Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: September 9, 2015

P. Hunt, Ed.
Oracle
M. Ansari
Cisco
I. Glazer
Salesforce

March 8, 2015

SCIM Event Notification
draft-hunt-scim-notify-00

Abstract

The System for Cross-Domain Identity Management (SCIM) specification is an HTTP based protocol that makes managing identities in multi-domain scenarios easier to support through a standardized HTTP service.

In a SCIM environment, changes to resources may be requested by multiple parties. As time goes by an interested subscriber may wish to be informed about resource state changes that are occurring at the SCIM service provider. This specification defines a hub notification service that can be used to publish and distribute events to interested registered subscribers.

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 September 9, 2015.

Table of Contents

1. Introduction and Overview

The System for Cross-Domain Identity Management (SCIM) specification is an HTTP based protocol that makes managing identities in multi-domain scenarios easier to support through a standardized HTTP service.

In a SCIM environment, changes to resources may be requested by multiple parties. As time goes by an interested subscriber may wish to be informed about resource state changes that are occurring at the SCIM service provider. This specification defines a hub notification service that can be used to publish and distribute events to interested registered subscribers. This specification's design is inspired by [pubsubhubbub], but is significantly adapted to use JSON and JWT tokens.

The goal of this specification is to avoid alternative approaches such as ATOM feeds or LDAP changelogs that require historical change history be maintained. These techniques not only causes scalability issues, but may also provide security risks over time because of the difficulties in filtering history and data for specific subscribers based on their authorized access to information.

This specification defines a Notification Hub which receives events from a SCIM Service Provider. The role of the notification hub is to offload the SCIM Service provider by taking care of:

- Re-publishing events to appropriate Feeds where the event is a match.
- Distributing events to registered subscribers using their registered notification mechanism.
- Persisting events as necessary until all registered subscribers have received the event.

The following diagram shows a typical Notification Hub and its service relationships with SCIM Service Providers and registered notification Subscribers.

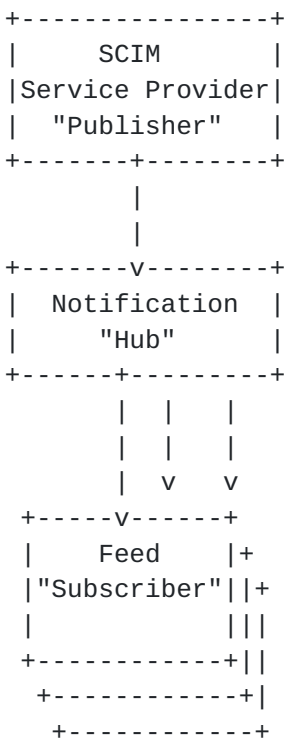

Figure 1: Notification Relationships

The specification supports two layers of authentication. The base layer uses normal HTTP Authentication techniques that may include TLS mutual-authentication and OAuth access token authorizations. The second layer involves the exchange of JWT keys that may be used to authenticate messages and encrypt content for registered subscribers.

For a feed subscriber, this specification makes it possible to receive update or change-of-state notifications that may be of interest. By providing state change event notifications, a large cross-section of subscribers can be developed to support use cases such as: work-flow co-ordination, overall enterprise identity governance co-ordination, and cross-domain replication. [DISCUSS: should we elaborate on this?]

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 interoperability 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

SCIM Service Provider The SCIM service provider as defined in SCIM core schema. A SCIM service provider publishes events to be distributed by a trusted Notification Hub.

Notification Hub A service provider in the same domain as the SCIM Service Provider. Because the "hub" manages individual subscribers, the "hub" must have access to all event message content so that it may encrypt it for specific subscribers when required.

Event An event is a SCIM change event that is to be distributed to one or more registered subscribers. An event is encapsulated as a JWT token and MAY be signed or encrypted using JWS/JWE for authentication and confidentiality reasons.

Feed A feed is a URI that describes the set of resources and events under which events may be issued. An interested client registers with the Notification Hub to subscribe to events associated with a 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. [This specification will define a delivery extension point.] Examples of possible delivery mechanisms include:

* Registered Callback - The Notification Hub will deliver events by using HTTP POST to a registered endpoint.

* Polling - The subscriber will periodically poll the notification hub for one or more events by performing an HTTP GET to a specified URI (mailbox endpoint).

* WebPush - An HTTP/2 based method for delivering real-time events. https://tools.ietf.org/wg/webpush/

* Platform/Mobile Notification Services (e.g. Apple Push Notification Service, Google Cloud Messaging, and Windows Notification Services). Future profiles that support delivery of SCIM events vis platform specific messaging services.

Subscriber A Subscriber registers to receive event notifications from a Notification Hub on behalf of a SCIM Service Provider.

2. Notification Process

When a resource has changed, a SCIM Service Provider constructs a JWT that describes the event. The service provider MAY include the affected feedURIs for the Notification Hub to process. The SCIM service provider publishes the event to the Notification Hub using either an HTTP POST, or will hold the event (along with other events) until the notification retrieves pending events via an HTTP GET. After an event is delivered to the Notification Hub, the service provider has no further obligation to maintain or distribute the event. The service provider MAY subsequently receive requests from subscribers for more information about the current state of a resource using a normal SCIM GET request.

An event is a set of attributes combined together in a JWT token. An event typically includes:

o Affected resource URI

o A list of attributes modified

o An event type (patch, create, delete)

o Optional attribute values for affected attributes. Note: JWT MUST be a JWE if raw data is included.

An event describes what state change has occurred at a publishing service provider resource. An event should not be interpreted as a command to change by the subscriber.

Upon receiving an event, the notification hub, categorizes the event and determine which feeds should be notified of the event. For each feed, the notification hub delivers the events to any registered

clients. If the event contains raw attribute values, the notification hub encrypts the JWT so that only the registered subscriber(s) may receive the event. To deliver the event, the notification hub uses the registered notification mechanism requested by the subscriber.

Upon receiving an event (or events for polling subscribers), the subscriber reads the JWT or JWE token and validates it if signed (a JWS). Based on the content of the JWT, the subscriber decides what if any action it needs to take in response to the event from the SCIM Service Provider. The subscriber MAY perform a SCIM GET request to the affected resource URI in order to confidentially obtain the current state of the affected resource.

3. SCIM Events

A SCIM Event notifies a subscriber of a possible change in state of a resource contained within a specified feed. The event specifies the URI of an affected resource and the type of event that has occurred. The event message may also contain additional metadata describing the attributes that may have been modified, and/or values representing the final state of the affected attribute(s).

3.1. Event Types

This specification defines 8 events. Additional event types may be defined using the Event Type IANA registration process described in Section 8.2.

ADD
: The specified resource URI was added to the feed. An add does not necessarily indicate a resource is new or has been recently created, but rather that it has been added to a feed. For example, an existing user has had a new role (e.g. CRM_User) added to their profile which has caused their resource to join a feed.

CREATE
: The new resource URI has been created at the service provider and has been added to the feed. When a CREATE event is sent, a corresponding ADD event is not issued. For example, a new user was created via HTTP POST, whose attribute profile met the criteria of a current feed.

ACTIVATE
: The specified resource (e.g. User) has been activated or is otherwise available for use. [TODO: this seems to be a higher

level concept that may consist of multiple attributes changing - How to differentiate between activate and MODIFY events]

MODIFY
The specified resource has been updated (e.g. one or more attributes has changed).

DEACTIVATE
The specified resource (e.g. User) has been deactivated. [TODO: Is there a corresponding MODIFY event?]

DELETE The specified resource has been deleted from the service provider and is also removed from the feed. When a DELETE is sent, a corresponding REMOVE is not issued.

REMOVE
The specified resource has been removed from the feed. Removal does not indicate that the resource was deleted or otherwise deactivated.

PASSWORD
The specified resource (e.g. User) has changed its password. If secure exchange is possible with the subscriber, the event may also include the raw password update text. A PASSWORD event MUST be transmitted in encrypted form (see Section 3.3).

CONFIRMATION A special event that is used during Polling Feed Registrations and Web Callback URI subscriptions to confirm successful configuration of an event feed. The contents of a CONFIRMATION event SHALL be defined by the registration process documented in following sections [TODO add reference]

3.2. Event Metadata

The following are attributes that may be included in an event message:

schemas A multi-valued String attribute that contains the value "urn:ietf:params:scim:schemas:notify:2.0:Event".

feedUris A multivalued String containing the URIs of the feeds the event is associated with. The notification hub may filter these values to be only those values relevant to a particular subscriber. In doing so, the notification hub is not obligated to deliver repeated events to the same subscriber. [DISCUSS: Is this problematic if the subscriber is using multiple endpoints?]

publisherUri

A single valued string containing the URI of the SCIM Service Provider publishing the event. Typically this is the SCIM Service Provider's root endpoint.

resourceUris
: A multi-valued string that contains the URIs of one or more affected resources in the event. For each resource URI included, the event must be identical.

type
: A single-valued string that contains the type of event as described in Section 3.1 or defined in the event extension registry in Section 8.2.

attributes
: A multi-valued list of affected SCIM attributes. Each attribute listed may be a fully-qualified attribute name or an attribute "path" as defined in Figure 7 of Section 3.3.2 of [I-D.ietf-scim-api]

values
: A JSON object structure containing the affected attributes and their associated values. If the "values" attribute is supplied, the event message MUST be encrypted. Service providers SHOULD take care to ensure that eligible subscribers are able to see attribute values. Alternatively, subscribers MAY use the resourceURIs to retrieve the final attribute values. When doing so, the SCIM service provider can then assess the subscribers right to obtain the actual attribute values.

  For a password change event, the clear text password attribute value MAY be included in the values attribute. "PASSWORD" event.

3.3. Event Message

An event message is a JSON structure consisting of the attributes described in the Event Metadata section above. The following is a non-normative example event that has been modified for readability:

```
{
  "schemas":["urn:ietf:params:scim:schemas:notify:2.0:Event"],
  "publisherUri":"https://scim.example.com",
  "feedUris":[
   "https://jhub.example.com/Feeds/98d52461fa5bbc879593b7754",
   "https://jhub.example.com/Feeds/5d7604516b1d08641d7676ee7"
  ],
  "resourceUris":[
    "https://scim.example.com/Users/44f6142df96bd6ab61e7521d9"
  ],
  "type":"CREATE",
  "attributes":["id","name","userName","password","emails"],
  "values":{
    "emails":[
     {"type":"work","value":"jdoe@example.com"}
    ],
    "password":"not4u2no",
    "userName":"jdoe",
    "id":"44f6142df96bd6ab61e7521d9",
    "name":{
      "givenName":"John",
      "familyName":"Doe"
    }
  }
}
```

Figure 2: Example Event JSON Data

When transmitted, the above JSON body must be converted into a JWT as per [I-D.ietf-oauth-json-web-token]. In this example, because the event contains attribute values, the token MUST be encrypted per JWE (see [I-D.ietf-jose-json-web-encryption]) before transmission.

The following is an example of a SCIM Event expressed in an unsecured JWT token. The JWT header of:

```
{"alg":"none"}
```

Base64url encoding of the octets of the UTF-8 representation of the header yields:

```
eyJhbGciOiJub25lIn0
```

The example JSON Event Data is encoded as follows:

eyJwdWJsaXNoZXJVcmkiOiJodHRwczovL3NjaW0uZXhhbXBsZS5jb20iLCJmZWV
kVXJpcyI6WyJodHRwczovL2podWIuZXhhbXBsZS5jb20vRmVlZHMvOThkNTI0Nj
FmYTViYmM4Nzk1OTNiNzc1NCIsImh0dHBzOi8vamh1Yi5leGFtcGxlLmNvbS9GZ
WVkcy81ZDc2MDQ1MTZiMWQwODY0MWQ3Njc2ZWU3Il0sInJlc291cmNlVXJpcyI6
WyJodHRwczovL3NjaW0uZXhhbXBsZS5jb20vVXNlcnMvNDRmNjE0MmRmOTZiZDZ
hYjYxZTc1MjFkOSJdLCJldmVudFR5cGVzIjpbIkNSRUFURSJdLCJhdHRyaWJ1dG
VzIjpbImlkIiwibmFtZSIsInVzZXJOYW1lIiwicGFzc3dvcmQiLCJlbWFpbHMiX
SwidmFsdWVzIjp7ImVtYWlscyI6W3sidHlwZSI6Indvcmsi LCJ2YWx1ZSI6Impk
b2VAZXhhbXBsZS5jb20ifV0sInBhc3N3b3JkIjoibm90NHUybm8iLCJ1c2VyTmF
tZSI6Impkb2UiLCJpZCI6IjQ0ZjYxNDJkZjk2YmQ2YWI2MWU3NTIxZDkiLCJuYW
1lIjp7ImdpdmVuTmFtZSI6IkpvaG4iLCJmYW1pbHlOYW1lIjoiRG9lIn19fQ

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

eyJhbGciOiJub25lIn0
.
eyJwdWJsaXNoZXJVcmkiOiJodHRwczovL3NjaW0uZXhhbXBsZS5jb20iLCJmZWV
kVXJpcyI6WyJodHRwczovL2podWIuZXhhbXBsZS5jb20vRmVlZHMvOThkNTI0Nj
FmYTViYmM4Nzk1OTNiNzc1NCIsImh0dHBzOi8vamh1Yi5leGFtcGxlLmNvbS9GZ
WVkcy81ZDc2MDQ1MTZiMWQwODY0MWQ3Njc2ZWU3Il0sInJlc291cmNlVXJpcyI6
WyJodHRwczovL3NjaW0uZXhhbXBsZS5jb20vVXNlcnMvNDRmNjE0MmRmOTZiZDZ
hYjYxZTc1MjFkOSJdLCJldmVudFR5cGVzIjpbIkNSRUFURSJdLCJhdHRyaWJ1dG
VzIjpbImlkIiwibmFtZSIsInVzZXJOYW1lIiwicGFzc3dvcmQiLCJlbWFpbHMiX
SwidmFsdWVzIjp7ImVtYWlscyI6W3sidHlwZSI6Indvcmsi LCJ2YWx1ZSI6Impk
b2VAZXhhbXBsZS5jb20ifV0sInBhc3N3b3JkIjoibm90NHUybm8iLCJ1c2VyTmF
tZSI6Impkb2UiLCJpZCI6IjQ0ZjYxNDJkZjk2YmQ2YWI2MWU3NTIxZDkiLCJuYW
1lIjp7ImdpdmVuTmFtZSI6IkpvaG4iLCJmYW1pbHlOYW1lIjoiRG9lIn19fQ
.

Figure 3: Example Unsecured Event Token

To create and or validate a signed or encrypted event token follow the instructions in section 7 of [I-D.ietf-oauth-json-web-token].

4. Discovery

4.1. SCIM Server Resource Link Headers

A SCIM Service Provider MAY provide resource link headers per [RFC5988] for the purpose of feed and endpoint discovery. When querying a SCIM Service Provider, the following resource link headers are defined:

rel=notifyHub

The URI provided is the base URI for a SCIM Notification Hub and its associated HTTP services.

rel=oauth
The URI provides is for an OAuth authorization endpoint that will authorize access to a SCIM feed. When requesting authorization, clients should request an OAuth scope of "notifyHub". This allows the client to register and request access to specific feeds. [TBD is this needed?]

rel=scimfeed
The URI provided is for a SCIM Feed for the current resource. The feed URI might be specific to the current resource, or for a larger set of resources (e.g. /Users). There may be more than one feed for any particular resource or set of resources.

4.2. Feeds Endpoint

A service provider may publish a set of Feed resources that each describe an available feed at a "/Feeds" endpoint. [TODO: Define the schema and resource type configuration for a feed based on SCIM Feed definitions in the next section]

5. SCIM Feeds

A SCIM Service 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 a feed or how feed URIs should be specified. Some possible methods for defining feeds include:

By Resource
Each resource might have its own 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 transmitted to subscriber. This type of feed is likely to have many notifications as the number of resources in an endpoint grows (e.g. /Users). Typically only privileged partners would be allowed to use this type of feed. For example an enterprise wishes to be notified of all 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 SCIM search filter. Based on 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 service provider to determine if any particular resource event matches the filter criteria. It may also be difficult for the service provider to notify subscribers of Feed additions and deletions as these will occur dynamically based on the filter.

By Group
A designated SCIM Group could be used to define the resources within a particular feed. [TODO define a FEED Group extensions that define the attributes and events included within a particular Feed Group]

5.1. Feed Discovery

[TBD] May be discovered by resource link headers, or by querying for SCIM Feed Groups [TBD].

6. Notification Hub

The notification hub is usually deployed as a separate but trusted server (in the same security domain) in relation to a SCIM Service Provider. A single server MAY support both SCIM services and the API described in the notification hub. The notification hub's API follows the SCIM API specification unless indicated otherwise. The notification hub provides 2 core services:

- A set of publisher endpoints that allows SCIM Service Providers to define feeds and to post SCIM Events.

- A set of subscriber endpoints that allows subscription clients to subscribe to feeds and to receive events.

6.1. Subscriber Services

Subscribers MAY manage their subscriptions using the notification hub's "/Subscriptions" endpoint. With the exception of the PATCH operation (which is not used), endpoints within the notification hub follow the same message format and API guidelines as per the SCIM API specification (see [I-D.ietf-scim-api]). The subscriptions endpoint supports HTTP GET, POST, PUT, and DELETE to manage the full life cycle of a subscription.

6.1.1. Subscription Metadata

The following attributes are used to define a feed subscription by a subscription client.

feedUri
: A string value containing the URI for a feed supported by the notification hub. REQUIRED.

mode
: A REQUIRED single-valued string which is a URI with one of the following values:

  "urn:ietf:params:scimnotify:api:messages:2.0:webCallback" - The notification hub will pass SCIM Events using HTTP POST to the callback URI specified in the attribute "eventUri".

  "urn:ietf:params:scimnotify:api:messages:2.0:poll" - The subscriber will poll for SCIM Events using HTTP GET at the URI specified in the attribute "eventUri"

eventUri
: When "mode" is set to "urn:ietf:params:scimnotify:api:messages:2.0:poll", "eventUri" specifies the endpoint where the subscriber will retrieve pending events. When set to "urn:ietf:params:scimnotify:api:messages:2.0:webCallback", it contains the URI where the notification hub will POST events.

feedJwk
: AN OPTIONAL JSON Web Key (see [I-D.ietf-jose-json-web-key]) that will be used to sign published events. If present, the subscriber can authenticate events relayed from the notification hub.

confidentialJwk
: The subscriber may provide a public JSON Web Key (see [I-D.ietf-jose-json-web-key]) that may be used by the notification hub to encrypt event messages for the subscriber.

pollInterval
: The optional period in seconds between event polls when the subscriber plans to poll for events from the notification hub. The interval is used by the hub to determine if a subscriber is offline or has otherwise failed over a number of intervals. The hub MAY then change the state of the feed and/or perform some out-of-band administrative alert.

state

An optional value which indicates the current state of the feed which is:

"on" - the default setting indicates the notification hub processing events and will pass them to the subscriber.

"verify" - the subscription is pending verification. While in "verify", published events SHALL NOT be stored or delivered to the subscriber.

"paused" - indicates the notification hub is temporarily suspending delivery to subscriber. While in "paused" events MAY be posted and will be delivered when state returns to "on".

"off" - indicates that the subscription is no longer passing events. While in off mode, the subscription is maintained, but new events are ignored and not processed.

"fail" - Indicates that the notification hub was unable to deliver events to the subscriber for an extended period of time, or due to a call failure to the registered web call back URI.

6.1.2. Request Subscription

To request a subscription, a client performs a SCIM POST to the /Subscriptions endpoint with a HTTP Body consisting of a JSON object based on the attributes described in Section 6.1.1. The request MUST include the "schemas" attribute with a value of: "urn:ietf:params:scim:schemas:notify:2.0:Subscription".

The following is a non-normative example subscription creation request.

```
POST /Subscriptions  HTTP/1.1
Host: notify.example.com
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Bearer h480djs93hd8
Content-Length: ...

{
  "schemas":
    ["urn:ietf:params:scim:schemas:notify:2.0:Subscription"],
  "feedUri":"https://example.com/v2/Feeds/548b7c3f77c8bab33a4fef40",
  "mode":"urn:ietf:params:scimnotify:api:messages:2.0:webCallback",
  "eventUri":"https://subscriber.com/Events",
  "state":"verify",
  "feedJwk":{
      "kty":"EC",
      "crv":"P-256",
      "x":"f83OJ3D2xF1Bg8vub9tLe1gHMzV76e8Tus9uPHvRVEU",
      "y":"x_FEzRu9m36HLN_tue659LNpXW6pCyStikYjKIWI5a0",
      "kid":"Public key for Example.com SCIM Publisher"
     }
}
```

Figure 4: Example Subscription Creation Request

In response to a successful subscription creation request, the server responds with HTTP Status 200 and a representation of the completed subscription. The following is a non-normative example response that has been formatted for display purposes:

```
HTTP/1.1 201 Created
Content-Type: application/scim+json
Location:
 https://example.com/v2/Subscriptions/548b7c3f77c8bab33a4fef40

{
  "schemas":
    ["urn:ietf:params:scim:schemas:notify:2.0:Subscription"],
  "feedUri":"https://example.com/v2/Feeds/548b7c3f77c8bab33a4fef40",
  "mode":"urn:ietf:params:scimnotify:api:messages:2.0:webCallback",
  "eventUri":"https://subscriber.com/Events",
  "feedJwk":{
      "kty":"EC",
      "crv":"P-256",
      "x":"f83OJ3D2xF1Bg8vub9tLe1gHMzV76e8Tus9uPHvRVEU",
      "y":"x_FEzRu9m36HLN_tue659LNpXW6pCyStikYjKIWI5a0",
      "kid":"Public key for Example.com SCIM Publisher"
     }
}
```

Upon receiving a successful subscription response, the subscribing client SHOULD check the subscription "state". If set to "verify", the client needs to complete the subscription verification process in the following section.

6.1.3. Subscription Verification

In order to avoid ongoing communication issues and to minimize requirements for notification hubs to maintain events indefinitely, a verification process is used to confirm that the requested event distribution endpoints are correct and that an event may be successfully delivered.

When using the WebCallback mode, or any other "push"-style communication scheme, the verification process also serves to verify that the identified endpoint consents to receiving events. This prevents a notification server from flooding an endpoint which did not actually request an event subscription.

To confirm a subscription, the notification hub SHALL POST (or otherwise send) an event to the endpoint identified by eventUri or as specified by the registered push extension. The event contains the following attributes:

type Set to the value of "CONFIRMATION"

publisherUri Set to the URI used to identify the notification hub.

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

confirmChallenge A random String value that the subscriber SHALL echo back in its response.

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

If a confidential JWK was supplied, then the event SHOULD be encrypted with the provided key. Successful parsing of the message confirms that both the endpoint is valid and the subscribers ability to parse the message.

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.

```
{
  "schemas":["urn:ietf:params:scim:schemas:notify:2.0:Event"],
  "publisherUri":"https://scim.example.com",
  "feedUris":[
   "https://notify.example.com/Feeds/98d52461fa5bbc879593b7754"

  ],
  "type":"CONFIRMATION",
  "confirmChallenge":"ca2179f4-8936-479a-a76d-5486e2baacd7",
  "expires":""

}
```

Upon receiving a subscription confirmation request, a consenting client responds with a confirmation that includes the original "confirmChallenge" value. A non-normative example of the response is:

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

{
  "schemas":["urn:ietf:params:scim:schemas:notify:2.0:Confirm"],
  "challengeResponse":"ca2179f4-8936-479a-a76d-5486e2baacd7"
  }
}
```

As per the above figure, upon receiving and parsing a confirmation event, the subscriber MUST respond by returning a JSON structure that includes the attribute "challengeResponse" with a matching value to "confirmChallenge" that was sent in the event. The response is not formatted as an event token but rather a JSON object with schemas set to "urn:ietf:params:scim:schemas:notify:2.0:Confirm". 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).

6.1.4. Cancel A Subscription

To cancel a subscription, the subscriber MAY perform a SCIM DELETE against the resource URI for the subscription. In the event the subscriber wants to temporarily suspend the subscription, it may modify the "state" attribute to a value of "off".

6.2. Publisher Services

With the exception of the PATCH operation (which is not used), the endpoints within the Notification Hub follow the same message format and API guidelines as per the SCIM API specification (see [I-D.ietf-scim-api]). Feed Registration supports HTTP GET, POST, PUT, and DELETE to manage the full life cycle of a Feed.

6.2.1. Feed Registration

To register a feed, a SCIM Service Provider makes a call to the Notify Hub's registration endpoint ("<Notification_base>/Feeds") by performing an HTTP POST containing a JSON structure based on the parameters defined in the following section. In response, the server will return a feed location and an optional public key which the publisher may use to encrypt posted events to the Notification Hub.

In the registration request, the "schemas" attribute MUST be included in the registration request and be set to: "urn:ietf:params:scim:schemas:notify:2.0:Feed". The following is a non-normative example of a request to create a new SCIM Feed. Note that additional spacing has been introduced for clarity.

```
POST /Feeds  HTTP/1.1
Host: notify.example.com
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Bearer h480djs93hd8
Content-Length: ...

{
  "schemas":["urn:ietf:params:scim:schemas:notify:2.0:Feed"],
  "feedName":"bjensen",
  "feedDescription":"Feed for changes related to bjensen",
  "feedData":{
    "$ref":
      "https://example.com/v2/Users/453a-919d-413861904646"
  }
  "mode":"post",
  "publisherJwk":{
      "kty":"EC",
      "crv":"P-256",
      "x":"f83OJ3D2xF1Bg8vub9tLe1gHMzV76e8Tus9uPHvRVEU",
      "y":"x_FEzRu9m36HLN_tue659LNpXW6pCyStikYjKIWI5a0",
      "kid":"Public key for Example.com SCIM Publisher"
     }
}
```

Figure 5: Example Feed Creation Request

In response to a successful registration request, the Notification Hub SHALL respond with the location of the created feed in the HTTP Location header, and the HTTP body SHALL contain a JSON structure with the accepted registration parameters and MAY include in its response, the following additional parameter:

confidentialJwk
: In its response, the Notification Hub may provide a public JSON Web Key (see [I-D.ietf-jose-json-web-key]) that may be used by the client to encrypt event messages for the notification hub. The key might be the notification hub's general public key, or it may be generated per registered feed. Accordingly, registering SCIM Service Providers should assume that each key returned MAY be specific to the corresponding registered feed.

```
HTTP/1.1 201 Created
Content-Type: application/scim+json
Location:
 https://example.com/v2/Feeds/548b7c3f77c8bab33a4fef40
ETag: 9d1c124149f522472e7a511c85b3a31b

{
  "schemas":["urn:ietf:params:scim:schemas:notify:2.0:Feed"],
  "id":"548b7c3f77c8bab33a4fef40",
  "feedName":"bjensen",
  "feedDescription":"Feed for changes related to bjensen",
  "feedData":{
    "type":"resource",
    "$ref":
      "https://example.com/v2/Users/453a-919d-413861904646"
  }
  "mode":"post",
  "state":"on",
  "publisherJwk":{
      "kty":"EC",
      "crv":"P-256",
      "x":"f83OJ3D2xF1Bg8vub9tLe1gHMzV76e8Tus9uPHvRVEU",
      "y":"x_FEzRu9m36HLN_tue659LNpXW6pCyStikYjKIWI5a0",
      "kid":"Public key for Example.com SCIM Publisher"
     }
  "confidentialJwk":{
     ...<feed public crypto key>...
  }
}
```

Figure 6: Example Feed Creation Response

6.2.1.1. Feed Request Parameters

To create a feed, the following parameters are used by the SCIM Service Provider to register a feed:

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.

feedDescription
: An optional string value containing text that describes the content of the feed. May be used in administrative user interfaces to assist subscribers in feed selection.

feedData
A single-valued complex attribute which contains feed information.

$ref A SCIM Reference attribute that contains a URI as defined by "feedType".

mode
A single-valued string which is one of the following values:

"post" - The publisher will pass SCIM Events using the HTTP POST option.

"poll" - The notification hub will poll for SCIM Events using HTTP GET at the specified URI ("feedPollUri").

The choice of "mode" is largely an internal decision based on the respective implementation architecture of the hub and its corresponding service providers that are publishing feeds. For many service providers with high rates of change, it may be limiting to store events for pick-up by a hub or subscriber (who is "polling"). In many cases service providers will normally prefer to simply POST events as generated to the hub so that they do not have to worry about persistence, etc. [TODO: is this a case for WebPUSH?]

feedPollUri
When "mode" is set to "poll", "feedPollUri" specifies the endpoint where the Notification hub will retrieve pending updates from the publishing SCIM Service Provider.

publisherJwk
The publishers optional JSON Web Key (see [I-D.ietf-jose-json-web-key]) that will be used to publish events. By registering a key, the Notification hub can authenticate events from SCIM Service Provider.

pollInterval
The optional period in seconds between event polls when the Notification Hub is set to poll for events from the SCIM Service Provider.

state
An optional value which indicates the current state of the feed which is:

"on" - the default setting indicates the notification hub is receiving events and will forward them to current feed

subscribers. If no subscribers exist or all subscribers have been notified, the events are deleted.

"pending" - indicates the notification hub is temporarily suspending delivery to subscribers. While in "pending" events may be posted and will be held for delivery by the hub until state returns to "on" (when events are subsequently delivered) or "fail".

"off" - indicates an administrator or publisher has requested the feed to stop delivery. While in off mode, the subscribers are maintained, but new events shall be ignored.

"fail" - usually used in connection with "polling" feeds. Indicates that the notification hub has been unable to retrieve events from the service provider for an extended period of time, or due to a call failure to the registered polling call back URI. [TODO: discuss whether a hub should continue to queue events in failed mode]

6.2.2. Feed Definition Operations

As with any SCIM resource, a notification Feed MAY be:

- Queried by using a SCIM HTTP GET request. In particular, subscribers may perform a GET to the "/Feeds" endpoint to discover available feeds.
- Updated by using a SCIM PUT request.
- Deleted using a SCIM DELETE request. Upon receiving a delete request, all corresponding notification subscriptions SHALL also be deleted. For this reason, instead of deletion, setting feed status to "off" is recommended.

6.2.3. Event POSTing

To create an event, a SCIM Event Publisher, simply performs an HTTP POST "/Events" appended to the Feed location URI. The body of the request includes a JSON object with the following attributes:

schemas A multi-valued string containing the value "

A non-normative example is as follows:

```
POST /Feeds/548b7c3f77c8bab33a4fef40/Events  HTTP/1.1
Host: notify.example.com
Accept: application/scim+json
Content-Type: application/scim+json
Content-Length: ...

{
"schemas":["urn:ietf:params:scim:schemas:notify:2.0:Feed"]
"eventToken":
  "eyJhbGciOiJub25lIn0
  .
  eyJwdWJsaXNoZXJVcmkiOiJodHRwczovL3NjaW0uZXhhbXBsZS5jb20iLCJmZWV
  kVXJpcyI6WyJodHRwczovL2podWIuZXhhbXBsZS5jb20vRmVlZHMvOThkNTI0Nj
  FmYTViYmM4Nzk1OTNiNzc1NCIsImh0dHBzOi8vamh1Yi5leGFtcGxlLmNvbS9GZ
  WVkcy81ZDc2MDQ1MTZiMWQwODY0MWQ3Njc2ZWU3Il0sInJlc291cmNlVXJpcyI6
  WyJodHRwczovL3NjaW0uZXhhbXBsZS5jb20vVXNlcnMvNDRmNjE0MmRmOTZiZDZ
  hYjYxZTc1MjFkOSJdLCJldmVudFR5cGVzIjpbIkNSRUFURSJdLCJhdHRyaWJ1dG
  VzIjpbImlkIiwibmFtZSIsInVzZXJOYW1lIiwicGFzc3dvcmQiLCJlbWFpbHMiX
  SwidmFsdWVzIjp7ImVtYWlscyI6W3sidHlwZSI6Indvcmsi LCJ2YWx1ZSI6Impk
  b2VAZXhhbXBsZS5jb20ifV0sInBhc3N3b3JkIjoibm90NHUybm8iLCJ1c2VyTmF
  tZSI6Impkb2UiLCJpZCI6IjQ0ZjYxNDJkZjk2YmQ2YWI2MWU3NTIxZDkiLCJuYW
  1lIjp7ImdpdmVuTmFtZSI6IkpvaG4iLCJmYW1pbHlOYW1lIjoiRG9lIn19fQ
  ."
}
```

In response, if the event token is validated, the server SHALL indicate successful submission by responding with:

```
HTTP/1.1 204 No Content
```

Since the normal operation of the notification hub is to forward events to registered subscribers, the Notification Hub is not obligated to inform the publisher of a permanent event URI that was created. Servers MAY allow HTTP clients to check for undelivered events by performing a GET against the same endpoint as the Event submission endpoint described above.

[TODO: Describe the error conditions and responses]

A SCIM Event Publisher MAY publish an event of type "CONFIRMATION" to the provided publication endpoint to confirm successful configuration. "CONFIRMATION" events SHALL NOT be passed on to subscribers.

6.2.4. Polling for Publisher Events

When a publisher registers a feed with a "mode" of "poll", the notification hub SHALL confirm configuration by performing an HTTP GET to the "feedPollUri" provided by the publisher during registration of the feed. On the first GET, the notification server should receive an event of type "CONFIRMATION". The confirmation event should contain an event with the "resourceUris" attribute set to a value that corresponds to the URI of the feed registration.

In its response to the notification polling for events by performing an HTTP Get to the "feedPollUri", the server shall construct a JSON message, with a schemas attribute consisting of the value: "urn:ietf:params:scim:api:messages:2.0:EventList" and a multivalued attribute of "eventTokens" containing one or more event tokens. The following non-normative example has been modified for brevity and readability:

```
HTTP/1.1 200 OK
Content-Type: application/scim+json
Location: https://example.com/FeedPoll/548b7c3f77c8bab33a4fef40
{
"schemas":["urn:ietf:params:scim:api:messages:2.0:EventList"],
"eventTokens":[
  "eyJhbGciOiJub25lIn0
  .
  eyJwdWJsaXNoZXJVcmkiOiJodHRwczovL3NjaW0uZXhhbXBsZS5jb20iLCJmZWV
  kVXJpcyI6WyJodHRwczovL2podWIuZXhhbXBsZS5jb20vRmVlZHMvOThkNTI0Nj
  FmYTViYmM4Nzk1OTNiNzc1NCIsImh0dHBzOi8vamh1Yi5leGFtcGxlLmNvbS9GZ
  WVkcy81ZDc2MDQ1MTZiMWQwODY0MWQ3Njc2ZWU3Il0sInJlc291cmNlVXJpcyI6
  WyJodHRwczovL3NjaW0uZXhhbXBsZS5jb20vVXNlcnMvNDRmNjE0MmRmOTZiZDZ
  hYjYxZTc1MjFkOSJdLCJldmVudFR5cGVzIjpbIkNSRUFURSJdLCJhdHRyaWJ1dG
  VzIjpbImlkIiwibmFtZSIsInVzZXJOYW1lIiwicGFzc3dvcmQiLCJlbWFpbHMiX
  SwidmFsdWVzIjp7ImVtYWlscyI6W3sidHlwZSI6Indvcmsi LCJ2YWx1ZSI6Impk
  b2VAZXhhbXBsZS5jb20ifV0sInBhc3N3b3JkIjoibm90NHUybm8iLCJ1c2VyTmF
  tZSI6Impkb2UiLCJpZCI6IjQ0ZjYxNDJkZjk2YmQ2YWI2MWU3NTIxZDkiLCJuYW
  1lIjp7ImdpdmVuTmFtZSI6IkpvaG4iLCJmYW1pbHlOYW1lIjoiRG9lIn19fQ
  ."]
}
```

6.3. Event Delivery

[TODO: Detail the event delivery extension point and define the following 3 methods]

6.3.1. Web Callback

6.3.2. Polling

6.3.3. Push Notification Extensions

7. Security Considerations

The synchronizing of User passwords between administrative domains is to be handled with great care. From a security perspective the re-use of passwords across service providers is to be highly discouraged. However, in the enterprise user experience, if the perception of the user is that service providers from multiple domains are part of a single corporate application environment, then password synchronization MAY be appropriate as part of an overall identity management and governance mechanism.

[TO BE COMPLETED]

8. IANA Considerations

8.1. SCIM Event Notification Mechanism Registry

TODO: Registration for Notification Mechanisms

8.2. SCIM Event Type Registry

TODO: Registration of Event Types

9. References

9.1. Normative References

[I-D.ietf-oauth-json-web-token]
Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", draft-ietf-oauth-json-web-token-32 (work in progress), December 2014.

[I-D.ietf-scim-api]
Hunt, P., Grizzle, K., Ansari, M., Wahlstroem, E., and C. Mortimore, "System for Cross-Domain Identity Management: Protocol", draft-ietf-scim-api-15 (work in progress), February 2015.

[I-D.ietf-scim-core-schema]
Hunt, P., Grizzle, K., Wahlstroem, E., and C. Mortimore, "System for Cross-Domain Identity Management: Core Schema", draft-ietf-scim-core-schema-17 (work in progress), March 2015.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC5988] Nottingham, M., "Web Linking", RFC 5988, October 2010.

9.2. Informative References

[I-D.ietf-jose-json-web-encryption]
Jones, M. and J. Hildebrand, "JSON Web Encryption (JWE)", draft-ietf-jose-json-web-encryption-40 (work in progress), January 2015.

[I-D.ietf-jose-json-web-key]
Jones, M., "JSON Web Key (JWK)", draft-ietf-jose-json-web-key-41 (work in progress), January 2015.

[I-D.ietf-jose-json-web-signature]
Jones, M., Bradley, J., and N. Sakimura, "JSON Web Signature (JWS)", draft-ietf-jose-json-web-signature-41 (work in progress), January 2015.

[pubsubhubbub]
Google, Inc, Google, Inc, Six Apart Ltd, and Notifixious Inc., "PubSubHubbub Core 0.4 -- Working Draft", .

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

Authors' Addresses

Phil Hunt (editor)
Oracle Corporation

Email: phil.hunt@yahoo.com

Morteza Ansari
Cisco

Email: morteza.ansari@cisco.com

Ian Glazer
Salesforce.com

Email: iglazer@salesforce.com