for "receiverUri". This endpoint does not need to be SCIM compliant and can be a generic (e.g. shared by all polliers) endpoint such as "https://events.bighost.com".

Stream Verification:

Same requirements are for Scenario 1 (see Section 4.1).

Delivery:

Delivery is accomplished by having the Event Receiver initiate an HTTP request that causes a response such as:

```
{
  "sets":{
    "4d3559ec67504aaba65d40b0363faad8":
      "eyJhbGciOiJub251In0
      .
      e3sgIAogICJqdGkiOiAiNGQzNTU5ZWZM2NzUwNGFhYmE2NWQ0MGiWmZyZmFhZDgiLAog
      ICJpYXQiOiAxNDU4NDk2NDA0LAogICJpc3MiOiAiaHR0cHM6Ly9zY2ltLmV4YW1wbGUu
      Y29tIiwgIAogICJhdWQiOiBbCiAgICJodHRwczovL3NjaW0uZXhhbXBsZS5jb20vRmVl
      ZHMvOTlkNTI0NjFmYTViYmM4NzklOTNiNzclNCIsCiAgICJodHRwczovL3NjaW0uZXhh
      bXBsZS5jb20vRmVlZHMvNWQ3NjA0NTE2YjFkMDg2NDZkNzY3NmVlNyIKICBdLCAgCiAg
      CiAgImV2ZW50cyI6IHsKICAgICJlcm46aWV0ZjpwYXJhbXM6c2NpbTpldmVudDpjcmVh
      dGUiOiB7CiAgICAgICJyZWYiOgogICAgICAgICJodHRwczovL3NjaW0uZXhhbXBsZS5j
      b20vVXNlcnMvNDRmNjE0MmRmOTZiZDZhYjYxZTclMjFkOSIsCiAgICAgICJhdHRYaWJl
      dGVzIjpbImlkIiwgIm5hbWUiLCaidXNlck5hbWUiLCaiaicGFzc3dvcmQiLCaiaZWlhaWxz
      Il0KICAgIH0KICB9Cn0",
    "<nextJti>": "<nextJwt>"
  },
  "since":1458496025
}
```

Figure 7: Example Polling Response

In the above JSON object is a JSON attribute "sets" whose value is a JSON object that contains a set of JSON attributes that correspond to each event's JTI value. the value for each attribute is the actual encoded SET.

In addition to the "sets" attribute, a "since" attribute indicates the timestamp of either the last event previously transmitted or potentially oldest event in the current payload (To be discussed).

In order to acknowledge receipt, the receiver must successfully parse each message and respond by doing an HTTP POST back to the events endpoint using something along the lines of the following JSON structure:

```
{
  "ack":[
    "39e48e70e9f84d90b5fdbf2fbd826219",
    "8e1ed13b871547ffa332f7027a0fdd91",
    "0a02c62529e34541a8b3c5c7941fa545"
  ]
  "setErrs":{
    "3d0c3cf797584bd193bd0fb1bd4e7d30":{
      "err":"dup",
      "description":"SET already received. Ignored."
    }
  }
}
```

Figure 8: Poll Acknowledgement Response

In the payload above the receiver indicates which SET event JTIs have been accepted, and which SETs had errors using "accepts" and "setErrs".

It is expected that because most errors are due to JWT crypto configuration errors, that most responses will tend to be all errors or all accepts.

If a transmitter receives what it deems an unrecoverable error, or a receiver fails to poll for events, the transmitter can set the stream state to "failed" with an appropriate error indicator.

4.3. Scenario 3[P2]: Cloud-to-Mobile Application PUSH

This scenario is a hybrid of scenario 1 and 2. The scenario uses mobile message delivery services (APNS, GMS, SMS) to deliver events. Typically a stream has only one subject in its feed. The events are used to notify client applications about changes to entitlements, or other configuration (e.g. new tenancy endpoints) that might be useful to user experience.

As in the polling method in Scenario 2, to acknowledge events, the mobile app will need to use the POST (as defined in Scenario 2) to acknowledge SET delivery. To be discussed, this might not be necessary if assured delivery is not required.

5. Security Considerations

None as this is a use case document to describe considerations.

6. Privacy Considerations

None as this is a use case document to describe considerations.

7. IANA Considerations

There are no IANA considerations.

8. References

8.1. Normative References

- [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>>.

8.2. Informative References

- [I-D.hunt-idevent-scim]
Hunt, P., Denniss, W., and M. Ansari, "SCIM Event Extension", draft-hunt-idevent-scim-00 (work in progress), March 2016.
- [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.
- [I-D.mcmurtry-scim-polling]
McMurtry, C., "SCIM Polling Protocol", draft-mcmurtry-scim-polling-01 (work in progress), April 2016.
- [idevent-scim]
Oracle Corporation, "SCIM Event Extensions (work in progress)".
- [openid-connect-core]
NRI, "OpenID Connect Core 1.0", Nov 2014.
- [RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <<http://www.rfc-editor.org/info/rfc3339>>.

- [RFC7159] 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>>.
- [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>>.
- [RFC761] USC, "TRANSMISSION CONTROL PROTOCOL", January 1980.
- [RFC7642] LI, K., Ed., Hunt, P., Khasnabish, B., Nadalin, A., and Z. Zeltsan, "System for Cross-domain Identity Management: Definitions, Overview, Concepts, and Requirements", RFC 7642, DOI 10.17487/RFC7642, September 2015, <<http://www.rfc-editor.org/info/rfc7642>>.
- [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 - Initial draft

Authors' Addresses

Phil Hunt (editor)
Oracle Corporation

Email: phil.hunt@yahoo.com

Morteza Ansari
Cisco

Email: moransar@cisco.com

Security Events Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 1, 2018

P. Hunt, Ed.
Oracle
W. Denniss
Google
M. Ansari
Cisco
M. Jones
Microsoft
June 30, 2017

Security Event Token (SET)
draft-ietf-secevent-token-02

Abstract

This specification defines the Security Event Token, which may be distributed via a protocol such as HTTP. The Security Event Token (SET) specification profiles the JSON Web Token (JWT), which can be optionally signed and/or encrypted. A SET describes a statement of fact from the perspective of an issuer that it intends to share with one or more receivers.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on January 1, 2018.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of

publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction and Overview	2
1.1. Notational Conventions	4
1.2. Definitions	4
2. The Security Event Token (SET)	5
2.1. Core SET Claims	8
2.2. Explicit Typing of SETs	10
2.3. Security Event Token Construction	10
3. Requirements for SET Profiles	12
4. Security Considerations	13
4.1. Confidentiality and Integrity	13
4.2. Delivery	13
4.3. Sequencing	13
4.4. Timing Issues	14
4.5. Distinguishing SETs from ID Tokens	14
4.6. Distinguishing SETs from Access Tokens	15
4.7. Distinguishing SETs from other kinds of JWTs	15
5. Privacy Considerations	16
6. IANA Considerations	16
6.1. JSON Web Token Claims Registration	16
6.1.1. Registry Contents	17
6.2. Media Type Registration	17
6.2.1. Registry Contents	17
7. References	18
7.1. Normative References	18
7.2. Informative References	19
Appendix A. Acknowledgments	20
Appendix B. Change Log	20
Authors' Addresses	22

1. Introduction and Overview

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

This specification profiles the use of JWT for the purpose of issuing security event tokens (SETs). This specification defines a base format upon which profiling specifications define actual events and their meanings. Unless otherwise specified, this specification uses non-normative example events intended to demonstrate how events may be constructed.

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

Security Events are not commands issued between parties. A security event is a statement of fact from the perspective of an issuer about the state of a security subject (e.g., a web resource, token, IP address, the issuer itself) that the issuer controls or is aware of, that has changed in some way (explicitly or implicitly). A security subject MAY be permanent (e.g., a user account) or temporary (e.g., an HTTP session) in nature. A state change could describe a direct change of entity state, an implicit change of state or other higher-level security statements such as:

- o The creation, modification, removal of a resource.
- o The resetting or suspension of an account.
- o The revocation of a security token prior to its expiry.
- o The logout of a user session. Or,
- o A cumulative conclusion such as to indicate that a user has taken over an email identifier that may have been used in the past by another user.

While subject state changes are often triggered by a user-agent or security-subsystem, the issuance and transmission of an event often occurs asynchronously and in a back-channel to the action which caused the change that generated the security event. Subsequently, an Event Receiver, having received a SET, validates and interprets the received SET and takes its own independent actions, if any. For example, having been informed of a personal identifier being associated with a different security subject (e.g., an email address is being used by someone else), the Event Receiver may choose to ensure that the new user is not granted access to resources associated with the previous user. Or, the Event Receiver may not have any relationship with the subject, and no action is taken.

While Event Receivers will often take actions upon receiving SETs, security events MUST NOT be assumed to be commands or requests. The intent of this specification is to define a way of exchanging statements of fact that subscribers may interpret for their own purposes. As such, SETs have no capability for error signaling other than to ensure the validation of a received SET.

1.1. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [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 document, all figures MAY contain spaces and extra line-wrapping for readability and space limitations. Similarly, some URIs contained within examples have been shortened for space and readability reasons.

1.2. Definitions

The following definitions are used with SETs:

Security Event Token (SET)

A SET is a JWT [RFC7519] that is distributed to one or more registered Event Receivers.

Event Transmitter

A service provider that delivers SETs to other providers known as Event Receivers.

Event Receiver

An Event Receiver is an entity that receives SETs through some distribution method.

Subject

A SET describes an event or state change that has occurred about a Subject. A Subject may be a principal (e.g., Section 4.1.2 [RFC7519]), a web resource, an entity such as an IP address, or the issuer itself that a SET might reference.

Profiling Specification A specification that uses the SET Token specification to define one or more event types and the associated claims included.

2. The Security Event Token (SET)

A SET is a data structure (in the form of a JWT [RFC7519]) representing one or more related security events about a Subject.

The schema and structure of a SET follows the JWT [RFC7519] specification. A SET has the following structure:

- o An outer JSON object that acts as the SET "envelope". The envelope contains a set of name/value pairs called the JWT Claims Set, typically common to every SET or common to a number of different Events within a single Profiling Specification or a related series of specifications. Claims in the envelope (the outer JSON structure) SHOULD be registered in the JWT Token Claims Registry Section 10.1 [RFC7519] or be Public Claims or Private Claims as also defined in [RFC7519].
- o Envelope claims that are profiled and defined in this specification are used to validate a SET and declare the contents of the event data included in the SET. The claim "events" identifies the event types expressed that are related to the Security Subject and MAY also include event-specific data.
- o Each JSON member of the "events" object is a name and value pair. The JSON attribute name is a URI String value that expresses an event type, and the corresponding value is a JSON object known as the event "payload". The payload JSON object contains claims typically unique to the event's URI type value and are not registered as JWT claims. These claims are defined by their associated Profiling Specification. An event with no payload claims SHALL be represented as the empty JSON object ("{}"). In many cases, one event URI expresses the primary event URI, while other events might be considered extensions that MAY be used to do things such as:
 - * A categorization event type to provide classification information (e.g., threat type or level).
 - * An enhancement of an existing specifications the arise over time.
 - * An extension needed to link a potential series of events.

- * Localized specific purpose event URI used between a particular Event Transmitter and Receiver.

The following is a non-normative example showing the JWT Claims Set for a hypothetical SCIM password reset SET. This example shows an additional events value ("https://example.com/scim/event/passwordResetExt") used to convey additional information -- in this case, the current count of reset attempts:

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

Figure 1: Example SCIM Password Reset Event

The event in the figure above expresses hypothetical password reset event for SCIM [RFC7644]. The JWT consists of:

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

In this example, the SCIM event indicates that a password has been updated and the current password reset count is 5. Notice that the

value for "resetAttempts" is actually part of its own JSON object associated with its own event URI attribute.

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

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

Figure 2: Example OpenID Back-Channel Logout Event

Note that the above SET has an empty JSON object and uses the JWT registered claims "sub" and "sid" to identify the subject that was logged-out.

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

```
{
  "jti": "fb4e75b5411e4e19b6c0fe87950f7749",

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

Figure 3: Example Consent Event

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

2.1. Core SET Claims

The following are claims that are based on [RFC7519] claim definitions and are profiled for use in an event token:

jti

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

iss

A single valued String containing the URI of the service provider publishing the SET (the issuer). This claim is REQUIRED. Note that when a SET is expressing an event about a Security Subject for which the SET issuer is not the issuer of the Security Subject, the conflict SHALL be resolved by including the Security Subject "iss" value within the event "payload" (see "events" claim).

aud

The syntax of the claim is as defined in Section 4.1.3 [RFC7519]. This claim contains one or more audience identifiers for the SET. This claim is RECOMMENDED.

iat

As defined by Section 4.1.6 [RFC7519], a value containing a NumericDate, which represents when the event was issued. Unless otherwise specified, the value SHOULD be interpreted as equivalent to the actual time of the event. This claim is REQUIRED.

nbf

Defined by Section 4.1.5 [RFC7519], a number whose value is a NumericDate. In the context of the SET token it SHALL be interpreted to mean a date in which the event is believed to have occurred (in the past) or will occur in the future. Note: there MAY be some cases where "nbf" is still smaller than "iat" such as when it took an extended time for a SET to be issued (for example after some analysis). This claim is OPTIONAL.

sub As defined by Section 4.1.2 [RFC7519], a String or URI value representing the principal or the subject of the SET. This is usually the entity whose "state" was changed. For example, an IP Address was added to a black list. A URI representing a user resource that was modified. A token identifier for a revoked token. If used, the Profile Specification SHOULD define the content and format semantics for the value. This claim is OPTIONAL, as the principal for any given profile may already be identified without the inclusion of a subject claim. Note that some SET profiles MAY choose to convey event subject information in the event payload, particularly if the subject information is relative to issuer information that is also conveyed in the event payload, which may be the case for some identity SET profiles.

exp As defined by [RFC7519], this claim is time on which the JWT MUST NOT be accepted for processing. In the context of a SET however, this notion does not apply since a SET reflects something that has already been processed and is historical in nature. While some specifications MAY have a need for this claim, its use in general cases is NOT RECOMMENDED.

The following are new claims defined by this specification:

events

The semantics of this claim is to define a set of event statements that each MAY add additional claims to fully describe a single logical event that has occurred (e.g. a state change to a subject). Multiple event statements of the same type SHALL NOT be accepted. The "events" claim SHOULD NOT be used to express multiple logical events.

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

txn

An OPTIONAL String value that represents a unique transaction identifier. In cases where multiple SETs are issued based on different event URIs, the transaction identifier MAY be used to correlate SETs to the same originating event or stateful change.

2.2. Explicit Typing of SETs

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

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

2.3. Security Event Token Construction

A SET is a JWT [RFC7519] that is constructed by building a JSON structure that constitutes an event object which is then used as the body of a JWT.

While this specification uses JWT to convey a SET, implementers SHALL NOT use SETs to convey authentication or authorization assertions.

The following is an example JWT Claims Set for a security event token (which has been formatted for readability):

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

Figure 4: Example Event Claims

When transmitted, the above JSON body must be converted into a JWT as per [RFC7519].

The following is an example of a SCIM Event expressed as an unsecured JWT. The JOSE Header is:

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

Base64url encoding of the octets of the UTF-8 representation of the JOSE Header yields:

```
eyJ0eXAiOiJzZW5ldmVudCtqd3QiLCJhbGciOiJub251In0
```

The example JWT Claims Set is encoded as follows:

```
eyJqdGkiOiI0ZDMlNTllYzY3NTA0YWFiYTY1ZDQwYjAzNjNmYWVkbGciOiJub251In0
ODQ5NjQwNCwiaXNzIjoiaHR0cHM6Ly9zY2ltLmV4YW1wbGUuY29tIiwiaXVkbGciOiJub251In0
dHBzOi8vc2NpbS5leGFtcGxlLmNvbS9GZWVkcY85OGQ1MjQ2MWZhNWJiYzg3OTU5M2I3
NzU0IiwiaHR0cHM6Ly9zY2ltLmV4YW1wbGUuY29tL0ZlZWRzLzVhbnZyYWN0UxNmIjZDA4
NjQxZDc2NzZlZTciXSwiZXZlbnRzIjp7InVybjppZXRmOnBhcmFtczpzY2ltOmV2ZW50
OmNyZWV0ZSI6eyJyZWYiOiJodHRwczovL3NjaW0uZXhhbXBsZS5jb20vVXNlcnMvNDRm
NjE0MmRmOTZiZDZhYjYxZTclMjFkOSIsImF0dHJpYnV0ZXMiOlsiaWQiLCJyZWllIiwiaXVkbGciOiJub251In0
dXNlck5hbWUiLCJwYXNzd29yZCI6ImVtYWlscyJdfX19
```

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

```
eyJ0eXAiOiJzZW5ldmVudCtqd3QiLCJhbGciOiJub251In0.eyJqdGkiOiI0ZDM1NTllYzY3NTA0YWFiYTY1ZDQwYjAzNjNmYWFKOCIsImh0dCI6MTQ1ODQ5NjQwNCwiaXNzIjoiaHR0cHM6Ly9zY2ltLmV4YW1wbGUuY29tIiwiaXVkiJpbImh0dHBzOi8vc2NpbS5leGFtcGxlLmNvbS9GZWVkcyc85OGQ1MjQ2MWZhNWJiYzg3OTU5M2I3NzU0IiwiaHR0cHM6Ly9zY2ltLmV4YW1wbGUuY29tL0ZlZWRzLzVkNzYwNDUxNmIxZDA4NjQxZDc2NzZlZTciXSwiZXZlbnRzIjp7InVybjppZXRmOnBhcmFtczpzY2ltOmV2ZW50OmNyZWFOZSI6eyJyZWYiOiJodHRwczovL3NjaW0uZXhhbXBsZS5jb20vVXNlcnMvNDRmNjE0MmRmOTZiZDZhYjYxZTclMjFkOSIsImF0dHJpYnV0ZXMiOlsiaWQiLCJyZWl1IiwidXNlck5hbWUiLCJwYXNzd29yZCIsImVtYWlscyJdfX19.
```

Figure 5: Example Unsecured Security Event Token

For the purpose of a simpler example in Figure 5, an unsecured token was shown. When SETs are not signed or encrypted, the Event Receiver MUST employ other mechanisms such as TLS and HTTP to provide integrity, confidentiality, and issuer validation, as needed by the application.

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

3. Requirements for SET Profiles

Profile Specifications for SETs define the syntax and semantics of SETs conforming to that SET profile and rules for validating those SETs. The syntax defined by profiling specifications includes what claims and event payload values are used by SETs utilizing the profile.

Defining the semantics of the SET contents for SETs utilizing the profile is equally important. Possibly most important is defining the procedures used to validate the SET issuer and to obtain the keys controlled by the issuer that were used for cryptographic operations used in the JWT representing the SET. For instance, some profiles may define an algorithm for retrieving the SET issuer's keys that uses the "iss" claim value as its input.

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

4. Security Considerations

4.1. Confidentiality and Integrity

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

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

Security Events distributed without authentication over the channel, such as via TLS ([RFC5246] and [RFC6125]), and/or OAuth 2.0 [RFC6749], or Basic Authentication [RFC7617], MUST be signed using JWS [RFC7515] so that individual events can be authenticated and validated by the Event Receiver.

4.2. Delivery

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

4.3. Sequencing

As defined in this specification, there is no defined way to order multiple SETs in a sequence. Depending on the type and nature of SET event, order may or may not matter. For example, in provisioning, event order is critical -- an object could not be modified before it was created. In other SET types, such as a token revocation, the order of SETs for revoked tokens does not matter. If however, the event was described as a log-in or logged-out status for a user subject, then order becomes important.

Profiling Specifications and implementers SHOULD take caution when using timestamps such as "iat" to define order. Distributed systems

will have some amount of clock-skew and thus time by itself will not guarantee order.

Specifications profiling SET SHOULD define a mechanism for detecting order or sequence of events. For example, the "txn" claim could contain an ordered value (e.g., a counter) that the issuer defines.

4.4. Timing Issues

When SETs are delivered asynchronously and/or out-of-band with respect to the original action that incurred the security event, it is important to consider that a SET might be delivered to a Subscriber in advance or well behind the process that caused the event. For example, a user having been required to logout and then log back in again, may cause a logout SET to be issued that may arrive at the same time as the user-agent accesses a web site having just logged-in. If timing is not handled properly, the effect would be to erroneously treat the new user session as logged out. Profiling Specifications SHOULD be careful to anticipate timing and subject selection information. For example, it might be more appropriate to cancel a "session" rather than a "user". Alternatively, the specification could use timestamps that allows new sessions to be started immediately after a stated logout event time.

4.5. Distinguishing SETs from ID Tokens

Because [RFC7519] states that "all claims that are not understood by implementations MUST be ignored", there is a consideration that a SET token might be confused with ID Token [OpenID.Core] if a SET is mistakenly or intentionally used in a context requiring an ID Token. If a SET could otherwise be interpreted as a valid ID Token (because it includes the required claims for an ID Token and valid issuer and audience claim values for an ID Token) then that SET profile MUST require that the "exp" claim not be present in the SET. Because "exp" is a required claim in ID Tokens, valid ID Token implementations will reject such a SET if presented as if it were an ID Token.

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

Note that the use of explicit typing, as described in Section 2.2, will not achieve disambiguation between ID Tokens and SETs, as the ID Token validation rules do not use the "typ" header parameter value.

4.6. Distinguishing SETs from Access Tokens

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

- o Prohibit use of the "exp" claim, as is done to prevent ID Token confusion.
- o Where possible, use a separate "aud" claim value to distinguish between the SET subscriber and the protected resource that is the audience of an access token.
- o Modify access token validation systems to check for the presence of the "events" claim as a means to detect security event tokens. This is particularly useful if the same endpoint may receive both types of tokens.
- o Employ explicit typing, as described in Section 2.2, and modify access token validation systems to use the "typ" header parameter value.

4.7. Distinguishing SETs from other kinds of JWTs

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

The most direct way to ensure that a SET is not confused with another kind of JWT is to have the JWT validation logic reject JWTs containing an "events" claim unless the JWT is intended to be a SET. This approach can be employed for new systems but may not be applicable to existing systems.

Another direct way to prevent confusion is to employ explicit typing, as described in Section 2.2, and modify applicable token validation

systems to use the "typ" header parameter value. This approach can be employed for new systems but may not be applicable to existing systems.

For many use cases, the simplest way to prevent substitution is requiring that the SET not include claims that are required for the kind of JWT that might be the target of an attack. For example, for [RFC8055], the "sip_callid" claim could be omitted and for [I-D.ietf-stir-passport], the "orig" claim could be omitted.

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

5. Privacy Considerations

If a SET needs to be retained for audit purposes, JWS MAY be used to provide verification of its authenticity.

Event Transmitters SHOULD attempt to specialize feeds so that the content is targeted to the specific business and protocol needs of subscribers.

When sharing personally identifiable information or information that is otherwise considered confidential to affected users, Event Transmitters and Receivers MUST have the appropriate legal agreements and user consent or terms of service in place.

The propagation of subject identifiers can be perceived as personally identifiable information. Where possible, Event Transmitters and Receivers SHOULD devise approaches that prevent propagation -- for example, the passing of a hash value that requires the subscriber to already know the subject.

6. IANA Considerations

6.1. JSON Web Token Claims Registration

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

6.1.1.1. Registry Contents

- o Claim Name: "events"
- o Claim Description: Security Event Object
- o Change Controller: IESG
- o Specification Document(s): Section 2 of [[this specification]]

- o Claim Name: "txn"
- o Claim Description: Transaction Identifier
- o Change Controller: IESG
- o Specification Document(s): Section 2 of [[this specification]]

6.2. Media Type Registration

6.2.1. Registry Contents

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

- o Type name: application
- o Subtype name: secevent+jwt
- o Required parameters: n/a
- o Optional parameters: n/a
- o Encoding considerations: 8bit; A SET is a JWT; JWT values are encoded as a series of base64url-encoded values (some of which may be the empty string) separated by period ('.') characters.
- o Security considerations: See the Security Considerations section of [[this specification]]
- o Interoperability considerations: n/a
- o Published specification: Section 2.2 of [[this specification]]
- o Applications that use this media type: TBD
- o Fragment identifier considerations: n/a
- o Additional information:
 - Magic number(s): n/a
 - File extension(s): n/a
 - Macintosh file type code(s): n/a

- o Person & email address to contact for further information: Michael B. Jones, mbj@microsoft.com
- o Intended usage: COMMON
- o Restrictions on usage: none
- o Author: Michael B. Jones, mbj@microsoft.com
- o Change controller: IESG
- o Provisional registration? No

7. References

7.1. Normative References

- [IANA.JWT.Claims]
IANA, "JSON Web Token Claims",
<<http://www.iana.org/assignments/jwt>>.
- [IANA.MediaTypees]
IANA, "Media Types",
<<http://www.iana.org/assignments/media-types>>.
- [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>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<http://www.rfc-editor.org/info/rfc5246>>.
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <<http://www.rfc-editor.org/info/rfc6125>>.
- [RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <<http://www.rfc-editor.org/info/rfc6749>>.
- [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>>.
- [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <<http://www.rfc-editor.org/info/rfc7525>>.

- [RFC7617] Reschke, J., "The 'Basic' HTTP Authentication Scheme", RFC 7617, DOI 10.17487/RFC7617, September 2015, <<http://www.rfc-editor.org/info/rfc7617>>.

7.2. Informative References

- [I-D.ietf-stir-passport]
Wendt, C. and J. Peterson, "Personal Assertion Token (PASSporT)", draft-ietf-stir-passport-11 (work in progress), February 2017.
- [I-D.sheffer-oauth-jwt-bcp]
Sheffer, Y., Hardt, D., and M. Jones, "JSON Web Token Best Current Practices", draft-sheffer-oauth-jwt-bcp-00 (work in progress), June 2017.
- [OpenID.Core]
Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and C. Mortimore, "OpenID Connect Core 1.0", November 2014, <http://openid.net/specs/openid-connect-core-1_0.html>.
- [RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types", RFC 2046, DOI 10.17487/RFC2046, November 1996, <<http://www.rfc-editor.org/info/rfc2046>>.
- [RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <<http://www.rfc-editor.org/info/rfc6838>>.
- [RFC7009] Lodderstedt, T., Ed., Dronia, S., and M. Scurtescu, "OAuth 2.0 Token Revocation", RFC 7009, DOI 10.17487/RFC7009, August 2013, <<http://www.rfc-editor.org/info/rfc7009>>.
- [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>>.

- [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>>.
- [RFC8055] Holmberg, C. and Y. Jiang, "Session Initiation Protocol (SIP) Via Header Field Parameter to Indicate Received Realm", RFC 8055, DOI 10.17487/RFC8055, January 2017, <<http://www.rfc-editor.org/info/rfc8055>>.
- [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 thank the members of the IETF SCIM working group, which began discussions of provisioning events starting with draft-hunt-scim-notify-00 in 2015.

The editors would like to thank the participants in the IETF id-event mailing list and related working groups for their support of this specification.

Appendix B. Change Log

From the original draft-hunt-idevent-token:

Draft 01 - PH - Renamed eventUris to events

Draft 00 - PH - First Draft

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

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

Draft 03 - PH

General edit corrections from Sarah Squire

Changed "event" term to "SET"

Corrected author organization for William Denniss to Google

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

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

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

Draft 04 - PH -

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

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

Draft 06 - PH -

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

Draft 07 -

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

Draft 08 - PH -

- o Added clarification to status of examples

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

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

Draft 01 - PH - Changes as follows:

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

Draft 02 - Changes are as follows:

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

Authors' Addresses

Phil Hunt (editor)
Oracle Corporation

Email: phil.hunt@yahoo.com

William Denniss
Google

Email: wdenniss@google.com

Morteza Ansari
Cisco

Email: morteza.ansari@cisco.com

Michael B. Jones
Microsoft

Email: mbj@microsoft.com
URI: <http://self-issued.info/>

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

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

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

Abstract

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

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 10, 2018.

Copyright Notice

Copyright (c) 2018 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

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

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

1. Introduction and Overview

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

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

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

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

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

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

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

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

1.1. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

For purposes of readability, examples are not URL encoded. Implementers MUST percent encode URLs as described in Section 2.1 of [RFC3986].

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

1.2. Definitions

The following definitions are used with SETs:

Security Event Token (SET)

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

SET Issuer

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

SET Recipient

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

Subject

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

Event Identifier

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

Event Payload

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

Profiling Specification

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

2. The Security Event Token (SET)

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

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

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

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

2.1. Illustrative Examples

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

2.1.1. SCIM Example

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

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

Figure 1: Example SCIM Password Reset Event

The JWT Claims Set usage consists of:

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

The SET contains two event payloads:

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

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

2.1.2. Logout Example

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

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

Figure 2: Example OpenID Back-Channel Logout Event

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

2.1.3. Consent Example

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

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

Figure 3: Example Consent Event

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

2.1.4. RISC Example

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

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

Figure 4: Example RISC Event

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

2.2. Core SET Claims

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

"iss" (Issuer) Claim

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

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

"iat" (Issued At) Claim

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

"jti" (JWT ID) Claim

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

"aud" (Audience) Claim

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

"sub" (Subject) Claim

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

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

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

"exp" (Expiration Time) Claim

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

The following new claims are defined by this specification:

"events" (Security Events) Claim

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

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

"txn" (Transaction Identifier) Claim

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

"toe" (Time of Event) Claim

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

2.3. Explicit Typing of SETs

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

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

2.4. Security Event Token Construction

This section describes how to construct a SET.

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

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

Figure 5: Example Event Claims

The JSON Claims Set is encoded per [RFC7519].

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

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

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

```
eyJ0eXAiOiJzZWNLdmVudCtqd3QiLCJhbGciOiJub25lIn0
```

The above example JWT Claims Set is encoded as follows:

[illegible]

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

```
eyJ0eXAiOiJzZWlnbmVudCtqd3QlLCJhbGciOiJub251In0.eyJqdGkiOiI0ZDMlNTllYyZyMjEwYWFkOCIsImhhbmCI6MTQ1ODQ5NjQwNCwiaXNzIjoiaHR0cHM6Ly9zY2ltLmV4YW1wbGUuY29tLiwiYXVkIjpjbImh0dHBzOi8vc2NpbS5leGFtcGxlLmNmVnBzS9GZVWkcy85OGQ1MjQ2MWZhNWJiYzg3OTU5M2I3NzU0IiwiaHR0cHM6Ly9zY2ltLmV4YW1wbGUuY29tL0ZlZWZRZlZVknZyYwNDUxNmIxZDA4NjQxZDc2NzZlZTciXSwiZXZlbnRzIjp7InVybjpppZXRmOnBhcmtFtczpzY2ltOmV2ZW50OmNyZWZ0ZSI6eyJyZWYiOiJodHRCzcovLzNjaW0uZXBhbXBzS5lb20vVXNlcmlcnMvNDRMNmE0MMhmRmOTZIzdzhYyYxZTclMjFkKOSisImF0dHJpYmV0ZXMiOlsiaWQiLCJuaW1lIiwidXNlcik5hbWUiLCJwYXNkd29yZCIsImVtYWIscyJdfX19.
```

Figure 6: Example Unsecured Security Event Token

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

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

3. Requirements for SET Profiles

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

Syntax

The syntax of the SETs defined, including:

Top-Level Claims

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

Event Payload

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

Semantics

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

Subject Identification

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

Validation

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

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

4. Preventing Confusion between SETs and other JWTs

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

4.1. Distinguishing SETs from ID Tokens

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

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

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

4.2. Distinguishing SETs from Access Tokens

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

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

4.3. Distinguishing SETs from other kinds of JWTs

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

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

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

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

[RFC8055], the "sip_callid" claim could be omitted and for [RFC8225], the "orig" claim could be omitted.

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

5. Security Considerations

5.1. Confidentiality and Integrity

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

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

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

5.2. Delivery

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

5.3. Sequencing

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

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

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

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

5.4. Timing Issues

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

5.5. Preventing Confusion

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

6. Privacy Considerations

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

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

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

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

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

7. IANA Considerations

7.1. JSON Web Token Claims Registration

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

7.1.1. Registry Contents

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

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

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

7.2. Structured Syntax Suffix Registration

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

7.2.1. Registry Contents

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

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

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

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

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

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

7.3. Media Type Registration

7.3.1. Registry Contents

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

- o Type name: application
- o Subtype name: secevent+jwt
- o Required parameters: n/a
- o Optional parameters: n/a
- o Encoding considerations: binary; A SET is a JWT; JWT values are encoded as a series of base64url-encoded values (some of which may be the empty string) separated by period ('.') characters.
- o Security considerations: See Section 5 of [[this specification]]
- o Interoperability considerations: n/a
- o Published specification: Section 2.3 of [[this specification]]
- o Applications that use this media type: Applications that exchange SETs
- o Fragment identifier considerations: n/a
- o Additional information:

Magic number(s): n/a

File extension(s): n/a

Macintosh file type code(s): n/a

- o Person & email address to contact for further information:
Michael B. Jones, mbj@microsoft.com
- o Intended usage: COMMON
- o Restrictions on usage: none
- o Author: Michael B. Jones, mbj@microsoft.com
- o Change controller: IESG
- o Provisional registration? No

8. References

8.1. Normative References

[IANA.JWT.Claims]

IANA, "JSON Web Token Claims",
<<http://www.iana.org/assignments/jwt>>.

[IANA.MediaType]

IANA, "Media Types",
<<http://www.iana.org/assignments/media-types>>.

- [IANA.StructuredSuffix]
IANA, "Structured Syntax Suffix",
<[https://www.iana.org/assignments/
media-type-structured-suffix/](https://www.iana.org/assignments/media-type-structured-suffix/)>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <<https://www.rfc-editor.org/info/rfc3629>>.
- [RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <<https://www.rfc-editor.org/info/rfc3986>>.
- [RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, DOI 10.17487/RFC5246, August 2008, <<https://www.rfc-editor.org/info/rfc5246>>.
- [RFC6125] Saint-Andre, P. and J. Hodges, "Representation and Verification of Domain-Based Application Service Identity within Internet Public Key Infrastructure Using X.509 (PKIX) Certificates in the Context of Transport Layer Security (TLS)", RFC 6125, DOI 10.17487/RFC6125, March 2011, <<https://www.rfc-editor.org/info/rfc6125>>.
- [RFC6749] Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <<https://www.rfc-editor.org/info/rfc6749>>.
- [RFC7515] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", RFC 7515, DOI 10.17487/RFC7515, May 2015, <<https://www.rfc-editor.org/info/rfc7515>>.
- [RFC7516] Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", RFC 7516, DOI 10.17487/RFC7516, May 2015, <<https://www.rfc-editor.org/info/rfc7516>>.
- [RFC7519] Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <<https://www.rfc-editor.org/info/rfc7519>>.

- [RFC7525] Sheffer, Y., Holz, R., and P. Saint-Andre, "Recommendations for Secure Use of Transport Layer Security (TLS) and Datagram Transport Layer Security (DTLS)", BCP 195, RFC 7525, DOI 10.17487/RFC7525, May 2015, <<https://www.rfc-editor.org/info/rfc7525>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.

8.2. Informative References

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

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

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

Appendix A. Acknowledgments

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

Appendix B. Change Log

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

From the original draft-hunt-idevent-token:

Draft 01 - PH - Renamed eventUris to events

Draft 00 - PH - First Draft

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

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

Draft 03 - PH

General edit corrections from Sarah Squire

Changed "event" term to "SET"

Corrected author organization for William Denniss to Google

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

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

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

Draft 04 - PH -

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

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

Draft 06 - PH -

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

Draft 07 -

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

Draft 08 - PH -

- o Added clarification to status of examples

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

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

Draft 01 - PH - Changes as follows:

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

Draft 02 - Changes are as follows:

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

Draft 03 - Changes are as follows:

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

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

Draft 04 - mbj - Changes were as follows:

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

Draft 05 - mbj - Changes were as follows:

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

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

Draft 06 - mbj - Changes were as follows:

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

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

Draft 08 - mbj - Changes were as follows:

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

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

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

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

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

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

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

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

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

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

Authors' Addresses

Phil Hunt (editor)
Oracle Corporation

Email: phil.hunt@yahoo.com

Michael B. Jones
Microsoft

Email: mbj@microsoft.com
URI: <http://self-issued.info/>

William Denniss
Google

Email: wdenniss@google.com

Morteza Ansari
Cisco

Email: morteza.ansari@cisco.com

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

M. Scurtescu
Google
June 29, 2017

Security Events RISC Use Cases
draft-scurtescu-secevent-risc-use-cases-00

Abstract

This document describes the RISC use cases for security events and helps with defining the requirements for token format and event distribution.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 31, 2017.

Copyright Notice

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

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

Table of Contents

1. Introduction	2
2. Definitions	2
3. Use Cases	2
3.1. Explicit IdP to RP	2
3.2. Explicit RP to IdP	3
3.3. Implicit IdP to RP	3
3.4. Implicit RP to IdP	4
3.5. Pseudo-implicit	4
3.6. Identity as a Service	4
3.7. Security as a Service	4
3.8. On-Premise RP	4
Author's Address	5

1. Introduction

2. Definitions

- o Transmitter - the entity that sends security events
- o Receiver - the entity that receives security events
- o IdP - Identity Provider, in most cases but not always this is the transmitter
- o RP - Relying Party, in most cases but not always this is the receiver
- o RISC - Risk and Incident Sharing and Coordination, see <http://openid.net/wg/risc/>
- o SCIM - System for Cross-domain Identity Management, see <http://www.simplecloud.info/>

3. Use Cases

3.1. Explicit IdP to RP

- o Transmitter: IdP
- o Receiver: RP

Simplest use case, IdPs send security events to relevant RPs.

RP can make control plane calls to the IdP and can authenticate with access tokens issued by IdP.

3.2. Explicit RP to IdP

- o Transmitter: RP
- o Receiver: IdP

The RP can also send RISC events back to IdP. We want to make it very easy for the RP to do that, no complicated registration steps and crypto of possible.

IdP can document well-known endpoint for data plane (where it receives events). RP can use access token when sending events on data plane and maybe does not need to sign SETs.

If RP is sophisticated and is exposing its own control plane then during RP stream registration with IdP (either manual or programmatic) it can advertise its own issuer and that issuer through .well-known can specify full transmitter functionality of RP.

3.3. Implicit IdP to RP

- o Transmitter: implicit IdP
- o Receiver: implicit RP

Example: Google and Amazon, Amazon account can be backed by gmail address. Amazon acts as implicit RP to Google in this case.

Google and Amazon need legal agreement, When Amazon account is created or updated with gmail address Amazon makes REST call to Google to enroll this new email address for RISC events. If enrollment succeeds then RISC events will flow bidirectionally (see next section, for simplicity only unidirectional is considered in this section).

Assumption: Amazon/RP is registered with Google/IdP as an OAuth 2 client and can use access tokens for control plane.

Open question: what are the implications of unverified email addresses?

Open question: discovery of hosted domains, how does Google know that example.com is managed by Oracle and that subject enrollment should be sent to them?

3.4. Implicit RP to IdP

- o Transmitter: implicit RP
- o Receiver: implicit IdP

No enrollment call is strictly necessary. The RP can start sending events to IdP as new identifiers show up.

3.5. Pseudo-implicit

Common email address or phone number used by two different RPs.

Example: Amazon and PayPal, both Amazon and PayPal each have an account with the same gmail address.

Mutual discovery by exchanging email address hashes.

Open question: legal and privacy implications

3.6. Identity as a Service

Example: Google Firebase, IdaaS manages large number of RPs and implements RP functionality on their behalf.

IdaaS should be able to manage SET distribution configuration for its RPs with a given IdP using the credentials already established between the RP and the IdP. Control plane operation to create/update stream allows that.

Assumption: IdaaS can impersonate RP at IdP (can obtain access token on behalf of RP)

3.7. Security as a Service

Similar to IdaaS described in previous section, but the service provider has its own set of credentials different from the credentials and RP is using. The SP cannot impersonate the RP at IdP. The IdP must define delegation rules and allow the SP to make requests on behalf of the RP.

3.8. On-Premise RP

The RP (receiver) is behind a firewall and cannot be reached through HTTP. The only way to deliver events is if the RP periodically polls an endpoint provided by the transmitter.

Author's Address

Marius Scurtescu
Google

Email: mscurtescu@google.com

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

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <http://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on December 31, 2017.

Copyright Notice

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

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of

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

Table of Contents

1. Introduction	2
2. Notational Conventions	3
3. Definitions	3
4. Event Stream Management	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.

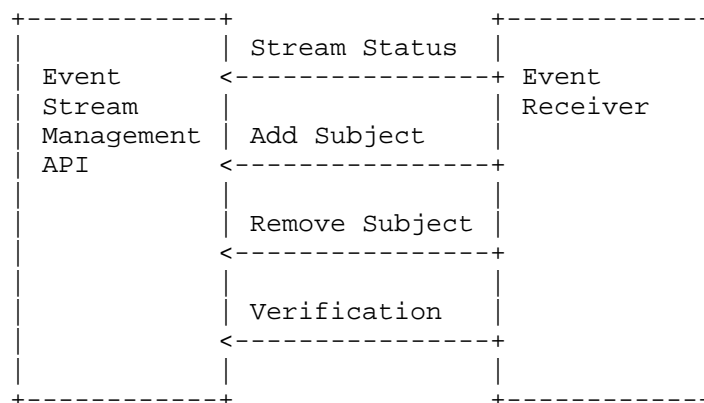


Figure 1: Event Stream Management API

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 eyJ0b2tldiI6ImV4YW1wbGUifQo=
```

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 status codes as follows:

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

Table 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 status codes as follows:

Code	Description
400	if the request body cannot be parsed or if the request is otherwise invalid
401	if authorization failed or it is missing
403	if the Event Receiver is not allowed to remove this particular subject
404	if the subject is not recognized by the Event Transmitter, the Event Transmitter may choose to stay silent in this case and respond with 204
429	if the Event Receiver is sending too many requests in a given 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 eyJ0b2tldiI6ImV4YW1wbGUifQo=

{
  "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 status codes as follows:

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

Table 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 eyJ0b2tlbiI6ImV4YWlwbGUifQo=
Content-Type: application/json; charset=UTF-8

{
  "state": "VGhpcyBpcyBhbiBleGFtcGx1IHNOYXRlIHZhbHVlLGo="
}
```

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": "VGhpcyBpcyBhbiBleGFtcGx1IHN0YXR1IHZhbHVlLgo=",
    },
  ],
}
```

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