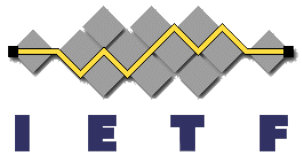
Control Plane Based on SCIM API

draft-hunt-secevents-stream-mgmt-api



Phil Hunt

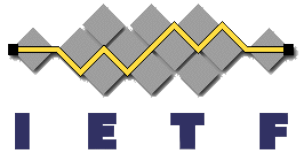
IETF100, Singapore

November, 2017

Why SCIM?

- Why not re-use?
- SCIM2 is a provisioning protocol
  - Based on JSON documents and RESTful API
  - Namespace extension model similar to SET
    - Used in IoT, Internet2, Applications, many none IDM areas
  - Resource life-cycle & referential integrity
- Good implementation and open source availability
  - > 20 SCIM2 libraries available
    - See: http://SimpleCloud.info

SCIM2 Industry Usage



Cloud Apps

ServiceNow

Concur

NetSuite

Jive

Workplace Facebook

Fusion Apps

GoTo Mtg

Google Apps

Workday

SalesForce

Docusign

Exchange

Dropbox

Sharepoint

WebEx

Cloud IDM

Oracle Identity Cloud & OIM

MS Cloud Security

Ping Identity

Microsoft Azure

Integrators

onelogin

Centrify

okta

gluu

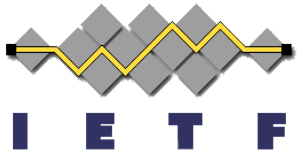
RADIANT LOGIC

One Global Identity to Drive Business in a Distributed World

SailPoint

Design Requirements

- A SECEVENTs Control Plane:
  - Provision and manage event streams
    - Endpoints, methods, events, security
  - Check functional status of streams
    - Is it working? Config problem or availability issue?
      - What failure errors matter?
  - Subject management
    - Provide extensible system for profile specific needs
    - Can model different types of subject stream relationships



draft-hunt-secevents-stream-mgmt-api

PROPOSAL

API Basics

- Resource as a life-cycle entity
  - E.g. a User or an EventStream
- Uses HTTP Methods as Operations defined by RFC7644
  - POST – To Create a Resource
  - GET – To Search and Retrieve
  - PUT – To Replace a Resource
  - PATCH – To Modify a Resource (JSON Patch like)
  - DELETE – To Remove a Resource
- JSON Based Documents as Resources
  - Defined by RFC7643
  - Attribute types, mutability, composites, visibility, canonicalization
  - Extensions
    - New resource types and object extensions
- Error Handling

Configuration Discovery

- /Schemas – Endpoint to look up schema URIs
  - Attributes in an object (e.g. EventStream)
  - Syntax, Mutability, Returnability, Canonicalization, Uniqueness, Optionality, Multi-valued, CaseExactness
    - Types: String, Boolean, Decimal, Integer, DateTime, Binary, Reference, and Complex (composite attributes)
- /ResourceTypes
  - Resource container endpoints (e.g. /EventStreams)
  - Primary Schema URI
  - Extension Schemas
- /ServiceProviderConfig
  - Basic SCIM service provider capabilities

Error Handling

- Robustness principle (as in Jon Postel/RFC793)
  - Try to accept what is understood (very different from XML/DSML/LDAP etc)
  - Service provider MAY interpret request, requestor accepts response
    - e.g. flexible negotiation of eventUris configured
- Basic HTTP Status Code Support
- HTTP 400 Errors
  - invalidFilter
  - tooMany
  - uniqueness
  - mutability
  - invalidSyntax
  - invalidPath
  - noTarget
  - invalidValue
  - invalidVers
  - sensitive (PII info)

Create Event Stream

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

{
  "schemas":["urn:ietf:params:scim:schemas:event:2.0:EventStream",
    "urn:ietf:params:set:method:HTTP:webCallback"],
  "feedName":"OIDCLogoutFeed",
  "eventUris_req":[
    "http://schemas.openid.net/event/backchannel-logout"
  ],
  "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",

  "urn:ietf:params:set:method:HTTP:webCallback":{
    ---method specific configuration items---
  }
}
```

Each delivery method can have its own specific configuration attributes*

Creation Response

```
HTTP/1.1 201 Created
Content-Type: application/scim+json
Location:
 https://example.com/v2/EventStreams/767aad7853d240debc8e3c962051c1c0
{
  "schemas":["urn:ietf:params:scim:schemas:event:2.0:EventStream"],
  "id":"767aad7853d240debc8e3c962051c1c0",
  "eventUris_req":["http://schemas.openid.net/event/backchannel-logout"],
  "eventUris":[
    "http://schemas.openid.net/event/backchannel-logout"
  ],
  "eventUris_avail":[
    "http://schemas.openid.net/event/backchannel-logout"
  ],
  "methodUri":"urn:ietf:params:set:method:HTTP:webCallback",
  "deliveryUri":"https://notify.examplerp.com/Events",
  "aud":"https://sets.myexamplerp.com",
  "status":"on",
  "maxDeliveryTime":3600,
  "minDeliveryInterval":0,
  "iss":"oidc.example.com"
  "iss_jwksUri":"https://example.com/keys/oidc-example-com.jwks"
  "description":"Logout events from oidc.example.com",
  "urn:ietf:params:set:method:HTTP:webCallback":{
     ---method specific configuration items---
  },
  "meta":{
     ... SCIM meta attributes ...
  }
}
```

Service Provider responds with accepted eventUris.

SP can advertise other available events

GET Status/Config Request

Retrieve Configuration

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

Check Status

```
GET /EventStreams/767aad7853d240debc8e3c962051c1c0&filter=status ne "on"
Host: example.com
Accept: application/json
Authorization: Bearer h480djs93hd8
```

Return only if there status is not "on" or totalResults=0

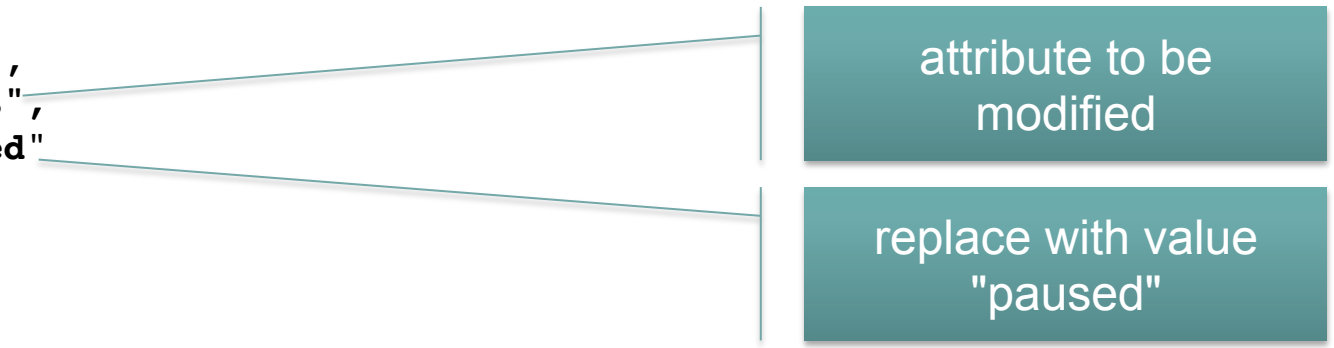
Status/Config Request

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

{
  "schemas":["urn:ietf:params:scim:schemas:event:2.0:EventStream"],
  "id":"767aad7853d240debc8e3c962051c1c0",
  "eventUris_req":[
    "http://schemas.openid.net/event/backchannel-logout"
  ],
  "eventUris":[
    "http://schemas.openid.net/event/backchannel-logout"
  ],
  "methodUri":"urn:ietf:params:set:method:HTTP:webCallback",
  "deliveryUri":"https://notify.examplerp.com/Events",
  "aud":"https://sets.myexamplerp.com",
  "status":"fail",
  "txErr":"connection",
  "txErrDesc":"TCP connect error to notify.examplerp.com.",
  "maxDeliveryTime":3600,
  "minDeliveryInterval":0,
  "description":"Logout events from oidc.example.com",
  "meta":{
     ... SCIM meta attributes ...
  }
}
```

Indicates a failure due to inability to connect

Updating Stream

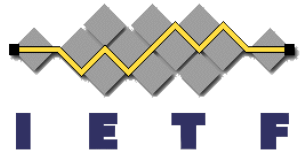
```
PATCH /EventStreams/767aad7853d240debc8e3c962051c1c0
Host: example.com
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Bearer h480djs93hd8

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



Subject Management

- Not necessarily a SECEVENTs requirement
  - RISC is an important use case
    - Should this be in RISC spec?
- 3 Example Models Documented in ID
  - Subjects as Attribute of Stream
  - Subjects as members of Group
  - Subjects as Resource (POST/DELETE Profile)

Subjects As Resource (POST Variant)



Request/Response for Subject Creation (register with an Event Stream):

```
POST /Subjects/ HTTP/1.1
Host: transmitter.example.com
Authorization: Bearer eyJ0b2tlbiI6ImV4YW1wbGUifQo=
{
 "email": "example.user@example.com"
 "streamId": "767aad7853d240debc8e3c962051c1c0",
 "schemas":["urn:ietf:params:scim:schemas:event:2.0
}

HTTP/1.1 201 Created
Content-Type: application/scim+json
Location:
    https://example.com/v2/Subjects/e3c962051c1c0
```

The subject's addition generates a record identifier so it is searchable and deleteable

To delete a subject use HTTP Delete

```
DELETE /Subjects/e3c962051c1c0
```

Subjects As Resource Variant

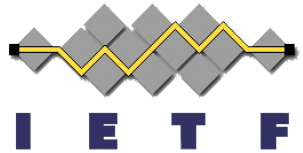
- Confirmation of membership
- Filter can match different types of subjects both simple and composite

```
GET /Subjects?filter=(streamId eq "e3c962051c1c0" and
email eq "example.user@example.com")
 → returns a match if present


GET /Subjects?filter=("iss" eq "gmail.com" and "sub" eq
"independentid")
 → returns all Streams with iss and sub match
 → can also add streamId qualifier
```

Stream Resource Variant

- Instead of "subjects" as its own endpoint, "subjects" defined as composite attribute of event stream resource enabling multiple types of subjects to be expressed
  - Email
  - OpenID (iss + sub)
  - SAML
  - Phone
  - User (as in SCIM User)
  - Group (as in SCIM Group)
  - URI (URI referenceable object – e.g. SOVRIN DID)
- Slightly more complex syntax (uses JSON Patch), but easier to map to internal systems …

Is EMail Subject in Stream?



REQUEST:

```
GET /EventStreams?filter=(subjects.value eq \
"alice@example.com")&attributes=id
Host: example.com
Accept: application/scim+json
Authorization: Bearer h480djs93hd8
```

RESPONSE: (no match)

```
HTTP/1.1 200 OK
Content-Type: application/scim+json
{
  "schemas":["urn:ietf:params:scim:api:messages:
2.0:ListResponse"],
  "totalResults":0,
  "Resources":[]
}
```

Is EMail Subject in Stream?

```
GET /EventStreams?filter=(subjects.value eq
"alice@example.com")&attributes=id
Host: example.com
Accept: application/scim+json
Authorization: Bearer h480djs93hd8
```

RESPONSE (Match Found):

```
HTTP/1.1 200 OK
Content-Type: application/scim+json
{
  "schemas":["urn:ietf:params:scim:api:messag
2.0:ListResponse"],
  "totalResults":1,
  "Resources":[
    {
      "id":"767aad7853d240debc8e3c962051c1c0",
    }
  ]
}
```

This is the "id" of the EventStream with a match

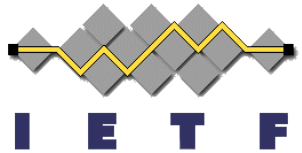
Confirming OIDC Subject in Stream

```
GET /EventStreams?filter=(subjects[value eq "123456"
and iss eq "op.example.com"])&attributes=id
Host: example.com
Accept: application/scim+json
Authorization: Bearer h480djs93hd8
```

Square brackets used to signal "inner" join..
Match of "value" and "iss" is against same record of subjects to avoid false matches

- "subjects" is a multi-valued composite attribute with sub-attributes
  - value – the value of the subject
  - iss – the issuer of the subject (used for OIDC)
  - type – the type of subject expressed in value

Adding Subject to Stream (EMail)




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

{
  "schemas":
    ["urn:ietf:params:scim:api:messages:2.0:PatchOp"],
  "Operations": [{
    "op":"add",
    "path":"subjects",
    "value":{
      "type":"EMAIL",
      "value":"alice@example.com"
    }
  }]
}
```

Since path is "subjects" value is a JSON object representing a record of subjects

Adding Subject to Stream (OIDC)

```
PATCH /EventStreams/767aad7853d240debc8e3c962051c1c0
Host: example.com
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Bearer h480djs93hd8
{
  "schemas":
    ["urn:ietf:params:scim:api:messages:2.0:PatchOp"],
  "Operations": [{
    "op":"add",
    "path":"subjects",
    "value":{
      "type":"OIDC",
      "value":"123456",
      "iss":"op.example.com"
    }
  }]
}
```

Supports OIDC's two part subject qualifier (iss and sub)

Removing Subject from Stream (OIDC)

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

{
  "schemas":
    ["urn:ietf:params:scim:api:messages:2.0:PatchOp"],
  "Operations": [{
    "op":"remove",
    "path":"subjects[value eq \"123456\" and iss eq
\"op.example.com\"]",
  }]
}
```

subject to be removed is selected by filter (as opposed to array index in JSON Patch)

Questions