Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: January 1, 2018

P. Hunt, Ed.
Oracle
M. Ansari
Cisco

June 30, 2017

SCIM Use Cases for SECEVENTS
draft-hunt-secevent-usecases-00

Abstract

This specification defines the SCIM use cases for the SECEVENTs working group.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction and Overview

SCIM is a system intended for provisioning identities (such as enterprise users or consumers) and other objects across security domains to a cloud based service providers. SCIM defines an extensible JSON [RFC7643] document format and profiles HTTP protocol [RFC7644]. In practice, SCIM service providers are applications supporting pre-provisioning support, or may be a service provider directory upon which applications are integrated.

This document defines the operational requirements SCIM deployers have for the use of triggers, as defined in the SCIM Use Cases specification [RFC7642], and used in the form of security events and the requirements for management based on SCIM architectural assumptions.

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 inter-operability 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

This specification assumes terminology defined in the Security Event Token specification[I-D.ietf-secevent-token] .

This specification assumes terminology defined in the SCIM specifications, specifically [RFC7643] and [RFC7644]

This specification defines the following terms:

Directory
: Defined as any centralized repository of security objects shared by multiple applications. A SCIM Directory, though not formally defined is simply a directory that supports SCIM protocol.

2. SCIM Background

The SCIM Core Schema specification [RFC7643] is a profile of JSON [RFC7159] that defines attribute types, mutability, data formats, composites, and multi-value attributes as well as SCIM Service Provider feature and schema discovery metadata. As core schema defines standard resource types: Users and Groups which are common to most service providers. Each resource type establishes a common set of attribute definitions that can be mapped to SAML [saml-core-2.0] and to OpenID Connect [openid-connect-core] as well as application specific attributes. The core schema specification provides an extension mechanism which has been popular in:

- Being extended to describe many security objects such as OAuth Clients, Applications, IoT objects, among others.

- Enabling localized extensions to standard resource types (e.g. Users) without compromising inter-operability of existing implementations.

The SCIM Protocol specification [RFC7644] describes a RESTful profile of HTTP [RFC7231] that defines create, read, update and delete life-cycle for resources. The processing rules follow Jon Postel's "Robustness Principle" (see Section 2.10 [RFC761]) which help avoid many of the failings of previous XML based approaches. In particular the use of robust RESTful JSON helped ensure client and server ability to deal with inter-domain differences in schema, data, and implementation avoiding a lot of per implementation/deployment custom connector approaches.

SCIM clients use HTTP requests to SCIM service providers as follows to:

- Query for resources (users and groups) based on filters using HTTP GET or confidentially using HTTP POST.

- Retrieve specific resources using HTTP GET.

- Create new resources using HTTP POST.

- Replace a resource using HTTP PUT.

- Update a resource using HTTP PATCH. And,

- Delete a resource using HTTP DELETE.

The SCIM Protocol defines capabilities for:

- Complex or composite attributes that contain multiple values and the need to select and update specific values. This includes how to express sub-attributes and values in filters and the ability to change them as part of a resource. An example of a composite attribute in SCIM is: addresses (e.g. street name, city, country). Note: In SECEVENTs a corresponding example complex/composite attribute is an OpenID Connect user which is identified by both 'sub' and 'iss'.

- How to handle attributes that are immutable or read-only in the context of operations like PUT. How to handle attributes that are hashed or write-only and cannot be retrieved.

- Flexibility for web applications to take what they want without having traditional schema enforcement as with XML Schema.

- How to handle identifiers between clients and service providers and across domains.

- Referential stability of resources over time.

Some other relevant information:

- SCIM Polling Draft form Craig McMurtry [I-D.mcmurtry-scim-polling]

- Early SCIM Events proposal [I-D.hunt-idevent-scim]

3. High-Level Requirements

3.1. SCIM Event Trigger Requirements

SCIM's need for Security Events arises from a requirement for triggers identified in the SCIM Use Case specification [RFC7642]. Clients and service providers that operate across security domains have independent resource management that causes co-ordination and governance challenges between domains. The use of triggers is intended to alert clients (e.g. enterprises) of state changes within service providers that may be of interest to SCIM clients that may need to be co-ordinated or reconciled across domains.

As a general example, a change to a resource that occurs within a cloud software as a service (SaaS) provider generates an Event to be sent to a registered recipient via an Event Stream. Upon receipt of the event, the receiver performs a SCIM GET to obtain additional information and then decide if a local update or other action is required.

3.2. SCIM Security Model Considerations

Authentication and Authorization

SCIM follows normal authentication and authorization practices for HTTP (See Sections 2 and 7 [RFC7644]). In typical deployed cases, access to SCIM endpoints is managed by OAuth authorization in both cross-domain provisioning, delegated administration, and self-service applications. Many integrators also support basic authentication, and TLS mutual authentication. SCIM is often accessed in a couple of ways:

* End-user servers (e.g. as facilitated via a /Me endpoint) via a self-service web application or Javascript client.

- Administrative - where an administrator identity has access to groups of objects they are entitled to administer.

- Server-to-server, where identity provisioning systems implementing management workflows initiate commands across domains using OAuth enabled authorization.

PII Confidentiality

Querying using personally identifiable information (PII) causes privacy concerns when using HTTP GET. In typical HTTP usage, since HTTP [RFC7231] does not allow for query payloads on an HTTP GET, query parameters and filters are typically passed as part of the URL. When queries contain PII (most will in the case of RISC), there are security issues (e.g. leakage via audit logs and browser histories) relating to passing filter terms that contain PII in URLs. See [RFC7642] Security Considerations, section 7.5.2. From the perspective of SECEVENTs, the SCIM community has the same PII requirement that the management of SECEVENT streams and delivery not pass PII in request URIs.

Scale, PII, and Multi-Valued Data

One of the concerns the SCIM working group had when developing SCIM was the challenge that Groups (e.g. a group of users) will tend to get very big at Internet scale. The bigger a Group gets, the more expensive it is to enumerate. With a high change rate it quickly become impractical to do a simple PUT to replace an entire Group object due to the likely number of independent update conflicts that would occur. To avoid this, implementers often:

- Severely restrict when clients are actually authorized to return large objects (million member groups).

- Set access policy to allow search filters that confirm membership but avoid returning the members attribute (to avoid enumeration of all values).

- Use HTTP PATCH (a derivative of JSON Patch) to remove or add specific subjects without having to know the entire contents (e.g. the group).

3.3. Control Plane Assumptions

In the original SCIM identity event proposals, "Control Plane" functionality was accomplished by SCIM. SCIM protocol was proposed to configure and provision "streams" that deliver events via other protocols or profiles. The SCIM proposal allowed Event Receivers to check for delivery problems by retrieving Stream "resources" (which contain the stream configuration attributes) of which "status" is an

attribute that could be used to report operational state of a stream. Updates to Stream resource enable Event Recipients to do things like rotate credentials, or suspend streams. To initiate a verification to test a stream is functional, the Receiver or an authorized administrator can modify the Stream resource to "request" a verify by changing the value of "status" to "verify". In SCIM the subjects in a stream can be identified by a number of methods:

- Members of a Group

- The addition of a "streams" attribute to Users and other objects that may be part of a stream.

- An attribute or filter condition. E.g. the members of a Stream are defined by those Users with entitlements or roles containing a specific value (e.g. "entitlements" eq "CRM").

The SCIM WG in re-using SCIM as the control plane had assumed the following is already defined (and any alternative proposal would have to support):

- Defined processing of attributes based on type, mutability, etc for each HTTP method. For example, the handling of omitted attributes in a PUT or POST operation. Is a value intended to be defaulted or set to null?

- Handling of extensibility semantics as defined in the SCIM specifications such as the definition of new resource types (objects) and addition of new attributes by other profiling specifications.

- The ability of a service provider to override or modify client provider asserted values.

- Identifier and resource URI stability and referential integrity.

- Querying of subjects using various standard identifiers such as "id", "emails", "telephoneNumbers", etc. The ability to express composite queries such as "sub" and "iss" in a query.

- Ability to add and remove subjects from a group while keeping enumeration of that group from the client. Ability to confirm membership in a group without enumeration (facilitated through support for write-only/compare-only schema or access control).

- Standardized error control, handling and processing rules. See Section 3.12 [RFC7644] and [RFC7231].

3.4. Network and Protocol Operational Considerations

The SCIM WG discussed that transmission (now called data-plane or stream) can have much simpler semantics and error conditions and thus did not need to profile JSON beyond simple SET transfer (no need for attribute types, filters, etc). The SCIM WG also anticipated some varied requirements for delivery that include:

- PUSH delivery via HTTP POST (the generally preferred ideal solution).

- POLLING (to enable delivery across firewalls) using HTTP GET.

- PUSH delivery via messaging systems like APNS, GMS, SMS, etc - many of these had to do with provisioning and entitlement signals for mobile applications (e.g. WebEx). For example user contacts synchronization where after a change to a user's contact list, an application can receive an Event notification through the mobile platform's messaging solution as a trigger to fetch changes.

3.5. Dynamic Filtering Considerations

When defining filtered Streams, SCIM has to consider some special cases when the contents of a Stream is based upon a filter (query) to define which affected resources are included. For example, if the contents of a Stream is defined as Events related to resources where "emails.value sw "A"" and a resource is deleted, then the deleted resource won't match the filter anymore but notification may still need to be sent.

3.6. Directory and Application Provisioning

Network relationships for connections are typically:

- Enterprise Directory to Cloud Directory.

- Cloud Directory to Cloud Directory.

- Enterprise or Cloud to Cloud Application (applications used by many users).

- Enterprise or Cloud to Mobile Application (applications running on a device controlled by a single user).

An enterprise directory is typically (but not always) legacy-LDAP. In the cloud, a directory is simply any shared centralized profile store (e.g. Google Dir, Azure Directory/OpenGraph, SCIM Directory, etc). Important: While for many organizations LDAP remains the

center of administrative control, it is important to note that cloud directories and applications hold significantly more PII than enterprise directories. This creates a challenge for enterprise organizations to ensure proper governance and management of data given that a lot of cloud data is independently managed and updated.

As with an enterprise directory, a cloud directory is often shared by multiple applications. Cloud directories not only contain entitlement information but now also contain CRM data, contact, credentials, personalization and localization data, social network data, etc (the list goes on). While some cloud providers centralize others are tenancy structured with different directory endpoints per tenancy (e.g. Oracle).

As described above, because data, particularly PII, is being independently managed across multiple domains, there is a need to generate change signals (events) from cloud based directories and applications back to the enterprise. This was originally identified in the SCIM Use Cases (see Section 2.2.1 [RFC7642]).

4. Use Cases

The following use cases are expressed in terms of the direction of flow of events. In typical SCIM cases, there is only 1-way event exchange. Typical usage of events is to act as a "trigger" (see [RFC7642]) to let a receiver know that an event has occurred in the transmitter's domain that may require action on the part of the receiver. Events can be simple resource changed events, to higher level account status and change events (e.g. account or password reset). While many events are similar to OpenID RISC proposed events, a major distinction is that SCIM events are often triggered by user, administrative, or workflow provisioning action rather than a risk analytical engine (e.g. that might detect suspicious activity).

4.1. Scenario 1[P0]: Cloud-to-Enterprise PUSH and Cloud-to-Cloud PUSH

Pre-conditions:

The Event Receiver already has SCIM access to the Event Transmitter service provider. This includes HTTP credentials and endpoint.

Event Receivers and Transmitters can agree out-of-band on SET/JWT security requirements including use of signing and/or encryption to be documented in a Stream Configuration.

```
                      +----------------+--------+
                      |      SCIM      |  SCIM  |
                      |Service Provider| Events |
                      |                | Stream |
                      +--------^-------+--------+
                           SCIM|          |Events via
                       Commands|          |HTTP POST
                               |          |
                               |          |
                               |          |
                               |          |
                             +-+----------v-+
                             | SCIM Client  |
                             | Provisioning |
                             |  Controller  |
                             +--------------+
```

Figure 1: SCIM Provisioining with PUSH Triggers

In Figure 1, the SCIM client initiates RESTful SCIM commands to a SCIM service provider. In addition to provisioning security objects such as Users and Groups, the client also uses SCIM to provision Event Streams in order to receive Events to an endpoint the provisioning controller requests. The service provider MUST be able to POST to the client's domain. Usually this means the client is able to have a public HTTP endpoint available to receive SET events.

Stream Creation Flow:

To create a Stream, the Event Receiver (or an administrator) uses their SCIM access credential to access the SCIM endpoint and creates a Stream resource configuration:

```
POST /Streams
Host: scim.bighost.com
Authorization: Bearer h480djs93hd8
{ "receiverId":"<client-id>",
  "method":"webCallBack",
  "receiverUri":"https://set.example.com/events/",
  "aud":"<client-id>",
  "type":"SCIM",
  "receiverJwkUri":"<receiver's public key url>",
  "authorization":"<btoken|BasicAuth>"
}
```

Figure 2: Stream Creation Operation

Note: If the Transmitter does not have an HTTP credential to send events, the receiver should include one in its registration POST request or negotiate one out-of-band.

In the stream configuration there is likely a definition as to what types of events (event families) and which subjects constitute the feed. In SCIM this will likely be a group of objects, or filter condition such as "roles" eq "CRM_Users". This is likely based on the relationship between parties that determines which entities are provisioned between domains.

Upon successful creation of the Stream, the SCIM Event Transmitter Responds with:

```
HTTP/1.1 201 Created
Location: https://events.bighost.com/Streams/2819c223-7f76-453a
{ "receiverId":"<client-id>",
  "method":"webCallBack",
  "receiverUri":"https://set.example.com/events/",
  "aud":"<client-id>",
  "type":"SCIM",
  "receiverJwkUri":"<receiver's public key url>",
  "authorization":"<btoken|BasicAuth>",
  "status":"on"
}
```

Note that in the above figure, the Location URI is the fixed reference to the Stream for as long as it exists. Administrative users and Event Receiver entities MAY use the location to check status or update configuration as needed.

Figure 3: Stream Creation Response

[[TBD, the event receiver, needs to issue the event transmitter a credential in order for it to issue HTTP POSTs to the Event Receivers callback endpoint. In some cases there may be an existing OpenID Connect relationship but in most cases this not expected - especially in directory-to-directory synchronization scenarios.]]

Stream Verification:

During the initial stream creation request and at any point the transmitter deems appropriate (e.g. as a ping), the transmitter verifies configuration by sending a verification event to the receiver that demonstrates the receiver:

- is willing accept the event, and

- is able to parse the event - especially if encrypted.

Conversely an Event Receiver should be able to initiate a verification request and may provide a confirmation challenge and nonce to verify the relationship from the Event Receiver's perspective.

Delivery:

Delivery is accomplished by doing a simple HTTP POST to the registered endpoint of the receiver. The payload of the POST is application/jwt and contains a single JWT (which is actually a SET).

Before responding with a 2xx success message, the receiver should ensure it was able to read and validate the SET. If the transmitter receives a 2xx response, the transmitter may assume the event was successfully delivered.

A set of Status 400 error conditions are defined which the receiver can use to indicate various JWT validation conditions.

4.2. Scenario 2[P0]: Cloud-to-Enterprise POLLING

Pre-conditions:

The Event Receiver already has SCIM access to the Event Transmitter service provider. This includes HTTP credentials and endpoint.

Event Receivers and Transmitters can agree out-of-band on SET/JWT security requirements including use of signing and/or encryption to be documented in a Stream Configuration.

The Event Receiver is unable to open an endpoint to receive SETs inside the firewall.

```
                    +----------------+--------+
                    |      SCIM      |  SCIM  |
                    |Service Provider| Events |
                    |                |        |
                    +--------^-------+--^-----+
                         SCIM|          |Events via
                     Commands|          |HTTP GET Long Poll
                             |          |& POST Acks
                Firewall     |          |
               +---------------------------------------+
                             |          |
                           +-+----------+-+
                           | SCIM Client  |
                           | Provisioning |
                           |  Controller  |
                           +--------------+
```

Figure 4: Event Delivery with Firewall

In Figure 4, the SCIM client initiates RESTful SCIM commands to a SCIM service provider. In addition to provisioning security objects such as Users and Groups, the client also uses SCIM to provision Event Streams in order to receive Events to an endpoint the provisioning controller requests. In this case, the SCIM Client "polls" for events using HTTP GET. The client MAY request immediate response based on a timed schedule, or the client MAY use HTTP Long Polling to wait for SETs as they become available.

Stream Creation Flow:

The Event Receiver uses their SCIM credential to access the SCIM service provider endpoint to create a Stream resource by performing a POST

```
POST /Streams
Host: scim.bighost.com
Authorization: Bearer h480djs93hd8
{ "receiverId":"<client-id>",
  "method":"POLLING",
  "aud":"<client-id>",
  "type":"SCIM",
  "receiverJwkUri":"<receiver's public key url>"
}
```

Figure 5: Create Polling Stream

It is assumed, but may not always be true. that the POLLING receiver can simply use their SCIM credential to perform HTTP GETs to the polling endpoint. Additional parameters will likely need to be defined to control polling rate, number of events in a message, etc.

Note, in the stream configuration there is likely a definition as to what types of events (event families) and which subjects constitute the feed. In SCIM this will likely be a group of objects, or filter condition such as "roles" eq "CRM_Users". This is likely based on the relationship between parties that determines which entities are provisioned between domains.

Upon successful creation of the stream, the transmitter responds to the receiver with:

```
HTTP/1.1 201 Created
Location: https://events.bighost.com/Streams/2819c223-7f76-453a
{ "receiverId":"<client-id>",
  "method":"POLLING",
  "receiverUri":"https://set.bighost.com/Events/2819c223-7f76-453a",
  "aud":"<client-id>",
  "type":"SCIM",
  "receiverJwkUri":"<receiver's public key url>",
  "status":"on"
}
```

Figure 6: Polling Stream Creation Response

In the above response, the transmitter indicates to the receiver where to poll for events by setting a value for "receiverUri". This endpoint does not need to be SCIM compliant and can be a generic (e.g. shared by all polliers) endpoint such as "https://events.bighost.com".

Stream Verification:

Same requirements are for Scenario 1 (see Section 4.1).

Delivery:

Delivery is accomplished by having the Event Receiver initiate an HTTP request that causes a response such as:

```
{
 "sets":{
 "4d3559ec67504aaba65d40b0363faad8":
   "eyJhbGciOiJub25lIn0
   .
   e3sgIAogICJqdGkiOiAiNGQzNTU5ZWM2NzUwNGFhYmE2NWQ0MGIwMzYzZmFhZDgiLAog
   ICJpYXQiOiAxNDU4NDk2NDA0LAogICJpc3MiOiAiaHR0cHM6Ly9zY2ltLmV4YW1wbGUu
   Y29tIiwgIAogICJhdWQiOiBbCiAgICJodHRwczovL3NjaW0uZXhhbXBsZS5jb20vRmVl
   ZHMvOThkNTI0NjFmYTViYmM4Nzk1OTNiNzc1NCIsCiAgICJodHRwczovL3NjaW0uZXhh
   bXBsZS5jb20vRmVlZHMvNWQ3NjA0NTE2YjFkMDg2NDFkNzY3NmVlNyIKICBdLCAgCiAg
   CiAgImV2ZW50cyI6IHsKICAgICJ1cm46aWV0ZjpwYXJhbXM6c2NpbTpldmVudDpjcmVh
   dGUiOiB7CiAgICAgICJyZWYiOgogICAgICAgICJodHRwczovL3NjaW0uZXhhbXBsZS5j
   b20vVXNlcnMvNDRmNjE0MmRmOTZiZDZhYjYxZTc1MjFkOSIsCiAgICAgICJhdHRyaWJ1
   dGVzIjpbImlkIiwgIm5hbWUiLCAidXNlck5hbWUiLCAicGFzc3dvcmQiLCAiZW1haWxz
   Il0KICAgIH0KICB9Cn0",
  "<nextJti>":"<nextJwt>"
 },
 "since":1458496025
}
```

Figure 7: Example Polling Response

In the above JSON object is a JSON attribute "sets" whose value is a JSON object that contains a set of JSON attributes that correspond to each event's JTI value. the value for each attribute is the actual encoded SET.

In addition to the "sets" attribute, a "since" attribute indicates the timestamp of either the last event previously transmitted or potentially oldest event in the current payload (To be discussed).

In order to acknowledge receipt, the receiver must successfully parse each message and respond by doing an HTTP POST back to the events endpoint using something along the lines of the following JSON structure:

```
   {
     "ack":[
       "39e48e70e9f84d90b5fdbf2fbd826219",
       "8e1ed13b871547ffa332f7027a0fdd91",
       "0a02c62529e34541a8b3c5c7941fa545"
     ]
     "setErrrs":{
       "3d0c3cf797584bd193bd0fb1bd4e7d30":{
         "err":"dup",
         "description":"SET already received. Ignored."
       }
     }
   }
```

Figure 8: Poll Acknowledgement Response

In the payload above the receiver indicates which SET event JTIs have been accepted, and which SETs had errors using "accepts" and "setErrs".

It is expected that because most errors are due to JWT crypto configuration errors, that most responses will tend to be all errors or all accepts.

If a transmitter receives what it deems an unrecoverable error, or a receiver fails to poll for events, the transmitter can set the stream state to "failed" with an appropriate error indicator.

4.3. Scenario 3[P2]: Cloud-to-Mobile Application PUSH

This scenario is a hybrid of scenario 1 and 2. The scenario uses mobile message delivery services (APNS, GMS, SMS) to deliver events. Typically a stream has only one subject in its feed. The events are used to notify client applications about changes to entitlements, or other configuration (e.g. new tenancy endpoints)that might be useful to user experience.

As in the polling method in Scenario 2, to acknowledge events, the mobile app will need to use the POST (as defined in Scenario 2) to acknowledge SET delivery. To be discussed, this might not be necessary if assured delivery is not required.

5. Security Considerations

None as this is a use case document to describe considerations.

6. Privacy Considerations

None as this is a use case document to describe considerations.

7. IANA Considerations

There are no IANA considerations.

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

8.2. Informative References

[I-D.hunt-idevent-scim] Hunt, P., Denniss, W., and M. Ansari, "SCIM Event Extension", draft-hunt-idevent-scim-00 (work in progress), March 2016.

[I-D.ietf-secevent-token] Hunt, P., Denniss, W., Ansari, M., and M. Jones, "Security Event Token (SET)", draft-ietf-secevent-token-00 (work in progress), January 2017.

[I-D.mcmurtry-scim-polling] McMurtry, C., "SCIM Polling Protocol", draft-mcmurtry-scim-polling-01 (work in progress), April 2016.

[idevent-scim] Oracle Corporation, "SCIM Event Extensions (work in progress)".

[openid-connect-core] NRI, "OpenID Connect Core 1.0", Nov 2014.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <http://www.rfc-editor.org/info/rfc3339>.

[RFC7159] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", RFC 7159, DOI 10.17487/RFC7159, March 2014, <http://www.rfc-editor.org/info/rfc7159>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <http://www.rfc-editor.org/info/rfc7231>.

[RFC761] USC, "TRANSMISSION CONTROL PROTOCOL", January 1980.

[RFC7642] LI, K., Ed., Hunt, P., Khasnabish, B., Nadalin, A., and Z. Zeltsan, "System for Cross-domain Identity Management: Definitions, Overview, Concepts, and Requirements", RFC 7642, DOI 10.17487/RFC7642, September 2015, <http://www.rfc-editor.org/info/rfc7642>.

[RFC7643] Hunt, P., Ed., Grizzle, K., Wahlstroem, E., and C. Mortimore, "System for Cross-domain Identity Management: Core Schema", RFC 7643, DOI 10.17487/RFC7643, September 2015, <http://www.rfc-editor.org/info/rfc7643>.

[RFC7644] Hunt, P., Ed., Grizzle, K., Ansari, M., Wahlstroem, E., and C. Mortimore, "System for Cross-domain Identity Management: Protocol", RFC 7644, DOI 10.17487/RFC7644, September 2015, <http://www.rfc-editor.org/info/rfc7644>.

[saml-core-2.0] Internet2, "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0", March 2005.

Appendix A. Acknowledgments

The editors would like to thanks the members of the SCIM WG which began discussions of provisioning events starting with: draft-hunt-scim-notify-00 in 2015.

The editor would like to thank the participants in the the SECEVENTS working group for their support of this specification.

Appendix B. Change Log

Draft 00 - PH - Initial draft

Authors' Addresses

Phil Hunt (editor)
Oracle Corporation

Email: phil.hunt@yahoo.com

Morteza Ansari
Cisco

Email: moransar@cisco.com