Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: April 2, 2017

P. Hunt, Ed.
Oracle
M. Scurtescu
Google
M. Ansari
Cisco

September 29, 2016

SET Token Distribution and Subscription Management Profile
draft-hunt-idevent-distribution-01

Abstract

The specification defines how a subscriber to a feed of security events (SETs) may query for, subscribe and receive SETs from a security event feed. The specification defines a single mandatory-to-implement method using HTTP Post to deliver events to registered subscribers and a registry for new methods.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction and Overview

This specification defines a method by which SETs (see [I-D.hunt-idevent-token]) can be delivered by publishers to feed subscribers using HTTP POST [RFC7231] as well as an extension

registry enabling other methods of delivery. This specification also defines how subscribers MAY query for available Feeds, and manage event Subscriptions using SCIM [RFC7644].

The following diagram shows a typical SET Feed Provider and the services provided to Subscribers. Arrow heads point to the service provider (the direction of an HTTP request):

```
        +------------+                        +-------------+
        |            |Feeds Catalog           |             |
        |            <------------------------+             |
        |   SCIM     |                        |    SET      |
        |   Feed     |Subscription Request    |    Feed     |
        |   Mgmt     <------------------------+  Subscriber |
        |            |                        |             |
        |            |Subscription Mgmt       |             |
        |            <------------------------+             |
        |            |                        |             |
        +------------+                        |             |
        +------------+                        |             |
        |            |                        |             |
        |   FEED     |SET Delivery            |             |
        |            +------------------------>             |
        |  Provider  |                        |             |
        |            |                        |             |
        +------------+                        +-------------+
```

Figure 1: Subscription Management and Delivery

A SET Feed Provider MAY be directly integrated into a source service that generates events, or it may be a separate service entity that off-loads event distribution from the event generator to act as its delegated publisher. For the purposes of this specification, while SET distribution may be handled separately, this specification will consider the method for how event generators send events to publishers as out-of-scope.

The specification uses SCIM protocol [RFC7644] to advertise available Feeds and to enable Subscribers to request, subscriber to, and manage Subscriptions.

The specification defines a registry by which multiple SET delivery methods can be registered. The specification includes a web callback method which uses HTTP POST [RFC7231] to deliver SETs to Subscribers.

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.hunt-idevent-token].

The following definitions are specific to Identity Event publishing:

Feed Provider
   The Feed Provider publishes SETs to be distributed to registered subscribers.

Feed
   A Feed is a URI that describes the set of resources and events under which events may be issued. An interested subscriber registers with the feed provider to subscribe to an event URI to receive SETs associated with a Feed. An individual Feed MAY have zero or more Subscriptions.

Subscription
   A Subscription contains the information needed by a Feed Provider (e.g. delivery endpoints, credentials) to deliver a Feed of SETs to an individual Subscriber. A Subscription has ONE Feed.

Notification Mechanism
   A URI that describes the chosen event notification mechanism. When subscribing to a feed, a client may choose a specific mechanism by which it wishes to receive notification events.

Subscriber
A Subscriber is an party or security entity registers in the form of a Subscription to receive SETs from a feed provider that are part of a Feed.

2. Event Notification Process

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

How Feeds are defined and the process by which events are identified for subscribers is out-of-scope of this specification.

When a SET is available for a subscriber, the Feed Provider attempts to deliver the SET based on the Subscriber's registered delivery mechanism:

o The subscriber provided a web-callback endpoint, the publisher uses an HTTP/1.1 POST to the endpoint to deliver the event to the registered subscriber;

o Or, the Feed Provider delivers the event through a different method not defined by this specification.

After a SET is delivered to all subscribers, Feed Providers do not typically maintain SETs or histories. As such, published SETs SHOULD be self-validating (e.g. signed).

If delivery to any particular subscriber has been delayed for an extended period of time, the Feed Provider MAY suspend the subscription and even stop maintaining outstanding SETs for the Subscriber at its discretion and available resources. See subscription "state" in Section 4.1.

Upon receiving a SET, the Subscriber reads the token and validates it. Based on the content of the token, the subscriber decides what if any action it needs to take in response to the SET. For example, in response to a SCIM event [idevent-scim] indicating a changed resource, the subscriber might perform a SCIM GET request (see Section 3.4 [RFC7644]) to the affected resource URI in order to confidentially obtain the current state of the affected SCIM resource.

The action a Subscriber takes in response to a SET MAY be substantially different than merely copying the action of the publisher. A single SET MAY trigger multiple receiver actions. For

example, upon receiving notification that a user resource has been added to a group, the Subscriber may first determine that the user does not exist in the Subscriber's domain. The Subscriber translates the event into two actions:

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

2. The Subscriber then enables the user for the application associated with membership in the Feed Publisher's group.

3. Event Feeds

An Feed is defined by a "feedUri". The Feed provides a stream of SETs to be delivered to registered Subscribers based on a Subscription. An individual Subscription contains the metadata about a particular Subscriber regarding their subscription to a particular "feedUri". Subscription metadata indicates the current subscription state indicating whether all events are delivered, pending, or whether delivery has failed. Subscription metadata also describes the method of event delivery, and any associated security and configuration information (see Section 4.1).

3.1. Feed Types

A Feed Provider MAY define Feeds based on a number of criteria. This specification does not specify or limit the basis for which a service provider defines the resources or entities contained in a Feed or how feed URIs should be specified. Some possible methods for defining entities covered by a Feed include:

By Resource or Subject
: A resource or subject might have its own associated event notification Feed. For example, a User's mobile application may require notification of changes or rights defined in a SCIM User resource associated with the mobile user.

By Endpoint
: A Feed might be defined by an endpoint where any event relating to a resource within an endpoint is delivered to a subscriber. This type of feed is likely to have many notifications as the number of resources in an endpoint grows (e.g. a SCIM "/Users") and SHOULD be used with caution. Typically only privileged partners would be allowed to use this type of feed. For example, an enterprise wishes to be notified of all change events to any of its users assuming all users within the endpoint are related to the subscribing enterprise.

By Filter
A Feed might define a collection of resources based on a filter that describes a set of matching criteria a resource may be included in a Feed. Note that this type of Feed may require extra processing by the Feed Provider to determine if any particular SET event matches the filter criteria. It may also be difficult for the Feed Provider to notify Subscribers of additions and deletions of resources to the Feed as the resources in the Feed MAY change based on the filter itself.

By Group
All entities or resources within some specified group. For example, all resources within a SCIM Group could be used to define the resources for which SETs will be issued within a particular Feed.

The list above is intended to show common use cases for defining Feeds. How Feeds are defined is out-of-scope of this specification.

3.2. Feed Metadata

Feed metadata consists of the following singular attributes:

feedName
A required string value containing a name for the feed. May be used in administrative user interfaces to assist subscribers in Feed selection. The value MUST be unique within a given administrative domain. This is a REQUIRED attribute.

feedUri
An attribute of type "String" that is a unique URI identifying the feed. This attribute characteristic "mutability" is "immutable" and SHALL NOT change once assigned. The value of this attribute MAY be the SCIM URI for the Feed resource (e.g. "https://scim.example.com/Feeds/88bc00de"). This is a REQUIRED attribute.

description
A "String" attribute that describes the purpose of the feed in human readable form. This is an OPTIONAL attribute.

events
An attribute whose value is a JSON object consisting of multi-valued JSON attributes where each attribute is the name of a primary event URI and each value represents an event extension to the primary event. An empty array SHALL indicate there are no extensions. When set, Feeds SHALL only provide the primary events

defined. However, a Feed Provider MAY provide additional extensions that are not declared. This is an OPTIONAL attribute.

The following is a non-normative example events claim:

```
"events":{
  "urn:ietf:params:scim:event:passwordReset":[
      "https://example.com/scim/event/passwordResetExt"],
  "https://specs.openid.net/logout":[]
}
```

Figure 2: Example Events Attribute

In the above example, the feed has two events defined. The first is a hypothetical password reset, and the second is a hypothetical OpenID Connect logout. The password reset event has one extension defined which is "https://example.com/scim/event/passwordResetExt".

type
An OPTIONAL String attribute that MAY have values such as:

resource Indicates that the Feed is for events related to a specific resource. In such cases, the value of the attribute "filter" is set to a specific resource URI or "/Me" .

endpoint Indicates that the Feed is for all events that occur for resources within a specific endpoint. In such cases, "filter" is set to an endpoint container for a group of resources (e.g. "/Users").

filter Indicates that events for a Feed will be selected based on events relating to the set of resources described by a filter. For example, the value of the attribute "filter" is a SCIM filter Section 3.4.2 [RFC7644] that describes a condition that selects a set of resources that match before or after a resource state change.

group Indicates that events for a Feed will be based on events relating to the set of resources listed in a group such as a SCIM GroupSection 4.2 [RFC7643].

The attribute is typically used by the Feed Publisher to determine the meaning and content of the Feed "filter" attribute.

filter
An OPTIONAL String value containing a filter whose syntax is defined by a profiling specification (e.g. SCIM) or the Feed

Publisher. For example in SCIM, a filter MAY be a filter Section 3.4.2.2 [RFC7644]), a resource, or a SCIM endpoint URI depending on the value of "type". And, if the SCIM Feed type is "resource", than the filter value is a URI for a SCIM resource.

The following multi-valued attributes are defined:

deliveryModes
One or more URIs representing the methods of delivery supported by the Feed Publisher. Values in this attribute correspond to the Subscription "methodUri" attribute (see Section 4.1).

3.3. SCIM Feed Management

When Feeds are managed within a SCIM service provider [RFC7644], Feed resources use schema defined in Section 3.2 and use a schema value of "urn:ietf:params:scim:schemas:event:2.0:Feed". The SCIM "ResourceType" definition defines the location of the SCIM service provider endpoint for "Feed" resources.

The Feed "ResourceType" definition is typically defined as follows:

```
{
  "schemas": ["urn:ietf:params:scim:schemas:core:2.0:ResourceType"],
  "id": "Feed",
  "name": "Feed",
  "endpoint": "/Feeds",
  "description": "Event Feeds",
  "schema": "urn:ietf:params:scim:schemas:event:2.0:Feed",
  "schemaExtensions": []
}
```

Figure 3: SCIM Feed ResourceType Definition

To retrieve information about a "Feed" or a number of feeds, subscribers or management clients MAY query the "/Feeds" endpoint as defined in Section 3.4 [RFC7644].

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

```
GET /Feeds/88bc00de776d49d5b535ede882d98f74
Host: example.com
Accept: application/scim+json
Authorization: Bearer h480djs93hd8
```

Figure 4: Example Feed GET Request

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

```
HTTP/1.1 200 OK
Content-Type: application/scim+json
Location:
 https://example.com/v2/Feeds/88bc00de776d49d5b535ede882d98f74
ETag: 9d1c124149f522472e7a511c85b3a31b

{
  "schemas":["urn:ietf:params:scim:schemas:event:2.0:Feed"],
  "id":"88bc00de776d49d5b535ede882d98f74",
  "feedName":"OIDCLogoutFeed",
  "feedUri":"https://oidc.example.com/",
  "description":"Logout events from oidc.example.com",
  "type":"resource",
  "events":[
    "https://specs.openid.net/logout":[]
  ]
  "meta":{
     ... SCIM meta attributes ...
  }
}
```

Figure 5: Example Feed GET Response

In the above example (Figure 5) we can observe that the Feed has only one event type, "https://specs.openid.net/logout" and has no extensions defined for the event (see empty square brackets). Note also, that no value for "filter" has been specified suggesting that the Feed will return events about all subjects of the publisher.

4. Subscriptions

A subscription represents an agreement to deliver SETs from a specified Feed URI from a Feed Provider to an individual Subscriber entity also known as the "audience". The method of delivery and the parameters for delivery are specified a set of parameters called subscription metadata (see Section 4.1).

4.1. Subscription Metadata

A subscription is defined by the following metadata:

feedUri
: A String value containing the URI for a feed supported by the feed provider. It describes the content of the feed and MAY also be a

resolvable URI where the feed meta data may be returned as a JSON object. REQUIRED.

methodUri
A REQUIRED single-valued string which is a URI with a prefix of "urn:ietf:params:set:method". This specification defines HTTP POST delivery Section 5:
"urn:ietf:params:set:method:HTTP:webCallback"
in which the Feed Provider delivers events using HTTP POST to a specified callback URI.

deliveryUri
A URI that describes the location SETs are delivered. Its format and usage requirements are defined by the associated "methodUri" specification.

aud
An OPTIONAL URI representing the audience of the subscription. The value SHALL be the value of "aud" when the subscriber receives SETs from the feed.

feedJwk
An OPTIONAL public JSON Web Key (see [RFC7517]) that will be used to sign published SETs. If present, the Subscriber can authenticate SETs relayed from the Feed Provider.

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

subStatus
An OPTIONAL value that indicates the current status of a Subscription:

"on" - indicates the Subscription has been verified and that the Feed Provider MAY pass SETs to the Subscriber.

"verify" - indicates the Subscription is pending verification. While in "verify", published SETs SHALL NOT be stored or delivered to the Subscriber. Once verified, the status returns to "on".

"paused" - indicates the Feed Provider is temporarily suspending delivery to Subscriber. While "paused", SETs SHOULD be retained and delivered when state returns to "on". Verification is NOT required when returning to "on".

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

"fail" - indicates that the feed provider was unable to deliver SETs to the Subscriber for an extended period of time, or due to a call failure to the registered web call back URI. Unlike paused status, a failed subscription no longer receives SETs, nor are they retained by the Feed Provider. Before returning to "on", a verification MUST be performed.

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

maxDeliveryTime
An OPTIONAL number indicating the maximum amount of time in seconds a SET MAY take for successful delivery. Upon reaching the maximum, the subscription "subStatus" is set to "failed". If undefined, there is no maximum time.

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

4.2. Subscription State Model

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

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

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

Figure 6: Subscription States at Feed Publisher

In the above diagram, the following actions impact the state of a Subscription. "subStatus" values are shown in the boxes, and change based on the following actions:

Create
   A Subscriber or an administrator creates a new subscription using SCIM as described in Section 4.3.2. The initial state is "verify".

Confirm
   The Feed Publisher sends a verification SET to the Subscriber which confirms with the correct response as described in Section 4.4. If it succeeds to deliver, the Feed Publisher mail retry or set state to "on".

Confirm Fail

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

Timeout
A Feed Publisher having not being able to deliver a SET over one or more retries which has reached a limit of attempts ("maxRetries") or time ("maxDeliveryTime") MAY set the subscription state to "fail". In general, the intention is to indicate the maximum number of retries or time a Feed Publisher is able to wait until SET event loss begins to occur resulting in the failed state.

Restart
An administrator having corrected the failed delivery condition modifies the Subscription state to "verify" (e.g. see Section 4.3.3).

Suspend and Resume
A Subscription MAY be suspended and resumed by updating the Subscription state to "paused" or "on". For example, see see Section 4.3.3. While suspended, the Feed Publisher MAY retain undelivered SETs for a period of time. If the Feed Publisher is no longer able to retain SETs, the subscription state SHOULD be set to "off" to indicate SETs are being lost.

Enable and Disable
A subscription MAY be disabled and enabled by updating the Subscription state to "off" or "on". For example, see see Section 4.3.3. While the Subscription is disabled, all SETs that occur at the Feed Publisher are lost.

4.3. SCIM Subscription Management

A Feed Publisher MAY use SCIM to support management of subscriptions. Typically this involves support for the Subscription Resource Type, and the corresponding SCIM operations to create, update, retrieve Subscription Resources. For SCIM service provider capability and schema discovery, see Section 4 [RFC7644].

4.3.1. SCIM Subscription Resource Type

When Subscriptions are managed within a SCIM service provider [RFC7644], Subscription resources use schema defined in Section 4.1 and use a schema value of "urn:ietf:params:scim:schemas:event:2.0:Subscription".

The SCIM Subscription "ResourceType" definition is defined as follows:

```
{
  "schemas": ["urn:ietf:params:scim:schemas:core:2.0:ResourceType"],
  "id": "Subscription",
  "name": "Subscription",
  "endpoint": "/Subscriptions",
  "description": "Subscribers to SET Feeds",
  "schema": "urn:ietf:params:scim:schemas:event:2.0:Subscription",
  "schemaExtensions": []
}
```

Figure 7: SCIM Subscription ResourceType Definition

To retrieve information about one or more Subscriptions, Subscribers or management clients MAY query the "/Subscriptions" endpoint as defined in Section 3.4 [RFC7644].

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

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

Figure 8: Example SCIM Subscription GET Request

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

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

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

Figure 9: Example Subscription GET Response

In the above example (Figure 9) observe that the subscription is for the SCIM "Feed" resource defined at "https://example.com/v2/Feeds/88bc00de776d49d5b535ede882d98f74". The current subscription state is "pending" which suggest the Subscription Verification (see Section 4.4) process has not yet completed. Since there is no value for "feedJwk",) or "confidentialJwk", the SETs will be sent without signing or encryption (plain text).

4.3.2. New Subscription Requests

To subscribe to a feed, the subscriber of management client uses the SCIM Create operation as defined in Section 3.3 [RFC7644]. SCIM subscription management service providers MAY have additional schema requirements which MAY be discovered using SCIM service configuration and schema discovery, see Section 4 [RFC7644].

In the following non-normative example, a new Subscription resource is requested. Note that the SCIM service provider automatically assigns the "id" attribute.

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

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

Figure 10: Example New Subscription Request in SCIM

In following non-normative response, the SCIM service provider has automatically assigned a resource location as well as an "id". Usually upon creation, the initial value of "subStatus" is "pending" indicating that the Subscription Verification (see Section 4.4) has not been completed.

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

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

Figure 11: Example Response to Subscription Request

4.3.3. Updating Subscriptions

To modify a Subscription, a Subscriber or authorized management client MAY use the SCIM PUT operation (see Section 3.5.1 [RFC7644]) and MAY use the SCIM PATCH operation (see Section 3.5.2 [RFC7644]) if supported by the SCIM Subscription server.

In the following non-normative example, the client is requesting that "subStatus" be changed to "paused" for the Subscription whose path is identified by the request URI path.

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

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

Figure 12: Example SCIM Subscription Update

4.4. Subscription Verification

In order to avoid ongoing communication issues and to minimize requirements for Feed Providers to maintain a series of SETs indefinitely, a verification process is used to confirm that the requested Subscription distribution endpoints are valid and that SETs may be successfully delivered. When a Subscription is created or modified, or goes into a failed or off state, the Feed Provider SHALL set the Subscription state attribute "subStatus" to "verify" and send a Verify SET message to the subscriber. If the SET is delivered successfully, the subscription state SHOULD be turned to "on". If however verification fails due to a timeout or connection failure, or any other cause, the Subscription status SHALL be set to "fail".

4.4.1. Verifying Subscriptions

The verification process serves to verify that the identified Subscriber is willing to receive SETs and is correctly configured. In the case of push style subscriptions, where the publisher initiates the action to deliver a SET, Verification can also serve to prevent a Feed Publication server from flooding an endpoint which did not actually request a Subscription.

A Feed Provider MAY send a Verify SET at any time in order to reverify connectivity and to assure the subscriber the subscription is valid (e.g. as a keep alive technique).

To confirm a subscription, the Feed Provider SHALL send a verification SET to the subscriber 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 of the feed publisher (see [I-D.hunt-idevent-token]).

aud MUST be set to a value that matches the subscription "feedUri" requested.

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

In the SET payload area, a specific delivery method MAY include an attribute that can be used to confirm the subscriber has successfully received and parsed the SET (e.g. such as the inclusion of a challenge attribute, see Section 5.3.3). If a confidential JWK was supplied, then the SET SHOULD be encrypted with the provided key. Successful parsing of the message confirms that provides confirmation of correct configuration and possession of keys.

Note that the verification event URI ("[[this RFC URL]]#verify") type is not normally listed as part of the definition of a Feed as it is not part of the normal information flow of a Feed. Any Feed MAY include a SET verification event whether listed or not in the Feed event metadata.

Upon receiving the SET, the subscriber acknowledges receipt as defined by the method profile (for example, see Section 5.3.3).

If the subscriber is unable to parse the verification SET, fails to return the correct challenge, or the SET is not delivered after a period of time. The Feed Publisher will set "subStatus" to "failed".

5. Event Delivery

5.1. Introduction to Event Delivery Methods

Each event delivery method SHOULD have the following information:

Description

The "methodUri" URI value for the delivery method and a description of the method.

Subscription Verification Procedure
The procedure that the configuration for a subscription is confirmed causing the subscription status to be set to "on".

Delivery Message Format
A description of an event delivery message and how to locate the event token(s) as well as any additional error signalling.

Delivery Procedure
The protocol procedure for a delivery request (push or poll), and the expected successful response.

Failure Conditions
A description of the failure conditions that might occur and the impact on the subscriptions operational status ("subStatus") if any.

This specification defines the first delivery method known as "HTTP Web Callback Method" which uses HTTP POST.

5.2. Delivery Processing

As mentioned in Section 4.1, the attribute "subStatus" defines the current state of a subscribers subscription. Figure 6 shows a state diagram for Subscriptions. The following describes that actions taken by the Feed Publisher based upon "subStatus".

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

Table 1: Delivery Processing By Status

5.3. HTTP Web Callback Method

5.3.1. Description

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

This delivery method is capable of delivering a single SET per HTTP POST request. Depending on the settings for the subscription metadata (see Section 4.1), the SET MAY be signed and/or encrypted as defined in [I-D.hunt-idevent-token].

The Subscription's "deliveryUri" attribute indicates the location of a Subscriber provided endpoint which can accept HTTP POST requests (e.g. "https://notify.examplerp.com/Events").

5.3.2. Delivery Message Format

The content-type for this method is "application/jwt" and consists of a single SET token (see [I-D.hunt-idevent-token]).

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

Figure 13: Example Web Callback POST Message

5.3.3. Subscription Verification

This profile specifies the verification method for HTTP POST and is based on the general verification method described in Section 4.4.1.

To confirm a subscription, the Feed Provider SHALL send a verification SET to the subscriber using the registered "methodUri" mechanism which in this case is "urn:ietf:params:set:method:HTTP:webCallback". The Verify SET contains the attributes listed in Section 4.4.1.

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

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

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

Figure 14: Example Verification SET with Challenge

The above SET is encoded as a JWT and transmitted to the Subscriber as shown in Figure 16.

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

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

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

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

Figure 15: Example Response to Verify SET with Challenge

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

5.3.4. Delivery Procedure

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

```
POST /Events  HTTP/1.1

Host: notify.examplerp.com
Accept: application/json
Content-Type: application/jwt
"eyJhbGciOiJub25lIn0
.
eyJwdWJsaXNoZXJVcmkiOiJodHRwczovL3NjaW0uZXhhbXBsZS5jb20iLCJmZWV
kVXJpcyI6WyJodHRwczovL2podWIuZXhhbXBsZS5jb20vRmVlZHMvOThkNTI0Nj
FmYTVmYmM4Nzk1OTNiNzc1NCIsImh0dHBzOi8vamh1Yi5leGFtcGxlLmNvbS9GZ
WVkcy81ZDc2MDQ1MTZiMWQwODY0MWQ3Njc2ZWU3Il0sInJlc291cmNlVXJpcyI6
WyJodHRwczovL3NjaW0uZXhhbXBsZS5jb20vVXNlcnMvNDRmNjE0MmRmOTZiZDZ
hYjYxZTc1MjFkOSJdLCJldmVudFR5cGVzIjpbIkNSRUFURSJdLCJhdHRyaWJ1dG
VzIjpbImlkIiwibmFtZSIsInVzZXJOYW1lIiwicGFzc3dvcmQiLCJlbWFpbHMiX
SwidmFsdWVzIjp7ImVtYWlscyI6W3sidHlwZSI6IndvcmsiLCJ2YWx1ZSI6Impk
b2VAZXhhbXBsZS5jb20ifV0sInBhc3N3b3JkIjoibm90NHUybm8iLCJ1c2VyTmF
tZSI6Impkb2UiLCJpZCI6IjQ0ZjYxNDJkZjk2YmQ2YWI2MWU3NTIxZDkiLCJuYW
1lIjp7ImdpdmVuTmFtZSI6IkpvaG4iLCJmYW1pbHlOYW1lIjoiRG9lIn19fQ
.
```

Figure 16: Example Web Callback POST Request

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

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

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

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

Table 2: HTTP Status 400 Errors

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

```
HTTP/1.1 202 Accepted
```

Figure 17: Example Successful Delivery Response

An HTTP Status 400 Bad Request response includes a JSON object which provides details about the error. The JSON object includes the JSON attributes:

err
   A value which is a keyword that describes the error (see Table 2).

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

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

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

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

}
```

Figure 18: Example Bad Request Response

6. Security Considerations

[TO BE COMPLETED]

7. IANA Considerations

7.1. Event Notification Mechanism Registry

[TODO: Registration for Notification Mechanisms]

7.2. SCIM Schema Registration

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

Schema URI	Name	ResourceType	Reference
urn:ietf:params:scim: schemas:event:2.0: Feed	SET Event Feed	Feed	Section 3.3
urn:ietf:params:scim: schemas:event:2.0: Subscription	SET Event Subscription	Subscription	Section 4.3

8. References

8.1. Normative References

[I-D.hunt-idevent-token]
Hunt, P., Denniss, W., Ansari, M., and M. Jones, "Security Event Token (SET)", draft-hunt-idevent-token-05 (work in progress), September 2016.

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

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC5988] Nottingham, M., "Web Linking", RFC 5988, DOI 10.17487/RFC5988, October 2010, <http://www.rfc-editor.org/info/rfc5988>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <http://www.rfc-editor.org/info/rfc7231>.

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

[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>.

8.2. Informative References

[I-D.ietf-webpush-protocol]
Thomson, M., Damaggio, E., and B. Raymor, "Generic Event Delivery Using HTTP Push", draft-ietf-webpush-protocol-02 (work in progress), November 2015.

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

[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>.

Appendix A. Contributors

Appendix B. Acknowledgments

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

Appendix C. Change Log

Draft 00 - PH - First Draft

Draft 01 - PH -

- o Removed the version from filename in GITHUB version
- o Aligned document with new SET terminology from I-D.hunt-idevent-token
- o Simplified draft to only define HTTP POST profile (TBD)
- o Removed webpush and polling modes (can be re-added later).
- o Added SCIM management definitions for Feeds
- o Added delivery information including errors
- o Added subscription management information (e.g. how to subscribe)
- o Updated reference to idevent-token to published IETF version
- o Added a state diagram for Subscriptions

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