Abstract

The SCIM 2.0 Core API defines a simple profile for searching for specific resource types using filters and qualifiers in combination with the HTTP GET verb. The Token Search specification defines the following additional features:

- Specification of search terms within an HTTP POST verb to avoid accidental leakage of confidential information via HTTP GET URLs,
- An optional result set token enabling clients to page through results in a state consistent fashion, and
- The ability to search across multiple resource types (endpoints) and return one or more resource types.

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."
Table of Contents

1. Introduction .................................................. 4
   1.1. Requirements Language .................................... 4
2. Search Extension ................................................. 4
   2.1. Search Tokens .............................................. 6
   2.2. Filter Processing Across Multiple Resource Types ......... 6
   2.3. Search Request Using HTTP POST .......................... 7
   2.4. Search Request Using HTTP GET ............................ 8
   2.5. Search Response ............................................ 8
      2.5.1. Server Processing ...................................... 8
      2.5.2. Client Processing .................................... 11
   2.6. Querying An Existing Result .............................. 11
   2.7. Abandoning Search Results ............................... 11
   2.8. ServiceProviderConfig Discovery .......................... 12
3. Acknowledgements ................................................. 12
4. IANA Considerations .............................................. 12
5. Security Considerations ......................................... 12
6. References ................................................................ 13
   6.1. Normative References ...................................... 13
   6.2. Informative References .................................. 14
Authors’ Addresses ................................................ 14
1. Introduction

The SCIM Core API is an application-level, RESTful service for provisioning and managing identity data on the web. The SCIM Core API specification[I-D.ietf-scim-api] defines methods for creation, modification, retrieval and discovery of resources. This specification extends SCIM Core API capabilities to support extended searching operations:

- the ability to query for one or more resources using a filter,
- the ability to search from any resource endpoint including the server root,
- the ability to support result sets which provide consistent search results across multiple requests,
- a POST Search profile which supports searching with parameters not included in the URL, and
- a token parameter which supports returning stateful paged results.

[Discuss: This extension specification does not propose any change in functionality to the existing GET search functions with the exception of making resourceType a searchable filter term and returning resourceType in all JSON resource representations]

[Note: this specification may be optionally combined with the SCIM core API spec]

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 [RFC2119].

Note: For readability and space reasons, some included examples and attribute values, and UUID identifiers are shortened.

2. Search Extension

The SCIM protocol specifies well known endpoints and HTTP methods for managing resources in the core schema. In SCIM Extended Search, a virtual resource known as a "search" is defined. A "search" is a RESTful representation of a search of a set of objects in a SCIM Service Provider. A SCIM extended "search" endpoint can be appended to any normal SCIM endpoint in order to define open scope and
specific resource scoped searches.

SCIM Extended Searches are defined (in ABNF [RFC5234]) as follows:

```
SCIMSEARCH = scimEndpoint [scimSearch] ["/" UUID] ["?" query]
```

```
scimEndpoint = "https://" authority [path-absolute] [scimVers]
               [scimResType] ["/" UUID]
```

```
scimSearch   = "/.search"

authority    = [ userinfo "@" ] host [ ":" port ]

scimVers     = "/V" 1*DIGIT

scimResType  = "/Users" / "/Groups" / "/" 1*extResChar

extResChar   = "-" / "." / "_" / DIGIT / ALPHA
```

Figure 1: ABNF for SCIM Search Endpoints

[Discussion: since a search within a "/Users" endpoint could be confused with a resource, do we want to have something more jarring like "xsearch" or ".search" (as in well-known urls)?)

Where:

```
scimSearch    Is a special path qualifier that indicates the
              operation is related to a search. When used with HTTP
              GET, scimSearch is OPTIONAL.
```

```
authority, path-absolute, userinfo, host, port  Are defined as per
       URI Syntax ABNF [RFC3986]
```

```
query         Is any SCIM query term as specified in section 3.2
              of the SCIM Core API [I-D.ietf-scim-api].
```

```
scimVers      Is the SCIM Service Provider API version.
```

```
scimResType   Is a SCIM resource type such as "Users", "Groups" or
              any other schema extended resource supported by the
              service provider.
```

```
UUID          Identifies a unique SCIM resource or SCIM search result
              that is retrievable from the SCIM Service Provider (see
              [RFC4122]).
```

Example SCIM Search endpoints include:
Token search is initiated by using either an HTTP POST or HTTP GET command (see next sections) to pass search parameters AND by setting the parameter "stateful" to "true". The server responds and returns results in a JSON result set along with a searchId token if stateful results are available.

2.1. Search Tokens

From the client’s perspective, as with any HTTP is a stateless protocol. When performing searches requiring several pages of results to be returned, there is always the possibility to have inconsistent results should the underlying data change between requests. SCIM Service Providers MAY optionally maintain a stateful representation of search results that may be accessed by the client using a token.

Once a search result “searchId” token has been created, a client may perform GET requests using the parameter "searchId" and "startIndex" to return additional result pages. Any other parameters and path are IGNORED when "searchId" is present. The searchId token allows the service provider to respond with consistent stateful results, using pagination commands as defined in section 3.2.2.3 of the SCIM Core API.

[Discussion: Can an existing search result be used in combination with a new search or ordering request to sub-search within an existing set of results? E.g. would it make sense to allow a PUT against an existing search result?]

2.2. Filter Processing Across Multiple Resource Types

When the SCIM Extended Search specification is supported, Filtering, as defined in the SCIM Core API, section 3.2.2.1 is REQUIRED.

When processing search operations across endpoints that MAY include
more than one SCIM resource type (e.g. a search from the server root endpoint), filters MUST be processed in the same fashion as outlined in 3.2.2.1 of the SCIM Core API. For filtered attributes that are not part of a particular resource type, the service provider SHALL treat the attribute as if there is no attribute value. For example, a presence or equality filter for an undefined attribute evaluates as FALSE.

2.3. Search Request Using HTTP POST

To create a new search result set, clients send an HTTP POST request to the desired SCIM resource endpoint (see Figure 1). The body of the POST request MAY include any of the parameters as defined in section 3.2 of the SCIM Core API.

An additional parameter "stateful" MAY be set to "true" in order to request stateful results from the server. Note that the presence of this parameter DOES NOT require that the server produce a stateful result set. Detection of stateful results is described in Section 2.5.1.

After receiving a HTTP POST request, a response is returned as specified in Section 2.5.

The following example shows an HTTP POST Search request with search parameters attributes, filter, and count included:

```plaintext
POST /.search
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Bearer h480djs93hd8
Content-Length: ...

{
    "schemas": ["urn:scim:schemas:core:1.0"],
    "attributes": ["displayName","username"],
    "filter": "displayName sw "smith"",
    "stateful": "true",
    "count": 10
}
```

Figure 2: Example POST Search Request

[TBD - are there any cases for persistent searches?]

[TBD - should there be additional options to control where result state is maintained, lifetime, etc. Controls whether subsequent gets cause re-queries or whether result set consistency is maintained for
the life of the result set]

2.4.  Search Request Using HTTP GET

When an HTTP GET request is made against a search endpoint (see Figure 1) a new search result is created. The parameters and arguments are as defined in section 3.2 of the SCIM Core API.

An additional parameter "stateful" MAY be set to "true" in order to request stateful results from the server. Note that the presence of this parameter DOES NOT require that the server produce a stateful result set. Detection of stateful results is described in Section 2.5.1.

After receiving an HTTP GET request, a response is returned as specified in Section 2.5.

The following example is the HTTP GET equivalent of the example shown in Figure 2.

GET /.search?attributes="displayName,username"
  &filter=displayName sw "smith"&count=10&stateful="true"
Host: example.com
Accept: application/json
Authorization: Bearer h480djs93hd8

Figure 3: Example GET Search Request

2.5.  Search Response

2.5.1.  Server Processing

If, on receiving a search request, only a single resource is matched, the server SHALL respond with an HTTP status code of 200 and include the single JSON formatted search result in the body of the response as specified in the SCIM Core API.

If multiple resources are to be returned in a single response, the server SHALL respond with an HTTP status code of 200 and a body containing an array of "Resources" in a JSON structure up to the default server limit OR the number of records specified in SCIM pagination parameter ‘count’.

If more results are available than returned and the client request included the parameter "stateful" whose value is "true", Service Providers MAY support stateful paged results by returning an HTTP status code of 201 and include an attribute "searchId" specifying a token that MAY be used to obtain additional results in a stateful manner.
If the client DID NOT specify "stateful" as "true", OR Servers NOT supporting stateful results MUST return an HTTP status of 200 and a page of results as per the normal HTTP GET request of the SCIM Core API.

[Is it more appropriate in token mode for status 200 to be returned?]

Regardless of the number of resources returned or the search endpoint, the server SHALL include the following meta attributes with each resource JSON structure:

- location: the permanent location URI of the returned object
- resourceType: a string (scimResType) representing the type of resource returned (e.g. "User", "Group").
A search response is shown with the first page of results. For brevity reasons, only two matches are shown: one User and one Group. A status of 201 and a location header is returned with the result set identifier indicating more results are available in stateful representation. A subsequent GET to the location allows multiple pages of results to be returned while preserving the original result set using a search session identifier. As described in section 3.2.2.3, the response includes pagination response attributes: itemsPerPage, totalResults, and startIndex.

HTTP/1.1 201 Created
Content-Type: application/json
Location: https://example.com/.search

```json
{
    "totalResults":100,
    "itemsPerPage":10,
    "startIndex":1,
    "searchId":"131dc0667e20"
    "schemas": ["urn:scim:schemas:core:1.0"],
    "Resources": [  
    {
        "meta":{
            "location":
            "https://example.com/Users/2819c223-7f76-413861904646",
            "resourceType": "User",
            "lastModified": ... 
        }
        "username": "jsmith",
        "displayName": "Smith, James"
    },
    {
        "meta":{
            "location":
            "https://example.com/Groups/c8596b90-7539-4f20968d1908",
            "resourceType": "Group",
            "lastModified": ... 
        }
        "displayName": "Smith Family"
    }
    ...
}
```

Figure 4: Example POST Search Response

[Discuss: Instead of returning a location identifier for paging, how about using a "token" for paging?]
2.5.2. Client Processing

If the initial search response has status 201 and a "searchId" attribute is present, SCIM clients MAY obtain additional page results in a stateful fashion using the SCIM Core API "startIndex" pagination parameter as described in the following Section 2.6. Alternatively, SCIM clients MAY ignore the stateful results location, and request additional paged results as per the normal SCIM Core API stateless functionality which causes new results to be generated with each page request.

2.6. Querying An Existing Result

A SCIM client MAY request additional results after an initial search request has been returned with a status 201 and a "searchId" token has been provided. The client requests additional pages by using the "searchId" parameter which is set to the provided token value, along with "startIndex" and "count" parameters.

SCIM clients SHOULD NOT expect that search results will be maintained indefinitely and SHOULD request additional results in a reasonable time frame.

In the code example below, a request is made to return results starting with the 11th item in the result set identified by the identifier 2a7229cc-d9dc-4153-131dc0667e20. The identifier is used to ensure consistent results in the event the underlying data has changed since the original query was posted.

```
GET /v2/.search?startIndex=11&count=10
&searchId=2a7229cc-d9dc-4153-131dc0667e20
HOST: example.com
Accept: application/json
Content-Type: application/json
Authorization: Bearer h480djs93hd8
```

Figure 5: Example Consistent Paged Result Set Query Request

After all results have been returned, OR after a reasonable period, the Service Provider MAY automatically delete the result set. A subsequent attempt to retrieve results from a deleted result set SHALL return an HTTP/1.1 404 NOT FOUND error.

2.7. Abandoning Search Results

If a client decides not to retrieve all results in a stateful search, a client SHOULD inform the Service Provider that it is finished with a result set by issuing an HTTP DELETE with the searchId token provided as a parameter to the endpoint where the original search was
Upon receiving a DELETE request, the server SHALL respond with status 200 (OK) if the search request was deleted or status 404 (NOT FOUND) if the result has already been deleted or otherwise expired.

DELETE /Users/.search?searchId=64b6efd7-6598-4dbc-ab3a-af38dad71493
Host: example.com
Authorization: Bearer h480djs93hd8

HTTP/1.1 200 OK

Figure 6: Cancelling Search Results Set

Servers MAY choose to automatically expire search results after all results have been returned or after a period of time in which no additional requests have been received for the search result. Upon automatically expiring a search result, any further request to return a result or delete a result SHOULD return a status of 404 (NOT FOUND).

2.8. ServiceProviderConfig Discovery

[TBD]

ServiceProviderConfig will require a new set of "search" attributes that indicate what is supported. For example:

- root search
- post search
- stateful paged search results

3. Acknowledgements

[TBD]

4. IANA Considerations

This memo includes no request to IANA.

5. Security Considerations

Implementers should consider that the creation of search results can lead to depletion of server resources creating a vector for denial-
of-service attacks. Implementers and deployers should take appropriate counter measures such as:

- limiting which clients may use extended search,
- limiting the number of extended searches any client MAY execute in a session or HTTP connection,
- encourage clients to delete search results using the HTTP DELETE command in order to free resources,
- limiting the time a search result is retained, and
- purging unused or expired search results.

[Others TBD]

6. References

6.1. Normative References

[I-D.ietf-scim-api]

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


6.2. Informative References

[I-D.narten-iana-considerations-rfc2434bis]
Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs",
draft-narten-iana-considerations-rfc2434bis-09 (work in progress), March 2008.

Authors’ Addresses

Phil Hunt (editor)
Oracle

Email: phil.hunt@yahoo.com

Kelly Grizzle
Sailpoint

Bjorn Aannestad
UnboundID

Morteza Ansari
CISCO

Dale Olds
VMWare