Abstract

The System for Cross-Domain Identity Management (SCIM) specification is an HTTP based protocol that makes managing identities in multi-domain scenarios easier to support through a standardized service. Examples include but are not limited to enterprise to cloud service providers, and inter-cloud based scenarios. This specification extends the SCIM Protocol to include support for HTTP SEARCH.

Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at http://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on November 29, 2015.

Copyright Notice

Copyright (c) 2015 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust’s Legal Provisions Relating to IETF Documents (http://trustee.ietf.org/license-info) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of
Internet-Draft  draft-hunt-scim-search  May 2015

Table of Contents

1. Introduction and Overview ....................................... 2
   1.1. Intended Audience ........................................ 3
   1.2. Notational Conventions .................................... 3
   1.3. Definitions ............................................... 4
2. Discovery of Search Support ..................................... 4
   2.1. Advertising Support in OPTIONS ............................ 4
   2.2. The Accept-Search Header ................................. 4
   2.3. Service Provider Configuration ............................ 4
3. SEARCH Method .................................................. 4
4. Stored Search Queries ........................................... 8
   4.1. Storing A Search ........................................... 8
   4.2. Retrieving A Stored Search ................................. 9
   4.3. Executing A Stored Search .................................. 10
   4.4. SCIM Schema Extension ...................................... 11
      4.4.1. Search Schema ......................................... 11
      4.4.2. Search Resource Type ................................... 11
5. Security Considerations ......................................... 12
6. Privacy Considerations .......................................... 12
7. IANA Considerations ............................................ 13
8. Normative References ............................................ 13
Appendix A. Change Log ............................................ 13
Author’s Address .................................................. 13

1. Introduction and Overview

The SCIM Protocol is an application-level HTTP protocol for provisioning and managing identity data on the web and in cross-domain environments such as enterprise to cloud, or inter-cloud scenarios. The protocol supports creation, modification, retrieval, and discovery of core identity resources such as Users and Groups, as well as custom resources and resource extensions. The definition of resources, attributes, and overall schema are defined in the SCIM Core Schema document (see [I-D.ietf-scim-core-schema]). The SCIM Protocol defines methods for creation, modification, deletion, and searching of SCIM resources (see [I-D.ietf-scim-api]).

In SCIM Protocol, searching is currently supported using HTTP GET and HTTP POST (see section 3.4.2 and 3.4.3 [I-D.ietf-scim-api]). While a common practice, the HTTP GET method of searching causes problems with the disclosure of sensitive information through URL leakage (see Section 7.5 [I-D.ietf-scim-api]). In contrast, using HTTP POST has the disadvantage of overloading POST to perform another function other than creating new SCIM resources. Further, HTTP POST requires
the client to add special URI path encodings so that a SCIM service provider can determine the clients intent to create a resource vs. searching for resources.

This draft proposes to use the HTTP SEARCH method defined in [I-D.snell-search-method] as an improved approach that reduces complexity for SCIM clients in the future. [[Note: the author acknowledges that adding a 3rd method does not reduce complexity. However one has to acknowledge water under the bridge and the costs of transition. The intent here is to show how SCIM would have been simpler had SEARCH been available.]]

One of the advantages of using HTTP SEARCH is that security control layers and gateways can easily differentiate from a general resource retrieval (e.g., via HTTP GET), or resource creation (e.g., via HTTP POST). The security system is not required to parse the content of the URI or of the body to understand that the request is a search operation.

1.1. Intended Audience

This document is intended as a guide to SCIM protocol usage for both SCIM HTTP service providers and HTTP clients who may provision information to service providers or retrieve information from them.

1.2. 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 interoperability 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.
1.3. Definitions

This specification uses the definitions from [I-D.ietf-scim-core-schema], and [I-D.ietf-scim-api].

2. Discovery of Search Support

2.1. Advertising Support in OPTIONS

A server can advertise its support for the SEARCH method by adding it to the listing of allowed methods in the "Allow" OPTIONS response header defined in HTTP/1.1. The SEARCH method MAY appear in the "Allow" header even if the Accept-Search header is absent, in which case the list of allowed patch documents is not advertised.

2.2. The Accept-Search Header

As defined in [I-D.snell-search-method], the "Accept-Search" response header MAY be used by service providers to directly signal support for the SEARCH method while identifying the SCIM content-type. For example:

Accept-Search: application/scim+json

2.3. Service Provider Configuration

Clients may discover service provider support for SEARCH by querying the service provider configuration as described in Section 5 and 8.5 of [I-D.ietf-scim-core-schema]. The attribute "search" has the following values:

- supported A boolean value indicating that the required features of this extension are supported.
- stored A boolean value indicating that stored search queries are supported.
- persistent A boolean value indicating the service provider supports persistent search using WebPUSH. [To be defined]

3. SEARCH Method

A SCIM server that supports HTTP SEARCH accepts and processes a request whose body is defined by SCIM POST based search (see Section 3.4.2 and 3.4.3 of [I-D.ietf-scim-api]). When using HTTP SEARCH, the client SHOULD NOT specify "/.search" as part of the request URI. The remaining content-type, request body and responses specifications are identical. The request path SHALL indicate the
desired search scope. For example, a search at the service provider’s root, indicates the client wants to search all resources. A search against a specific resource indicates the client wishes to search or match against a specific resource.

Not including the impacts of independent operations by other SCIM client with a SCIM service provider, the SCIM SEARCH request is considered idempotent. A repeated search request should reflect the same result unless a parameter (e.g., paging) or the underlying data has been changed independently by another client.

The following is a non-normative example of a SCIM HTTP SEARCH request. Note that some data has been removed and spacing added for readability:

```
SEARCH /Users
Host: example.com
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Bearer h480djs93hd8

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:SearchRequest"],
  "attributes": ["displayName", "userName"],
  "filter":
    "displayName sw "smith\"",
  "startIndex": 1,
  "count": 10
}
```

Figure 1: Example HTTP SEARCH Request
A SEARCH query response is shown with the first page of results. For brevity reasons, only two matches are shown:

HTTP/1.1 200 OK
Content-Type: application/scim+json
Location: https://example.com/Users

```json
{
    "schemas": ["urn:ietf:params:scim:api:messages:2.0:ListResponse"],
    "totalResults":100,
    "itemsPerPage":10,
    "startIndex":1,
    "Resources": [ 
        { 
            "id":"2819c223-7f76-413861904646",
            "userName":"jsmith",
            "displayName":"Smith, James"
        },
        { 
            "id":"c8596b90-7539-4f20968d1908",
            "userName":"alice123",
            "displayName":"Smith, Alice"
        },
        ...
    ]
}
```

Figure 2: Example SEARCH Response
The following is a non-normative example of a SCIM HTTP SEARCH request that tests a specific attribute match condition of a SCIM resource. Note that some data has been removed and spacing added for readability:

```
SEARCH /Users/2819c223-7f76-413861904646
Host: example.com
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Bearer h480djs93hd8

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:SearchRequest"],
  "attributes": ["id"],
  "filter":
    "entitlements.value eq \"CRM_User\"",
  "startIndex": 1,
  "count": 10
}
```

Figure 3: Example HTTP SEARCH Against A Specific Resource

A successful response is shown for the match request above (see Figure 3).

```
HTTP/1.1 200 OK
Content-Type: application/scim+json
Location: https://example.com/Users

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:ListResponse"],
  "totalResults": 1,
  "itemsPerPage": 10,
  "startIndex": 1,
  "Resources": [
    {
      "id": "2819c223-7f76-413861904646"
    }
  ]
}
```

Figure 4: Example Successful Matched Response
The response when no match is made to the request above (see Figure 3). Note that while the request is successful, no match is made, so there are zero results.

HTTP/1.1 200 OK
Content-Type: application/scim+json
Location: https://example.com/Users

```json
{
    "schemas": ["urn:ietf:params:scim:api:messages:2.0:ListResponse"],
    "totalResults": 0,
    "itemsPerPage": 10,
    "startIndex": 1,
    "Resources": []
}
```

Figure 5: Example No-Match Response

A service provider that receives an HTTP SEARCH request that does not support HTTP SEARCH method (this extension) SHOULD return HTTP Status 405 (Method Not Allowed) as defined in Section 6.5.5 of [RFC7231].

[[Editors note: some SCIM service providers might return status 501 which indicates the server does not support the technology. Discuss]]

4. Stored Search Queries

A SCIM SEARCH service provider MAY support stored queries. Queries that have been stored in the endpoint "Searches" may be referenced by id when the body of a search request uses the schemas value of: "urn:ietf:params:scim:api:messages:2.0:StoredSearch". The attributes supported for a stored search request are:

- **id** A SCIM id that is the id of a SCIM resource with schema of "urn:ietf:params:scim:schemas:core:2.0:StoredSearchRequest".

- **$ref** A URI pointing to an HTTP retrievable resource that contains a JSON object defined by the schema: "urn:ietf:params:scim:schemas:core:2.0:StoredSearchRequest". The resource MAY be locally defined or MAY be retrieved from an external SCIM service provider.

4.1. Storing A Search
The following is a non-normative example of a client creating a stored search query:

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

{
    "schemas":
        ["urn:ietf:params:scim:schemas:core:2.0:StoredSearchRequest"],
    "attributes": ["userName"],
    "filter":
        "entitlements.value eq "CRM_User"
}  

Figure 6: Stored Search Create Request

The following is a non-normative example of a stored search creation response.

HTTP/1.1 201 Created
Content-Type: application/scim+json
Location: https://example.com/v2/Searches/86c25f16-d9dd-4a68-a421-1355aa5ae8f0

{
    "schemas":
        ["urn:ietf:params:scim:schemas:core:2.0:StoredSearchRequest"],
    "id": "86c25f16-d9dd-4a68-a421-1355aa5ae8f0",
    "attributes": ["userName"],
    "filter":
        "entitlements.value eq "CRM_User"
}  

Figure 7: Search Creation Response

4.2. Retrieving A Stored Search
To retrieve a known stored search, the client does an HTTP GET:

GET /Searches/86c25f16-d9dd-4a68-a421-1355aa5ae8f0

Host: example.com
Accept: application/scim+json
Authorization: Bearer h480djs93hd8

Figure 8: Retrieving A Stored Search Using GET

The service provider responds with the stored search definition resource (some spacing and "..." added for clarity and brevity):

HTTP/1.1 200 OK
Content-Type: application/scim+json
Location: https://example.com/v2/Searches/86c25f16-d9dd-4a68-a421-1355aa5ae8f0

{
  "schemas": [
    "urn:ietf:params:scim:schemas:core:2.0:StoredSearchRequest"],
  "id": "86c25f16-d9dd-4a68-a421-1355aa5ae8f0",
  "attributes": ["userName"],
  "filter": "entitlements.value eq "CRM_User"," meta": { ... }
}

Figure 9: Response to GET for Stored Search

4.3. Executing A Stored Search

The following is an example of a client executing a stored search against a specific User resource. The "id" specified in the request is the id of the stored request (see Figure 7).

SEARCH /Users/2819c223-7f76-413861904646
Host: example.com
Accept: application/scim+json
Content-Type: application/scim+json
Authorization: Bearer h480djs93hd8

{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:StoredSearch"],
  "id": "86c25f16-d9dd-4a68-a421-1355aa5ae8f0"
}

Figure 10: Executing A Stored Search with SEARCH
If the resource matches the filter in the stored query, the response is returned as defined in the stored request with the attributes userName and id.

HTTP/1.1 200 OK
Content-Type: application/scim+json
Location: https://example.com/.search
{
  "schemas": ["urn:ietf:params:scim:api:messages:2.0:ListResponse"],
  "totalResults": 1,
  "Resources": [  
    {  
      "id": "2819c223-7f76-413861904646",
      "userName": "jsmith",
    }  
  ]
}

Figure 11: Example Response to Stored Search Execution

4.4. SCIM Schema Extension

This section defines the schema extensions necessary for defining the store search resources.

4.4.1. Search Schema

A stored search request has a schemas value of 
"urn:ietf:params:scim:schemas:core:2.0:StoredSearchRequest" and the attributes defined are identical to those defined in Section 3.2.3 of [I-D.ietf-scim-api]. In addition to the SearchRequest attributes, a StoredSearchRequest also has the common attributes id, externalId, and meta as defined in Section 3.1 of [I-D.ietf-scim-core-schema].

4.4.2. Search Resource Type

A stored search has a resource type of "Search" with an endpoint of "/Searches". The value of "schema" is "urn:ietf:params:scim:schemas:core:2.0:StoredSearchRequest". As present, no schema extensions are defined.
Example representation of a stored search resource type:

```json
{
  "schemas": ["urn:ietf:params:scim:schemas:core:2.0:ResourceType"],
  "id":"Search",
  "name":"Search",
  "endpoint": "/Searches",
  "description": "User Account",
  "schema":
  "meta": {
    "location":"https://example.com/v2/ResourceTypes/Search",
    "resourceType": "ResourceType"
  }
}
```

Figure 12: Resource Type Representation

5. Security Considerations

This specification is an extension of SCIM Protocol and carries the same security considerations. See [I-D.ietf-scim-api] and [I-D.ietf-scim-core-schema]. This specification is also subject to the security considerations in [RFC7231].

[[Discussion item to be removed prior to publication: When a search is not fully supported it is important that the entire operation fail. For example, a gateway that does not support SEARCH shall normally fail the request. Headers for filters were also considered as well as a body on GET requests. Both of these approaches have the unwanted effect of misleading the server to return results without processing the filter. It is important that if the request proceeds that the filter MUST be present to conform to the clients intent to match the result. For example, a client that is a security system that is asking if a certain user has a certain attribute condition might be mislead into thinking that condition is true if the server fails by returning results as a simple GET.]]

6. Privacy Considerations

This specification is an extension of SCIM Protocol and carries the same privacy considerations. See [I-D.ietf-scim-api] and [I-D.ietf-scim-core-schema].

In order to improve privacy, SCIM service providers MAY deprecate use of filters with the SCIM GET command in favour of the SCIM SEARCH command.
7. IANA Considerations


8. Normative References

[I-D.ietf-scim-api]

[I-D.ietf-scim-core-schema]

[I-D.snell-search-method]


Appendix A. Change Log

Draft 00 - PH - Initial Draft

Author’s Address

Phil Hunt (editor)
Oracle Corporation

Email: phil.hunt@yahoo.com