Management API for SET Event Streams

Annabelle Backman – Amazon

IETF 99 – July, 2017
Specification Purpose

• In Scope:
  • Query stream status
  • Adding subjects to stream
  • Removing subjects from stream
  • Requesting verification of stream

• Out of Scope:
  • Receiver registration
  • Credential management
  • Delivery method configuration
  • Identification/authorization of receivers
Don’t Reinvent the Wheel

- RISC
- OIDC
- SCIM

Events

SET

- OpenID Connect
- OAuth 2.0
- SCIM

???
SET Event Streams

- Relationship between receiver and transmitter
- Specifies delivery method
- One per receiver and transmitter
Subject Identifier Object

- Set of claims in a JSON object
- Uniquely identifies subject
- SET profiling specs should declare these
Subject Identifier Object: Examples

{
    "iss": "http://oidc.example.com",
    "sub": "3435648645"
}

{
    "phone": "+12065551212"
}
Stream Management API

• Provided by transmitter
• Called by receiver
• Four HTTP endpoints:
  • Event stream management
  • Add subject
  • Remove subject
  • Verification
• May be shared by multiple streams
Reading Stream Configuration

• HTTP GET to Event Stream Management Endpoint
• Transmitter returns configuration as JSON object
Read Config.: Example Request

GET /set/stream HTTP/1.1
Host: transmitter.example.com
Authorization: Bearer eyJ0b2tlbiI6ImV4YW1wbGUifQo=
HTTP/1.1 200 OK
Server: transmitter.example.com
Content-Type: application/json; charset=UTF-8

{
    "aud": "http://www.example.com",
    "delivery": {
        "delivery_method": "https://.../http-push",
        "url": "https://receiver.example.com/events"
    },
    "events": [
        "https://schemas.openid.net/.../account-at-risk",
        "https://schemas.openid.net/.../account-locked",
        ...
    ]
}
Adding a Subject

- HTTP POST to Add Subject Endpoint
  - Contains Subject Identifier Object
- Transmitter returns 204 No Content response
- Transmitter may silently ignore request
  - Transmitter must not inform receiver
Add Subject: Example Request

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

{
    "email": "example.user@example.com"
}
Add Subject: Example Response

HTTP/1.1 204 No Content
Server: transmitter.example.com
Removing a Subject

• HTTP POST to Remove Subject Endpoint
  • Contains Subject Identifier Object

• Transmitter returns 204 No Content response
Remove Subject: Example Request

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

{
   "phone_number": "+12065551212"
}
Remove Subject: Example Response

HTTP/1.1 204 No Content
Server: transmitter.example.com
Verifying Stream Configuration

• HTTP POST to Verification Endpoint
  • Contains JSON object with optional state parameter
• Transmitter returns 204 No Content response
• Transmitter sends verification event over stream
• Verification is asynchronous
Verification: Example Request

POST /set/verify HTTP/1.1
Host: transmitter.example.com
Authorization: Bearer eyJ0b2tlbiI6ImV4YW1wbGUifQo=
Content-Type: application/json; charset=UTF-8

{
  "state": "VGhpcyBpcyBhbiBleGFtcGxlIHN0YXRl="
}

Verification: Example Response

HTTP/1.1 204 No Content
Server: transmitter.example.com
Verification: Example Event

{
  "jti": "123456",
  "iss": "https://transmitter.example.com",
  "aud": "receiver.example.com",
  "iat": "1493856000",
  "events": [
    "urn:ietf:params:...:core:verify": {
      "state": "VGhpcyBpcyBhbiBleGFtcGxlIHN0YXRl="
    },
  ]
}
Error Reporting

- API uses HTTP error codes

<table>
<thead>
<tr>
<th>Error Status</th>
<th>Usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>400</td>
<td>Request is invalid or cannot be parsed</td>
</tr>
<tr>
<td>401</td>
<td>Authorization failed</td>
</tr>
<tr>
<td>403</td>
<td>Receiver is not allowed to add/remove specified subject</td>
</tr>
<tr>
<td>404</td>
<td>Subject is not recognized</td>
</tr>
<tr>
<td>429</td>
<td>Receiver is sending too many requests</td>
</tr>
</tbody>
</table>
Questions