SET Event Distribution

Phil Hunt, Oracle
Marius Scurtescu, Google
October 2016

Introduction

- SET Event Tokens are a profile of JWT
  - Can be delivered by any number of means
  - Are URL safe – can be used in the 'front' channel
- Distribution draft
  - https://tools.ietf.org/html/draft-hunt-idevent-distribution-01
  - Assured/secured delivery in 'back' channel
    - Simple HTTP POST
  - Defines feed discovery and subscriptions mgmt
    - Uses SCIM (RFC7643/RFC7644)

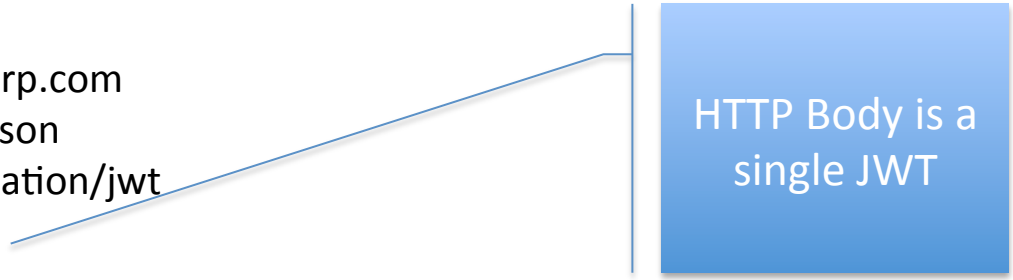
SET Delivery

- Thinking is to establish a registry for future methods
  - Share common metadata
  - Share common security / privacy concerns
- Initial draft will contain HTTPS POST (webcallback) method
  - Each POST delivers 1 message
  - HTTP Response is used to confirm receipt

Delivery Web Callback

POST /Events HTTP/1.1

Host: notify.examplerp.com
Accept: application/json
Content-Type: application/jwt



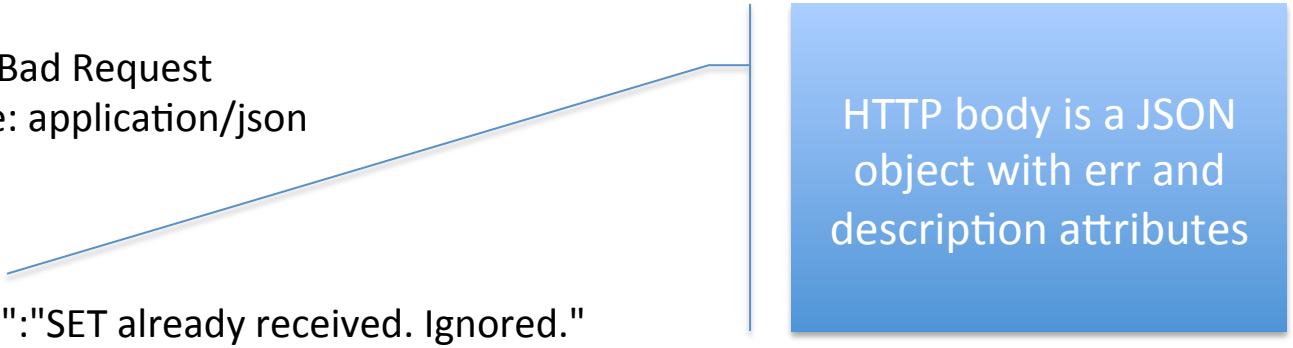
eyJhbGciOiJub25lIn0
.
eyJwdWJsaXNoZXJVcmkiOiJodHRwczovL3NjaW0uZXhhbXBsZS5jb20iLCJmZWV
kVXJpcyI6WyJodHRwczovL2podWIuZXhhbXBsZS5jb20vRmVlZHMvOThkNTI0Nj
FmYTViYmM4Nzk1OTNiNzc1NCIsImh0dHBzOi8vamh1Yi5leGFtcGxlLmNvbS9GZ
WVkcy81ZDc2MDQ1MTZiMWQwODY0MWQ3Njc2ZWU3Il0sInJlc291cmNlVXJpcyI6
WyJodHRwczovL3NjaW0uZXhhbXBsZS5jb20vVXNlcnMvNDRmNjE0MmRmOTZiZDZ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

POST Request Processing

- The POST contains one JWT
- Before responding the receiver must parse and validate the JWT
- The receiver acknowledges receipt by responding with HTTP Status 202 "Accepted"
- In addition to HTTP errors, the following 400 bad request errors are defined:
  - jwtParse, jwtHdr, jwtCrypto, jws, jwe, jwtAud, jwtIss, setType, setParse, setData, dup

Example Error Response

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

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

 }
```



Discussion - Batching

- Why Not
  - HTTP request/response processing very fast if connections properly managed
    - Very small perceived performance gain
  - Batching introduces complicated error handling semantics when a single event fails to validate
    - SCIM had large batch concerns, but adoption has been strongly in favour of single op per request. Bulk has not been adopted in production (AFAIK)
- Why?
  - Mass events such as domain or group level events where may people are affected brings a new use case to re-consider

How Do We Manage This?

- SCIM Provides RESTful capability to:
  - Lookup Feeds
  - Register for and manage subscriptions
  - Check the operational status of a subscription
- Why SCIM?
  - SCIM libraries are available for apps to implement server and client – you don't need a directory
  - Semantics on issues like nulls, defaults, schema are mature
  - Being considered as part of the FastFed work
  - We needed to profile/re-use something
    - The IESG was concerned about yet another protocol had recommended NETCONF as an alternative

Data Model

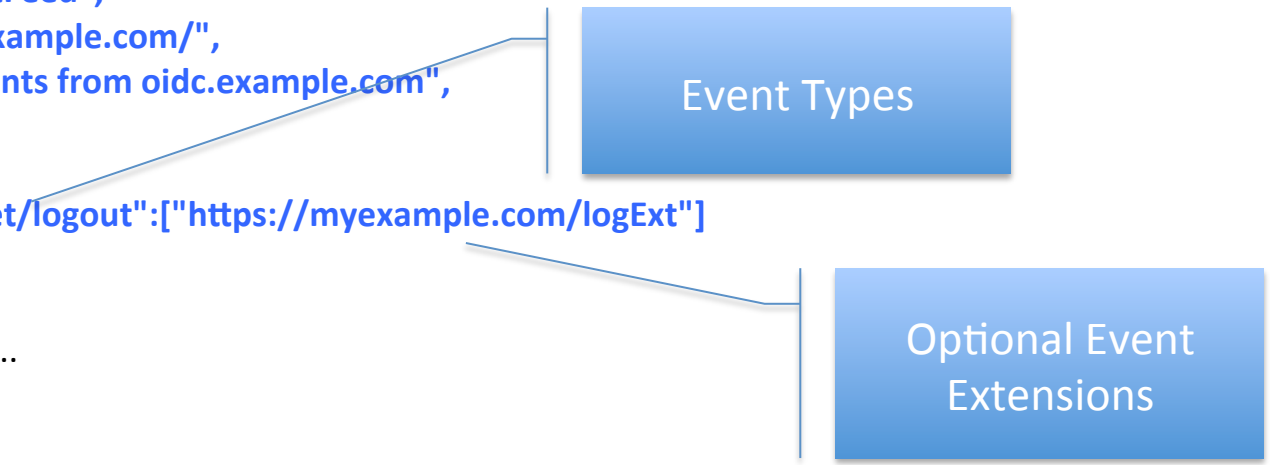
- Feeds
- Subscriptions

Feeds

- We do not define how feeds are actually built
- Simply track the metadata to allow subscribers to discover
- Attributes
  - feedName, feedUri, description, events, type, filter, deliveryModes (methods)

Example Feed Inquiry

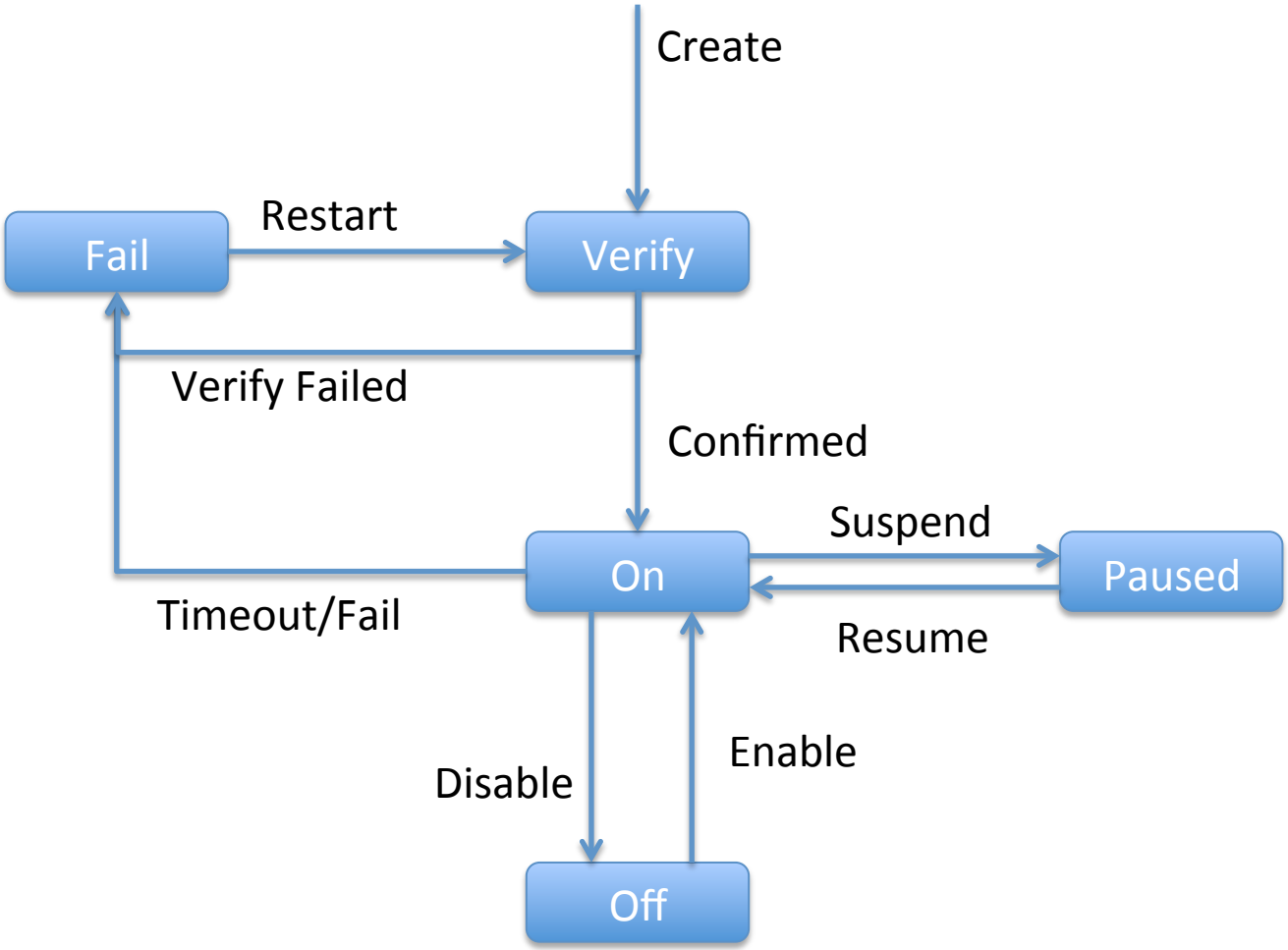
```
GET /Feeds/88bc00de776d49d5b535ede882d98f74
  Host: example.com
  Accept: application/scim+json
  Authorization: Bearer h480djs93hd8

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

{
  "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":["https://myexample.com/logExt"]
  ]
  "meta":{
     ... SCIM meta attributes ...
  }
}
```



Subscriptions

- Used to manage a subscription for a particular subscribing client
- Indicates method, endpoints, feed, status, JWT config, operational params
- Attributes:
  - feedUri, methodUri, deliveryUri, aud, feedJwk, confidentialJwk, subStatus, maxRetries, maxDeliveryTime, minDeliveryInterval

Subscription State



Subscribing to a Feed (SCIM Create)

```
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"
}
```

Subscription Response

```
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":"verify",
  "maxDeliveryTime":3600,
  "minDeliveryInterval":0,
  "description":"Logout events from oidc.example.com",
  "meta":{
     ... SCIM meta attributes ...
  }
}
```

Updating a Subscription (SCIM PATCH)

```
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"
  }]
}
```

Verifying a Subscription

- Confirms correct endpoint config
- Co-ordinate the start of feed / DevOps Co-ord
- Confirm clients acceptance of subscription
  - Prevent unauthorized subscription
  - DoS Prevention

Verification SET


```
{
    "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"
    }
  }
```

Verification Response (HTTP POST)

- To confirm, subscriber returns a simple JSON structure with the confirm challenge
  - Confirms correct JWT processing and endpoint
  - Upon receipt, publisher updates subStatus to "on"

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

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

Discussion