Internet Engineering Task Force
Internet-Draft
Intended status: Standards Track
Expires: December 23, 2019

M. Wahl, Ed.
Microsoft Corporation

June 21, 2019

SCIM Profile for Provisioning Users Into Relying Party Applications
draft-wahl-scim-profile-00

Abstract

This document specifies a profile of the System for Cross-Domain Identity Management Protocol (SCIM). This profile defines how an identity provider, acting as a SCIM client, can notify a relying party application of changes to user accounts.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction 2
   1.1. Requirements Language 4
2. General Considerations 4
   2.1. Endpoints and Payload 4
   2.2. Authentication and Authorization 5
   2.3. SCIM Servers With Multiple Identity Providers and Multiple tenants 5
   2.4. Case Sensitivity in Payloads 6
   2.5. Overlapping and Throttling Requests 6
   2.6. Bulk Operations and Service Provider Configuration . . . 7
   2.7. Error Results 7
   2.8. Non-Requirements 7
3. Minimum Schema and Schema Discovery 7
   3.1. Additional Considerations for the ID Attribute 9
   3.2. Additional Considerations ExternalId Attribute 9
   3.3. Resources for Schema Discovery 10
4. Operations on Users 12
   4.1. When a User Account is added in the Identity Provider . . 12
      4.1.1. Optionally locating a User by ExternalId 12
      4.1.2. Creating a User with POST 13
   4.2. When a User Account's Attributes Change 14
      4.2.1. Retrieving a User by ID 15
      4.2.2. Modifying a User with PATCH 16
      4.2.3. Indicating User Sign In Blocked 18
   4.3. When the User's Account is to be Removed 19
      4.3.1. Removing a User with DELETE 19
   4.4. User Account Retrieval from the Relying Party 20
      4.4.1. Retrieving Multiple Users in a Single Request 20
5. Security Considerations 20
6. IANA Considerations 21
7. References 21
   7.1. Normative References 21
   7.2. Informative References 22
Author's Address 22

1. Introduction

Relying party applications implement single-sign on protocols, such as SAML or OpenID Connect, for user authentication from an identity provider, and often store one or more attributes about each of user identities from those identity providers. The term relying party is defined in OpenID Connect [OPENID.OIDC]. The single sign in flows between an identity provider and a relying party often allow the identity provider to include one or more attributes about a user signing in, such as their name, at the time of the sign in event. However, those attributes might also change, and those changes be

relevant, to a relying party application, even when no user is signing in. For example, if an identity provider deletes a user, then the relying party application would never see a subsequent sign in event from that user.

The SCIM protocol RFC 7644 [RFC7644] is an application-level, REST protocol for provisioning and managing identity data on the web. SCIM can be leveraged for many use cases, including transfer of attributes about users to a relying party application (see RFC 7643 [RFC7643] section 3). This profile describes how SCIM can be used between an identity provider and a relying party application, so the identity provider can notify a relying party application acting as a SCIM server of changes to user accounts, out of band from the normal sign-in flow of those users. . This profile defines the interactions between an identity provider as the initiator, a SCIM client, and a replying party application, as the SCIM server, in the following scenario:

o A replying party application has a database of user resources with attributes of those users relevant to that application. It also trusts one or more identity providers to authenticate users on its behalf and provide claims about that user. It also copies the values from some of those claims from those identity providers into its database.

o An identity provider has its own distinct database of user resources. That database is the source of its claims which it provides to a relying party application. Normally this is done through a sign-in flow using a protocol such as SAML or Open ID Connect.

o The identity provider also wishes to keep the relying party application up to date, even when no user is signing in to that replying party, and so uses SCIM to send changes to the application.

Based on this profile, the identity provider acts as a SCIM client and sends the changes via the SCIM protocol to the application acting as a SCIM server. The relying party application represents each user account in its database which is visible to the identity provider, as a SCIM User resource. This enables the relying party application to have a more up-to-date copy of users with its identity provider, than if it was only being updated when a user signs in.

For example, if the identity provider deletes a user, this deletion event can be transferred to the relying party application via SCIM, so that the application also cleans up any data associated with that user -- who won't be accessing that application in future. Or, for

another example, if the user changes their name in the identity provider, then this change can be sent to the application, so that the user's name is displayed correctly within the application to other users.

This profile is not intended to be a comprehensive or bidirectional replication protocol; instead, it provides basic consistency for user resources sent from an identity provider to a relying party, necessary for a user to be able to sign into the relying party. Management of other resource types besides users is outside the scope of this profile.

1.1. Requirements Language

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 RFC 2119 [RFC2119].

2. General Considerations

A SCIM client in an identity provider will, either upon specific changes in its database (e.g., when there is a new user), or at intervals, send queries and changes to the relying party application's SCIM server. This section covers the basic requirements for a SCIM client and server implementation. The sections following this one cover the schema requirements, the pattern of operations in this profile, and security considerations.

2.1. Endpoints and Payload

A SCIM client and server MUST support HTTPS. A SCIM client and SCIM server MUST have previously agreed on the HTTPS URI for the SCIM server's "Users" endpoint, below the Base URI (Base URI is defined in section 1.3 of RFC 7644 [RFC7644]). Additionally, the client and server MAY have also previously agreed on the HTTPS URIs for the SCIM server's "ResourceTypes" and "Schemas" endpoints. These URI MAY contain the string "v2", for example "https://api.example.com/scim/tenant/example.org/v2/Users" might be a base endpoint for Users in a particular application.

A SCIM client implementing this profile MUST use JSON-structured request bodies and expect JSON-structured responses using the UTF-8 RFC 3629 [RFC3629] encoding, in the Content-Type "application/scim+json".

These are to be sent via HTTPS using TLS, RFC 8446 [RFC8446].

2.2. Authentication and Authorization

This profile is intended for service-to-service communication: a SCIM client is not acting under the context of an individual person, it is operating on behalf of the identity provider.

A SCIM client and a SCIM server SHOULD support one or more authentication mechanisms to authenticate a SCIM client to a SCIM server. One mechanism SHOULD be bearer tokens.

If a SCIM client supports bearer tokens, then the client SHOULD include the token in the Authorization header of each request, as described in section 2.1 of RFC 6750 [RFC6750]. This profile does not cover establishing common authentication keys, which is assumed to occur out of band.

A SCIM server does not need to support operations by un-authenticated clients. A SCIM server MUST only permit a client to query, create, update and delete one or more User resources, after the token is validated.

2.3. SCIM Servers With Multiple Identity Providers and Multiple tenants

If a SCIM server supports multiple tenants, then the operations defined in this profile are relative to a single tenant. If a SCIM server is part of an application which is affiliated with multiple independent identity providers, and the operator of that SCIM server permits each tenant to choose its identity providers, then that SCIM server MUST ensure that the operations performed by one SCIM client on behalf of one tenant do not affect another tenant.

A SCIM server SHOULD give each SCIM client a distinct identity and authorization information.

A SCIM server SHOULD ensure that within a single tenant, or within the application if the application is not multi-tenanted, each SCIM client's updates do not affect those of other SCIM clients.

There are several ways for a SCIM server to achieve that. For example, a SCIM server might have each tenant or each SCIM client use a unique User endpoint.

For example, if the application had tenants example.com and example.org, and identity providers A and B, and tenant example.com used identity providers A and B but tenant example.org only allowed identity provider B, then identity provider A would be instructed to use endpoints /example.com/Users and /example.org/Users, and identity provider B would be instructed to use endpoint /example.org/Users.

Or, in another approach, a SCIM server MAY record which SCIM client identity created each user resource, and only permit user resources to be queried, updated or deleted by the same client.

2.4. Case Sensitivity in Payloads

As defined in RFC 7644 [RFC7644], attribute names and attribute operators used in filters are case insensitive.

An implementation of this protocol SHOULD treat all other ABNF US-ASCII strings as case insensitive. For example, in the following PATCH body

```
  {
     "schemas":
      ["urn:ietf:params:scim:api:messages:2.0:PatchOp"],
     "Operations":[{
     "op":"remove",
     "path":"emails[type eq \"work\"]"
     }]
   }
```

The case of these strings "schemas", "Operations", "op", "remove" and "path" are not significant.

Most attribute values are case insensitive when matching, however the values of the "id" attribute and the "externalID" attribute are case sensitive.

2.5. Overlapping and Throttling Requests

A SCIM client MAY send multiple requests to a SCIM server, and need not wait for one a response prior to sending another requests, with one exception. A SCIM client SHOULD NOT send another PATCH request for a particular User resource, if there was a PATCH request sent for that same resource in the last 60 seconds for which the client did not receive a response. A SCIM server need not process multiple operations in the order they were sent by the client.

A SCIM server MAY throttle clients who send too many simultaneous requests or too many requests in a short time interval, by returning status code 429, as defined in RFC 6585 [RFC6585], to subsequent requests. A SCIM client which receives status code 429 SHOULD wait before retrying that request.

2.6. Bulk Operations and Service Provider Configuration

A SCIM server MAY support bulk operations, and if it does SHOULD support querying the service provider configuration. A SCIM server is not required to support bulk operations or the service provider configuration, and a server which does not support them SHOULD return either an error or an empty result for a query for the ServiceProviderConfig.

A SCIM client MAY query a SCIM server to determine if it supports bulk operations, by sending a GET for ServiceProviderConfig, as described in section 4 RFC 7644 [RFC7644]. If a SCIM server returns an error to this query, then a SCIM client SHOULD assume that SCIM server does not support bulk operations, and send subsequent requests not using bulk operations.

2.7. Error Results

If a SCIM server implementing this profile encounters an error when processing a request from a client, it SHOULD transmit the error in the HTTP status code of the response, as described in section 3.2 of RFC 7644 [RFC7644] and RFC 6585 [RFC6585].

2.8. Non-Requirements

A SCIM server implementing this profile MAY implement these or other features, but a SCIM client SHOULD NOT assume they are supported unless it has determined otherwise.

o HTTP cookies for authentication, or authentication not related to OAuth (from RFC 7644 [RFC7644] section 2)

o sorting, paging or queries via POST (from RFC 7644 [RFC7644] sections 3.4.2 and 3.4.3)

o updates via PUT (from RFC 7644 [RFC7644] sections 3.5.1)

o the Me URI fragment (from RFC 7644 [RFC7644] sections 3.11)

o eTags (from RFC 7644 [RFC7644] sections 3.14)

o anonymous requests (from RFC 7644 [RFC7644] sections 7.6)

3. Minimum Schema and Schema Discovery

SCIM defines a user schema in RFC 7643 [RFC7643]. A SCIM server implementing this profile need not implement the full set of

attributes of that schema, since those attributes are not necessarily relevant to the application.

A SCIM server that supports a client creating users MUST recognize the User resource schema, as described in section 4.1 of RFC 7643 [RFC7643], but does not need to support all of the attributes.

A SCIM server SHOULD recognize the Enterprise User schema extension as described in section 4.3 of RFC 7643 [RFC7643].

From these schemas,

o A SCIM server MUST provide the attribute "ID" for all users.

o A SCIM server that supports a client creating users SHOULD support the attribute "userName" being supplied by a SCIM client when creating a user, or when the client is modifying a user.

o A SCIM server SHOULD support the attribute "active", as it allows a SCIM client to temporarily de-activate a user.

o A SCIM server SHOULD support the attribute "displayName", with values of at least 128 characters in length, being included when a user resource is created and subsequently changed.

A SCIM server SHOULD NOT require any additional attributes besides userName to be supplied by a SCIM client when creating a user.

If email addresses are relevant to a SCIM server application,

o That SCIM server SHOULD support the multi-valued attribute "emails", for a client providing one value.

o That SCIM server MAY support a client providing more than one value. As there is no way to indicate this at present, a SCIM client SHOULD NOT require a SCIM server to support multiple values.

o That SCIM server SHOULD support each value of "emails" having sub-attributes "primary", "type" and "value", and SHOULD support sub-attribute "type" having a value of "work".

For other attributes, if they are relevant to the application, and can be supplied by the identity provider, then that application's SCIM server MAY support storing them. In particular,

o A SCIM server MAY support the attribute "externalId", with values of at least 64 characters in length.

- A SCIM server MAY support the complex multi-valued attributes "addresses" and "phoneNumbers", for a client providing one value of each, and MAY support a client providing more than one value of each attribute.

- A SCIM server MAY support the complex attribute "manager".

- A SCIM server MAY support the complex attribute "name", and if it does, SHOULD support the sub-attributes "formatted", "familyName", "givenName", "middleName", "honorificPrefix" and "honorificSuffix".

- A SCIM server MAY support the attributes "title" and "preferredLanguage".

- A SCIM server MAY support the attribute "department" in the enterprise 2.0 user schema.

3.1. Additional Considerations for the ID Attribute

A SCIM server SHOULD construct a value for the "ID" attribute which is unique across all users which that SCIM server has in its database, and is likely to have in the future. In addition, the value SHOULD NOT include any characters which would require escaping if they were included in a HTTPS URI, and be limited to 64 US-ASCII characters in length. For example, a universally unique identifier as described in RFC 4122 [RFC4122] would be a better choice for an id value than an email address or person's display name.

A SCIM server SHOULD NOT change the value of the "ID" attribute once it has been assigned to a User. Each subsequent GET of that user SHOULD return the same value.

A SCIM client MUST NOT assign any semantics to the value of the "ID" attribute which is receives from a SCIM server, other than it is unique in that server.

3.2. Additional Considerations ExternalId Attribute

A SCIM client MAY construct a value for the "externalId" attribute for its users. If it does, the values of that attribute SHOULD be unique across all users which the identity provider has in its database, and is likely to have in the future. In addition, the value SHOULD NOT include any characters which would require escaping if they were to be included in a HTTPS URI, and be limited to 64 US-ASCII characters in length. For example, a universally unique identifier as described in RFC 4122 [RFC4122] would be a better

choice for an externalId value than an email address or person's display name.

A SCIM server need not support the "externalId" attribute. If it does, it SHOULD NOT assign any semantics to the value of the "externalId" attributes.

3.3. Resources for Schema Discovery

A SCIM server SHOULD support schema discovery through the "Schemas" and "ResourceTypes" endpoint URIs, as described in section 4 RFC 7644 [RFC7644], that specifies the attributes of the "User" resource type which it supports.

A SCIM server that supports more than the minimum attributes of this profile SHOULD publish through schema discovery any additional attributes which it permits an identity provider to send.

A SCIM client MAY query a SCIM server to determine its schema, by querying either of those collections with GET requests. If that SCIM server returns an error to either query, then the SCIM client SHOULD assume that SCIM server only supports the attributes "userName", "ID", "active", "displayName" and "emails".

A SCIM server MUST ignore any additional resource types or attributes in the returned schema which it does not understand.

For example, if "https://example.com/Schemas" was the endpoint for "Schemas", and an authorized SCIM client sent

```
GET /Schemas HTTP/1.1
Authorization: Bearer h480djs93hd8
```

then a SCIM server might return

```
   HTTP/1.1 200 OK
   Content-Type: application/scim+json
   Content-Length: ...

   {
   "schemas":["urn:ietf:params:scim:api:messages:2.0:ListResponse"],
   "totalResults": 1,
   "Resources":[
   {
    "id":"urn:ietf:params:scim:schemas:core:2.0:User",
```

```
         "name":"User",
         "description":"User Account",
         "attributes":[
           {
             "name":"userName",
             "type":"string",
             "multiValued":false,
             "required":true,
             "caseExact":false,
             "mutability":"readWrite",
             "returned":"default",
             "uniqueness":"server"
           },

           {
             "name":"displayName",
             "type":"string",
             "multiValued":false,
             "required":false,
             "caseExact":false,
             "mutability":"readWrite",
             "returned":"default",
             "uniqueness":"none"
           },

           {
             "name":"active",
             "type":"boolean",
             "multiValued":false,
             "required":false,
             "mutability":"readWrite",
             "returned":"default"
           },

           ...

         ]
        },
        "meta":{
           "resourceType":"Schema"
         }
        }
        ]
        }
```

4. Operations on Users

4.1. When a User Account is added in the Identity Provider

When a user account is added in the identity provider, then

- o The identity provider's SCIM client MAY query a SCIM server to locate the user by externalId. This step is OPTIONAL and a SCIM client MAY omit it. A SCIM server is not required to support querying by externalId, and if it does not, SHOULD return an error for a filter on the "Users" endpoint with the externalId attribute.

- o If the user does already not exist in that SCIM server, or the client did not query for it, then the identity provider's SCIM client MAY send a POST to create the user in that SCIM server.

4.1.1. Optionally locating a User by ExternalId

If a SCIM server supports the "externalId" attribute, a SCIM client MAY query a user by filtering for a value of the "externalId" attribute. A SCIM server that implements this profile and supports the "externalId" attribute SHOULD support filtering, as described in section 3.4.2.22 of RFC 7644 [RFC7644]. That SCIM server MAY support filtering by an equality match of the "externalId" attribute.

```
GET /Users?filter=externalId%20eq%201-2 HTTP/1.1
Authorization: Bearer h480djs93hd8
```

A SCIM client MAY query a SCIM server for a value of "externalId". If the SCIM server supports this query and the attribute value is not present on any current User in that SCIM server, the query response is a successful response with 0 result resources.

```
   HTTP/1.1 200 OK
   Content-Type: application/scim+json
   Content-Length: ...

   {
   "schemas":["urn:ietf:params:scim:api:messages:2.0:ListResponse"],
   "totalResults": 0,
   "Resources":[]
   }
```

Otherwise, if there is a matching entry, then the query response is a successful response with one result resource.

4.1.2. Creating a User with POST

To create a user, a SCIM client sends a POST request to the SCIM server's "Users" endpoint. The order of fields in the request for the attributes and metadata is not significant.

For example, if a SCIM client has determined a SCIM server supports the attributes "email", "name" and "department", the SCIM client would send

```
POST /Users HTTP/1.1
Host: example.com
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Bearer h480djs93hd8
Content-Length: ...

{
  "schemas":["urn:ietf:params:scim:schemas:core:2.0:User",
    "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User"],
  "active":true,
  "displayName":"Babs Jensen",
  "emails":[{"primary":true,"type":"work",
    "value":"babs@example.com"}],
  "meta":{"resourceType":"User"},
  "userName":"bjensen@example.com",
  "name":{
    "formatted":"Ms. Barbara J Jensen III",
    "familyName":"Jensen",
    "givenName":"Barbara"
  },
  "urn:ietf:params:scim:schemas:extension:enterprise:2.0:User":{
    "department":"Retail"
  }
}
```

In addition to the attributes, a SCIM server MAY supply, and a SCIM server SHOULD permit, the POST payload to contain the "schemas" and "meta" fields. A SCIM client MUST NOT send the "id" attribute in the POST request.

If the creation was successful, then that server SHOULD return HTTP status code 201 with a representation of the User resource. The returned resource SHOULD have an "id" attribute. The returned resource MAY contain some of the attributes supplied by that SCIM client, but some attributes might not be returned if they were not stored by that server, or if that server chooses not to return them. The returned resource MAY have additional attributes generated by that server.

```
HTTP/1.1 201 Created
Content-Type: application/scim+json
Content-Length: ...

{
  "schemas":["urn:ietf:params:scim:schemas:core:2.0:User"],
 "id":"48af03ac28ad4fb88478"
}
```

If a user resource earlier existed but had been deleted, then a POST request for a new User resource with the same userName, externalId or other attributes as the previously deleted resource SHOULD NOT fail due to attribute conflict. If it does fail due to an attribute conflict, that server SHOULD return status code 409.

4.2. When a User Account's Attributes Change

When inside of an identity provider, one or more of a user's attributes, such as display name, changes to a new value, then

The identity provider's SCIM client MAY query a SCIM server with a GET to retrieve the user's current attributes, as known to that server. This step is OPTIONAL and a SCIM client MAY omit it.

The identity provider's SCIM client MAY send a PATCH to update the user in a SCIM server.

If a SCIM client chooses to perform the query prior to the PATCH, then the sequence of operations would resemble

```
+--------+                                   +--------+
|        |                                   |        |
|        |--(1) GET Request with id of user->|        |
|        |                                   |        |
|        |<-(2) GET Response of that user----|        |
|        |                                   |        |
|  IDP   |                                   |   RP   |
|        |                                   |        |
|        |--(3) PATCH Request for that user->|        |
|        |                                   |        |
|        |<-(4) PATCH Response---------------|        |
|        |                                   |        |
+--------+                                   +--------+
```

4.2.1. **Retrieving a User by ID**

A SCIM client MAY query a SCIM server for a user created earlier, by constructing an URI from the "Users" endpoint and the value of the "id" attribute returned by that server in an earlier POST response. For example,

```
GET /Users/6ba7b810-9dad-11d1-80b4-00c04fd430c8 HTTP/1.1
Authorization: Bearer h480djs93hd8
```

If that user still exists in that SCIM server, that SCIM server would return it.

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

{
"schemas":["urn:ietf:params:scim:schemas:core:2.0:User"],
"id":"5d48a0a8e9f04aa38008",
"externalId":"58342554-38d6-4ec8-948c-50044d0a33fd",
"meta": {
 "resourceType":"User"
},
"userName":"bjensen@example.com",
"name": {
 "formatted":"givenName familyName",
 "familyName":"familyName",
 "givenName":"givenName",
},
"active": true,
"emails":[{
 "value":"bjensen@example.com",
 "type":"work",
 "primary": true
}]
}
```

A SCIM client MAY use the returned user resource to determine if a change it intends to send has already been made to a user.

If the user no longer exists, a SCIM server SHOULD return a 404 error.

If a SCIM client expected that the user was present, but received a 404 error, then that SCIM client MAY attempt to re-create a replacement user with a POST operation, it cannot PATCH a deleted user. Note that the new user SHOULD have a different "ID" attribute value.

4.2.2. Modifying a User with PATCH

A SCIM client sends a PATCH operation to change one or more attributes of the user. A SCIM server implementing this profile SHOULD support PATCH, as described in section 3.5 of RFC 7644 [RFC7644].

o A SCIM server SHOULD support the "replace" operation for the "userName" attribute.

- A SCIM server SHOULD support the "add", "remove" and "replace" operations for the attributes "active" and "displayName". A SCIM client MUST only send the "add" operation for one of these attributes if it has not previously set or retrieved that value for that attribute, and MUST only send the "remove" operation for one of these attributes if it has previously set or retrieved a value for that attribute.

- If a SCIM server supports storing email addresses, then that SCIM server SHOULD support the "add", "remove" and "replace" operations for attribute "emails", and SHOULD support "replace" changing just the value sub-attribute.

- A SCIM server MAY support operations on other attributes within the PATCH.

- A SCIM client MUST NOT assume the response contains any or all of the user attributes, as that SCIM server MAY choose to not return the entire user resource and all its attributes in the response.

```
PATCH /Users/6ba7b810-9dad-11d1-80b4-00c04fd430c8 HTTP/1.1
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",
    "value":{
      "emails":[
        {
          "value":"bjensen@example.com",
          "type":"work",
          "primary":true
        }
      ]
  }]
}
```

A SCIM client MAY include multiple operations for different attributes in a single PATCH.

```
      PATCH /Users/6ba7b810-9dad-11d1-80b4-00c04fd430c8 HTTP/1.1
      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":"emails[type eq \"work\"].value",
           "value":"bjensen@example.com"
           },
           {
         "op":"Replace",
         "path":"name.familyName",
         "value":"Jensen"
           }
        ]
      }
```

4.2.3. Indicating User Sign In Blocked

An identity provider MAY wish to indicate that a user is unable to sign in, and so is temporarily unable to access the application. A SCIM client would send a change of the "active" attribute to the value false to indicate the user will be unable to sign in. Later, if the user is made able to sign in, that SCIM client would send a new request to change the value of the "active" attribute of true.

```
      PATCH /Users/6ba7b810-9dad-11d1-80b4-00c04fd430c8 HTTP/1.1
      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",
          "value":{
            "active":false
        }]
      }
```

4.3. When the User's Account is to be Removed

The identity provider MAY wish to indicate to the application that a user will never be signing in again. As there is no undo, this SHOULD only be done when the user account has been purged, e.g., if the user account has been soft-deleted in the identity provider but might be restored, the "active" attribute SHOULD be used instead of a delete to indicate the user's undoable change of status.

4.3.1. Removing a User with DELETE

A SCIM client MAY delete a user it had created earlier, using the DELETE operation.

```
      DELETE /Users/6ba7b810-9dad-11d1-80b4-00c04fd430c8 HTTP/1.1
      Host: example.com
      Authorization: Bearer h480djs93hd8
```

If the delete was successful, a SCIM server SHOULD return status code 200.

A SCIM server MAY return an error 404 if the user was already deleted, and SHOULD return a different error if the delete could not be performed.

4.4. User Account Retrieval from the Relying Party

At intervals, a SCIM client MAY wish to query a SCIM server to ensure that SCIM server's copy of the user resources matches those expected by that SCIM client. For example, the identity provider might wish to reconcile with the application so that it can determine if one or more changes are needed. This can be done by either querying for each individual user with a GET operation, or retrieving multiple users in a single request.

Querying for an individual user was shown in the earlier section, Retrieving a User by ID.

4.4.1. Retrieving Multiple Users in a Single Request

A SCIM client MAY request to retrieve all users visible to it, by sending a GET request for the "Users" endpoint.

```
GET /Users HTTP/1.1
Authorization: Bearer h480djs93hd8
```

That SCIM server SHOULD either return the result, return the result in multiple pages, or reject the request if the response would be too large to return.

```
   HTTP/1.1 200 OK
   Content-Type: application/scim+json
   Content-Length: ...

   {
   "schemas":["urn:ietf:params:scim:api:messages:2.0:ListResponse"],
   "totalResults": 1,
   "Resources":[{
    ...
   }]
   }
```

5. Security Considerations

The security requirements of sections 7.1, 7.2, 7.3, 7.4, 7.5, 7.7 and 7.8 of RFC 7644 [RFC7644] apply to implementations of this profile. A SCIM server following this protocol SHOULD NOT support

anonymous requests, so section 7.6 of that RFC would not be applicable.

If a SCIM server in an application is affiliated with multiple independent identity providers, then multiple SCIM clients could be making changes and querying the same SCIM server. The application security model will take into consideration whether user resources created from one identity provider are intended be visible to a client acting on behalf of a different identity provider.

6. IANA Considerations

There are no IANA considerations in this document.

7. References

7.1. 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, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/info/rfc3629>.

[RFC6585] Nottingham, M. and R. Fielding, "Additional HTTP Status Codes", RFC 6585, DOI 10.17487/RFC6585, April 2012, <https://www.rfc-editor.org/info/rfc6585>.

[RFC6750] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, DOI 10.17487/RFC6750, October 2012, <https://www.rfc-editor.org/info/rfc6750>.

[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, <https://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, <https://www.rfc-editor.org/info/rfc7644>.

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/info/rfc8446>.

7.2. Informative References

[OPENID.OIDC]
Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and C. Mortimore, "OpenID Connect Core 1.0 incorporating errata set 1", November 2014, <https://openid.net/specs/openid-connect-core-1_0.html>.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <https://www.rfc-editor.org/info/rfc4122>.

Author's Address

Mark Wahl (editor)
Microsoft Corporation
1 Microsoft Way
Redmond, WA 98052
US

Email: mwahl@microsoft.com