SECEvents Control Plane Requirements for RISC

RISC F2F @ Amazon

Phil Hunt

Sept 25, 2017

Introduction

- We have a couple of very different proposals
  - https://www.ietf.org/archive/id/draft-hunt-idevent-distribution-01.txt
    - Unpublished: draft-hunt-secevent-stream-mgmt
  - https://tools.ietf.org/html/draft-scurtescu-secevent-event-stream-mgmt-api-00.txt

Tech Objectives

- A standard Control Plane to:
  - 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 extensions
    - Flexible model for defining contents of stream
    - Can subjects can be reported in a stream?
    - Is a subject registered?
    - Register or unregister subject (subjects?)

Parties (Users)

- Multiple entities may access a Stream CAPI
  - Monitor
    - A receiver may wish to check after a timeout period
    - An operations center monitors state of stream
  - Control
    - A security admin to alter or update configuration (credential roll-over, end-point change etc)
  - Manage
    - An automated provisioning system
      - E.g. setting up new federation (FASTFED?)
      - Configuring a new RP or a new IDP
    - Add/remove subjects from a stream

Architecture

- Simple/Single Service
  - All components of control implemented on a single web API server server
- Layered Services
  - Security functions and access control performed at a gateway layer and at web API server
  - Gateways look at TLS, HTTP headers, not the body
  - Application process HTTP headers and body
- Distributed vs. Single
  - A single endpoint for all streams (container) vs. full path for each stream
  - Actual endpoints might be in one location or available in multiple locations globally

URIs for Streams

- Individual stream paths must be possible
  - simple archs can use a common "container" URI
  - Do we allow for this [YES]
- Multi-credential access per stream configuration URI must be possible but not required
  - requirement for multiple actors
  - availability monitor not always the Event Receiver
  - provisioning and administration may be done by people not "servers"
- Use HTTP Methods not endpoints to define operations against stream URI resources? [YES]
  - multiple endpoints provide some flexibility but complicate security

RISC Subject Management

- 3 Variants
  - POST Variants
    - Operations implemented as POST operation
    - Two styles, POST+Command & HTTP Method
  - Modify Stream Variant

POST + Command Variant

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

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

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

{
 "phone_number": "+1 206 555 0123"
}
```

HTTP POST + DELETE Variants

- Use RESTful (HTTP Method) Model

```
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:Subject"]
}

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
```

HTTP POST+DELETE Variant

- Confirmation of membership

```
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 is a composite attribute 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 (JSON Patch), but easier to map to internal systems …

Confirming 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
```

** NO STREAMS FOUND THAT MATCH **

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

Confirming 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

** 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",
    }
  ]
}
```

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)

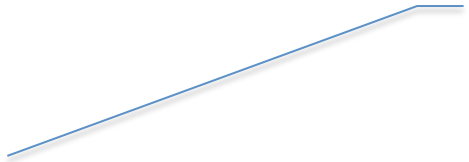
```
PATCH /EventStreams/767aad7853d240debc8e3c962051c1c0
Host: example.com
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Bearer h480djs93hd8

{
  "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)