Network Working Group
Internet Draft
Intended status: Proposed Standard
Expires: November 2014

M. Wahl
Microsoft
May 7, 2014

SCIM Profile For Enhancing Just-In-Time Provisioning
draft-wahl-scim-jit-profile-02.txt

Status of this Memo

Copyright Notice

Abstract

This document specifies a profile of the System for Cross-Domain Identity Management Protocol (SCIM). Servers which implement protocols such as SAML or OpenID Connect receive user identities through those protocols and often cache them, and this profile of SCIM defines how an identity provider can notify a SCIM server of changes to user accounts.

Table of Contents

1. Introduction

The SCIM protocol [1] is an application-level, REST protocol for provisioning and managing identity data on the web. SCIM can be leveraged for numerous use cases, including transfer of attributes to a relying party web site (see [3] section 3).

This profile of SCIM illustrates the interactions between a SCIM client and a SCIM server, in the following scenario:

- The SCIM client has an associated database (SCIM client database) of user records, and that SCIM client database is leveraged by an identity provider for user authentication.

- The SCIM server has a different associated database (SCIM server database) of user records, and that SCIM server database is leveraged by a service provider (an application).

- The service provider trusts the identity provider to authenticate users, and a user's username and other attributes as stored in the SCIM client database are transferred using a federation or authentication protocol (such as SAML or OpenID Connect -- not SCIM) from the identity provider to the service provider each time a user logs into the service provider.

- Optionally, when the service provider receives a user identity from the identity provider in that federation or authentication protocol, and the service provider cannot find a user record with matching username in the SCIM server database, then the service provider creates a new record in the SCIM server database.

- An identity management system associated with the SCIM client database makes changes to users in the SCIM client database, for instance to de-activate a user, or change the user's display names. These changes are of interest to the service provider as it enables the application to be responsive to user changes even when the user is not logged in.

This profile enables the SCIM client to notify the SCIM server of changes to users in the SCIM client database, so that the SCIM server can make corresponding changes in the SCIM server database, which are part of the service provider. For example, if the identity provider deletes a user, this deletion event can be transferred to the service provider via SCIM, so that the service provider can clean up any data associated with a user who won't be accessing that service provider again. Or if the user changes their username, then this can be made known to the service provider, so that subsequent requests by that user will be associated to the same account in the SCIM server database.

This profile is not intended to be a comprehensive replication protocol; instead, it provides basic consistency for user records for in two domain's databases, for all users who choose to access the service provider. This profile also does not cover establishing common index keys of usernames between a SCIM client and a SCIM server. Finally, management of other object types besides users, and additional attributes beyond basic user status and name, is outside the scope of this profile.

1.1. Conventions used in this document

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 [2].

Throughout this document, values are quoted to indicate that they are to be taken literally. When using these values in protocol messages, the quotes MUST NOT be used as part of the value.

2. Events in the SCIM Client Database

A SCIM client will, either upon specific change in the SCIM client database, or at intervals, provide one or more changes to the SCIM server.

2.1. User is added to the SCIM client database

This profile provides two options for the SCIM client, and support for user creation via SCIM in the SCIM server is OPTIONAL.

2.1.1. SCIM client does not support SCIM user creation

The SCIM client does not notify the SCIM server of this event. (In this profile, user creation is assumed to occur out of band from SCIM, such as through a "just in time" operation in a federation protocol such as SAML [6].)

2.1.2. SCIM client supports SCIM user creation

Support for this procedure is OPTIONAL. When a user is added in the SCIM client database, then the SCIM client can perform the following procedure.

- o The SCIM client will attempt to locate the user in the SCIM server using the user’s username, as described in section 3.1 of this document.

- o If the user exists in the SCIM server database, then the procedure ends.

- Otherwise, if the user’s record was not found in the SCIM server, then the SCIM client will send a POST, as described in section 3.4 of this document, to create the user representation in the SCIM server.

- If the SCIM server returns a 400-series error indication from the POST, then the SCIM client SHOULD NOT retry the operation.

2.2. User’s username changes

When a user's username changes in the SCIM client database, then the SCIM client will perform the following procedure.

- The SCIM client will attempt to locate the user in the SCIM server using the old username, as described in section 3.1 of this document.

- If the user could not be located (no matching record is returned from the GET request), then the procedure ends.

- Otherwise, if the user’s record was found in the SCIM server, then the SCIM client will send a patch, as described in section 3.2 of this document, to set the value of the username attribute to the new username.

- If the SCIM server returns a 400-series error indication from the patch, then the SCIM client SHOULD NOT retry the operation. However, this will indicate to the SCIM client that the representations of the user between the SCIM client database and the SCIM server database are inconsistent and an administrator might be needed to reconcile the difference (e.g., if there was already another user in the SCIM server database who was from another identity provider but had the same user name).

2.3. User’s display name or other descriptive attributes change

When one or more of a user's descriptive attribute such as display name changes to a new value in the SCIM client database, then the SCIM client will perform the following procedure.

- The SCIM client will attempt to locate the user in the SCIM server using the user’s username, as described in section 3.1 of this document.

- If the user could not be located (no matching record is returned from the GET request), then the procedure ends.

- o Otherwise, if the user’s record was found in the SCIM server, then the SCIM client will send a patch, as described in section 3.2 of this document, to set the value of the intended attribute, such as "displayName", or OPTIONALLY the value(s) of sub-attribute(s) of an attribute the "name" attribute, to the new value.

- o If the SCIM server returns a 400-series error indication from the patch, then the SCIM client SHOULD NOT retry the operation.

Note that this profile does not define process for a SCIM client to perform a removal of a user’s attributes.

2.4. User’s account is disabled

When a user's account is disabled in the SCIM client database, then the SCIM client will perform the following procedure.

- o The SCIM client will attempt to locate the user in the SCIM server using the user’s username, as described in section 3.1 of this document.

- o If the user could not be located (no matching record is returned from the GET request), then the procedure ends.

- o If the user’s record was found in the SCIM server, and the GET returned the "active" attribute type in that record and that attribute had the value false, then the procedure ends.

- o Otherwise, then the SCIM client will send a patch, as described in section 3.2 of this document, to set the value of the active attribute to false.

- o If the SCIM server returns a 400-series error indication from the patch, then the SCIM client SHOULD NOT retry the operation. However, this will indicate to the SCIM client that the representations of the user between the SCIM client database and the SCIM server database are inconsistent and an administrator might be needed to determine why a user could not be disabled in the target system.

2.5. User’s account is re-enabled

When a user's account is re-enabled in the SCIM client database after having previously been disabled, then the SCIM client will perform the following procedure.

- The SCIM client will attempt to locate the user in the SCIM server using the user's username, as described in section 3.1 of this document.
- If the user could not be located (no matching record is returned from the GET request), then the procedure ends.
- If the user's record was found in the SCIM server, and the GET returned the "active" attribute type in that record and that attribute had the value true, then the procedure ends.
- Otherwise, then the SCIM client will send a patch, as described in section 3.2 of this document, to set the value of the active attribute to true.
- If the SCIM server returns a 400-series error indication from the patch, then the SCIM client SHOULD NOT retry the operation. However, this will indicate to the SCIM client that the representations of the user between the SCIM client database and the SCIM server database are inconsistent and an administrator might be needed to determine why a user could not be enabled in the target system.

2.6. User's account is purged

When a user's account is purged in the SCIM client, then the SCIM client will perform the following procedure.

- The SCIM client will attempt to locate the user in the SCIM server using the user's username, as described in section 3.1 of this document.
- If the user could not be located (no matching record is returned from the GET request), then the procedure ends.
- Otherwise, then the SCIM client will send a delete, as described in section 3.3 of this document.
- If the SCIM server returns a 400-series error indication from the delete, then the SCIM client SHOULD NOT retry the operation.

3. SCIM Interaction Profile

A SCIM client is REQUIRED to be configured with the following configuration settings prior to communication with the relying party application of the SCIM server:

- o Confidential client authentication material (for example, an token to authenticate the SCIM client to a SCIM server, or a client identifier and client secret password to authenticate the SCIM client to an OAuth2 server)

- o If the client does not have a valid token, an OAuth2 server HTTPS URL (for example "https://example.com/TBD/oauthbase/token") along with any supporting data needed to validate the authenticity of the responding HTTP server

- o SCIM endpoint HTTPS URL prefix (for example, "https://example.com/TBD/scimbase/"), along with any supporting data needed to validate the authenticity of the responding HTTP server

The interactions in this section require the SCIM client to have a valid OAuth2 token, such as a bearer token [8]. If the SCIM client does not have a bearer token, it MUST obtain one using either an OAuth Refresh token or the procedure described in section 5 of this document to obtain an access token.

If a SCIM client supports multiple tenants, the SCIM client SHOULD maintain distinct set of configuration settings for each tenant.

3.1. Locating a user by their user name

In order to modify or delete a user record in a SCIM server, the SCIM client needs to first discover the id of that record as stored in the SCIM server. This is done by searching for the userName attribute, which is defined in section 6.1 of the SCIM Schema [3]. This will also cause an ETag, if versioning is required by the SCIM server, to be returned.

The client can issue a SCIM query request for the namespace ending with /Users with a query parameter of a filter for userName matching for equality the user name. For example, a search for a user name of "matt@example.com" (lines wrapped for clarity):

```
GET /TBD/scimbase/Users?filter=username%20eq%20%22matt@example.co
m%22&attributes=userName,active HTTP/1.1
Host: example.com
Accept: application/json
Authorization: Bearer deadbeef
```

The SCIM server when performing this search MUST use a case insensitive match for the user. Note that as userName is required to be unique across all users known to the SCIM server, at most one result resource would be returned.

If found, the server will respond with a HTTP 200 message containing a single result resource, with one or more attributes:

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

   {
     "schemas":["urn:scim:schemas:core:2.0:ListResponse"],
     "totalResults":1,
     "Resources":[
       {
         "schemas":["urn:scim:schemas:core:2.0:User"],
         "id":"2819c223-7f76-453a-919d-413861904646",
...
         "userName":"matt@example.com",
         "meta":{
            "resourceType":"User",
...

            "version":"W\/\"e180ee84f0671b1\""
         }
       }
     ]
   }
```

If not found, the SCIM server will respond with a HTTP 200 message containing zero result resources.

If the SCIM server requires the SCIM client to support versioning with ETag, then the SCIM server MUST include a version attribute in the meta section of the result resource.

3.2. Modifying a user

For this interaction, the SCIM client needs the id, and OPTIONALLY the version, of the user as represented in the SCIM server database. If the SCIM client does not already have id, the SCIM client can obtain the id of the user as described in section 3.1. If the SCIM client can locate the user record, then the client will modify the attributes of the user in the SCIM server by issuing a POST with an override to PATCH.

If the SCIM server returned a version attribute in response to the GET request from section 3.1, then the SCIM client MUST include an If-Match header in the POST.

The body of the POST request will be a JSON array with one or more elements, each a structure with an "op" key, a "path" key with a value of such as "userName", "displayName", "active" or "name", and "value" key. For example (lines wrapped for clarity)

```
POST /TBD/scimbase/Users/acbf3ae7-8463-4692-b4fd-9b4da3f908ce
HTTP/1.1
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Bearer deadbeef
X-HTTP-Method-Override: PATCH
Content-Length: 70

{
  "op":"replace",
  "path":"displayName",
  "value":"Babs Jensen"
}
```

If successful, the SCIM server will return either a 200 or a 204 status code in the response.

If the POST request included an If-Match header and the SCIM server returns status code 412 indicating precondition failed, then the SCIM client SHOULD re-retrieve the resource using GET, and then re-issue the POST, updating the If-Match header with the new value.

3.3. Deleting a user

For this interaction, the SCIM client needs the id of the user. If it does not already have it, it can obtain the id of the user as described in section 3.1.

If the SCIM client can locate the user record, then the client can request deletion of user in the server by issuing a POST with an override to DELETE.

If the SCIM server returned a version attribute in response to the GET request from section 3.1, then the SCIM client MUST include an If-Match header in the POST.

For example (lines wrapped for clarity)

```
POST /TBD/scimbase/Users/acbf3ae7-8463-4692-b4fd-9b4da3f908ce
HTTP/1.1
Host: example.com
Authorization: Bearer deadbeef
X-HTTP-Method-Override: DELETE
```

If the user cannot be found, the server will return error code 404. Otherwise, if the user is deleted, then the server will return error code 200.

3.4. Creating a User

Support for this operation is OPTIONAL.

The SCIM client issues a POST (without an override) to the Users subpath. The body of the POST request MUST contain the "schemas" and "username" attributes, MAY contain the "displayName", "active" and "name" attributes, and MAY contain additional attributes supported by the SCIM server.

For example (lines wrapped for clarity):

```
POST /TBD/scimbase/Users
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Bearer deadbeef
Content-Length: 100

{
  "schemas": ["urn:scim:schemas:core:2.0:User"],
  "userName": "bjensen@example.com",
  "displayName": "Babs Jensen"
}
```

If the operation was successful, the SCIM server will respond with status code 201 and a response resource containing at least the "id" attribute.

The SCIM server MUST return an error if the requested username would match (ignoring case) with the username of another user resource already present.

4. Schema Profile

A server that implements this profile is REQUIRED to recognize and store in its database the "userName" attribute of the SCIM User Schema. (This attribute is described in section 6 of the SCIM Schema [3].) This values of this attribute are REQUIRED to be unique across all objects queryable by a SCIM client.

The SCIM server MUST recognize and SHOULD store the "displayName" and "active" attributes of the SCIM User Schema. The SCIM server MUST recognize and MAY store the "name" attribute with components "givenName", "middleName" and "familyName" sub-attributes of the SCIM User Schema. If the SCIM server does not store one or more of those attributes, then any changes to them requested by the SCIM client SHOULD be silently discarded.

The SCIM server MUST recognize the "schemas" attribute of the SCIM Core Schema (in section 5.2 of the SCIM Schema [3]), but the value is not modified by the SCIM client in this profile.

The User password and externalId attributes, and other resources, are not used in this profile.

4.1. Relationship to SAML

This section is informational and only applicable to an identity provider or a service provider which implements SAML [6] for user identification.

The value supplied by the SCIM client in the "username" attribute SHOULD be the same as the value included by the identity provider as the subject name identifier inside a SAML assertion.

For example, if an identity provider uses emailAddress format subject name identifiers, then after the SCIM client provisions a user with username scott@example.org, the identity provider could send in a SAML message (lines wrapped for clarity):

```
     <Response ...>
   ...
     <Assertion ...>
   ...
      <Subject>
       <NameID
        Format="urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress">
       scott@example.org
       </NameID>
   ...
      </Subject>
   ...
     </Assertion>
     </Response>
```

4.2. Relationship to OpenID Connect

This section is informational and only applicable to an identity provider or a service provider which implements OpenID Connect [7] for user identification.

The value supplied by the SCIM client in the "username" attribute SHOULD be the same as the value included by the identity provider as the subject identifier ("sub" field) in the ID token.

The value supplied by the SCIM client in the "displayName" attribute SHOULD be the same as in the OpenID Connect "name" claim. Similarly, the SCIM "name" attribute "givenName" sub-attribute corresponds to the OpenID Connect "given_name" claim, the SCIM "name" attribute "familyName" sub-attribute to the "family_name" claim, and the SCIM "name" attribute "middleName" sub-attribute to the "middle_name" claim.

For example, if an identity provider has a user with these attributes that it returns in an OpenID Connect userinfo response:

```
     {
      "sub": "janedoe@example.com",
      "name": "Jane Doe",
      "given_name": "Jane",
      "family_name": "Doe",
      "middle_name": "Barbara",

   ...
     }
```

then that identity provider could provision a user to a SCIM server as (lines wrapped for clarity):

```
   POST /TBD/scimbase/Users
   Host: example.com
   Accept: application/json
   Content-Type: application/json
   Authorization: Bearer deadbeef
   Content-Length: 213

   {
     "schemas": ["urn:scim:schemas:core:2.0:User"],
     "userName": "janedoe@example.com",
     "displayName": "Jane Doe",
     "name": {
       "familyName": "Doe",
       "givenName": "Barbara",
       "middleName": "Jane"
     }
   }
```

5. SCIM Client Authentication

How the SCIM client locates an OAuth endpoint and is registered to that server is currently outside the scope of this document.

5.1. Obtaining an OAuth Bearer Token

A SCIM client can obtain a bearer token from the OAuth server to which it has been registered by generating a token request. The format of the request is a POST, as described in sections 3.2.1 and 4.4.2 of OAuth2 [4], with parameter grant_type having value "client_credentials".

If the SCIM client authenticates itself to the OAuth endpoint using a username and password, then the POST header MUST include an Authorization header, with a value encoding the combination of a client username and client password as described in section 2.3.1 of OAuth2.

For example, for a SCIM client with username "s6BhdRkqt3" and client secret "gX1fBat3bV" to request a token,

```
POST /TBD/oauthbase/token HTTP/1.1
Host: example.com
Content-Type: application/x-www-form-urlencoded
Authorization: Basic czZCaGRSa3F0MzpnWDFmQmF0M2JW
Accept: application/json

grant_type=client_credentials
```

A successful response is a JSON encoded structure containing an access_token field, as shown in section 4.4.3 of OAuth2 [4]. The value access_token is the OAuth bearer token used in subsequent SCIM interactions. The bearer token MUST be valid for at least 60 minutes from the time it is issued.

```
     HTTP/1.1 200 OK
     Content-Type: application/json;charset=UTF-8
     Cache-Control: no-store
     Pragma: no-cache

     {
       "access_token":"2YotnFZFEjr1zCsicMWpAA",
...
     }
```

6. Security Considerations

Both the SCIM and OAuth2 interactions are to be protected using TLS. The OAuth URL, and the SCIM endpoint URL, MUST both be HTTPS URLs.

As described in OAuth2 [4], access token credentials MUST be kept confidential in transit and storage, and only shared among the authorization server, the resource server the access token is valid for, and the client to whom the access token is issued.

For both SCIM and OAuth2 interactions, the client MUST negotiate sufficient TLS protection mechanisms to ensure that the content cannot be modified during transmission, prior to sending the HTTP payload. Furthermore, the client MUST validate the HTTP server authenticity prior to sending the first HTTP request, to avoid disclosing its client secret or bearer token to an unauthorized server.

7. IANA Considerations

There are no IANA considerations in this document.

8. References

8.1. Normative References

[1]   Grizzle, K., Hunt, P., Ansari, M., Wahlstroem, E., Mortimore, C., "System for Cross-Domain Identity Management:Protocol", draft-ietf-scim-api-04, April 2014.

[2]   Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[3]   Grizzle, K., Hunt, P., Wahlstroem, E., Mortimore, C., "System for Cross-Domain Identity Management: Core Schema", draft-ietf-scim-core-schema-04, April 2014.

[4]   Hardt, D., "The OAuth 2.0 Authorization Framework", RFC 6749, October 2012.

8.2. Informative References

[5]   Hunt, P., Khasnabish, B., Nadalin, A., Li, K., Zeltsan, Z., "SCIM Use Cases", draft-ietf-scim-use-cases-01, March 2014.

[6]   Cantor, S., Kemp, J., Philpott, R., Maler, E., "Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0", http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf , March 2005.

[7]   Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., Mortimore, C., "OpenID Connect Basic Client Profile 1.0", http://openid.net/specs/openid-connect-basic-1_0.html , February 2014.

[8]   Jones, M., Hardt, D., "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, October 2012.

Authors' Addresses

Mark Wahl

Microsoft Corporation
1 Microsoft Way
Redmond WA 98052 USA

Email: mark.wahl@microsoft.com