scim
Internet-Draft
Intended status: Standards Track
Expires: August 16, 2014

B. Greevenbosch
Huawei Technologies

February 12, 2014

SCIM and vCard mapping
draft-greevenbosch-scim-vcard-mapping-03

Abstract

This document defines a mapping between SCIM and vCard.

Note

Discussion and suggestions for improvement are requested, and should be sent to scim@ietf.org.

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 August 16, 2014.

Copyright Notice

Table of Contents

1. Introduction

The SCIM core schema [I-D.ietf-scim-core-schema] defines a platform neutral data and extension model for representing users of cloud services. SCIM core also defines XML and JSON serialisations of the abstract schema.

This document defines a mapping between SCIM and vCard [RFC6350]. The mapping may serve several purposes:

- To provide a unified conversion mechanism between SCIM and vCard.
- To identify properties that are defined in vCard, but are missing in SCIM.
- To identify SCIM attributes that may be useful in vCard too.

2. Requirements notation

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].

3. Mapping from SCIM to vCard

When mapping SCIM attributes to vCard, they may either become mapped to vCard properties or to vCard attribues associated with vCard properties. Section 3.1 defines the mappings to the vCard properties, whereas Section 3.2 defines mappings to vCard attributes.

In addition, in accordance to [RFC6350], the vCard representation MUST include the mandatory fields:

- VERSION
- FN

3.1. Mapping of SCIM attributes to vCard properties

Table 1 describes a mapping from SCIM attributes to the vCard properties.

In the table, the cardinality of the SCIM attribute is prefixed by an "S", whereas the cardinality of the vCard property is prefixed by a "v". The further notation has been adopted from [RFC6350] as follows:

1	Exactly one instance MUST be present.
*1	Exactly one instance MAY be present.
1*	One or more instances MUST be present.
*	One or more instances MAY be present.

SCIM fields that have no vCard equivalent MUST be omitted in the vCard result.

The reverse mapping from vCard to SCIM is defined in Section 4. The reason for having two tables is that some mappings are not invertible.

SCIM attribute	vCard property	Cardinality	Notes

id	UID	S1 v*1	See Section 5 for conversion from SCIM id space to vCard UID space.
externalId		S*1	
meta/created		S*1	
meta/lastModified	REV	S*1 v*1	
meta/location		S*1	No direct vCard equivalent. Candidates could be SOURCE and ORG-DIRECTORY.
meta/version		S*1	
meta/attributes		S*1	
userName		S1	
name/formatted	FN	S*1 v1*	
name/familyName	N (family names)	S*1 v*1	Combined with other name attributes in a single N element.
name/givenName	N (given names)	S*1 v*1	Combined with other name attributes in a single N element.
name/middleName	N (additional names)	S*1 v*1	

name/honorificPrefix	N (honorific prefixes)	S*1 v*1	Combined with other name attributes in a single N element.
name/honorificSuffix	N (honorific suffixes)	S*1 v*1	Combined with other name attributes in a single N element.
displayName		S*1	
nickName	NICKNAME	S*1 v*	
profileUrl	URL	S*1 v*	Multiple fields in SCIM better?
emails	EMAIL	S* v*	See Table 2 for the conversion of a possible "type" attribute.
phoneNumbers (type="work")	TEL (TYPE="voice,work")	S* v*	
phoneNumbers (type="home")	TEL (TYPE="voice,home")	S* v*	
phoneNumbers (type="mobile")	TEL (TYPE="voice,cell")	S* v*	
phoneNumbers (type="fax")	TEL (TYPE="fax")	S* v*	
phoneNumbers (type="pager")	TEL (TYPE="pager")	S* v*	
phoneNumbers (type="other")	TEL (no TYPE)	S* v*	

phoneNumbers (no type)	TEL (no TYPE)	S* v*	
ims	IMPP	S* v*	
photos	PHOTO	S* v*	URL of a web location where the photo can be retrieved.
addresses	ADR	S* v*	See [RFC6350] for the internal coding of the ADR property.
addresses/formatted	ADR (LABEL)	S* v*	
addresses/streetAddress	ADR (street address)	S* v*	Combined with other address attributes into a single ADR element.
addresses/locality	ADR (locality)	S* v*	Combined with other address attributes into a single ADR element.
addresses/region	ADR (region)	S* v*	Combined with other address attributes into a single ADR element.
addresses/postalCode	ADR (postal code)	S* v*	Combined with other address

			attributes into a single ADR element.
addresses/country	ADR (country)	S* v*	Combined with other address attributes into a single ADR element.
groups/value		S*	ID of the group
groups/$ref		S*	URI of the group
entitlements		S*	Hard to map as it is proprietary by nature.
roles	ROLE	S* v*	Consider distinction with the "userType" attribute.
x509Certificates	KEY	S* v*	Care is required: keys may not have the same usage.
employeeNumber		S*1	
title	TITLE	S*1 v*	
userType	ROLE	S*1 v*	Consider distinction with the "roles" attribute.
preferredLanguage	LANG	S*1 v*	Language tag according to

			[RFC5646].
locale		S*1	
timezone	TZ	S*1 v*	
active		S*1	
password		S*1	
costCenter		S*1	
organization	ORG	S*1 v*	Use the hierarchical order defined in vCard.
division	ORG	S*1 v*	Use the hierarchical order defined in vCard.
department	ORG	S*1 v*	Use the hierarchical order defined in vCard.
manager/managerId		S*1	SCIM specific ID, related to "id" attribute. The vCard RELATED property could be used, but a TYPE "manager" may need definition. In SCIM, "managerID" is mandatory if "manager"

				is included.
manager/$ref			S*1	The URI of the SCIM resource representing the User's manager.
manager/displayName			S*1	This field is optional in SCIM, also when "manager" is included.
members/$ref	MEMBER		S* v*	Contains the URIs of the SCIM resources associated with the members of the group.
members/id	MEMBER		S* v*	Contains the IDs of the SCIM resources associated with the members of the group.

Table 1: SCIM to vCard mapping

3.2. Mapping of SCIM attributes to vCard parameters

In addition to SCIM properties, SCIM attributes may also need to be converted to vCard parameters. Table 2 contains the related mappings.

SCIM attribute	SCIM value	vCard parameter	vCard value	Notes
type	home	TYPE	home	May be combined

				with other types in vCard
type	work	TYPE	work	May be combined with other types in vCard
type	mobile	TYPE	cell	May be combined with other types in vCard
type	fax	TYPE	fax	May be combined with other types in vCard
type	pager	TYPE	pager	May be combined with other types in vCard
type	other			Omitted in vCard
type	aim	TYPE	x-aim	Only for "ims"
type	gtalk	TYPE	x-gtalk	Only for "ims"
type	icq	TYPE	x-icq	Only for "ims"
type	xmpp	TYPE	x-xmpp	Only for "ims"
type	msn	TYPE	x-msn	Only for "ims"
type	skype	TYPE	x-skype	Only for "ims"
type	qq	TYPE	x-qq	Only for "ims"
type	yahoo	TYPE	x-yahoo	Only for "ims"
type	photo			Only for "photo", vCard parameter can be omitted.
type	yahoo	TYPE	x-thumbnail	Only for "thumbnail"
primary	true	PREF	1	
primary	false			Omitted in vCard.

Table 2: Mapping of SCIM attributes to vCard parameters

4. Mapping from vCard properties to SCIM attributes

4.1. Mapping of vCard properties

Table 3 describes a mapping from vCard properties to SCIM attributes. For the cardinalities, the same notation from Section 3 is used.

Notice that the attributes "uid" and "userName" are mandatory in a SCIM representation, whereas they may not be available in the vCard. It is left to the application to generate sensible values for these fields.

vCard property	SCIM attribute	Cardinality in vCard/SCIM	Notes
SOURCE		v*	Similar to SCIM meta/location.
KIND		v*1	In vCard can have the values "individual", "group", "org" and "location". The value "application" was added by [RFC6473].
XML		v*	Purpose: to include extended XML-encoded vCard data in a plain vCard.
FN	names/formatted	v1* S*1	
N (family names)	names/familyName	v*1 S*1	

N (given names)	names/givenName	v*1 S*1	
N (additional names)	names/middleName	v*1 S*1	
N (honorific prefixes)	names/honorificPrefix	v*1 S*1	
N (honorific suffixes)	names/honorificSuffix	v*1 S*1	
NICKNAME	nickName	v* S*1	
PHOTO	photos	v* S*	URL of a web location where the photo can be retrieved.
BDAY		v*1	
ANNIVERSARY		v*1	
GENDER		v*1	Can have the values "M"ale, "F"emale, "O"ther, "N"one or not applicable or "U"nknown.
ADR (LABEL)	addresses/formatted	v* S*	
ADR (post office box)	addresses/streetAddress	v* S*	
ADR (extended address)		v*	
ADR (street address)	addresses/streetAddress	v* S*	
ADR (locality)	addresses/locality	v* S*	
ADR (region)	addresses/region	v* S*	

ADR (postal code)	addresses/postalCode	v* S*	
ADR (country)	addresses/country	v* S*	
TEL (TYPE="textphone")	phoneNumbers, type="other"	v* S*	See Table 4 for related type mapping.
EMAIL	emails	v* S*	Can have TYPE="work", TYPE="home".
IMPP	ims	v* S*	
LANG	preferredLanguage	v* S*1	
TZ	timezone	v* S*1	
GEO		v*	GPS coordinates
TITLE	title	v* S*1	
ROLE	roles	v* S*1	
LOGO		v*	
ORG	organization	v* S*1	
MEMBER	members/id	v* S*	Contains a vCard ID of a member of this group. The vCard MUST have KIND="group". ID must be converted.
RELATED		v*	Contains a vCard ID of another related vCard. Can have many

			TYPE values, such as "friend", "neighbor" and "spouse".
CATEGORIES		v*	Contains not necessarily unified tags.
NOTE		v*	Any text.
PRODID		v*1	ID for producer of vCard.
REV		v*1	Purpose: to specify revision information about the current vCard.
SOUND		v*	
UID	externalId	v*1 S*1	See Section 5 for conversion from vCard UID space to SCIM id space.
CLIENTPIDMAP		v*	Link between local PID and global URI.
URL	profileUrl	v* S*1	
VERSION		v1	Version of vCard specification.
KEY	x509Certificates?	v* S*	Care is

			required: keys may not have the same usage.
FBURL		v*	Purpose: to specify the URI for the busy time associated with the object that the vCard represents.
CALADRURI		v*	Purpose: to specify the calendar user address to which a scheduling request should be sent for the object represented by the vCard.
CALURI		v*	Purpose: to specify the URI for a calendar associated with the object represented by the vCard.
BIRTHPLACE		v*1	Defined in [RFC6474].
DEATHDATE		v*1	Defined in [RFC6474].
DEATHPLACE		v*1	Defined in [RFC6474].

EXPERTISE		v*	Defined in [RFC6715].
HOBBY		v*	Defined in [RFC6715].
INTEREST		v*	Defined in [RFC6715].
ORG-DIRECTORY		v*	Defined in [RFC6715].

Table 3: vCard to SCIM mapping

4.2. Mapping of vCard parameters

Table 4 describes how vCard parameters are mapped to SCIM.

vCard parameter	vCard parameter value	SCIM representation	Notes
TYPE	cell	"type": "mobile"	
TYPE	fax	"type": "fax"	
TYPE	pager	"type": "pager"	
TYPE	text	"type": "other"	
TYPE	textphone	"type": "other"	
TYPE	video	"type": "video"	
TYPE	voice		Omitted in SCIM

Table 4: Mapping of vCard parameters

5. Mapping between SCIM and vCard IDs

A SCIM specific prefix could be used to indicate the conversion from SCIM IDs to vCard UIDs. A "Service Provider" specific part would

need to be included in the vCard UID, as the SCIM ID is unique within the Service Provider's space only. The following format is proposed:

UID:scim:[serviceProviderID]:123456789

Conversion from vCard to SCIM may be done similarly, i.e. by adding a prefix to the vCard UID. The SCIM schema document mentions for the SCIM ID: "This identifier MUST be unique across the Service Provider's entire set of Resources", so as long as the vCard UID indeed is globally unique, and the service provider uses the prefix for vCard acquired resources only, the rule should hold.

Notice that the above mechanism allows looping. For example, converting SCIM -> vCard -> SCIM would lead to another SCIM ID in the second representation as in the first. This indeed reflects the possible loss of information in the conversion process. It is RECOMMENDED to avoid this kind of chained conversion.

Because of the format of the vCard UID after conversion from SCIM, the SCIM service provider can detect above mentioned chained conversion, as well as the original vCard ID. The actions the service provider may take upon such detection may for example include using the original SCIM data instead, or using smarter mapping by analysing the original and the new import. This kind of mechanisms is left out of scope of this document.

6. Differences between vCard and SCIM

This section contains a non-exhaustive list of differences between vCard and SCIM.

- In vCard, a group property can be established. This property contains the IDs of its members. In SCIM however, the group/ membership relation can be signalled in two directions: just like vCard the group object can signal its members through the "members" attribute, but the member objects can also point to the groups they are part of, through the "groups" attribute.

- In SCIM, relations between objects can be established either through their IDs or through their URIs. vCard only uses IDs to signal relationships between entities.

7. Examples

7.1. Mapping from SCIM to vCard

Figure 2 contains the result after converting the SCIM data from Figure 1 to vCard.

Notice that the following fields have been omitted during conversion:

- userName
- locale
- active
- password
- groups
- meta fields except for "lastModified"

```
{
 "schemas": ["urn:scim:schemas:core:2.0:User"],
 "id": "2819c223-7f76-453a-919d-413861904646",
 "externalId": "701984",
 "userName": "bjensen@example.com",
 "name": {
   "formatted": "Ms. Barbara J Jensen III",
   "familyName": "Jensen",
   "givenName": "Barbara",
   "middleName": "Jane",
   "honorificPrefix": "Ms.",
   "honorificSuffix": "III"
 },
 "displayName": "Babs Jensen",
 "nickName": "Babs",
 "profileUrl": "https://login.example.com/bjensen",
 "emails": [
   {
     "value": "bjensen@example.com",
     "type": "work",
     "primary": true
   },
   {
     "value": "babs@jensen.org",
     "type": "home"
   }
 ],
 "addresses": [
   {
```

```
      "type": "work",
      "streetAddress": "100 Universal City Plaza",
      "locality": "Hollywood",
      "region": "CA",
      "postalCode": "91608",
      "country": "USA",
      "formatted": "100 Universal City Plaza\nHollywood, CA 91608 USA",
      "primary": true
    },
    {
      "type": "home",
      "streetAddress": "456 Hollywood Blvd",
      "locality": "Hollywood",
      "region": "CA",
      "postalCode": "91608",
      "country": "USA",
      "formatted": "456 Hollywood Blvd\nHollywood, CA 91608 USA"
    }
  ],
  "phoneNumbers": [
    {
      "value": "555-555-5555",
      "type": "work"
    },
    {
      "value": "555-555-4444",
      "type": "mobile"
    }
  ],
  "ims": [
    {
      "value": "someaimhandle",
      "type": "aim"
    }
  ],
  "photos": [
    {
      "value": "https://photos.example.com/profilephoto/72930000000Ccne/F",
      "type": "photo"
    },
    {
      "value": "https://photos.example.com/profilephoto/72930000000Ccne/T",
      "type": "thumbnail"
    }
  ],
  "userType": "Employee",
  "title": "Tour Guide",
  "preferredLanguage":"en_US",
```

```
 "locale": "en_US",
 "timezone": "America/Los_Angeles",
 "active":true,
 "password":"t1meMa$heen",
 "groups": [
   {
     "value": "e9e30dba-f08f-4109-8486-d5c6a331660a",
     "$ref": "https://example.com/v1/Groups/e9e30dba-f08f-4109-8486-d5c6a331660a
",
     "display": "Tour Guides"
   },
   {
     "value": "fc348aa8-3835-40eb-a20b-c726e15c55b5",
     "$ref": "https://example.com/v1/Groups/fc348aa8-3835-40eb-a20b-c726e15c55b5
",
     "display": "Employees"
   },
   {
     "value": "71ddacd2-a8e7-49b8-a5db-ae50d0a5bfd7",
     "$ref": "https://example.com/v1/Groups/71ddacd2-a8e7-49b8-a5db-ae50d0a5bfd7
",
     "display": "US Employees"
   }
 ],
 "x509Certificates": [
   {
     "value": "MIIDQzCCAqygAwIBAgICEAAwDQYJKoZIhvcNAQEFBQAwTjELMAkGA1UEBhMCVVMx
               EzARBgNVBAgMCkNhbGlmb3JuaWExFDASBgNVBAoMC2V4YW1wbGUuY29tMRQwEgYD
               VQQDDAtleGFtcGxlLmNvbTAeFw0xMTEwMjIwNjI0MzFaFw0xMjEwMDQwNjI0MzFa
               MH8xCzAJBgNVBAYTAlVTMRMwEQYDVQQIDApDYWxpZm9ybmlhMRQwEgYDVQQKDAtl
               eGFtcGxlLmNvbTEhMB8GA1UEAwwYTXMuIEJhcmJhcmEgSiBKZW5zZW4gSUlJMSIw
               IAYJKoZIhvcNAQkBFhNiamVuc2VuQGV4YW1wbGUuY29tMIIBIjANBgkqhkiG9w0B
               AQEFAAOCAQ8AMIIBCgKCAQEA7Kr+Dcds/JQ5GwejJFcBIP682X3xpjis56AK02bc
               1FLgzdLI8auoR+cC9/Vrh5t66HkQIOdA4unHh0AaZ4xL5PhVbXIPMB5vAPKpzz5i
               PSi8xO8SL7I7SDhcBVJhqVqr3HgllEG6UClDdHO7nkLuwXq8HcISKkbT5WFTVfFZ
               zidPl8HZ7DhXkZIRtJwBweq4bvm3hM1Os7UQH05ZS6cVDgweKNwdLLrT51ikSQG3
               DYrl+ft781UQRIqxgwqCfXEuDiinPh0kkvIi5jivVu1Z9QiwlYEdRbLJ4zJQBmDr
               SGTMYn4lRc2HgHO4DqB/bnMVorHB0CC6AV1QoFK4GPe1LwIDAQABo3sweTAJBgNV
               HRMEAjAAMCwGCWCGSAGG+EIBDQQfFh1PcGVuU1NMIEdlbmVyYXRlZCBDZXJ0aWZp
               Y2F0ZTAdBgNVHQ4EFgQU8pD0U0vsZIsaA16lL8En8bx0F/gwHwYDVR0jBBgwFoAU
               dGeKitcaF7gnzsNwDx708kqaVt0wDQYJKoZIhvcNAQEFBQADgYEAA81SsFnOdYJt
               Ng5Tcq+/ByEDrBgnusx0jloUhByPMEVkoMZ3J7j1ZgI8rAbOkNngX8+pKfTiDz1R
               C4+dx8oU6Za+4NJXUjlL5CvV6BEYb1+QAEJwitTVvxB/A67g42/vzgAtoRUeDov1
               +GFiBZ+GNF/cAYKcMtGcrs2i97ZkJMo="
   }
 ],
 "meta": {
   "resourceType": "User",
   "created": "2010-01-23T04:56:22Z",
   "lastModified": "2011-05-13T04:42:34Z",
   "version": "W\/\"a330bc54f0671c9\"",
```

```
   "location": "https://example.com/v1/Users/2819c223-7f76-453a-919d-41386190464
6"
 }
}
```

Figure 1: Original SCIM data

```
BEGIN:VCARD
VERSION:4.0
UID:"scim:provider.example:org:2819c223-7f76-453a-919d-413861904646"
FN:Ms. Barbara J Jensen III
N:Jensen;Barbera;Jane;Ms.;III
NICKNAME:Babs
URL:"https://login.example.com/bjensen"
EMAIL;TYPE=work;PREF=1:bjensen@example.com
EMAIL;TYPE=home:babs@jensen.org
ADR;LABEL="100 Universal City Plaza\nHollywood, CA 91608 USA";TYPE=work
 :;;100 Universal City Plaza;Hollywood;CA;91608;USA
ADR;LABEL="456 Hollywood Blvd\nHollywood, CA 91608 USA";type=home:;;456
 Hollywood Blvd;Hollywood;CA;91608;USA
TEL;TYPE=voice,work:555-555-5555
TEL;TYPE=cell:555-555-4444
IMPP;TYPE=x-aim:someaimhandle
PHOTO:"https://photos.example.com/profilephoto/72930000000Ccne/F"
PHOTO;TYPE=x-thumbnail:"https://photos.example.com/profilephoto/7293000
 0000Ccne/T"
ROLE:Employee
TITLE:Tour Guide
LANG:en-US
TZ:America/Los_Angeles
KEY:MIIDQzCCAqygAwIBAgICEAAwDQYJKoZIhvcNAQEFBQAwTjELMAkGA1UEBhMCVVMx
 EzARBgNVBAgMCkNhbGlmb3JuaWExFDASBgNVBAoMC2V4YW1wbGUuY29tMRQwEgYD
 VQQDDAtleGFtcGxlLmNvbTAeFw0xMTEwMjIwNjI0MzFaFw0xMjEwMDQwNjI0MzFa
 MH8xCzAJBgNVBAYTAlVTMRMwEQYDVQQIDApDYWxpZm9ybmlhMRQwEgYDVQQKDAtl
 eGFtcGxlLmNvbTEhMB8GA1UEAwwYTXMuIEJhcmJhcmEgSiBKZW5zZW4gSUlJMSIw
 IAYJKoZIhvcNAQkBFhNiamVuc2VuQGV4YW1wbGUuY29tMIIBIjANBgkqhkiG9w0B
 AQEFAAOCAQ8AMIIBCgKCAQEA7Kr+Dcds/JQ5GwejJFcBIP682X3xpjis56AK02bc
 1FLgzdLI8auoR+cC9/Vrh5t66HkQIOdA4unHh0AaZ4xL5PhVbXIPMB5vAPKpzz5i
 PSi8xO8SL7I7SDhcBVJhqVqr3HgllEG6UClDdHO7nkLuwXq8HcISKkbT5WFTVfFZ
 zidPl8HZ7DhXkZIRtJwBweq4bvm3hM1Os7UQH05ZS6cVDgweKNwdLLrT51ikSQG3
 DYrl+ft781UQRIqxgwqCfXEuDiinPh0kkvIi5jivVu1Z9QiwlYEdRbLJ4zJQBmDr
 SGTMYn4lRc2HgHO4DqB/bnMVorHB0CC6AV1QoFK4GPe1LwIDAQABo3sweTAJBgNV
 MIIDQzCCAqygAwIBAgICEAAwDQYJKoZIhvcNAQEFBQAwTjELMAkGA1UEBhMCVVMx
 EzARBgNVBAgMCkNhbGlmb3JuaWExFDASBgNVBAoMC2V4YW1wbGUuY29tMRQwEgYD
 VQQDDAtleGFtcGxlLmNvbTAeFw0xMTEwMjIwNjI0MzFaFw0xMjEwMDQwNjI0MzFa
 MH8xCzAJBgNVBAYTAlVTMRMwEQYDVQQIDApDYWxpZm9ybmlhMRQwEgYDVQQKDAtl
 eGFtcGxlLmNvbTEhMB8GA1UEAwwYTXMuIEJhcmJhcmEgSiBKZW5zZW4gSUlJMSIw
 IAYJKoZIhvcNAQkBFhNiamVuc2VuQGV4YW1wbGUuY29tMIIBIjANBgkqhkiG9w0B
 AQEFAAOCAQ8AMIIBCgKCAQEA7Kr+Dcds/JQ5GwejJFcBIP682X3xpjis56AK02bc
```

```
  1FLgzdLI8auoR+cC9/Vrh5t66HkQIOdA4unHh0AaZ4xL5PhVbXIPMB5vAPKpzz5i
  PSi8xO8SL7I7SDhcBVJhqVqr3HgllEG6UClDdHO7nkLuwXq8HcISKkbT5WFTVfFZ
  zidPl8HZ7DhXkZIRtJwBweq4bvm3hM1Os7UQH05ZS6cVDgweKNwdLLrT51ikSQG3
  DYrl+ft781UQRIqxgwqCfXEuDiinPh0kkvIi5jivVu1Z9QiwlYEdRbLJ4zJQBmDr
  SGTMYn4lRc2HgHO4DqB/bnMVorHB0CC6AV1QoFK4GPe1LwIDAQABo3sweTAJBgNV
  HRMEAjAAMCwGCWCGSAGG+EIBDQQfFh1PcGVuU1NMIEdlbmVyYXRlZCBDZXJ0aWZp
  Y2F0ZTAdBgNVHQ4EFgQU8pD0U0vsZIsaA16lL8En8bx0F/gwHwYDVR0jBBgwFoAU
  dGeKitcaF7gnzsNwDx708kqaVt0wDQYJKoZIhvcNAQEFBQADgYEAA81SsFnOdYJt
  Ng5Tcq+/ByEDrBgnusx0jloUhByPMEVkoMZ3J7j1ZgI8rAbOkNngX8+pKfTiDz1R
  C4+dx8oU6Za+4NJXUjlL5CvV6BEYb1+QAEJwitTVvxB/A67g42/vzgAtoRUeDov1
  +GFiBZ+GNF/cAYKcMtGcrs2i97ZkJMo=
REF:"2011-05-13T04:42:34Z"
END:VCARD
```

Figure 2: After conversion to vCard

7.2. Mapping from vCard to SCIM

Figure 4 contains the result after converting the vCard data from Figure 3 to SCIM.

The following vCard attributes have been omitted in the SCIM representation:

- GENDER
- BDAY

The mandatory "uid" and "userName" attributes have been added to the SCIM representation, although they have not been defined in the vCard.

```
BEGIN:VCARD
VERSION:4.0
FN:Vincent van Gogh
N:van Gogh;Vincent;;;
GENDER:M
BDAY:18530330
ROLE;LANGUAGE="en":painter
LANG;PREF=1:nl
LANG;PREF=2:fr
ADR;LABEL="Vincent van Gogh\n54 Rue Lepic\n75018 Paris\nFrance";LANGUAG
E="fr";TYPE=home:;3th floor;54 Rue Lepic;Paris;;75018;France
TEL;TYPE="work,voice";PREF=1:+33-1-123456
TEL;TYPE="home,voice";PREF=2:+33-1-654321
EMAIL;TYPE=home:vangogh@example.com
URL;TYPE=work:"http://www.vangogh.example.com"
TZ:+0100
END:VCARD
```

Figure 3: Original SCIM data

```
{
  "schemas": ["urn:scim:schemas:core:2.0:User"],
  "id": "xyz",
  "userName": "vangogh@example.com",
  "name": {
    "formatted": "Vincent van Gogh",
    "familyName": "van Gogh",
    "givenName": "Vincent",
  },
  "roles": [
    {
      "value": "painter"
    }
  ],
  "preferredLanguage": "nl",
  "adresses": [
    {
      "type": "home",
      "streetAddress": "54 Rue Lepic",
      "locality": "Paris",
      "postalCode": "75018",
      "country": "France",
      "formatted": "Vincent van Gogh\n54 Rue Lepic\n75018 Paris\nFrance"
    }
  ],
  "phoneNumbers": [
    {
      "value": "+33-1-123456",
      "type": "work"
    },
    {
      "value": "+33-1-654321",
      "type": "home"
    }
  ],
  "emails": [
    {
      "value": "vangogh@example.com",
      "type": "home"
    }
  ],
  "timezone": "+0100"
}
```

Figure 4: Original SCIM data

8. Open issues

The following issues require further consideration:

- It may be feasible to leave out the conversion between SCIM ids and vCarD UIDs, as they may be dependent on the particular application that is importing the information.

- It is unclear on whether the SCIM ID can include alphanumeric characters or is restricted to numeric characters only. The examples in [I-D.ietf-scim-core-schema] seem to indicate that they consist of hexadecimal numbers, with dashes at appropriate places. If this is the case, then during the conversion from vCard UIDs to SCIM IDs would include conversion of alphanumeric characters to hexadecimal values.

- For SCIM fields that have no equivalent vCard attributes, vCard attributes of the form "x-..." could be defined. Alternatively, vCard attributes could be defined, and registered with IANA.

- The "id" and "userName" fields are mandatory in SCIM. However, a vCard does not have to contain similar information. Creating a sensible value of these fields may be left to the SCIM application that is importing the vCard, or guidelines could be defined.

9. IANA Considerations

A "manager" TYPE for the RELATED vCard property may need registration.

10. Security Considerations

The mapping between vCard and SCIM may be useful for easily transferring data for one system towards another. However, it also has privacy implications. Therefore, it is important that user consensus is acquired where applicable.

For this document, some decisions were made concerning mapping between attributes and properties with similar, but not equal, semantics. This was done in a best effort manner. However one should realise that during the mapping process some accuracy from the original data may be lost.

Conversion from SCIM to vCard and subsequently back to SCIM, as well as conversion from vCard to SCIM and subsequently back to vCard SHOULD be avoided.

11. Acknowledgements

Thanks to Kepeng Li for providing feedback and suggestions. Thanks to Paul Madsen and Phil Hunt for providing similar mapping drafts [draft-scim-saml2-binding] and [I-D.hunt-scim-directory], which have served as inspiration for this document. Michael Angstadt and Dany Cauchie provided valuable review comments.

12. References

12.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC5646] Phillips, A. and M. Davis, "Tags for Identifying Languages", BCP 47, RFC 5646, September 2009.

[RFC6350] Perreault, S., "vCard Format Specification", RFC 6350, August 2011.

[RFC6473] Saint-Andre, P., "vCard KIND:application", RFC 6473, December 2011.

[RFC6474] Li, K. and B. Leiba, "vCard Format Extensions: Place of Birth, Place and Date of Death", RFC 6474, December 2011.

[RFC6715] Cauchie, D., Leiba, B., and K. Li, "vCard Format Extensions: Representing vCard Extensions Defined by the Open Mobile Alliance (OMA) Converged Address Book (CAB) Group", RFC 6715, August 2012.

[I-D.ietf-scim-core-schema] Mortimore, C., Harding, P., Madsen, P., and T. Drake, "System for Cross-Domain Identity Management: Core Schema", draft-ietf-scim-core-schema-02 (work in progress), August 2013.

12.2. Informative References

[I-D.hunt-scim-directory] Hunt, P., "SCIM Directory Services", draft-hunt-scim-directory-00 (work in progress), September 2012.

[draft-scim-saml2-binding] Madsen, P., "SAML 2.0 Binding for SCIM", draft-scim-saml2-binding-02 (work in progress), April 2011.

Author's Address

Bert Greevenbosch
Huawei Technologies Co., Ltd.
Huawei Industrial Base F1-8
Bantian, Longgang District
Shenzhen 518129
P.R. China

Phone: +86-755-28979133
Email: bert.greevenbosch@huawei.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 16, 2014

K. Grizzle
SailPoint
P. Hunt, Ed.
Oracle
M. Ansari
Cisco
E. Wahlstroem
Technology Nexus
C. Mortimore
Salesforce
February 12, 2014

System for Cross-Domain Identity Management:Protocol
draft-ietf-scim-api-03

Abstract

The System for Cross-Domain Identity Management (SCIM) specification is designed to make managing user identity in cloud based applications and services easier. The specification suite seeks to build upon experience with existing schemas and deployments, placing specific emphasis on simplicity of development and integration, while applying existing authentication, authorization, and privacy models. It’s intent is to reduce the cost and complexity of user management operations by providing a common user schema and extension model, as well as binding documents to provide patterns for exchanging this schema using standard protocols. In essence, make it fast, cheap, and easy to move users in to, out of, and around the cloud.

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 August 16, 2014.

Table of Contents

1. Introduction and Overview

The SCIM Protocol is an application-level, REST protocol for provisioning and managing identity data on the web. The protocol supports creation, modification, retrieval, and discovery of core identity resources; i.e., Users and Groups, as well as custom resource extensions.

1.1. Intended Audience

This document is intended as a guide to SCIM API usage for both identity service providers and clients.

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 [RFC3896].

1.3. Definitions

Base URL: The SCIM REST API is always relative to a Base URL. The Base URL MUST NOT contain a query string as clients may append additional path information and query parameters as part of forming the request. Example: https://example.com/scim/v2/

2. Authentication and Authorization

The SCIM protocol does not define a scheme for authentication and authorization therefore implementers are free to choose mechanisms appropriate to their use cases. The choice of authentication

mechanism will impact interoperability. It is RECOMMENDED that clients be implemented in such a way that new authentication schemes can be deployed. Implementers SHOULD support existing authentication /authorization schemes. In particular, OAuth2[RFC6750] is RECOMMENDED. Appropriate security considerations of the selected authentication and authorization schemes SHOULD be taken. Because this protocol uses HTTP response status codes as the primary means of reporting the result of a request, servers are advised to respond to unauthorized or unauthenticated requests using the 401 response code in accordance with section 10.4.2 of Section 10.4.2 [RFC2616].

All examples assume OAuth2 bearer token [RFC6750]; e.g.,

```
GET /Users/2819c223-7f76-453a-919d-413861904646 HTTP/1.1
Host: example.com
Authorization: Bearer h480djs93hd8
```

The context of the request (i.e. the user for whom data is being requested) MUST be inferred by service providers.

3. API

The SCIM protocol specifies well known endpoints and HTTP methods for managing resources defined in the core schema; i.e., "User" and "Group" resources correspond to "/Users" and "/Groups" respectively. Service providers that support extended resources SHOULD define resource endpoints using the established convention; pluralize the resource name defined in the extended schema by appending an ’s’. Given there are cases where resource pluralization is ambiguous; e.g., a resource named "Person" is legitimately "Persons" and "People" clients SHOULD discover resource endpoints via the "/ ResourceTypes" endpoint .

GET Retrieves a complete or partial resource.

POST Create new resource, perform an extended Search, or bulk modify resources.

PUT Modifies a resource with a complete, client specified resource (replace).

PATCH Modifies a resource with a set of client specified changes (partial update).

DELETE Deletes a resource.

Resource	Endpoint	Operations	Description
User	/Users	GET (Section 3.2.1), POST (Section 3.1), PUT (Section 3.3.1), PATCH (Section 3.3.2), DELETE (Section 3.4)	Retrieve/Add/Modify Users
Group	/Groups	GET (Section 3.2.1), POST (Section 3.1), PUT (Section 3.3.1), PATCH (Section 3.3.2), DELETE (Section 3.4)	Retrieve/Add/Modify Groups
Service Provider Configuration	/ServiceProviderConfigs	GET (Section 3.2.1)	Retrieve the service provider's configuration
Resource Type	/ResourceTypes	GET (Section 3.2.1)	Retrieve the supported resource types
Schema	/Schemas	GET (Section 3.2.1)	Retrieve a resource's schema
Bulk	/Bulk	POST (Section 3.5)	Bulk modify resources
Search	[prefix]/.search	POST (Section 3.2.3)	Perform a search at system root or within a resource endpoint for one or more resource types using POST.

Table 1: Defined endpoints

All requests to the service provider are made via Section 9 [RFC2616] on a URL derived from the Base URL. Responses are returned in the body of the HTTP response, formatted as JSON. Response and error codes SHOULD be transmitted via the HTTP status code of the response (if possible), and SHOULD also be specified in the body of the response.

3.1. Creating Resources

To create new resources, clients send POST requests to the resource endpoint; i.e., "/Users" or "/Groups".

Successful resource creation is indicated with a 201 ("Created") response code. Upon successful creation, the response body MUST contain the newly created resource. Since the server is free to alter and/or ignore POSTed content, returning the full representation can be useful to the client, enabling it to correlate the client and server views of the new resource. When a resource is created, its URI must be returned in the response Location header.

If the service provider determines creation of the requested resource conflicts with existing resources; e.g., a "User" resource with a duplicate "userName", the service provider MUST return a 409 error and SHOULD indicate the conflicting attribute(s) in the body of the response.

Below, the client sends a POST request containing a user

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

{
  "schemas":["urn:scim:schemas:core:2.0:User"],
  "userName":"bjensen",
  "externalId":"bjensen",
  "name":{
    "formatted":"Ms. Barbara J Jensen III",
    "familyName":"Jensen",
    "givenName":"Barbara"
  }
}
```

The server signals a successful creation with a status code of 201. The response includes a Location header indicating the User URI, and a representation of that user in the body of the response.

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: https://example.com/v2/Users/2819c223-7f76-453a-919d-413861904646
ETag: W/"e180ee84f0671b1"

{
  "schemas":["urn:scim:schemas:core:2.0:User"],
  "id":"2819c223-7f76-453a-919d-413861904646",
  "externalId":"bjensen",
  "meta":{
    "resourceType":"User",
    "created":"2011-08-01T21:32:44.882Z",
    "lastModified":"2011-08-01T21:32:44.882Z",
    "location":"https://example.com/v2/Users/2819c223-7f76-453a-919d-41386190464
6",
    "version":"W\/\"e180ee84f0671b1\""
  },
  "name":{
    "formatted":"Ms. Barbara J Jensen III",
    "familyName":"Jensen",
    "givenName":"Barbara"
  },
  "userName":"bjensen"
}
```

3.1.1. Resource Types

When adding a resource to a specific endpoint, the meta attribute "resourceType" SHALL be set by the service provider to the corresponding resource Type for the endpoint. For example, "/Users" will set "resourceType" to "User", and "/Groups" will set "resourceType" to "Group".

3.2. Retrieving Resources

"User" and "Group" resources are retrieved via opaque, unique URLs or via Query. Service providers MAY choose to respond with a sub-set of resource attributes, though MUST minimally return the resource id and meta attributes.

3.2.1. Retrieving a known Resource

To retrieve a known resource, clients send GET requests to the resource endpoint; e.g., "/Users/{id}" or "/Groups/{id}".

If the resource exists the server responds with a status code of 200 and includes the result in the body of the response.

The below example retrieves a single User via the "/Users" endpoint.

```
GET /Users/2819c223-7f76-453a-919d-413861904646
Host: example.com
Accept: application/json
Authorization: Bearer h480djs93hd8
```

The server responds with:

```
HTTP/1.1 200 OK
Content-Type: application/json
Location: https://example.com/v2/Users/2819c223-7f76-453a-919d-413861904646
ETag: W/"f250dd84f0671c3"

{
  "schemas":["urn:scim:schemas:core:2.0:User"],
  "id":"2819c223-7f76-453a-919d-413861904646",
  "externalId":"bjensen",
  "meta":{
    "resourceType":"User",
    "created":"2011-08-01T18:29:49.793Z",
    "lastModified":"2011-08-01T18:29:49.793Z",
    "location":"https://example.com/v2/Users/2819c223-7f76-453a-919d-41386190464
6",
    "version":"W\/\"f250dd84f0671c3\""
  },
  "name":{
    "formatted":"Ms. Barbara J Jensen III",
    "familyName":"Jensen",
    "givenName":"Barbara"
  },
  "userName":"bjensen",
  "phoneNumbers":[
    {
      "value":"555-555-8377",
      "type":"work"
    }
  ],
  "emails":[
    {
      "value":"bjensen@example.com",
      "type":"work"
    }
  ]
}
```

3.2.2. List/Query Resources

SCIM defines a standard set of operations that can be used to filter, sort, and paginate response results. The operations are specified by adding query parameters to the resource's endpoint. Service providers MAY support additional query parameters not specified here, and Providers SHOULD ignore any query parameters they don't recognize.

List and query responses MUST be identified using the following URI: "urn:scim:schemas:core:2.0:ListResponse". The following attributes are defined for list and query responses:

totalResults The total number of results returned by the list or query operation. This may not be equal to the number of elements in the resources attribute of the list response if pagination (Section 3.2.2.4) is requested. REQUIRED.

Resources A multi-valued list of complex objects containing the requested resources. This may be a subset of the full set of resources if pagination (Section 3.2.2.4) is requested. REQUIRED.

startIndex The 1-based index of the first result in the current set of list results. REQUIRED if pagination (Section 3.2.2.4) is requested.

itemsPerPage The number of resources returned in a list response page. REQUIRED if pagination (Section 3.2.2.4) is requested.

The below example returns the userName for all Users:

```
GET /Users?attributes=userName
Host: example.com
Accept: application/json
Authorization: Bearer h480djs93hd8
```

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

{
  "schemas":["urn:scim:schemas:core:2.0:ListResponse"],
  "totalResults":2,
  "Resources":[
    {
      "userName":"bjensen"
    },
    {
      "userName":"jsmith"
    }
  ]
}
```

3.2.2.1. Query Endpoints

Queries MAY be performed against a SCIM resource object or a resource type endpoint. For example:

"/Users/{userid}"

"/Users"

"/Groups"

A server MAY support searches against the server root (e.g. "/"). A search against a server root indicates that ALL resources within the server SHALL be included subject to filtering. A filter expression using "meta.resourceType" MAY be used to restrict results to one or more specific resource types (e.g. "User").

When processing search operations across endpoints that 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 Section 3.2.2.2. 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.

3.2.2.2. Filtering

Filtering is OPTIONAL. Clients may request a subset of resources by specifying the 'filter' URL query parameter containing a filter expression. When specified only those resources matching the filter expression SHALL be returned. The expression language that is used in the filter parameter supports references to attributes and

literals. The literal values can be strings enclosed in double quotes, numbers, date times enclosed in double quotes, and Boolean values; i.e., true or false. String literals MUST be valid [RFC4627].

The attribute name and attribute operator are case insensitive. For example, the following two expressions will evaluate to the same logical value:

filter=userName Eq "john"

filter=Username eq "john"

The filter parameter MUST contain at least one valid Boolean expression. Each expression MUST contain an attribute name followed by an attribute operator and optional value. Multiple expressions MAY be combined using the two logical operators. Furthermore expressions can be grouped together using "()".

The operators supported in the expression are listed in the following table.

Operator	Description	Behavior
eq	equal	The attribute and operator values must be identical for a match.
ne	not equal	The attribute and operator values are not identical.
co	contains	The entire operator value must be a substring of the attribute value for a match.
sw	starts with	The entire operator value must be a substring of the attribute value, starting at the beginning of the attribute value. This criterion is satisfied if the two strings are identical.
ew	ends with	The entire operator value must be a substring of the attribute value, matching at the end of the attribute value. This criterion is satisfied if the two strings are identical.
pr	present (has value)	If the attribute has a non-empty value, or if it contains a non-empty node for complex attributes there is a match.
gt	greater than	If the attribute value is greater than operator value, there is a match. The

		actual comparison is dependent on the attribute type. For string attribute types, this is a lexicographical comparison and for DateTime types, it is a chronological comparison.
ge	greater than or equal	If the attribute value is greater than or equal to the operator value, there is a match. The actual comparison is dependent on the attribute type. For string attribute types, this is a lexicographical comparison and for DateTime types, it is a chronological comparison.
lt	less than	If the attribute value is less than operator value, there is a match. The actual comparison is dependent on the attribute type. For string attribute types, this is a lexicographical comparison and for DateTime types, it is a chronological comparison.
le	less than or equal	If the attribute value is less than or equal to the operator value, there is a match. The actual comparison is dependent on the attribute type. For string attribute types, this is a lexicographical comparison and for DateTime types, it is a chronological comparison.

Table 2: Attribute Operators

Operator	Description	Behavior
and	Logical And	The filter is only a match if both expressions evaluate to true.
or	Logical or	The filter is a match if either expression evaluates to true.
not	Not function	The filter is a match if the expression evaluates to false.

Table 3: Logical Operators

Operator	Description	Behavior
()	Precedence grouping	Boolean expressions may be grouped using parentheses to change the standard order of operations; i.e., evaluate OR logical operators before logical AND operators.
[]	Complex attribute filter grouping	Service providers MAY support complex filters where expressions MUST be applied to the same value of a parent attribute specified immediately before the left square bracket ("["). The expression within square brackets ("[" and "]") MUST be a valid filter expression based upon sub-attributes of the parent attribute. Nested expressions MAY be used. See examples below.

Table 4: Grouping Operators

Filters MUST be evaluated using standard order of operations [Order-Operations]. Attribute operators have the highest precedence, followed by the grouping operator (i.e, parentheses), followed by the logical AND operator, followed by the logical OR operator.

If the specified attribute in a filter expression is a multi-valued attribute, the resource MUST match if any of the instances of the given attribute match the specified criterion; e.g. if a User has multiple emails values, only one has to match for the entire User to match. For complex attributes, a fully qualified Sub-Attribute MUST be specified using standard attribute notation (Section 3.8). For example, to filter by userName the parameter value is userName and to filter by first name, the parameter value is name.givenName.

Providers MAY support additional filter operations if they choose. Providers MUST decline to filter results if the specified filter operation is not recognized and return a HTTP 400 error with an appropriate human readable response. For example, if a client specified an unsupported operator named 'regex' the service provider should specify an error response description identifying the client error; e.g., 'The operator 'regex' is not supported.'

String type attributes are case insensitive by default unless the attribute type is defined as a caseExact string. Attribute operators 'eq', 'co', and 'sw' MUST perform caseIgnore matching for all string attributes unless the attribute is defined as caseExact. By default all string attributes are caseIgnore.

Clients MAY search by schema or schema extensions by using a filter expression including the "schemas" attribute.

The following are examples of valid filters. Some attributes (e.g. rooms and rooms.number) are hypothetical extensions and are not part of SCIM core schema:

```
filter=userName eq "bjensen"

filter=name.familyName co "O'Malley"

filter=userName sw "J"

filter=title pr

filter=meta.lastModified gt "2011-05-13T04:42:34Z"

filter=meta.lastModified ge "2011-05-13T04:42:34Z"

filter=meta.lastModified lt "2011-05-13T04:42:34Z"

filter=meta.lastModified le "2011-05-13T04:42:34Z"

filter=title pr and userType eq "Employee"

filter=title pr or userType eq "Intern"

filter=schemas eq "urn:scim:schemas:extension:enterprise:2.0:User"

filter=userType eq "Employee" and (emails co "example.com" or emails
  co "example.org")

filter=userType ne "Employee" and not (emails co "example.com" or
  emails co "example.org")

filter=userType eq "Employee" and (emails.type eq "work")

filter=userType eq "Employee" and emails[type eq "work" and
  value co "@example.com"]

filter=emails[type eq "work" and value co "@example.com"] or ims[type
  eq "xmpp" and value co "@foo.com"]

filter=addresses[state eq "CA" and rooms[type eq "bedroom" and
  number gt 2]]
```

3.2.2.3. Sorting

Sort is OPTIONAL. Sorting allows clients to specify the order in which resources are returned by specifying a combination of sortBy and sortOrder URL parameters.

sortBy: The sortBy parameter specifies the attribute whose value SHALL be used to order the returned responses. If the sortBy attribute corresponds to a singular attribute, resources are sorted according to that attribute's value; if it's a multi-valued attribute, resources are sorted by the value of the primary attribute, if any, or else the first value in the list, if any. If the attribute is complex the attribute name must be a path to a sub-attribute in standard attribute notation (Section 3.8) ; e.g., "sortBy=name.givenName". For all attribute types, if there is no data for the specified "sortBy" value they are sorted via the "sortOrder" parameter; i.e., they are ordered last if ascending and first if descending.

sortOrder: The order in which the sortBy parameter is applied. Allowed values are "ascending" and "descending". If a value for sortBy is provided and no sortOrder is specified, the sortOrder SHALL default to ascending. String type attributes are case insensitive by default unless the attribute type is defined as a case exact string. "sortOrder" MUST sort according to the attribute type; i.e., for "caseIgnore" attributes, sort the result using case insensitive, unicode alphabetic sort order, with no specific locale implied and for caseExact attribute types, sort the result using case sensitive, Unicode alphabetic sort order.

3.2.2.4. Pagination

Pagination parameters can be used together to "page through" large numbers of resources so as not to overwhelm the client or service provider. Pagination is not session based hence clients SHOULD never assume repeatable results. For example, a request for a list of 10 resources beginning with a startIndex of 1 may return different results when repeated as a resource in the original result could be deleted or new ones could be added in-between requests. Pagination parameters and general behavior are derived from the OpenSearch Protocol [OpenSearch].

The following table describes the URL pagination parameters.

Parameter	Description	Default
startIndex	The 1-based index of the first search result.	1
count	Non-negative Integer. Specifies the desired maximum number of search results per page; e.g., 10.	None. When specified the service provider MUST not return more results than specified though MAY return fewer results. If unspecified, the maximum number of results is set by the service provider.

Table 5: Pagination Request parameters

The following table describes the query response pagination attributes specified by the service provider.

Element	Description
itemsPerPage	Non-negative Integer. Specifies the number of search results returned in a query response page; e.g., 10.
totalResults	Non-negative Integer. Specifies the total number of results matching the client query; e.g., 1000.
startIndex	The 1-based index of the first result in the current set of search results; e.g., 1.

Table 6: Pagination Response Elements

For example, to retrieve the first 10 Users set the startIndex to 1 and the count to 10.

```
GET /Users?startIndex=1&count=10
Host: example.com
Accept: application/json
Authorization: Bearer h480djs93hd8
```

```
{
  "totalResults":100,
  "itemsPerPage":10,
  "startIndex":1,
  "schemas":["urn:scim:schemas:core:2.0"],
  "Resources":[{
    ...
  }]
}
```

Given the example above, to continue paging set the startIndex to 11 and re-fetch; i.e., /Users?startIndex=11&count=10

3.2.3. Querying Resources Using HTTP POST

Clients MAY execute queries without passing parameters on the URL by using the HTTP POST verb combined with the '/.search' path extension. The inclusion of '/.search' on the end of a valid SCIM endpoint SHALL be used to indicate the HTTP POST verb is intended to be a query operation.

To create a new search result set, a SCIM client sends an HTTP POST request to the desired SCIM resource endpoint (ending in '/.search'). The body of the POST request MAY include any of the parameters as defined in Section 3.2.2.

Search requests MUST be identified using the following URI: 'urn:scim:schemas:core:2.0:SearchRequest'. The following attributes are defined for search requests:

attributes A multi-valued list of strings indicating the names of resource attributes to return in the response. Attribute names MUST be in standard attribute notation (Section 3.8) form. See additional retrieval query parameters (Section 3.7). OPTIONAL.

filter The filter string used to request a subset of resources. The filter string MUST be a valid filter (Section 3.2.2.2) expression. OPTIONAL.

sortBy A string indicating the attribute whose value SHALL be used to order the returned responses. The sortBy attribute MUST be in standard attribute notation (Section 3.8) form. See sorting (Section 3.2.2.3). OPTIONAL.

sortOrder A string indicating the order in which the sortBy parameter is applied. Allowed values are "ascending" and "descending". See sorting (Section 3.2.2.3). OPTIONAL.

startIndex An integer indicating the 1-based index of the first search result. See pagination (Section 3.2.2.4). OPTIONAL.

count An integer indicating the desired maximum number of search results per page. See pagination (Section 3.2.2.4). OPTIONAL.

After receiving a HTTP POST request, a response is returned as specified in Section 3.2.2.

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

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

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

Figure 1: Example POST Search Request

A search response is shown with the first page of results. For brevity reasons, only two matches are shown: one User and one Group.

```
HTTP/1.1 200 OK
Content-Type: application/json
Location: https://example.com/.search
{
  "schemas": ["urn:scim:schemas:core:2.0:ListResponse"],
  "totalResults":100,
  "itemsPerPage":10,
  "startIndex":1,
  "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 2: Example POST Search Response

3.3. Modifying Resources

Resources can be modified in whole or in part via PUT or PATCH, respectively. Implementers MUST support PUT as specified in Section 9.6 [RFC2616] . Resources such as Groups may be very large hence implementers SHOULD support PATCH [RFC5789] to enable partial resource modifications.

3.3.1. Modifying with PUT

PUT performs a full update. Clients MAY retrieve the entire resource in advance, add the desired modifications and use HTTP PUT which will overwrite all previously stored data. Since the PUT request performs a full update, clients MAY send attributes of the retrieved resource and the service provider MUST process according to attribute mutability as follows:

readWrite, writeOnly Any values provided SHALL replace the existing attribute values. Omitting the attribute or specific values means the attribute or specific value SHALL be removed;

immutable If values are provided for elements already set in the attribute they MUST match existing data or an error is returned. If the service provider has no existing values, a new value(s) MAY be specified; and,

readOnly Any values provided (e.g. meta.resourceType) SHALL be ignored.

If an attribute is "required", the client MUST specify the attribute in the PUT request.

If a value provided for an immutable attribute with an existing value is NOT matched, the server SHALL respond with an HTTP response code of 400 and an apprpriate human readable message indicating an attempt to change an immutable attribute.

Unless otherwise specified a successful PUT operation returns a 200 OK response code and the entire resource within the response body, enabling the client to correlate the client’s and Provider’s views of the updated resource. Example:

```
PUT /Users/2819c223-7f76-453a-919d-413861904646
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Bearer h480djs93hd8
If-Match: W/"a330bc54f0671c9"

{
  "schemas":["urn:scim:schemas:core:2.0:User"],
  "id":"2819c223-7f76-453a-919d-413861904646",
  "userName":"bjensen",
  "externalId":"bjensen",
  "name":{
    "formatted":"Ms. Barbara J Jensen III",
    "familyName":"Jensen",
    "givenName":"Barbara",
    "middleName":"Jane"
  },
  "emails":[
    {
        "value":"bjensen@example.com"
    },
    {
        "value":"babs@jensen.org"
    }
  ]
}
```

The service responds with the entire, updated User

```
HTTP/1.1 200 OK
Content-Type: application/json
ETag: W/"b431af54f0671a2"
Location:"https://example.com/v2/Users/2819c223-7f76-453a-919d-413861904646"
{
  "schemas":["urn:scim:schemas:core:2.0:User"],
  "id":"2819c223-7f76-453a-919d-413861904646",
  "userName":"bjensen",
  "externalId":"bjensen",
  "name":{
    "formatted":"Ms. Barbara J Jensen III",
    "familyName":"Jensen",
    "givenName":"Barbara",
    "middleName":"Jane"
  },
  "emails":[
    {
        "value":"bjensen@example.com"
    },
    {
        "value":"babs@jensen.org"
    }
  ],
  "meta": {
    "resourceType":"User",
    "created":"2011-08-08T04:56:22Z",
    "lastModified":"2011-08-08T08:00:12Z",
    "location":"https://example.com/v2/Users/2819c223-7f76-453a-919d-41386190464
6",
    "version":"W\/\"b431af54f0671a2\""
  }
}
```

3.3.2. Modifying with PATCH

PATCH is OPTIONAL. PATCH enables clients to send only those attributes requiring modification, reducing network and processing overhead. Attributes may be deleted, replaced, merged, or added in a single request.

The body of a PATCH request MUST contain a partial resource with the desired modifications. The server MUST return either a 200 OK response code and the entire resource (subject to the "attributes" query parameter - see Additional Retrieval Query Parameters (Section 3.7)) within the response body, or a 204 No Content response code and the appropriate response headers for a successful PATCH request. The server MUST return a 200 OK if the "attributes" parameter is specified on the request.

The server MUST process a PATCH request by first removing any attributes specified in the meta.attributes Sub-Attribute (if present) and then merging the attributes in the PATCH request body into the resource.

The meta.attributes Sub-Attribute MAY contain a list of attributes to be removed from the resource. If the PATCH request body contains an attribute that is present in the meta.attributes list, the attribute on the resource is replaced with the value from the PATCH body. If the attribute is complex the attribute name must be a path to a Sub-Attribute in standard attribute notation (Section 3.8); e.g., name.givenName.

Attributes that exist in the PATCH request body but not in the meta.attributes Sub-Attribute will be either be updated or added to the resource according to the following rules.

Singular attributes: Singular attributes in the PATCH request body replace the attribute on the resource.

Complex attributes: Complex Sub-Attribute values in the PATCH request body are merged into the complex attribute on the resource.

Multi-valued attributes: An attribute value in the PATCH request body is added to the value collection if the value does not exist and merged if a matching value is present. Values are matched by comparing the value Sub-Attribute from the PATCH request body to the value Sub-Attribute of the resource. Attributes that do not have a value Sub-Attribute; e.g., addresses, or do not have unique value Sub-Attributes cannot be matched and must instead be deleted then added. Specific values can be removed from a resource by adding an "operation" Sub-Attribute with the value "delete" to the attribute in the PATCH request body. As with adding/updating attribute value collections, the value to delete is determined by comparing the value Sub-Attribute from the PATCH request body to the value Sub-Attribute of the resource. Attributes that do not have a value Sub-Attribute or that have a non-unique value Sub-Attribute are matched by comparing all Sub-Attribute values from the PATCH request body to the Sub-Attribute values of the resource. A delete operation is ignored if the attribute’s name is in the meta.attributes list. If the requested value to delete does not match a unique value on the resource the server MAY return a HTTP 400 error.

The following example shows how to add a member to a group:

```
PATCH /Groups/acbf3ae7-8463-4692-b4fd-9b4da3f908ce
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Bearer h480djs93hd8
If-Match: W/"a330bc54f0671c9"

{
  "schemas": ["urn:scim:schemas:core:2.0:Group"],
  "members": [
    {
      "display": "Babs Jensen",
      "$ref": "https://example.com/v2/Users/2819c223-7f76-453a-919d-413861904646
",
      "value": "2819c223-7f76-453a-919d-413861904646"
    }
  ]
}
```

The "display" Sub-Attribute in this request is optional since the value attribute uniquely identifies the user to be added. If the user was already a member of this group, no changes should be made to the resource and a success response should be returned. The server responds with either the entire updated Group or no response body:

```
HTTP/1.1 204 No Content
Authorization: Bearer h480djs93hd8
ETag: W/"b431af54f0671a2"
Location: "https://example.com/v2/Groups/acbf3ae7-8463-4692-b4fd-9b4da3f908ce"
```

The following example shows how to remove a member from a group. As with the previous example, the "display" Sub-Attribute is optional. If the user was not a member of this group, no changes should be made to the resource and a success response should be returned.

Note that server responses have been omitted for the rest of the PATCH examples.

```
PATCH /Groups/acbf3ae7-8463-4692-b4fd-9b4da3f908ce
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Bearer h480djs93hd8
If-Match: W/"a330bc54f0671c9"

{
  "schemas": ["urn:scim:schemas:core:2.0:Group"],
  "members": [
    {
      "display": "Babs Jensen",
      "$ref": "https://example.com/v2/Users/2819c223-7f76-453a-919d-413861904646
",
      "value": "2819c223-7f76-453a-919d-413861904646",
      "operation": "delete"
    }
  ]
}
```

The following example shows how to remove all members from a group:

```
PATCH /Groups/acbf3ae7-8463-4692-b4fd-9b4da3f908ce
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Bearer h480djs93hd8
If-Match: W/"a330bc54f0671c9"

{
  "schemas": ["urn:scim:schemas:core:2.0:Group"],
  "meta": {
    "attributes": [
      "members"
    ]
  }
}
```

The following example shows how to replace all of the members of a group with a different members list:

```
PATCH /Groups/acbf3ae7-8463-4692-b4fd-9b4da3f908ce
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Bearer h480djs93hd8
If-Match: W/"a330bc54f0671c9"

{
  "schemas": ["urn:scim:schemas:core:2.0:Group"],
  "meta": {
    "attributes": [
      "members"
    ]
  },
  "members": [
    {
      "display": "Babs Jensen",
      "$ref": "https://example.com/v2/Users/2819c223-7f76-453a-919d-413861904646
",
      "value": "2819c223-7f76-453a-919d-413861904646"
    },
    {
      "display": "James Smith",
      "$ref": "https://example.com/v2/Users/08e1d05d-121c-4561-8b96-473d93df9210
",
      "value": "08e1d05d-121c-4561-8b96-473d93df9210"
    }
  ]
}
```

The following example shows how to add a member to and remove a member from a Group in a single request:

```
PATCH /Groups/acbf3ae7-8463-4692-b4fd-9b4da3f908ce
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Bearer h480djs93hd8
If-Match: W/"a330bc54f0671c9"

{
  "schemas": ["urn:scim:schemas:core:2.0:Group"],
  "members": [
    {
      "display": "Babs Jensen",
      "$ref": "https://example.com/v2/Users/2819c223-7f76-453a-919d-413861904646
",
      "value": "2819c223-7f76-453a-919d-413861904646",
      "operation": "delete"
    },
    {
      "display": "James Smith",
      "$ref": "https://example.com/v2/Users/08e1d05d-121c-4561-8b96-473d93df9210
",
      "value": "08e1d05d-121c-4561-8b96-473d93df9210"
    }
  ]
}
```

The following example shows how to change a User's primary email. If the User already has the email address, it is made the primary address and the current primary address (if present) is made non-primary. If the User does not already have the email address, it is added and made the primary address.

```
PATCH /Users/2819c223-7f76-453a-919d-413861904646
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Bearer h480djs93hd8
If-Match: W/"a330bc54f0671c9"

{
  "schemas": ["urn:scim:schemas:core:2.0:User"],
  "emails": [
    {
      "value": "bjensen@example.com",
      "primary": true
    }
  ]
}
```

The following example shows how to change a User's address. Since address does not have a value Sub-Attribute, the existing address must be removed and the modified address added.

```
PATCH /Users/2819c223-7f76-453a-919d-413861904646
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Bearer h480djs93hd8
If-Match: W/"a330bc54f0671c9"

{
  "schemas": ["urn:scim:schemas:core:2.0:User"],
  "addresses": [
    {
      "type": "work",
      "streetAddress": "100 Universal City Plaza",
      "locality": "Hollywood",
      "region": "CA",
      "postalCode": "91608",
      "country": "US",
      "formatted": "100 Universal City Plaza\nHollywood, CA 91608 US",
      "primary": true,
      "operation": "delete"
    },
    {
      "type": "work",
      "streetAddress": "911 Universal City Plaza",
      "locality": "Hollywood",
      "region": "CA",
      "postalCode": "91608",
      "country": "US",
      "formatted": "911 Universal City Plaza\nHollywood, CA 91608 US",
      "primary": true
    }
  ]
}
```

The following example shows how to change a User's nickname:

```
PATCH /Users/2819c223-7f76-453a-919d-413861904646
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Bearer h480djs93hd8
If-Match: W/"a330bc54f0671c9"

{
  "schemas": ["urn:scim:schemas:core:2.0:User"],
  "nickName": "Barbie"
}
```

The following example shows how to remove a User's nickname:

```
PATCH /Users/2819c223-7f76-453a-919d-413861904646
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Bearer h480djs93hd8
If-Match: W/"a330bc54f0671c9"

{
  "schemas": ["urn:scim:schemas:core:2.0:User"],
  "meta": {
    "attributes": [
      "nickName"
    ]
  }
}
```

The following example shows how to change a User's familyName. This only updates the familyName and formatted on the "name" complex attribute. Any other name Sub-Attributes on the resource remain unchanged.

```
PATCH /Users/2819c223-7f76-453a-919d-413861904646
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Bearer h480djs93hd8
If-Match: W/"a330bc54f0671c9"

{
  "schemas": ["urn:scim:schemas:core:2.0:User"],
  "name": {
    "formatted": "Ms. Barbara J Jensen III",
    "familyName": "Jensen"
  }
}
```

The following example shows how to remove a complex Sub-Attribute and an extended schema attribute from a User.

```
PATCH /Users/2819c223-7f76-453a-919d-413861904646
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Bearer h480djs93hd8
If-Match: W/"a330bc54f0671c9"

{
  "schemas": ["urn:scim:schemas:core:2.0:User"],
  "meta": {
    "attributes": [
      "name.formatted",
      "urn:hr:schemas:user:age"
    ]
  }
}
```

3.4. Deleting Resources

Clients request resource removal via DELETE. Service providers MAY choose not to permanently delete the resource, but MUST return a 404 error code for all operations associated with the previously deleted Id. Service providers MUST also omit the resource from future query results. In addition the service provider MUST not consider the deleted resource in conflict calculation. For example if a User resource is deleted, a CREATE request for a User resource with the same userName as the previously deleted resource should not fail with a 409 error due to userName conflict.

```
DELETE /Users/2819c223-7f76-453a-919d-413861904646
Host: example.com
Authorization: Bearer h480djs93hd8
If-Match: W/"c310cd84f0281b7"
```

In response to a successful delete, the server SHALL respond with successful HTTP status 204 (No Content). A non-normative example response:

```
HTTP/1.1 204 No Content
```

Example: client attempt to retrieve the previously deleted User

```
GET /Users/2819c223-7f76-453a-919d-413861904646
Host: example.com
Authorization: Bearer h480djs93hd8
```

Server Response:

```
HTTP/1.1 404 NOT FOUND

{
  "schemas": ["urn:scim:schemas:core:2.0:Error"],
  "Errors":[
    {
      "description":"Resource 2819c223-7f76-453a-919d-413861904646 not found",
      "code":"404"
    }
  ]
}
```

3.5. Bulk

Bulk is OPTIONAL. The bulk operation enables clients to send a potentially large collection of resource operations in a single request. The body of a a bulk operation contains a set of HTTP resource operations using one of the API supported HTTP methods; i.e., POST, PUT, PATCH or DELETE.

Bulk requests are identified using the following URI: 'urn:scim:schemas:core:2.0:BulkRequest'. Bulk responses are identified using the following URI:

'urn:scim:schemas:core:2.0:BulkResponse'. Bulk requests and bulk responses share many attributes. Unless otherwise specified, each attribute below is present in both bulk requests and bulk responses.

The following Singular Attribute is defined in addition to the common attributes defined in SCIM core schema.

failOnErrors An Integer specifying the number of errors that the service provider will accept before the operation is terminated and an error response is returned. OPTIONAL in a request. Not valid in a response.

The following Complex Multi-valued Attribute is defined in addition to the common attributes defined in core schema.

Operations Defines operations within a bulk job. Each operation corresponds to a single HTTP request against a resource endpoint. REQUIRED.

method The HTTP method of the current operation. Possible values are POST, PUT, PATCH or DELETE. REQUIRED.

bulkId The transient identifier of a newly created resource, unique within a bulk request and created by the client. The bulkId serves as a surrogate resource id enabling clients to uniquely identify newly created resources in the Response and cross reference new resources in and across operations within a bulk request. REQUIRED when method is POST.

version The current resource version. Version is REQUIRED if the service provider supports ETags and the method is PUT, DELETE, or PATCH.

path The resource's relative path. If the method is POST the value must specify a resource type endpoint; e.g., /Users or /Groups whereas all other method values must specify the path to a specific resource; e.g., /Users/2819c223-7f76-453a-919d-413861904646. REQUIRED in a request.

data The resource data as it would appear for a single POST, PUT or PATCH resource operation. REQUIRED in a request when method is POST, PUT and PATCH.

location The resource endpoint URL. REQUIRED in a response, except in the event of a POST failure.

status A complex type that contains information about the success or failure of one operation within the bulk job. REQUIRED in a response.

code The HTTP response code that would have been returned if a a single HTTP request would have been used. REQUIRED.

description A human readable error message. REQUIRED when an error occurred.

If a bulk job is processed successfully the HTTP response code 200 OK MUST be returned, otherwise an appropriate HTTP error code MUST be returned.

The service provider MUST continue performing as many changes as possible and disregard partial failures. The client MAY override this behavior by specifying a value for failOnErrors attribute. The failOnErrors attribute defines the number of errors that the service provider should accept before failing the remaining operations returning the response.

To be able to reference a newly created resource the attribute bulkId MUST be specified when creating new resources. The bulkId is defined by the client as a surrogate identifier in a POST operation. The service provider MUST return the same bulkId together with the newly created resource. The bulkId can then be used by the client to map the service provider id with the bulkId of the created resource.

There can be more then one operation per resource in each bulk job. The Service client MUST take notice of the unordered structure of JSON and the service provider can process operations in any order. For example, if the Service client sends two PUT operations in one request, the outcome is non-deterministic.

The service provider response MUST include the result of all processed operations. A location attribute that includes the resource’s end point MUST be returned for all operations excluding failed POSTs. The status attribute includes information about the success or failure of one operation within the bulk job. The attribute status MUST include the code attribute that holds the HTTP response code that would have been returned if a single HTTP request would have been used. If an error occurred the status MUST also include the description attribute containing a human readable explanation of the error.

```
   "status": {
     "code": "201"
   }
```

   The following is an example of a status in a failed operation.

```
"status": {
  "code": "400",
  "description": "Request is unparseable, syntactically incorrect, or violates s
chema."
}
```

   The following example shows how to add, update, and remove a user. The failOnErrors attribute is set to ’1’ indicating the service provider should return on the first error. The POST operation’s bulkId value is set to ’qwerty’ enabling the client to match the new User with the returned resource id ’92b725cd-9465-4e7d-8c16-01f8e146b87a’.

```
   POST /v2/Bulk
   Host: example.com
   Accept: application/json
   Content-Type: application/json
   Authorization: Bearer h480djs93hd8
   Content-Length: ...

   {
     "schemas": ["urn:scim:schemas:core:2.0:BulkRequest"],
     "failOnErrors":1,
     "Operations":[
       {
         "method":"POST",
         "path":"/Users",
         "bulkId":"qwerty",
         "data":{
           "schemas": ["urn:scim:schemas:core:2.0:User"],
           "userName":"Alice"
         }
       },
       {
         "method":"PUT",
         "path":"/Users/b7c14771-226c-4d05-8860-134711653041",
         "version":"W\/\"3694e05e9dff591\"",
         "data":{
           "schemas": ["urn:scim:schemas:core:2.0:User"],
           "id":"b7c14771-226c-4d05-8860-134711653041",
           "userName":"Bob"
```

```
        }
      },
      {
        "method":"PATCH",
        "path":"/Users/5d8d29d3-342c-4b5f-8683-a3cb6763ffcc",
        "version":"W\/\"edac3253e2c0ef2\"",
        "data":{
          "schemas":["urn:scim:schemas:core:2.0:User"],
          "id":"5d8d29d3-342c-4b5f-8683-a3cb6763ffcc",
          "userName":"Dave",
          "meta":{
            "attributes":[
              "nickName"
            ]
          }
        }
      },
      {
        "method":"DELETE",
        "path":"/Users/e9025315-6bea-44e1-899c-1e07454e468b",
        "version":"W\/\"0ee8add0a938e1a\""
      }
    ]
  }
```

The service provider returns the following response.

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

{
    "schemas": ["urn:scim:schemas:core:2.0:BulkResponse"],
    "Operations": [
        {
            "location": "https://example.com/v2/Users/92b725cd-9465-4e7d-8c16-01
f8e146b87a",
            "method": "POST",
            "bulkId": "qwerty",
            "version": "W\/\"oY4m4wn58tkVjJxK\"",
            "status": {
                "code": "201"
            }
        },
        {
            "location": "https://example.com/v2/Users/b7c14771-226c-4d05-8860-13
4711653041",
            "method": "PUT",
            "version": "W\/\"huJj29dMNgu3WXPD\"",
            "status": {
                "code": "200"
            }
        },
        {
            "location": "https://example.com/v2/Users/5d8d29d3-342c-4b5f-8683-a3
cb6763ffcc",
            "method": "PATCH",
            "version": "W\/\"huJj29dMNgu3WXPD\"",
            "status": {
                "code": "200"
            }
        },
        {
            "location": "https://example.com/v2/Users/e9025315-6bea-44e1-899c-1e
07454e468b",
            "method": "DELETE",
            "status": {
                "code": "204"
            }
        }
    ]
}
```

The following response is returned if an error occurred when attempting to create the User 'Alice'. The service provider stops processing the bulk operation and immediately returns a response to the client. The response contains the error and any successful results prior to the error.

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

{
  "schemas": ["urn:scim:schemas:core:2.0:BulkResponse"],
  "Operations": [
    {
      "method": "POST",
      "bulkId": "qwerty",
      "status": {
        "code": "400",
        "description": "Request is unparseable, syntactically incorrect, or viol
ates schema."
      }
    }
  ]
}
```

If the failOnErrors attribute is not specified or the service provider has not reached the error limit defined by the client the service provider will continue to process all operations. The following is an example in which all operations failed.

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

{
  "schemas": ["urn:scim:schemas:core:2.0:BulkResponse"],
  "Operations": [
    {
      "method": "POST",
      "bulkId": "qwerty",
      "status": {
        "code": "400",
        "description": "Request is unparseable, syntactically incorrect, or viol
ates schema."
      }
    },
    {
      "location": "https://example.com/v2/Users/b7c14771-226c-4d05-8860-13471165
3041",
      "method": "PUT",
      "status": {
        "code": "412",
        "description": "Failed to update as user changed on the server since you
 last retrieved it."
      }
    },
    {
      "location": "https://example.com/v2/Users/5d8d29d3-342c-4b5f-8683-a3cb6763
ffcc",
      "method": "PATCH",
      "status": {
        "code": "412",
        "description": "Failed to update as user changed on the server since you
 last retrieved it."
      }
    },
    {
      "location": "https://example.com/v2/Users/e9025315-6bea-44e1-899c-1e07454e
468b",
      "method": "DELETE",
      "status": {
        "code": "404",
        "description": "Specified resource; e.g., User, does not exist."
      }
    }
  ]
}
```

The client can, within one bulk operation, create a new User, a new Group and add the newly created User to the newly created Group. In order to add the new User to the Group the client must use the surrogate id attribute, bulkId, to reference the User. The bulkId attribute value must be pre-pended with the literal "bulkId:"; e.g., if the bulkId is ’qwerty’ the value is "bulkId:qwerty". The service

provider MUST replace the string "bulkId:qwerty" with the permanent resource id once created.

The following example creates a User with the userName ’Alice’ and a Group with the displayName ’Tour Guides’ with Alice as a member.

```
POST /v2/Bulk
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Bearer h480djs93hd8
Content-Length: ...

{
  "schemas": ["urn:scim:schemas:core:2.0:BulkRequest"],
  "Operations": [
    {
      "method": "POST",
      "path": "/Users",
      "bulkId": "qwerty",
      "data": {
        "schemas": ["urn:scim:schemas:core:2.0:User"],
        "userName": "Alice"
      }
    },
    {
      "method": "POST",
      "path": "/Groups",
      "bulkId": "ytrewq",
      "data": {
        "schemas": ["urn:scim:schemas:core:2.0:Group"],
        "displayName": "Tour Guides",
        "members": [
          {
            "type": "user",
            "value": "bulkId:qwerty"
          }
        ]
      }
    }
  ]
}
```

The service provider returns the following response.

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

{
  "schemas": ["urn:scim:schemas:core:2.0:BulkResponse"],
  "Operations": [
    {
      "location": "https://example.com/v2/Users/92b725cd-9465-4e7d-8c16-01f8e146
b87a",
      "method": "POST",
      "bulkId": "qwerty",
      "version": "W\/\"4weymrEsh5O6cAEK\"",
      "status": {
        "code": "201"
      }
    },
    {
      "location": "https://example.com/v2/Groups/e9e30dba-f08f-4109-8486-d5c6a33
1660a",
      "method": "POST",
      "bulkId": "ytrewq",
      "version": "W\/\"lha5bbazU3fNvfe5\"",
      "status": {
        "code": "201"
      }
    }
  ]
}
```

A subsequent request for the 'Tour Guides' Group ('e9e30dba-f08f-4109-8486-d5c6a331660a') returns the following:

```
GET /v2/Groups/e9e30dba-f08f-4109-8486-d5c6a331660a
Host: example.com
Accept: application/json
Authorization: Bearer h480djs93hd8
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Location: https://example.com/v2/Groups/e9e30dba-f08f-4109-8486-d5c6a331660a
ETag: W/"lha5bbazU3fNvfe5"

{
  "schemas": ["urn:scim:schemas:core:2.0:Group"],
  "id": "e9e30dba-f08f-4109-8486-d5c6a331660a",
  "displayName": "Tour Guides",
  "meta": {
    "resourceType": "Group",
    "created": "2011-08-01T18:29:49.793Z",
    "lastModified": "2011-08-01T20:31:02.315Z",
    "location": "https://example.com/v2/Groups/e9e30dba-f08f-4109-8486-d5c6a3316
60a",
    "version": "W\/\"lha5bbazU3fNvfe5\""
  },
  "members": [
    {
      "value": "92b725cd-9465-4e7d-8c16-01f8e146b87a",
      "$ref": "https://example.com/v2/Users/92b725cd-9465-4e7d-8c16-01f8e146b87a
",
      "type": "User"
    }
  ]
}
```

Extensions that include references to other resources MUST be handled in the same way by the service provider. The following example uses the bulkId attribute within the enterprise extension managerId attribute.

```
POST /v2/Bulk
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Bearer h480djs93hd8
Content-Length: ...

{
  "schemas": ["urn:scim:schemas:core:2.0:BulkRequest"],
  "Operations": [
    {
      "method": "POST",
      "path": "/Users",
      "bulkId": "qwerty",
      "data": {
        "schemas": ["urn:scim:schemas:core:2.0:User"],
        "userName": "Alice"
      }
    },
    {
      "method": "POST",
      "path": "/Users",
      "bulkId": "ytrewq",
      "data": {
        "schemas": [
          "urn:scim:schemas:core:2.0:User",
          "urn:scim:schemas:extension:enterprise:2.0:User"
        ],
        "userName": "Bob",
        "urn:scim:schemas:extension:enterprise:2.0:User": {
          "employeeNumber": "11250",
          "manager": {
            "managerId": "batchId:qwerty",
            "displayName": "Alice"
          }
        }
      }
    }
  ]
}
```

The service provider MUST try to resolve circular cross references between resources in a single bulk job but MAY stop after a failed attempt and instead return the status code 409 Conflict. The following example exhibits the potential conflict.

```
POST /v2/Bulk
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Bearer h480djs93hd8
Content-Length: ...

{
  "schemas": ["urn:scim:schemas:core:2.0:BulkRequest"],
  "Operations": [
    {
      "method": "POST",
      "path": "/Groups",
      "bulkId": "qwerty",
      "data": {
        "schemas": ["urn:scim:schemas:core:2.0:Group"],
        "displayName": "Group A",
        "members": [
          {
            "type": "group",
            "value": "bulkId:ytrewq"
          }
        ]
      }
    },
    {
      "method": "POST",
      "path": "/Groups",
      "bulkId": "ytrewq",
      "data": {
        "schemas": ["urn:scim:schemas:core:2.0:Group"],
        "displayName": "Group B",
        "members": [
          {
            "type": "group",
            "value": "bulkId:qwerty"
          }
        ]
      }
    }
  ]
}
```

If the service provider resolved the above circular references the following is returned from a subsequent GET request.

```
GET /v2/Groups?filter=displayName sw 'Group'
Host: example.com
Accept: application/json
Authorization: Bearer h480djs93hd8
```

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

{
  "schemas": ["urn:scim:schemas:core:2.0:ListResponse"],
  "totalResults": 2,
  "Resources": [
    {
      "id": "c3a26dd3-27a0-4dec-a2ac-ce211e105f97",
      "schemas": ["urn:scim:schemas:core:2.0:Group"],
      "displayName": "Group A",
      "meta": {
        "resourceType": "Group",
        "created": "2011-08-01T18:29:49.793Z",
        "lastModified": "2011-08-01T18:29:51.135Z",
        "location": "https://example.com/v2/Groups/c3a26dd3-27a0-4dec-a2ac-ce211
e105f97",
        "version": "W\/\"mvwNGaxB5SDq074p\""
      },
      "members": [
        {
          "value": "6c5bb468-14b2-4183-baf2-06d523e03bd3",
          "$ref": "https://example.com/v2/Groups/6c5bb468-14b2-4183-baf2-06d523e
03bd3",
          "type": "Group"
        }
      ]
    },
    {
      "id": "6c5bb468-14b2-4183-baf2-06d523e03bd3",
      "schemas": ["urn:scim:schemas:core:2.0:Group"],
      "displayName": "Group B",
      "meta": {
        "resourceType": "Group",
        "created": "2011-08-01T18:29:50.873Z",
        "lastModified": "2011-08-01T18:29:50.873Z",
        "location": "https://example.com/v2/Groups/6c5bb468-14b2-4183-baf2-06d52
3e03bd3",
        "version": "W\/\"wGB85s2QJMjiNnuI\""
      },
      "members": [
        {
          "value": "c3a26dd3-27a0-4dec-a2ac-ce211e105f97",
          "$ref": "https://example.com/v2/Groups/c3a26dd3-27a0-4dec-a2ac-ce211e1
05f97",
          "type": "Group"
        }
      ]
    }
  ]
}
```

The service provider MUST define the maximum number of operations and maximum payload size a client may send in a single request. If either limits are exceeded the service provider MUST return the HTTP response code 413 Request Entity Too Large. The returned response MUST specify the limit exceeded in the body of the error response.

The following example the client sent a request exceeding the service provider's max payload size of 1 megabyte.

```
POST /v2/Bulk
Host: example.com
Accept: application/json
Content-Type: application/json
Authorization: Bearer h480djs93hd8
Content-Length: 4294967296

...
```

```
HTTP/1.1 413 Request Entity Too Large
Content-Type: application/json
Location: https://example.com/v2/Bulk/yfCrVJhFIJagAHj8

{
  "schemas":["urn:scim:schemas:core:2.0:Error"],
  "Errors":[
    {
      "description":"The size of the bulk operation exceeds the maxPayloadSize (
1048576).",
      "code":"413"
    }
  ]
}
```

3.6. Data Input/Output Formats

Clients MUST specify the format in which the data is submitted via the Section 14.17 HTTP header content-type [RFC2616] and MAY specify the desired response data format via an HTTP Accept Header; e.g.,"Accept: application/json" or via URI suffix; e.g.,

```
GET /Users/2819c223-7f76-453a-919d-413861904646.json
Host: example.com
```

Service providers MUST support the Accept Headers "Accept: application/json" for [RFC4627]. The format defaults to JSON if no format is specified.

Singular attributes are encoded as string name-value-pairs in JSON; e.g.,

"attribute": "value"

Multi-valued attributes in JSON are encoded as arrays; e.g.,

"attributes": ["value1", "value2"]

Elements with nested elements are represented as objects in JSON; e.g,

"attribute": { "subattribute1": "value1", "subattribute2": "value2" }

3.7. Additional retrieval query parameters

Clients MAY request a partial resource representation on any operation that returns a resource within the response by specifying the URL query parameter 'attributes'. When specified, each resource returned MUST contain the minimal set of resource attributes and MUST contain no other attributes or Sub-Attributes than those explicitly requested. The query parameter attributes value is a comma separated list of resource attribute names in standard attribute notation (Section 3.8) form (e.g. userName, name, emails).

```
GET /Users/2819c223-7f76-453a-919d-413861904646?attributes=userName
Host: example.com
Accept: application/json
Authorization: Bearer h480djs93hd8
```

Giving the response

```
HTTP/1.1 200 OK
Content-Type: application/json
Location: https://example.com/v2/Users/2819c223-7f76-453a-919d-413861904646
ETag: W/"a330bc54f0671c9"

{
  "schemas":["urn:scim:schemas:core:2.0:User"],
  "id":"2819c223-7f76-453a-919d-413861904646",
  "userName":"bjensen",
  "meta":{
    "resourceType": "User",
    "created":"2011-08-01T18:29:49.793Z",
    "lastModified":"2011-08-01T18:29:49.793Z",
    "location":"https://example.com/v2/Users/2819c223-7f76-453a-919d-41386190464
6",
    "version":"W\/\"a330bc54f0671c9\""
  }
}
```

3.8. Attribute Notation

All operations share a common scheme for referencing simple and complex attributes. In general, attributes are identified by prefixing the attribute name with its schema URN separated by a ':' character; e.g., the core User resource attribute 'userName' is identified as 'urn:scim:schemas:core:2.0:User:userName'. Clients MAY omit core schema attribute URN prefixes though MUST fully qualify extended attributes with the associated resource URN; e.g., the attribute 'age' defined in 'urn:hr:schemas:user' is fully encoded as 'urn:hr:schemas:user:age'. A Complex attributes' Sub-Attributes are referenced via nested, dot ('.') notation; i.e., {urn}:{Attribute name}.{Sub-Attribute name}. For example, the fully qualified path for a User's givenName is urn:scim:schemas:core:2.0:User:name.givenName All facets (URN, attribute and Sub-Attribute name) of the fully encoded Attribute name are case insensitive.

3.9. HTTP Response Codes

The SCIM Protocol uses the response status codes defined in HTTP Section 10 [RFC2616] to indicate operation success or failure. In addition to returning a HTTP response code implementers MUST return the errors in the body of the response in the client requested format containing the error response and, per the HTTP specification, human-readable explanations. Error responses are identified using the following URI: 'urn:scim:schemas:core:2.0:Error'. The following multi-valued attribute is defined in addition to those attributes defined in SCIM Core Schema:

Errors The list of errors encountered by the service provider. The value attribute is a complex type with the following sub-attributes.

description A human-readable explanation of the error. REQUIRED.

code A string indicating the HTTP response code. REQUIRED.

Implementers SHOULD handle the identified errors as described below.

Code	Applicability	Suggested Explanation
307 TEMPORARY REDIRECT	GET, POST, PUT, PATCH, DELETE	The client is directed to repeat the same HTTP request at the location identified. The client SHOULD NOT use the location provided in the response as a permanent reference to the resource and SHOULD continue to use the original request URI [I-D.ietf-httpbis-p2-semantics].
308 PERMANENT REDIRECT	GET, POST, PUT, PATCH, DELETE	The client is directed to repeat the same HTTP request at the location identified. The client SHOULD use the location provided in the response as the permanent reference to the resource [I-D.reschke-http-status-308].
400 BAD REQUEST	GET, POST, PUT, PATCH, DELETE	Request is unparseable, syntactically incorrect, or violates schema
401 UNAUTHORIZED	GET, POST, PUT, PATCH, DELETE	Authorization failure
403 FORBIDDEN	GET, POST, PUT, PATCH, DELETE	Server does not support requested operation
404 NOT FOUND	GET, PUT, PATCH, DELETE	Specified resource; e.g., User, does not exist
409 CONFLICT	POST, PUT, PATCH, DELETE	The specified version number does not match the resource’s latest version number or a service provider refused to create a new, duplicate resource
412 PRECONDITION FAILED	PUT, PATCH,D ELETE	Failed to update as resource {id} changed on the server last retrieved

413 REQUEST ENTITY TOO LARGE	POST	{"maxOperations": 1000,"maxPayload": 1048576}
500 INTERNAL SERVER ERROR	GET, POST, PUT, PATCH, DELETE	An internal error. Implementers SHOULD provide descriptive debugging advice
501 NOT IMPLEMENTED	GET, POST, PUT, PATCH, DELETE	Service Provider does not support the request operation; e.g., PATCH

Table 7: Defined error cases

Error example in response to a non-existent GET request.

```
HTTP/1.1 404 NOT FOUND

{
  "schemas": ["urn:scim:schemas:core:2.0:Error"],
  "Errors":[
    {
      "description":"Resource 2819c223-7f76-453a-919d-413861904646 not found",
      "code":"404"
    }
  ]
}
```

3.10. API Versioning

The Base URL MAY be appended with a version identifier as a separate segment in the URL path. At this time the only valid identifier is 'v1'. If specified, the version identifier MUST appear in the URL path immediately preceding the resource endpoint and conform to the following scheme: the character 'v' followed by the desired SCIM version number; e.g., a version 'v1' User request is specified as /v2 /Users. When specified service providers MUST perform the operation using the desired version or reject the request. When omitted service providers SHOULD perform the operation using the most recent API supported by the service provider.

3.11. Versioning Resources

The API supports resource versioning via standard HTTP ETagsSection 14.19 [RFC2616]. Service providers MAY support weak ETags as the preferred mechanism for performing conditional retrievals and ensuring clients do not inadvertently overwrite each others changes, respectively. When supported SCIM ETags MUST be

specified as an HTTP header and SHOULD be specified within the 'version' attribute contained in the resource's 'meta' attribute.

Example:

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

{
  "schemas":["urn:scim:schemas:core:2.0:User"],
  "userName":"bjensen",
  "externalId":"bjensen",
  "name":{
    "formatted":"Ms. Barbara J Jensen III",
    "familyName":"Jensen",
    "givenName":"Barbara"
  }
}
```

The server responds with an ETag in the response header and meta structure.

```
HTTP/1.1 201 Created
Content-Type: application/json
Location: https://example.com/v2/Users/2819c223-7f76-453a-919d-413861904646
ETag: W/"e180ee84f0671b1"

{
  "schemas":["urn:scim:schemas:core:2.0:User"],
  "id":"2819c223-7f76-453a-919d-413861904646",
  "meta":{
    "resourceType":"User",
    "created":"2011-08-01T21:32:44.882Z",
    "lastModified":"2011-08-01T21:32:44.882Z",
    "location":"https://example.com/v2/Users/2819c223-7f76-453a-919d-41386190464
6",
    "version":"W\/\"e180ee84f0671b1\""
  },
  "name":{
    "formatted":"Ms. Barbara J Jensen III",
    "familyName":"Jensen",
    "givenName":"Barbara"
  },
  "userName":"bjensen"
}
```

With the returned ETag, clients MAY choose to retrieve the resource only if the resource has been modified.

Conditional retrieval example using If-None-Match Section 14.26 [RFC2616] header:

```
GET /Users/2819c223-7f76-453a-919d-413861904646?attributes=displayName
Host: example.com
Accept: application/json
Authorization: Bearer h480djs93hd8
If-None-Match: W/"e180ee84f0671b1"
```

If the resource has not changed the service provider simply returns an empty body with a 304 "Not Modified" response code.

If the service providers supports versioning of resources the client MUST supply an If-Match Section 14.24 [RFC2616] header for PUT and PATCH operations to ensure that the requested operation succeeds only if the supplied ETag matches the latest service provider resource; e.g., If-Match: W/"e180ee84f0671b1"

3.12. HTTP Method Overloading

In recognition that some clients, servers and firewalls prevent PUT, PATCH and DELETE operations a client MAY override the POST operation by specifying the custom header "X-HTTP-Method-Override" with the desired PUT, PATCH, DELETE operation. For example:

```
POST /Users/2819c223-7f76-453a-919d-413861904646
X-HTTP-Method-Override: DELETE
```

4. Multi-Tenancy

A single service provider may expose the SCIM protocol to multiple clients. Depending on the nature of the service, the clients may have authority to access and alter resources initially created by other clients. Alternatively, clients may expect to access disjoint sets of resources, and may expect that their resources are inaccessible by other clients. These scenarios are called "multi-tenancy", where each client is understood to be or represent a "tenant" of the service provider. Clients may also be multi-tenanted.

The following common cases may occur:

1. All clients share all resources (no tenancy)

2. Each single client creates and accesses a private subset of resources (1 client:1 Tenant)

3. Sets of clients share sets of resources (M clients:1 Tenant)

4. One client to Multiple Tenants (1 client:M Tenants)

Service providers may implement any subset of the above cases.

Multi-Tenancy is OPTIONAL. The SCIM protocol does not define a scheme for multi-tenancy.

The SCIM protocol does not prescribe the mechanisms whereby clients and service providers interact for:

- Registering or provisioning Tenants
- Associating a subset of clients with a subset of the Tenants
- Indicating which tenant is associated with the data in a request or response, or indicating which Tenant is the subject of a query
- Implementers are encouraged to use mechanisms which comply with RESTful conventions.

4.1. Associating Clients to Tenants

The service provider MAY use the authentication mechanism (Section 2) to determine the identity of the client, and thus infer the associated Tenant.

For implementations where a client is associated with more than one Tenant, the service provider MAY use one of the following methods for explicit specification of the Tenant.

If any of these methods of allowing the client to explicitly specify the Tenant are employed, the service provider should ensure that access controls are in place to prevent or allow cross-tenant use cases.

The service provider should consider precedence in cases where a client may explicitly specify a Tenant while being implicitly associated with a different Tenant.

4.1.1. URL Prefix Example

https://www.example.com/Tenants/{tenant_id}/v2/Users

4.1.2. Subdomain Example

https://{tenant_id}.example.com/v2/Groups

4.1.3. HTTP Header

The service provider may recognize a {tenant_id} provided by the client in the HTTP Header "SCIM_TENANT_ID" as the indicator of the desired target Tenant.

In all of these methods, the {tenant_id} is a unique identifier for the Tenant as defined by the service provider.

4.2. SCIM Identifiers with Multiple Tenants

Considerations for a Multi-Tenant Implementation:

The service provider may choose to implement SCIM ids which are unique across all resources for all Tenants, but this is not required.

The externalId, defined by the client, is required to be unique ONLY within the resources associated with the associated Tenant.

5. Security Considerations

The SCIM Protocol is based on HTTP and thus subject to the security considerations found in Section 15 of [RFC2616]. SCIM resources (e.g., Users and Groups) can contain sensitive information. Therefore, SCIM clients and service providers MUST implement TLS. Which version(s) ought to be implemented will vary over time, and depend on the widespread deployment and known security vulnerabilities at the time of implementation. At the time of this writing, TLS version 1.2 [RFC5246]] is the most recent version, but has very limited actual deployment, and might not be readily available in implementation toolkits. TLS version 1.0 [[RFC2246]] is the most widely deployed version, and will give the broadest interoperability.

6. References

6.1. Normative References

[I-D.ietf-httpbis-p2-semantics]
Fielding, R. and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", draft-ietf-httpbis-p2-semantics-25 (work in progress), November 2013.

[I-D.reschke-http-status-308]
Reschke, J., "The Hypertext Transfer Protocol (HTTP) Status Code 308 (Permanent Redirect)", draft-reschke-http-status-308-07 (work in progress), March 2012.

[IANA.Language]
Internet Assigned Numbers Authority (IANA), "Language Subtag Registry", 2005.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, January 1999.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, June 1999.

[RFC3896] Nicklass, O., "Definitions of Managed Objects for the DS3/E3 Interface Type", RFC 3896, September 2004.

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC5789] Dusseault, L. and J. Snell, "PATCH Method for HTTP", RFC 5789, March 2010.

[RFC6750] Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, October 2012.

6.2. Informative References

[OpenSearch]
Clinton, D., "OpenSearch Protocol 1.1, Draft 5", .

[Order-Operations]
Wikipedia, "Order of Operations: Programming Languages", .

Appendix A. Contributors

Samuel Erdtman (samuel@erdtman.se)

Patrick Harding (pharding@pingidentity.com)

Appendix B. Acknowledgments

The editors would like to acknowledge the contribution and work of the past draft editors:

Trey Drake, UnboundID

Chuck Mortimore, Salesforce

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

Appendix C. Change Log

[[This section to be removed prior to publication as an RFC]]

Draft 02 - KG - Addition of schema extensibility

Draft 03 - PH - Revisions based on following tickets:

24 - Add filter negation

39 - Clarification on response for DELETE

42 - Make root searches optional

49 - Add "ew" filter

50 - Filters for multi-valued complex attributes

51 - Search by Schema

53 - Standard use of term client (some was consumer)

55 - Redirect support (3xx)

56 - Make manager attribute consistent with other $ref attrs

57 - Update all "/v1" examples to ’/v2"

59 - Fix capitalization per IETF editor practices

60 - Changed <eref> tags to normal <xref> and <reference> tags

Authors' Addresses

Kelly Grizzle
SailPoint

Email: kelly.grizzle@sailpoint.com

Phil Hunt (editor)
Oracle Corporation

Email: phil.hunt@yahoo.com

Morteza Ansari
Cisco

Email: morteza.ansari@cisco.com

Erik Wahlstroem
Technology Nexus

Email: erik.wahlstrom@nexussafe.com

Chuck Mortimore
Salesforce.com

Email: cmortimore@salesforce.com

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 16, 2014

K. Grizzle
SailPoint
P. Hunt, Ed.
Oracle
E. Wahlstroem
Technology Nexus
C. Mortimore
Salesforce

February 12, 2014

System for Cross-Domain Identity Management: Core Schema
draft-ietf-scim-core-schema-03

Abstract

The System for Cross-Domain Identity Management (SCIM) specification is designed to make managing user identity in cloud based applications and services easier. The specification suite builds upon experience with existing schemas and deployments, placing specific emphasis on simplicity of development and integration, while applying existing authentication, authorization, and privacy models. Its intent is to reduce the cost and complexity of user management operations by providing a common user schema and extension model, as well as binding documents to provide patterns for exchanging this schema using standard protocols. In essence, make it fast, cheap, and easy to move identity in to, out of, and around the cloud.

This document provides a platform neutral schema and extension model for representing users and groups in JSON format. This schema is intended for exchange and use with cloud service providers. Additional binding documents provide a standard REST API, SAML binding, and use cases.

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 August 16, 2014.

Table of Contents

1. Requirements Notation and 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] .

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. Overview

While there are existing standards for describing and exchanging user information, many of these standards can be difficult to implement and/or use; e.g., their wire protocols do not easily traverse firewalls and/or are not easily layered onto existing web protocols. As a result, many cloud providers implement non-standard APIs for managing users within their services. This increases both the cost and complexity associated with organizations adopting products and services from multiple cloud providers as they must perform redundant integration development. Similarly, cloud services providers seeking to interoperate with multiple application marketplaces or cloud identity providers must be redundantly integrated.

SCIM seeks to simplify this problem through a simple to implement specification suite that provides a common user schema and extension model, as well as binding documents to provide patterns for exchanging this schema via a REST API. It draws inspiration and best practice, building upon existing user APIs and schemas from a wide variety of sources including, but not limited to, existing APIs exposed by cloud providers, PortableContacts, and LDAP directory services.

This document provides a platform neutral schema and extension model for representing users and groups in JSON format. This schema is

intended for exchange and use with cloud service providers. Additional binding documents provide a standard REST API, SAML binding, and use cases.

2.1. Definitions

Service Provider: An HTTP web application that provides identity information via the SCIM protocol.

Client: A website or application that uses the SCIM protocol to manage identity data maintained by the service provider. The client initiates SCIM HTTP requests to a target service provider.

Resource: The service provider managed artifact containing one or more attributes; e.g., "User" or "Group".

Resource Type: A type of a resource that is managed by a service provider. The resource type defines the resource name, endpoint URL, Schemas, and other meta-data which indicate where a resource is managed and how it is composed; e.g., "User" or "Group".

Schema: A collection of Attribute Definitions that describe the contents of an entire or partial resource; e.g., "urn:scim:schemas:core:2.0:User".

Singular Attribute: A resource attribute that contains 0..1 values; e.g., "displayName".

Multi-valued Attribute: A resource attribute that contains 0..n values; e.g., "emails".

Simple Attribute: A singular or multi-valued attribute whose value is a primitive; e.g., "String".

Complex Attribute: A singular or multi-valued attribute whose value is a composition of one or more simple attributes; e.g., "addresses".

Sub-Attribute: A simple attribute contained within a complex attribute.

3. SCIM Schema Structure

SCIM schema provides a minimal core schema for representing users and groups (resources), encompassing common attributes found in many existing deployments and schemas.

A resource is a collection of attributes identified by one or more schemas. Minimally, an attribute consists of the attribute name and at least one simple or complex value either of which may be multi-valued. SCIM schema defines the data type, plurality and other distinguishing features of an attribute. Unless otherwise specified all attributes are modifiable by consumers. Attribute definitions contain a property "mutability" that determines whether an attribute is: "readOnly", "writeOnly", "immutable", or "readWrite"(the default). Additionally, service providers MAY define other properties such as returnability Resource's schema endpoint (Section 5.2).

A JSON (JavaScript Object Notation) format [RFC4627] is defined. Attribute names SHOULD be camelCased. SCIM resources represented in JSON MUST specify schema via the "schemas" attributeSection 5.2.

3.1. Attribute Data Types

Attribute data types are derived from JSON [RFC4627] and unless otherwise specified have the following characteristics (see Section 11 for attribute characteristic definitions):

- are optional ("required=false").
- are case insensitive ("caseExact=false"),
- are modifiable ("mutability=readWrite"),
- are returned in response to queries ("returned=default"),
- are not unique ("uniqueness=none"), and,
- of type String (Section 3.1.1).

The JSON format defines a limited set of data types, hence, where appropriate, alternate JSON representations derived from XML Schema [XML-Schema] are defined below. SCIM extensions SHOULD NOT introduce new data types.

3.1.1. String

A sequence of zero or more Unicode characters. The JSON format is defined in Section 2.5 [RFC4627] . A "String" attribute MAY specify a required data format. Additionally, when canonical values are specified service providers SHOULD conform to those values if appropriate, but MAY provide alternate "String" values to represent additional values.

3.1.2. Boolean

The literal "true" or "false". The JSON format is defined in Section 2.1 [RFC4627] .

3.1.3. Decimal

A real number with at least one digit to the left and right of the period. The JSON format is defined in Section 2.4 [RFC4627] .

3.1.4. Integer

A decimal number with no fractional digits. The JSON format is defined in Section 2.4 [RFC4627] with the additional constraint that the value MUST NOT contain fractionial or exponent parts.

3.1.5. DateTime

A DateTime value (e.g. 2008-01-23T04:56:22Z). The attribute value MUST be encoded as a valid xsd:dateTime as specified in Section 3.2.7 [XML-Schema] .

Values represented in JSON MUST conform to the XML constraints above and are represented as a JSON String per Section 2.5 [RFC4627].

3.1.6. Binary

Arbitrary binary data. The attribute value MUST be encoded as a valid xsd:base64Binary as specified in Section 3.2.16 [XML-Schema] .

Values represented in JSON MUST conform to the XML constraints above and are represented as a JSON String perSection 2.5 [RFC4627].

3.1.7. Reference

A reference to a SCIM resource. The value MUST be the absolute or relative URI of the target resource. Relative URIs should be resolved as specified in Section 5.2 [RFC3986]. The base URI for relative URI resolution MUST include all URI components and path segments up to but not including the Endpoint URI; e.g., the base URI for a request to "https://example.com/v2/Users/2819c223-7f76-453a-919d-413861904646" would be "https://example.com/v2/" and the relative URI for this resource would be "Users/2819c223-7f76-453a-919d-413861904646".

Performing a GET operation on a reference URI MUST return the target resource or an appropriate HTTP response code. The service provider

MAY optionally choose to enforce referential integrity for references.

By convention, a reference is commonly represented as a "$ref" sub-attribute in complex or multi-valued attributes, however this is OPTIONAL.

3.1.8. Complex

A singular or multi-valued attribute whose value is a composition of one or more simple Attributes. The JSON format is defined in Section 2.2 [RFC4627] .

3.2. Multi-valued Attributes

Multi-valued attributes are unordered lists of attributes. Each attribute MAY contain Sub-Attributes and therefore multi-valued attributes may contain complex attributes. The sub-attributes below are considered normative and when specified SHOULD be used as defined.

type A label indicating the attribute’s function; e.g., "work" or "home".

primary A boolean value indicating the ’primary’ or preferred attribute value for this attribute, e.g. the preferred mailing address or primary e-mail address. The primary attribute value "true" MUST appear no more than once.

display A human readable name, primarily used for display purposes and has a mutability of "immutable".

operation The operation to perform on the multi-valued attribute during a PATCH request. The only valid value is "delete", which signifies that this instance should be removed from the resource.

value The attribute’s significant value; e.g., the e-mail address, phone number, etc. Attributes that define a "value" sub-attribute MAY be alternately represented as a collection of primitive types. For example:

```
{
  "emails": [
    {"value":"bjensen@example.com"},
    {"value":"babs@example.com"}
  ]
}
```

May also be represented as:

```
{
  "emails": ["bjensen@example.com","babs@example.com"]
}
```

$ref The reference URI of the target resource, if the attribute is a reference.

When returning multi-valued attributes, service providers SHOULD canonicalize the value returned, if appropriate (e.g. for e-mail addresses and URLs). Service providers MAY return the same value more than once with different types (e.g. the same e-mail address may used for work and home), but SHOULD NOT return the same (type, value) combination more than once per Attribute, as this complicates processing by the Consumer.

4. Schema Extension Model

SCIM schema follows an object extension model similar to ObjectClasses used in LDAP. Unlike LDAP there is no inheritance model; all extensions are additive (similar to LDAP Auxiliary Object Class [RFC4512]). Each value indicates additive schema that may exist in a SCIM representation as specified by extensions not defined in this suite. Schema extensions MUST NOT redefine any attributes defined in this specification and SHOULD follow conventions defined in this specification. Each schema extension must identify a URI used to identify the extension. The JSON format MUST use the "schemas" attributeSection 5.2 to distinguish extended resources and attributes.

5. SCIM Core Schema

5.1. Common Schema Attributes

Each SCIM resource (Users, Groups, etc.) includes the below common attributes. These attributes MUST be included in all resources, including any extended resource types. It is not necessary to specify the schemas attribute if the resource is fully defined in this document as the core schema is implicitly included.

id Unique identifier for the SCIM resource as defined by the service provider. Each representation of the resource MUST include a non-empty "id" value. This identifier MUST be unique across the service provider’s entire set of resources. It MUST be a stable, non-reassignable identifier that does not change when the same resource is returned in subsequent requests. The value of the "id" attribute is always issued by the service provider and MUST

never be specified by the client. bulkId: is a reserved keyword and MUST NOT be used in the unique identifier. REQUIRED and has a mutability of "readOnly".

externalId An identifier for the resource as defined by the client. The "externalId" may simplify identification of the resource between client and service provider by allowing the client to refer to the resource with its own identifier, obviating the need to store a local mapping between the local identifier of the resource and the identifier used by the service provider. Each resource MAY include a non-empty externalId value. The value of the "externalId" attribute is always issued by the client and can never be specified by the service provider. The service provider MUST always interpret the externalId as scoped to the client's tenant.

meta A complex attribute containing resource metadata. All sub-attributes are OPTIONAL

resourceType The name of the resource type of the resource. Attribute has mutability of "readOnly".

created The DateTime the resource was added to the service provider. The attribute MUST be a DateTime. Attribute has mutability of "readOnly".

lastModified The most recent DateTime the details of this resource were updated at the service provider. If this resource has never been modified since its initial creation, the value MUST be the same as the value of created. The attribute MUST be a DateTime. Attribute has mutability of "readOnly".

location The URI of the resource being returned. This value MUST be the same as the Location HTTP response header. Attribute has mutability of "readOnly".

version The version of the resource being returned. This value must be the same as the ETag HTTP response header. Attribute has mutability of "readOnly".

attributes The names of the attributes to remove from the resource during a PATCH operation.

5.2. "schemas" Attribute

SCIM supports resources of different types, with extensible schemas. Each resource MUST be indicated using fully qualified URLs.

A "schemas" attribute is used to indicate the version of SCIM schema as well as any schema extensions.

schemas The schemas attribute is an array of strings which allows introspection of the supported schema version for a SCIM representation as well any schema extensions supported by that representation. Each String value must be a unique URI. This specification defines URIs for User, Group, and a standard enterprise-user extension. All representations of SCIM schema MUST include a non-zero value array with value(s) of the URIs supported by that representation. The schemas attribute for a resource MUST only contain values defined as "schema" and "schemaExtensions" for the resource's resource type. Duplicate values MUST NOT be included. Value order is not specified and MUST not impact behavior. REQUIRED.

6. SCIM User Schema

SCIM provides a schema for representing Users, identified using the following URI: "urn:scim:schemas:core:2.0:User". The following attributes are defined in addition to those attributes defined in SCIM Core Schema:

6.1. Singular Attributes

userName Unique identifier for the user, typically used by the user to directly authenticate to the service provider. Often displayed to the user as their unique identifier within the system (as opposed to id or externalId, which are generally opaque and not user-friendly identifiers). Each User MUST include a non-empty userName value. This identifier MUST be unique across the client's entire set of Users. RECOMMENDED.

name The components of the user's real name. Service providers MAY return just the full name as a single string in the formatted sub-attribute, or they MAY return just the individual component attributes using the other sub-attributes, or they MAY return both. If both variants are returned, they SHOULD be describing the same name, with the formatted name indicating how the component attributes should be combined.

formatted The full name, including all middle names, titles, and suffixes as appropriate, formatted for display (e.g. "Ms. Barbara Jane Jensen, III.").

familyName The family name of the User, or last name in most Western languages (e.g. "Jensen" given the full name "Ms. Barbara Jane Jensen, III.").

givenName The given name of the User, or first name in most Western languages (e.g. "Barbara" given the full name "Ms. Barbara Jane Jensen, III.").

middleName The middle name(s) of the User (e.g. "Jane" given the full name "Ms. Barbara Jane Jensen, III.").

honorificPrefix The honorific prefix(es) of the User, or title in most Western languages (e.g. "Ms." given the full name "Ms. Barbara Jane Jensen, III.").

honorificSuffix The honorific suffix(es) of the User, or suffix in most Western languages (e.g. "III." given the full name "Ms. Barbara Jane Jensen, III.").

displayName The name of the user, suitable for display to end-users. Each user returned MAY include a non-empty displayName value. The name SHOULD be the full name of the User being described if known (e.g. "Babs Jensen" or "Ms. Barbara J Jensen, III"), but MAY be a username or handle, if that is all that is available (e.g. "bjensen"). The value provided SHOULD be the primary textual label by which this User is normally displayed by the service provider when presenting it to end-users.

nickName The casual way to address the user in real life, e.g. "Bob" or "Bobby" instead of "Robert". This attribute SHOULD NOT be used to represent a User's username (e.g. bjensen or mpepperidge).

profileUrl A fully qualified URL to a page representing the user's online profile.

title The user's title, such as "Vice President".

userType Used to identify the organization to user relationship. Typical values used might be "Contractor", "Employee", "Intern", "Temp", "External", and "Unknown" but any value may be used.

preferredLanguage Indicates the user's preferred written or spoken language. Generally used for selecting a localized User interface. Valid values are concatenation of the ISO 639-1 two

letter language code per [ISO639-2], an underscore, and the ISO 3166-1 2 letter country code[ISO3166]; e.g., 'en_US' specifies the language English and country US.

locale Used to indicate the User's default location for purposes of localizing items such as currency, date time format, numerical representations, etc. A locale value is a concatenation of the ISO 639-1 two letter language code[ISO639-2], an underscore, and the ISO 3166-1 2 letter country code[ISO3166]; e.g., 'en_US' specifies the language English and country US.

timezone The User's time zone in the "Olson" timezone database format[Olson-TZ]; e.g.,'America/Los_Angeles'.

active A Boolean value indicating the user's administrative status. The definitive meaning of this attribute is determined by the service provider. As a typical example, a value of true infers the user is able to login while a value of false implies the user's account has been suspended.

password The user's clear text password. This attribute is intended to be used as a means to specify an initial password when creating a new User or to reset an existing User's password. Password policies and the ability to update or set passwords are out of scope of this document. The mutability of this attribute is "writeOnly" indicating the value MUST NOT be returned by a service provider in any form.

6.2. Multi-valued Attributes

The following multi-valued attributes are defined.

emails E-mail addresses for the User. The value SHOULD be canonicalized by the service provider, e.g. "bjensen@example.com" instead of "bjensen@EXAMPLE.COM". Canonical type values of "work", "home", and "other".

phoneNumbers Phone numbers for the user. The value SHOULD be canonicalized by the service provider according to format in [RFC3966] e.g. 'tel:+1-201-555-0123'. Canonical type values of "work", "home", "mobile", "fax", "pager", and "other".

ims Instant messaging address for the user. No official canonicalization rules exist for all instant messaging addresses, but service providers SHOULD, when appropriate, remove all whitespace and convert the address to lowercase. Instead of the standard canonical values for type, this attribute defines the following canonical values to represent currently popular IM

services: "aim", "gtalk", "icq", "xmpp", "msn", "skype", "qq", "yahoo", and "other".

photos URL of a photo of the User. The value SHOULD be a canonicalized URL, and MUST point to an image file (e.g. a GIF, JPEG, or PNG image file) rather than to a web page containing an image. Service providers MAY return the same image at different sizes, though it is recognized that no standard for describing images of various sizes currently exists. Note that this attribute SHOULD NOT be used to send down arbitrary photos taken by this user, but specifically profile photos of the user suitable for display when describing the user. Instead of the standard canonical values for type, this attribute defines the following canonical values to represent popular photo sizes: "photo", "thumbnail".

addresses A physical mailing address for this user. Canonical type values of "work", "home", and "other". The value attribute is a complex type with the following sub-attributes. All sub-attributes are OPTIONAL.

formatted The full mailing address, formatted for display or use with a mailing label. This attribute MAY contain newlines.

streetAddress The full street address component, which may include house number, street name, P.O. box, and multi-line extended street address information. This attribute MAY contain newlines.

locality The city or locality component.

region The state or region component.

postalCode The zipcode or postal code component.

country The country name component. When specified the value MUST be in ISO 3166-1 alpha 2 "short" code format[ISO3166]; e.g., the United States and Sweden are "US" and "SE", respectively.

groups A list of groups that the user belongs to, either thorough direct membership, nested groups, or dynamically calculated. The values are meant to enable expression of common group or role based access control models, although no explicit authorization model is defined. It is intended that the semantics of group membership and any behavior or authorization granted as a result of membership are defined by the service provider. The canonical types "direct" and "indirect" are defined to describe how the

group membership was derived. Direct group membership indicates the user is directly associated with the group and SHOULD indicate that clients may modify membership through the "Group" resource. Indirect membership indicates user membership is transitive or dynamic and implies that clients cannot modify indirect group membership through the "Group" resource but MAY modify direct group membership through the "Group" resource which MAY influence indirect memberships. If the SCIM service provider exposes a Group resource, the "value" sub-attribute MUST be the "id" and the "$ref" sub-attribute must be the URI of the corresponding "Group" resources to which the user belongs. Since this attribute has a mutability of "readOnly", group membership changes MUST be applied via the Group Resource (Section 8). The attribute has a mutability of "readOnly".

entitlements A list of entitlements for the user that represent a thing the user has. An entitlement MAY be an additional right to a thing, object, or service. No vocabulary or syntax is specified and service providers and clients are expected to encode sufficient information in the value so as to accurately and without ambiguity determine what the user has access to. This value has NO canonical types though type may be useful as a means to scope entitlements.

roles A list of roles for the user that collectively represent who the user is; e.g., "Student, Faculty". No vocabulary or syntax is specified though it is expected that a role value is a String or label representing a collection of entitlements. This value has NO canonical types.

x509Certificates A list of certificates issued to the User. Values are Binary (Section 3.1.6) and DER encoded x509. This value has NO canonical types.

7. SCIM Enterprise User Schema Extension

The following SCIM extension defines attributes commonly used in representing users that belong to, or act on behalf of a business or enterprise. The enterprise user extension is identified using the following schema URI:
"urn:scim:schemas:extension:enterprise:2.0:User".

The following Singular Attributes are defined:

employeeNumber Numeric or alphanumeric identifier assigned to a person, typically based on order of hire or association with an organization.

costCenter Identifies the name of a cost center.

organization Identifies the name of an organization.

division Identifies the name of a division.

department Identifies the name of a department.

manager The user's manager. A complex type that optionally allows service providers to represent organizational hierarchy by referencing the "id" attribute of another User.

value The "id" of the SCIM resource representing the user's manager. RECOMMENDED.

$ref The URI of the SCIM resource representing the User's manager. RECOMMENDED.

displayName The displayName of the user's manager. This attribute is OPTIONAL and mutability is "readOnly".

8. SCIM Group Schema

SCIM provides a schema for representing groups, identified using the following schema URI: "urn:scim:schemas:core:2.0:Group".

Group resources are meant to enable expression of common group or role based access control models, although no explicit authorization model is defined. It is intended that the semantics of group membership and any behavior or authorization granted as a result of membership are defined by the service provider are considered out of scope for this specification.

The following singular attribute is defined in addition to the common attributes defined in SCIM core schema:

displayName A human readable name for the Group. REQUIRED.

The following multi-valued attribute is defined in addition to the common attributes defined in SCIM Core Schema:

members A list of members of the Group. While values MAY be added or removed, sub-attributes of members are "immutable". The "value" sub-attribute must be the "id" and the "$ref" sub-attribute must be the URI of a SCIM resource, either a "User", or a "Group". The intention of the "Group" type is to allow the service provider to support nested groups. Service providers MAY require clients to provide a non-empty members value based on the

"required" sub attribute of the "members" attribute in the "Group" resource schema.

9. Service Provider Configuration Schema

SCIM provides a schema for representing the service provider's configuration identified using the following schema URI: "urn:scim:schemas:core:2.0:ServiceProviderConfig"

The service provider configuration resource enables a service provider to discovery of SCIM specification features in a standardized form as well as provide additional implementation details to clients. All attributes are READ-ONLY (a mutability of "readOnly"). Unlike other core resources, the "id" attribute is not required for the service provider configuration resource.

The following Singular Attributes are defined in addition to the common attributes defined in Core Schema:

documentationUrl An HTTP addressable URL pointing to the service provider's human consumable help documentation.

patch A complex type that specifies PATCH configuration options. REQUIRED.

supported Boolean value specifying whether the operation is supported. REQUIRED.

bulk A complex type that specifies BULK configuration options. REQUIRED

supported Boolean value specifying whether the operation is supported. REQUIRED.

maxOperations An integer value specifying the maximum number of operations. REQUIRED.

maxPayloadSize An integer value specifying the maximum payload size in bytes. REQUIRED.

filter A complex type that specifies FILTER options. REQUIRED.

supported Boolean value specifying whether the operation is supported. REQUIRED.

maxResults Integer value specifying the maximum number of resources returned in a response. REQUIRED.

changePassword A complex type that specifies Change Password configuration options. REQUIRED.

supported Boolean value specifying whether the operation is supported. REQUIRED.

sort A complex type that specifies Sort configuration options. REQUIRED.

supported Boolean value specifying whether sorting is supported. REQUIRED.

etag A complex type that specifies Etag configuration options. REQUIRED.

supported Boolean value specifying whether the operation is supported. REQUIRED.

The following multi-valued attribute is defined in addition to the common attributes defined in core schema:

authenticationSchemes A complex type that specifies supported Authentication Scheme properties. This attribute defines the following canonical values to represent common schemes: "oauth", "oauth2", "oauthbearertoken", "httpbasic", and "httpdigest". To enable seamless discovery of configuration, the service provider SHOULD, with the appropriate security considerations, make the authenticationSchemes attribute publicly accessible without prior authentication. REQUIRED.

name The common authentication scheme name; e.g., HTTP Basic. REQUIRED.

description A description of the Authentication Scheme. REQUIRED.

specUrl A HTTP addressable URL pointing to the Authentication Scheme’s specification. OPTIONAL.

documentationUrl A HTTP addressable URL pointing to the Authentication Scheme’s usage documentation. OPTIONAL.

10. ResourceType Schema

The "ResourceType" schema specifies the meta-data about a resource type. Resource type resources are READ-ONLY and identified using the following schema URI: "urn:scim:schemas:core:2.0:ResourceType". Unlike other core resources, all attributes are REQUIRED unless

otherwise specified. The "id" attribute is not required for the resource type resource.

The following Singular Attributes are defined:

id The resource type's server unique id. Often this is the same value as the "name" attribute. OPTIONAL

name The resource type name. When applicable service providers MUST specify the name specified in the core schema specification; e.g., "User" or "Group". This name is referenced by the "meta.resourceType" attribute in all resources.

description The resource type's human readable description. When applicable service providers MUST specify the description specified in the core schema specification.

endpoint The resource type's HTTP addressable endpoint relative to the Base URL; e.g., "/Users".

schema The resource type's primary schema URI; e.g., "urn:scim:schemas:core:2.0:User". This MUST be equal to the "id" attribute of the associated "Schema" resource.

schemaExtensions A list of URIs of the resource type's schema extensions. OPTIONAL.

schema The URI of an extended schema; e.g., "urn:edu:2.0:Staff". This MUST be equal to the "id" attribute of a "Schema" resource. REQUIRED.

required A Boolean value that specifies whether the schema extension is required for the resource type. If true, a resource of this type MUST include this schema extension and include any attributes declared as required in this schema extension. If false, a resource of this type MAY omit this schema extension. REQUIRED.

11. Schema Schema

The "Schema" schema specifies the attribute(s) and meta-data that constitute a "Schema" resource. Schema resources have mutability of "readOnly" and identified using the following URI: "urn:scim:schemas:core:2.0:Schema". Unlike other core resources the "Schema" resource MAY contain a complex object within a sub-attribute and all attributes are REQUIRED unless otherwise specified.

The following Singular Attributes are defined:

id The unique URI of the schema. When applicable service providers MUST specify the URI specified in the core schema specification; e.g., "urn:scim:schemas:core:2.0:User". Unlike most other schemas, which use some sort of a GUID for the "id", the schema "id" is a URI so that it can be registered and is portable between different service providers and clients.

name The schema's human readable name. When applicable service providers MUST specify the name specified in the core schema specification; e.g., "User" or "Group". OPTIONAL.

description The schema's human readable description. When applicable service providers MUST specify the description specified in the core schema specification. OPTIONAL.

The following multi-valued attribute is defined:

attributes A complex type that specifies the set of resource attributes.

name The attribute's name.

type The attribute's data type; e.g., "String".

multiValued Boolean value indicating the attribute's plurality.

description The attribute's human readable description. When applicable service providers MUST specify the description specified in the core schema specification.

required A Boolean value that specifies if the attribute is required.

caseExact A Boolean value that specifies if the String attribute is case sensitive.

mutability A single keyword indicating what types of modifications an attribute MAY accept as follows:

readOnly The attribute MAY NOT be modified.

readWrite The attribute MAY be updated and read at any time. DEFAULT.

immutable The attribute MAY be defined at resource creation (e.g. POST) or at record replacement via request (e.g. a PUT). The attribute MAY NOT be updated.

writeOnly The attribute MAY be updated at any time. Attribute values MAY NOT be returned (e.g. because the value is a stored hash). Note: an attribute with mutability of "writeOnly" usually also has a returned setting of "never".

returned A single keyword that indicates when an attribute and associated values are returned in response to a GET request or in response to a PUT, POST, or PATCH request. Valid keywords are:

always The attribute is always returned regardless of the contents of the "attributes" parameter. For example, "id" is always returned to identify a SCIM resource.

never The attribute is never returned. This may occur because the original attribute value is not retained by the service provider (e.g. such as with a hashed value). A service provider MAY allow attributes to be used in a search filter.

default The attribute is returned by default in all SCIM operation responses where attribute values are returned. If the GET request "attributes" parameter is specified, attribute values are only returned if the attribute is named in the attributes parameter. DEFAULT.

request The attribute is returned in response to any PUT, POST, or PATCH operations if the attribute was specified by the client (for example, the attribute was modified). The attribute is returned in a SCIM query operation only if specified in the "attributes" parameter.

uniqueness A single keyword value that specifies how the service provider enforces uniqueness of attribute values. A server MAY reject an invalid value based on uniqueness by returning HTTP Response code 400 (Bad Request). A client MAY enforce uniqueness on the client-side to a greater degree than the service provider enforces. For example, a client could make a value unique while the server has uniqueness of "none". Valid keywords are:

none The values are not intended to be unique in any way. DEFAULT.

server The value SHOULD be unique within the context of the current SCIM endpoint (or tenancy) but MAY not be globally unique (e.g. a "username", email address, or other server generated key or counter). No two resources on the same server SHOULD possess the same value.

global The value SHOULD be globally unique (e.g. an email address, a GUID, or other value). No two resources on any server SHOULD possess the same value.

referenceTypes The names of the resource types that may be referenced; e.g., "User". This is only applicable for attributes that are of the "reference" Section 3.1.7 data type.

The following multi-valued attributes are defined. There are no canonical type values defined and the primary value serves no useful purpose.

name The attribute's name.

type The attribute's data type; e.g., String.

description The attribute's human readable description. When applicable service providers MUST specify the description specified in the core schema specification.

required A Boolean value that specifies if the attribute is required.

caseExact A Boolean value that specifies if the String attribute is case sensitive.

referenceTypes The names of the resource types that may be referenced; e.g., User. This is only applicable for attributes that are of the "reference" Section 3.1.7 data type.

canonicalValues A collection of canonical values. When applicable service providers MUST specify the canonical types specified in the core schema specification; e.g.,"work","home". OPTIONAL.

mutability A single keyword indicating what types of modifications an attribute MAY accept as follows:

readOnly The attribute MAY NOT be modified.

readWrite The attribute MAY be updated and read at any time. DEFAULT.

immutable The attribute MAY be defined at resource creation (e.g. POST) or at record replacement via request (e.g. a PUT). The attribute MAY NOT be updated.

writeOnly The attribute MAY be updated at any time. Attribute values MAY NOT be returned (e.g. because the value is a stored hash). Note: an attribute with mutability of "writeOnly" usually also has a returned setting of "never".

returned A single keyword that indicates when an attribute and associated values are returned in response to a GET request or in response to a PUT, POST, or PATCH request. Valid keywords are:

always The attribute is always returned regardless of the contents of the "attributes" parameter. For example, "id" is always returned to identify a SCIM resource.

never The attribute is never returned. This may occur because the original attribute value is not retained by the service provider (e.g. such as with a hashed value). A service provider MAY allow attributes to be used in a search filter.

default The attribute is returned by default in all SCIM operation responses where attribute values are returned. If the GET request "attributes" parameter is specified, attribute values are only returned if the attribute is named in the attributes parameter. DEFAULT.

request The attribute is returned in response to any PUT, POST, or PATCH operations if the attribute was specified by the client (for example, the attribute was modified). The attribute is returned in a SCIM query operation only if specified in the "attributes" parameter.

uniqueness A single keyword value that specifies how the service provider enforces uniqueness of attribute values. A server MAY reject an invalid value based on uniqueness by returning HTTP Response code 400 (Bad Request). A client MAY enforce uniqueness on the client-side to a greater degree than the service provider enforces. For example, a client could make a value unique while the server has uniqueness of "none". Valid keywords are:

none The values are not intended to be unique in any way. DEFAULT.

server The value SHOULD be unique within the context of the current SCIM endpoint (or tenancy) but MAY not be globally unique (e.g. a "username", email address, or

other server generated key or counter). No two resources on the same server SHOULD possess the same value.

global The value SHOULD be globally unique (e.g. an email address, a GUID, or other value). No two resources on any server SHOULD possess the same value.

12. JSON Representation

12.1. Minimal User Representation

The following is a non-normative example of the minimal required SCIM representation in JSON format.

```
{
  "schemas": ["urn:scim:schemas:core:2.0:User"],
  "id": "2819c223-7f76-453a-919d-413861904646",
  "userName": "bjensen@example.com",
  "meta": {
    "resourceType": "User",
    "created": "2010-01-23T04:56:22Z",
    "lastModified": "2011-05-13T04:42:34Z",
    "version": "W\/\"3694e05e9dff590\"",
    "location": "https://example.com/v2/Users/2819c223-7f76-453a-919d-4138619046
46"
  }
}
```

12.2. Full User Representation

The following is a non-normative example of the fully populated SCIM representation in JSON format.

```
{
  "schemas": ["urn:scim:schemas:core:2.0:User"],
  "id": "2819c223-7f76-453a-919d-413861904646",
  "externalId": "701984",
  "userName": "bjensen@example.com",
  "name": {
    "formatted": "Ms. Barbara J Jensen III",
    "familyName": "Jensen",
    "givenName": "Barbara",
    "middleName": "Jane",
    "honorificPrefix": "Ms.",
    "honorificSuffix": "III"
  },
  "displayName": "Babs Jensen",
  "nickName": "Babs",
  "profileUrl": "https://login.example.com/bjensen",
```

```
  "emails": [
    {
      "value": "bjensen@example.com",
      "type": "work",
      "primary": true
    },
    {
      "value": "babs@jensen.org",
      "type": "home"
    }
  ],
  "addresses": [
    {
      "type": "work",
      "streetAddress": "100 Universal City Plaza",
      "locality": "Hollywood",
      "region": "CA",
      "postalCode": "91608",
      "country": "USA",
      "formatted": "100 Universal City Plaza\nHollywood, CA 91608 USA",
      "primary": true
    },
    {
      "type": "home",
      "streetAddress": "456 Hollywood Blvd",
      "locality": "Hollywood",
      "region": "CA",
      "postalCode": "91608",
      "country": "USA",
      "formatted": "456 Hollywood Blvd\nHollywood, CA 91608 USA"
    }
  ],
  "phoneNumbers": [
    {
      "value": "555-555-5555",
      "type": "work"
    },
    {
      "value": "555-555-4444",
      "type": "mobile"
    }
  ],
  "ims": [
    {
      "value": "someaimhandle",
      "type": "aim"
    }
  ],
```

```
  "photos": [
    {
      "value": "https://photos.example.com/profilephoto/72930000000Ccne/F",
      "type": "photo"
    },
    {
      "value": "https://photos.example.com/profilephoto/72930000000Ccne/T",
      "type": "thumbnail"
    }
  ],
  "userType": "Employee",
  "title": "Tour Guide",
  "preferredLanguage":"en_US",
  "locale": "en_US",
  "timezone": "America/Los_Angeles",
  "active":true,
  "password":"t1meMa$heen",
  "groups": [
    {
      "value": "e9e30dba-f08f-4109-8486-d5c6a331660a",
      "$ref": "https://example.com/v2/Groups/e9e30dba-f08f-4109-8486-d5c6a331660
a",
      "display": "Tour Guides"
    },
    {
      "value": "fc348aa8-3835-40eb-a20b-c726e15c55b5",
      "$ref": "https://example.com/v2/Groups/fc348aa8-3835-40eb-a20b-c726e15c55b
5",
      "display": "Employees"
    },
    {
      "value": "71ddacd2-a8e7-49b8-a5db-ae50d0a5bfd7",
      "$ref": "https://example.com/v2/Groups/71ddacd2-a8e7-49b8-a5db-ae50d0a5bfd
7",
      "display": "US Employees"
    }
  ],
  "x509Certificates": [
    {
      "value": "MIIDQzCCAqygAwIBAgICEAAwDQYJKoZIhvcNAQEFBQAwTjELMAkGA1UEBhMCVVMx
                EzARBgNVBAgMCkNhbGlmb3JuaWExFDASBgNVBAoMC2V4YW1wbGUuY29tMRQwEgYD
                VQQDDAtleGFtcGxlLmNvbTAeFw0xMTEwMjIwNjI0MzFaFw0xMjEwMDQwNjI0MzFa
                MH8xCzAJBgNVBAYTAlVTMRMwEQYDVQQIDApDYWxpZm9ybmlhMRQwEgYDVQQKDAtl
                eGFtcGxlLmNvbTEhMB8GA1UEAwwYTXMuIEJhcmJhcmEgSiBKZW5zZW4gSUlJMSIw
                IAYJKoZIhvcNAQkBFhNiamVuc2VuQGV4YW1wbGUuY29tMIIBIjANBgkqhkiG9w0B
                AQEFAAOCAQ8AMIIBCgKCAQEA7Kr+Dcds/JQ5GwejJFcBIP682X3xpjis56AK02bc
                1FLgzdLI8auoR+cC9/Vrh5t66HkQIOdA4unHh0AaZ4xL5PhVbXIPMB5vAPKpzz5i
                PSi8xO8SL7I7SDhcBVJhqVqr3HgllEG6UClDdHO7nkLuwXq8HcISKkbT5WFTVfFZ
                zidPl8HZ7DhXkZIRtJwBweq4bvm3hM1Os7UQH05ZS6cVDgweKNwdLLrT51ikSQG3
                DYrl+ft781UQRIqxgwqCfXEuDiinPh0kkvIi5jivVu1Z9QiwlYEdRbLJ4zJQBmDr
                SGTMYn4lRc2HgHO4DqB/bnMVorHB0CC6AV1QoFK4GPe1LwIDAQABo3sweTAJBgNV
```

```
                HRMEAjAAMCwGCWCGSAGG+EIBDQQfFh1PcGVuU1NMIEdlbmVyYXRlZCBDZXJ0aWZp
                Y2F0ZTAdBgNVHQ4EFgQU8pD0U0vsZIsaA16lL8En8bx0F/gwHwYDVR0jBBgwFoAU
                dGeKitcaF7gnzsNwDx708kqaVt0wDQYJKoZIhvcNAQEFBQADgYEAA81SsFnOdYJt
                Ng5Tcq+/ByEDrBgnusx0jloUhByPMEVkoMZ3J7j1ZgI8rAbOkNngX8+pKfTiDz1R
                C4+dx8oU6Za+4NJXUjlL5CvV6BEYb1+QAEJwitTVvxB/A67g42/vzgAtoRUeDov1
                +GFiBZ+GNF/cAYKcMtGcrs2i97ZkJMo=="
    }
  ],
  "meta": {
    "resourceType": "User",
    "created": "2010-01-23T04:56:22Z",
    "lastModified": "2011-05-13T04:42:34Z",
    "version": "W\/\"a330bc54f0671c9\"",
    "location": "https://example.com/v2/Users/2819c223-7f76-453a-919d-4138619046
46"
  }
}
```


12.3. Enterprise User Extension Representation

The following is a non-normative example of the fully populated User using the enterprise User extension in JSON format.


```
{
  "schemas": ["urn:scim:schemas:core:2.0:User", "urn:scim:schemas:extension:ente
rprise:2.0:User"],
  "id": "2819c223-7f76-453a-919d-413861904646",
  "externalId": "701984",
  "userName": "bjensen@example.com",
  "name": {
    "formatted": "Ms. Barbara J Jensen III",
    "familyName": "Jensen",
    "givenName": "Barbara",
    "middleName": "Jane",
    "honorificPrefix": "Ms.",
    "honorificSuffix": "III"
  },
  "displayName": "Babs Jensen",
  "nickName": "Babs",
  "profileUrl": "https://login.example.com/bjensen",
  "emails": [
    {
      "value": "bjensen@example.com",
      "type": "work",
      "primary": true
    },
    {
      "value": "babs@jensen.org",
      "type": "home"
    }
```

```
  ],
  "addresses": [
    {
      "streetAddress": "100 Universal City Plaza",
      "locality": "Hollywood",
      "region": "CA",
      "postalCode": "91608",
      "country": "USA",
      "formatted": "100 Universal City Plaza\nHollywood, CA 91608 USA",
      "type": "work",
      "primary": true
    },
    {
      "streetAddress": "456 Hollywood Blvd",
      "locality": "Hollywood",
      "region": "CA",
      "postalCode": "91608",
      "country": "USA",
      "formatted": "456 Hollywood Blvd\nHollywood, CA 91608 USA",
      "type": "home"
     }
  ],
  "phoneNumbers": [
    {
      "value": "555-555-5555",
      "type": "work"
    },
    {
      "value": "555-555-4444",
      "type": "mobile"
    }
  ],
  "ims": [
    {
      "value": "someaimhandle",
      "type": "aim"
    }
  ],
  "photos": [
    {
      "value": "https://photos.example.com/profilephoto/72930000000Ccne/F",
      "type": "photo"
    },
    {
      "value": "https://photos.example.com/profilephoto/72930000000Ccne/T",
      "type": "thumbnail"
    }
  ],
```

```
  "userType": "Employee",
  "title": "Tour Guide",
  "preferredLanguage":"en_US",
  "locale": "en_US",
  "timezone": "America/Los_Angeles",
  "active":true,
  "password":"t1meMa$heen",
  "groups": [
    {
      "value": "e9e30dba-f08f-4109-8486-d5c6a331660a",
      "$ref": "/Groups/e9e30dba-f08f-4109-8486-d5c6a331660a",
      "display": "Tour Guides"
    },
    {
      "value": "fc348aa8-3835-40eb-a20b-c726e15c55b5",
      "$ref": "/Groups/fc348aa8-3835-40eb-a20b-c726e15c55b5",
      "display": "Employees"
    },
    {
      "value": "71ddacd2-a8e7-49b8-a5db-ae50d0a5bfd7",
      "$ref": "/Groups/71ddacd2-a8e7-49b8-a5db-ae50d0a5bfd7",
      "display": "US Employees"
    }
  ],
  "x509Certificates": [
    {
      "value": "MIIDQzCCAqygAwIBAgICEAAwDQYJKoZIhvcNAQEFBQAwTjELMAkGA1UEBhMCVVMx
                EzARBgNVBAgMCkNhbGlmb3JuaWExFDASBgNVBAoMC2V4YW1wbGUuY29tMRQwEgYD
                VQQDDAtleGFtcGxlLmNvbTAeFw0xMTEwMjIwNjI0MzFaFw0xMjEwMDQwNjI0MzFa
                MH8xCzAJBgNVBAYTAlVTMRMwEQYDVQQIDApDYWxpZm9ybmlhMRQwEgYDVQQKDAtl
                eGFtcGxlLmNvbTEhMB8GA1UEAwwYTXMuIEJhcmJhcmEgSiBKZW5zZW4gSUlJMSIw
                IAYJKoZIhvcNAQkBFhNiamVuc2VuQGV4YW1wbGUuY29tMIIBIjANBgkqhkiG9w0B
                AQEFAAOCAQ8AMIIBCgKCAQEA7Kr+Dcds/JQ5GwejJFcBIP682X3xpjis56AK02bc
                1FLgzdLI8auoR+cC9/Vrh5t66HkQIOdA4unHh0AaZ4xL5PhVbXIPMB5vAPKpzz5i
                PSi8xO8SL7I7SDhcBVJhqVqr3HgllEG6UClDdHO7nkLuwXq8HcISKkbT5WFTVfFZ
                zidPl8HZ7DhXkZIRtJwBweq4bvm3hM1Os7UQH05ZS6cVDgweKNwdLLrT51ikSQG3
                DYrl+ft781UQRIqxgwqCfXEuDiinPh0kkvIi5jivVu1Z9QiwlYEdRbLJ4zJQBmDr
                SGTMYn4lRc2HgHO4DqB/bnMVorHB0CC6AV1QoFK4GPe1LwIDAQABo3sweTAJBgNV
                HRMEAjAAMCwGCWCGSAGG+EIBDQQfFh1PcGVuU1NMIEdlbmVyYXRlZCBDZXJ0aWZp
                Y2F0ZTAdBgNVHQ4EFgQU8pD0U0vsZIsaA16lL8En8bx0F/gwHwYDVR0jBBgwFoAU
                dGeKitcaF7gnzsNwDx708kqaVt0wDQYJKoZIhvcNAQEFBQADgYEAA81SsFnOdYJt
                Ng5Tcq+/ByEDrBgnusx0jloUhByPMEVkoMZ3J7j1ZgI8rAbOkNngX8+pKfTiDz1R
                C4+dx8oU6Za+4NJXUjlL5CvV6BEYb1+QAEJwitTVvxB/A67g42/vzgAtoRUeDov1
                +GFiBZ+GNF/cAYKcMtGcrs2i97ZkJMo="
    }
  ],
  "urn:scim:schemas:extension:enterprise:2.0:User": {
    "employeeNumber": "701984",
```

```
    "costCenter": "4130",
    "organization": "Universal Studios",
    "division": "Theme Park",
    "department": "Tour Operations",
    "manager": {
      "managerId": "26118915-6090-4610-87e4-49d8ca9f808d",
      "$ref": "/Users/26118915-6090-4610-87e4-49d8ca9f808d",
      "displayName": "John Smith"
    }
  },
  "meta": {
    "resourceType": "User",
    "created": "2010-01-23T04:56:22Z",
    "lastModified": "2011-05-13T04:42:34Z",
    "version": "W\/\"3694e05e9dff591\"",
    "location": "https://example.com/v2/Users/2819c223-7f76-453a-919d-4138619046
46"
  }
}
```

12.4. Group Representation

The following is a non-normative example of SCIM Group representation in JSON format.

```
{
  "schemas": ["urn:scim:schemas:core:2.0:Group"],
  "id": "e9e30dba-f08f-4109-8486-d5c6a331660a",
  "displayName": "Tour Guides",
  "members": [
    {
      "value": "2819c223-7f76-453a-919d-413861904646",
      "$ref": "https://example.com/v2/Users/2819c223-7f76-453a-919d-413861904646
",
      "display": "Babs Jensen"
    },
    {
      "value": "902c246b-6245-4190-8e05-00816be7344a",
      "$ref": "https://example.com/v2/Users/902c246b-6245-4190-8e05-00816be7344a
",
      "display": "Mandy Pepperidge"
    }
  ],
  "meta": {
    "resourceType": "Group",
    "created": "2010-01-23T04:56:22Z",
    "lastModified": "2011-05-13T04:42:34Z",
    "version": "W\/\"3694e05e9dff592\"",
    "location": "https://example.com/v2/Groups/e9e30dba-f08f-4109-8486-d5c6a3316
60a"
  }
}
```

12.5. Service Provider Configuration Representation

The following is a non-normative example of the SCIM service provider configuration representation in JSON format.

```
{
  "schemas": ["urn:scim:schemas:core:2.0:ServiceProviderConfig"],
  "documentationUrl":"http://example.com/help/scim.html",
  "patch": {
    "supported":true
  },
  "bulk": {
    "supported":true,
    "maxOperations":1000,
    "maxPayloadSize":1048576
  },
  "filter": {
    "supported":true,
    "maxResults": 200
  },
  "changePassword" : {
    "supported":true
  },
  "sort": {
    "supported":true
  },
  "etag": {
    "supported":true
  },
  "authenticationSchemes": [
    {
      "name": "OAuth Bearer Token",
      "description": "Authentication Scheme using the OAuth Bearer Token Standar
d",
      "specUrl":"http://tools.ietf.org/html/draft-ietf-oauth-v2-bearer-01",
      "documentationUrl":"http://example.com/help/oauth.html",
      "type":"oauthbearertoken",
      "primary": true
    },
    {
      "name": "HTTP Basic",
      "description": "Authentication Scheme using the Http Basic Standard",
      "specUrl":"http://www.ietf.org/rfc/rfc2617.txt",
      "documentationUrl":"http://example.com/help/httpBasic.html",
      "type":"httpbasic"
     }
  ],
  "meta": {
    "location":"https://example.com/v2/ServiceProviderConfig",
```

```
    "resourceType": "ServiceProviderConfig",
    "created": "2010-01-23T04:56:22Z",
    "lastModified": "2011-05-13T04:42:34Z",
    "version": "W\/\"3694e05e9dff594\""
  }
}
```

12.6. Resource Type Representation

The following is a non-normative example of the SCIM resource type representation in JSON format.

```
{
  "schemas": ["urn:scim:schemas:core:2.0:ResourceType"],
  "id":"User",
  "name":"User",
  "endpoint": "/Users",
  "description": "Core User",
  "schema": "urn:scim:schemas:core:2.0:User",
  "schemaExtensions": [
    {
      "schema": "urn:scim:schemas:extension:enterprise:2.0:EnterpriseUser",
      "required": true
    }
  ],
  "meta": {
    "location":"https://example.com/v2/ResourceTypes/User",
    "resourceType": "ResourceType",
    "created": "2010-01-23T04:56:22Z",
    "lastModified": "2011-05-13T04:42:34Z",
    "version": "W\/\"3694e05e9dff595\""
  }
}
```

12.7. Schema Representation

The following is intended as normative example of the SCIM Schema representation in JSON format. Where permitted individual values and schema MAY change. Included but not limited to, are schemas for User, Group, and enterprise user.

```
{[
  {
    "id" : "urn:scim:schemas:core:2.0:User",
    "name" : "User",
    "description" : "Core User",
    "attributes" : [
      {
```

```
        "name" : "id",
        "type" : "string",
        "multiValued" : false,
        "description" : "Unique identifier for the SCIM resource as defined by t
he Service Provider. Each representation of the resource MUST include a non-empt
y id value. This identifier MUST be unique across the Service Provider's entire
set of resources. It MUST be a stable, non-reassignable identifier that does not
 change when the same resource is returned in subsequent requests. The value of
the id attribute is always issued by the Service Provider and MUST never be spec
ified by the Service Consumer. REQUIRED.",
        "required" : true,
        "caseExact" : false,
        "mutability" : "readOnly",
        "returned" : "always",
        "uniqueness" : "server"
      },
      {
        "name" : "externalId",
        "type" : "string",
        "multiValued" : false,
        "description" : "An identifier for the Resource as defined by the Servic
e Consumer.",
        "required" : true,
        "caseExact" : false,
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      },
      {
        "name" : "userName",
        "type" : "string",
        "multiValued" : false,
        "description" : "Unique identifier for the User typically used by the us
er to directly authenticate to the service provider. Each User MUST include a no
n-empty userName value.  This identifier MUST be unique across the Service Consu
mer's entire set of Users.  REQUIRED",
        "required" : true,
        "caseExact" : false,
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "server"
      },
      {
        "name" : "name",
        "type" : "complex",
        "multiValued" : false,
        "description" : "The components of the user's real name. Providers MAY r
eturn just the full name as a single string in the formatted sub-attribute, or t
hey MAY return just the individual component attributes using the other sub-attr
ibutes, or they MAY return both. If both variants are returned, they SHOULD be d
escribing the same name, with the formatted name indicating how the component at
tributes should be combined.",
        "required" : false,
        "caseExact" : false,
        "subAttributes" : [
          {
            "name" : "formatted",
            "type" : "string",
            "multiValued" : false,
            "description" : "The full name, including all middle names, titles,
and suffixes as appropriate, formatted for display (e.g. Ms. Barbara J Jensen, I
II.).",
            "required" : false,
```

```
            "caseExact" : false,
            "mutability" : "readWrite",
```

```
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "familyName",
            "type" : "string",
            "multiValued" : false,
            "description" : "The family name of the User, or Last Name in most W
estern languages (e.g. Jensen given the full name Ms. Barbara J Jensen, III.).",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "givenName",
            "type" : "string",
            "multiValued" : false,
            "description" : "The given name of the User, or First Name in most W
estern languages (e.g. Barbara given the full name Ms. Barbara J Jensen, III.)."
,
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "middleName",
            "type" : "string",
            "multiValued" : false,
            "description" : "The middle name(s) of the User (e.g. Robert given t
he full name Ms. Barbara J Jensen, III.).",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "honorificPrefix",
            "type" : "string",
            "multiValued" : false,
            "description" : "The honorific prefix(es) of the User, or Title in m
ost Western languages (e.g. Ms. given the full name Ms. Barbara J Jensen, III.).
",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
```

```
            "name" : "honorificSuffix",
            "type" : "string",
            "multiValued" : false,
            "description" : "The honorific suffix(es) of the User, or Suffix in
most Western languages (e.g. III. given the full name Ms. Barbara J Jensen, III.
).",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          }
        ],
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      },
      {
        "name" : "displayName",
        "type" : "string",
        "multiValued" : false,
        "description" : "The name of the User, suitable for display to end-users
. The name SHOULD be the full name of the User being described if known",
        "required" : false,
        "caseExact" : false,
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      },
      {
        "name" : "nickName",
        "type" : "string",
        "multiValued" : false,
        "description" : "The casual way to address the user in real life, e.g. \
"Bob\" or \"Bobby\" instead of \"Robert\".  This attribute SHOULD NOT be used to
 represent a User's username (e.g. bjensen or mpepperidge)",
        "required" : false,
        "caseExact" : false,
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      },
      {
        "name" : "profileUrl",
        "type" : "string",
        "multiValued" : false,
        "description" : "A fully qualified URL to a page representing the User's
 online profile",
        "required" : false,
        "caseExact" : false,
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      },
```

```
      {
        "name" : "title",
        "type" : "string",
        "multiValued" : false,
        "description" : "The user’s title, such as \"Vice President.\"",
        "required" : false,
        "caseExact" : false,
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      },
      {
        "name" : "userType",
        "type" : "string",
        "multiValued" : false,
        "description" : "Used to identify the organization to user relationship.
 Typical values used might be \"Contractor\", \"Employee\", \"Intern\", \"Temp\"
, \"External\", and \"Unknown\" but any value may be used ",
        "required" : false,
        "caseExact" : false,
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      },
      {
        "name" : "preferredLanguage",
        "type" : "string",
        "multiValued" : false,
        "description" : "Indicates the User’s preferred written or spoken langua
ge.  Generally used for selecting a localized User interface. e.g., ’en_US’ spec
ifies the language English and country US.",
        "required" : false,
        "caseExact" : false,
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      },
      {
        "name" : "locale",
        "type" : "string",
        "multiValued" : false,
        "description" : "Used to indicate the User’s default location for purpos
es of localizing items such as currency, date time format, numerical representat
ions, etc.",
        "required" : false,
        "caseExact" : false,
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      },
      {
        "name" : "timezone",
        "type" : "string",
        "multiValued" : false,
```

```
        "description" : "The User's time zone in the \"Olson\" timezone database
 format [19]; e.g.,'America/Los_Angeles'",
        "required" : false,
        "caseExact" : false,
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      },
      {
        "name" : "active",
        "type" : "boolean",
        "multiValued" : false,
        "description" : "A Boolean value indicating the User's administrative st
atus.",
        "required" : false,
        "caseExact" : false,
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      },
      {
        "name" : "password",
        "type" : "string",
        "multiValued" : false,
        "description" : "The User's clear text password.  This attribute is inte
nded to be used as a means to specify an initial password when creating a new Us
er or to reset an existing User's password.",
        "required" : false,
        "caseExact" : false,
        "mutability" : "writeOnly",
        "returned" : "never",
        "uniqueness" : "none"
      },
      {
        "name" : "emails",
        "type" : "complex",
        "multiValued" : true,
        "description" : "E-mail addresses for the user. The value SHOULD be cano
nicalized by the Service Provider, e.g. bjensen@example.com instead of bjensen@E
XAMPLE.COM. Canonical Type values of work, home, and other.",
        "required" : false,
        "caseExact" : false,
        "subAttributes" : [
          {
            "name" : "value",
            "type" : "string",
            "multiValued" : false,
            "description" : "E-mail addresses for the user. The value SHOULD be
canonicalized by the Service Provider, e.g. bjensen@example.com instead of bjens
en@EXAMPLE.COM. Canonical Type values of work, home, and other.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
```

```
          {
            "name" : "display",
            "type" : "string",
            "multiValued" : false,
            "description" : "A human readable name, primarily used for display p
urposes. READ-ONLY.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "type",
            "type" : "string",
            "multiValued" : false,
            "description" : "A label indicating the attribute's function; e.g.,
'work' or 'home'.",
            "required" : false,
            "caseExact" : false,
            "canonicalValues" : [
              "work",
              "home",
              "other"
            ],
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "primary",
            "type" : "boolean",
            "multiValued" : false,
            "description" : "A Boolean value indicating the 'primary' or preferr
ed attribute value for this attribute, e.g. the preferred mailing address or pri
mary e-mail address. The primary attribute value 'true' MUST appear no more than
 once.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          }
        ],
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      },
      {
        "name" : "phoneNumbers",
        "type" : "complex",
        "multiValued" : true,
        "description" : "Phone numbers for the User.  The value SHOULD be canoni
calized by the Service Provider according to format in RFC3966 [20] e.g. 'tel:+1
-201-555-0123'.  Canonical Type values of work, home, mobile, fax, pager and oth
er.",
```

```
        "required" : false,
        "caseExact" : false,
        "subAttributes" : [
          {
            "name" : "value",
            "type" : "string",
            "multiValued" : false,
            "description" : "Phone number of the User",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "display",
            "type" : "string",
            "multiValued" : false,
            "description" : "A human readable name, primarily used for display p
urposes. READ-ONLY.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "type",
            "type" : "string",
            "multiValued" : false,
            "description" : "A label indicating the attribute’s function; e.g.,
’work’ or ’home’ or ’mobile’ etc.",
            "required" : false,
            "caseExact" : false,
            "canonicalValues" : [
              "work",
              "home",
              "mobile",
              "fax",
              "pager",
              "other"
            ],
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "primary",
            "type" : "boolean",
            "multiValued" : false,
```

```
            "description" : "A Boolean value indicating the 'primary' or preferr
ed attribute value for this attribute, e.g. the preferred phone number or primar
y phone number. The primary attribute value 'true' MUST appear no more than once
.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          }
        ],
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      },
      {
        "name" : "ims",
        "type" : "complex",
        "multiValued" : true,
        "description" : "Instant messaging addresses for the User.",
        "required" : false,
        "caseExact" : false,
        "subAttributes" : [
          {
            "name" : "value",
            "type" : "string",
            "multiValued" : false,
            "description" : "Instant messaging address for the User.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "display",
            "type" : "string",
            "multiValued" : false,
            "description" : "A human readable name, primarily used for display p
urposes. READ-ONLY.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "type",
            "type" : "string",
            "multiValued" : false,
            "description" : "A label indicating the attribute's function; e.g.,
'aim', 'gtalk', 'mobile' etc.",
            "required" : false,
```

```
            "caseExact" : false,
            "canonicalValues" : [
              "aim",
              "gtalk",
              "icq",
              "xmpp",
              "msn",
              "skype",
              "qq",
              "yahoo"
            ],
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "primary",
            "type" : "boolean",
            "multiValued" : false,
            "description" : "A Boolean value indicating the ’primary’ or preferr
ed attribute value for this attribute, e.g. the preferred messenger or primary m
essenger. The primary attribute value ’true’ MUST appear no more than once.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          }
        ],
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      },
      {
        "name" : "photos",
        "type" : "complex",
        "multiValued" : true,
        "description" : "URLs of photos of the User.",
        "required" : false,
        "caseExact" : false,
        "subAttributes" : [
          {
            "name" : "value",
            "type" : "string",
            "multiValued" : false,
            "description" : "URL of a photo of the User.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
```

```
            "uniqueness" : "none"
          },
          {
            "name" : "display",
            "type" : "string",
            "multiValued" : false,
            "description" : "A human readable name, primarily used for display p
urposes. READ-ONLY.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "type",
            "type" : "string",
            "multiValued" : false,
            "description" : "A label indicating the attribute’s function; e.g.,
’photo’ or ’thumbnail’.",
            "required" : false,
            "caseExact" : false,
            "canonicalValues" : [
              "photo",
              "thumbnail"
            ],
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "primary",
            "type" : "boolean",
            "multiValued" : false,
            "description" : "A Boolean value indicating the ’primary’ or preferr
ed attribute value for this attribute, e.g. the preferred photo or thumbnail. Th
e primary attribute value ’true’ MUST appear no more than once.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          }
        ],
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      },
      {
        "name" : "addresses",
        "type" : "complex",
        "multiValued" : true,
```

```
        "description" : "A physical mailing address for this User, as described
in (address Element). Canonical Type Values of work, home, and other. The value
attribute is a complex type with the following sub-attributes.",
        "required" : false,
        "caseExact" : false,
        "subAttributes" : [
          {
            "name" : "formatted",
            "type" : "string",
            "multiValued" : false,
            "description" : "The full mailing address, formatted for display or
use with a mailing label. This attribute MAY contain newlines.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "streetAddress",
            "type" : "string",
            "multiValued" : false,
            "description" : "The full street address component, which may includ
e house number, street name, PO BOX, and multi-line extended street address info
rmation. This attribute MAY contain newlines.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "locality",
            "type" : "string",
            "multiValued" : false,
            "description" : "The city or locality component.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "region",
            "type" : "string",
            "multiValued" : false,
            "description" : "The state or region component.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
```

```
          {
            "name" : "postalCode",
            "type" : "string",
            "multiValued" : false,
            "description" : "The zipcode or postal code component.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "country",
            "type" : "string",
            "multiValued" : false,
            "description" : "The country name component.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "type",
            "type" : "string",
            "multiValued" : false,
            "description" : "A label indicating the attribute’s function; e.g.,
’work’ or ’home’.",
            "required" : false,
            "caseExact" : false,
            "canonicalValues" : [
              "work",
              "home",
              "other"
            ],
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          }
        ],
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      },
      {
        "name" : "groups",
        "type" : "complex",
        "multiValued" : true,
        "description" : "A list of groups that the user belongs to, either thoro
ugh direct membership, nested groups, or dynamically calculated",
```

```
        "required" : false,
        "caseExact" : false,
        "subAttributes" : [
          {
            "name" : "value",
            "type" : "string",
            "multiValued" : false,
            "description" : "The identifier of the User’s group.",
            "readOnly" : false,
            "required" : false,
            "caseExact" : false,
            "mutability" : "readOnly",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "$ref",
            "type" : "string",
            "multiValued" : false,
            "description" : "The URI of the corresponding Group resource to whic
h the user belongs",
            "readOnly" : false,
            "required" : false,
            "caseExact" : false,
            "mutability" : "readOnly",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "display",
            "type" : "string",
            "multiValued" : false,
            "description" : "A human readable name, primarily used for display p
urposes. READ-ONLY.",
            "readOnly" : true,
            "required" : false,
            "caseExact" : false,
            "mutability" : "readOnly",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "type",
            "type" : "string",
            "multiValued" : false,
            "description" : "A label indicating the attribute’s function; e.g.,
’direct’ or ’indirect’.",
            "readOnly" : false,
            "required" : false,
            "caseExact" : false,
            "canonicalValues" : [
```

```
              "direct",
              "indirect"
            ],
            "mutability" : "readOnly",
            "returned" : "default",
            "uniqueness" : "none"
          }
        ],
        "mutability" : "readOnly",
        "returned" : "default",
        "uniqueness" : "none"
      },
      {
        "name" : "entitlements",
        "type" : "complex",
        "multiValued" : true,
        "description" : "A list of entitlements for the User that represent a th
ing the User has.",
        "required" : false,
        "caseExact" : false,
        "subAttributes" : [
          {
            "name" : "value",
            "type" : "string",
            "multiValued" : false,
            "description" : "The value of an entitlement.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "display",
            "type" : "string",
            "multiValued" : false,
            "description" : "A human readable name, primarily used for display p
urposes. READ-ONLY.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "type",
            "type" : "string",
            "multiValued" : false,
            "description" : "A label indicating the attribute’s function.",
            "required" : false,
```

```
            "caseExact" : false,
            "canonicalValues" : [],
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "primary",
            "type" : "boolean",
            "multiValued" : false,
            "description" : "A Boolean value indicating the 'primary' or preferr
ed attribute value for this attribute. The primary attribute value 'true' MUST a
ppear no more than once.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          }
        ],
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      },
      {
        "name" : "roles",
        "type" : "complex",
        "multiValued" : true,
        "description" : "A list of roles for the User that collectively represen
t who the User is; e.g., 'Student', 'Faculty'.",
        "required" : false,
        "caseExact" : false,
        "subAttributes" : [
          {
            "name" : "value",
            "type" : "string",
            "multiValued" : false,
            "description" : "The value of a role.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "display",
            "type" : "string",
            "multiValued" : false,
            "description" : "A human readable name, primarily used for display p
urposes. READ-ONLY.",
            "required" : false,
            "caseExact" : false,
```

```
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "type",
            "type" : "string",
            "multiValued" : false,
            "description" : "A label indicating the attribute’s function.",
            "required" : false,
            "caseExact" : false,
            "canonicalValues" : [],
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "primary",
            "type" : "boolean",
            "multiValued" : false,
            "description" : "A Boolean value indicating the ’primary’ or preferr
ed attribute value for this attribute. The primary attribute value ’true’ MUST a
ppear no more than once.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          }
        ],
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      },
      {
        "name" : "x509Certificates",
        "type" : "complex",
        "multiValued" : true,
        "description" : "A list of certificates issued to the User.",
        "required" : false,
        "caseExact" : false,
        "subAttributes" : [
          {
            "name" : "value",
            "type" : "string",
            "multiValued" : false,
            "description" : "The value of a X509 certificate.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
```

```
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "display",
            "type" : "string",
            "multiValued" : false,
            "description" : "A human readable name, primarily used for display p
urposes. READ-ONLY.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "type",
            "type" : "string",
            "multiValued" : false,
            "description" : "A label indicating the attribute’s function.",
            "required" : false,
            "caseExact" : false,
            "canonicalValues" : [],
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "primary",
            "type" : "boolean",
            "multiValued" : false,
            "description" : "A Boolean value indicating the ’primary’ or preferr
ed attribute value for this attribute. The primary attribute value ’true’ MUST a
ppear no more than once.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          }
        ],
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      }
    ],
    "meta" : {
      "resourceType" : "Schema",
      "created" : "2010-01-23T04:56:22Z",
      "lastModified" : "2014-02-04T00:00:00Z",
      "version" : "W/\"3694e05e9dff596\"",
```

```
      "location" : "https://example.com/v2/Schemas/urn:scim:schemas:core:2.0:Use
r"
    }
  },
  {
    "id" : "urn:scim:schemas:core:2.0:Group",
    "name" : "Group",
    "description" : "Core Group",
    "attributes" : [
      {
        "name" : "id",
        "type" : "string",
        "multiValued" : false,
        "description" : "Unique identifier for the SCIM resource as defined by t
he Service Provider. Each representation of the resource MUST include a non-empt
y id value. This identifier MUST be unique across the Service Provider's entire
set of resources. It MUST be a stable, non-reassignable identifier that does not
 change when the same resource is returned in subsequent requests. The value of
the id attribute is always issued by the Service Provider and MUST never be spec
ified by the Service Consumer. REQUIRED.",
        "required" : true,
        "caseExact" : false,
        "mutability" : "readWrite",
        "returned" : "always",
        "uniqueness" : "server"
      },
      {
        "name" : "externalId",
        "type" : "string",
        "multiValued" : false,
        "description" : "An identifier for the Resource as defined by the Servic
e Consumer.",
        "required" : true,
        "caseExact" : false,
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      },
      {
        "name" : "displayName",
        "type" : "string",
        "multiValued" : false,
        "description" : "A human readable name for the Group.  REQUIRED.",
        "required" : false,
        "caseExact" : false,
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      },
      {
        "name" : "members",
        "type" : "complex",
        "multiValued" : false,
        "description" : "A list of members of the Group.",
        "required" : false,
        "caseExact" : false,
```

```
        "subAttributes" : [
          {
            "name" : "value",
            "type" : "string",
            "multiValued" : false,
            "description" : "The identifier of the member of this Group.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "immutable",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "$ref",
            "type" : "string",
            "multiValued" : false,
            "description" : "The URI of the corresponding to the member resource
 of this Group.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "immutable",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "type",
            "type" : "string",
            "multiValued" : false,
            "description" : "A label indicating the type of resource; e.g., 'Use
r' or 'Group'.",
            "required" : false,
            "caseExact" : false,
            "canonicalValues" : [
              "User",
              "Group"
            ],
            "mutability" : "immutable",
            "returned" : "default",
            "uniqueness" : "none"
          }
        ],
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      }
    ],
    "meta" : {
      "resourceType" : "Schema",
      "created" : "2010-01-23T04:56:22Z",
      "lastModified" : "2014-02-04T00:00:00Z",
```

```
      "version" : "W/\"3694e05e9dff596\"",
      "location" : "https://example.com/v2/Schemas/urn:scim:schemas:core:2.0:Gro
up"
    }
  },
  {
    "id" : "urn:scim:schemas:extension:enterprise:2.0:User",
    "name" : "EnterpriseUser",
    "description" : "Enterprise User",
    "attributes" : [
      {
        "name" : "employeeNumber",
        "type" : "string",
        "multiValued" : false,
        "description" : "Numeric or alphanumeric identifier assigned to a person
, typically based on order of hire or association with an organization.",
        "required" : false,
        "caseExact" : false,
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      },
      {
        "name" : "costCenter",
        "type" : "string",
        "multiValued" : false,
        "description" : "Identifies the name of a cost center.",
        "required" : false,
        "caseExact" : false,
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      },
      {
        "name" : "organization",
        "type" : "string",
        "multiValued" : false,
        "description" : "Identifies the name of an organization.",
        "required" : false,
        "caseExact" : false,
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      },
      {
        "name" : "division",
        "type" : "string",
        "multiValued" : false,
        "description" : "Identifies the name of a division.",
        "required" : false,
```

```
        "caseExact" : false,
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      },
      {
        "name" : "department",
        "type" : "string",
        "multiValued" : false,
        "description" : "Identifies the name of a department.",
        "required" : false,
        "caseExact" : false,
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      },
      {
        "name" : "manager",
        "type" : "complex",
        "multiValued" : false,
        "description" : "The User's manager. A complex type that optionally allo
ws Service Providers to represent organizational hierarchy by referencing the \"
id\" attribute of another User.",
        "required" : false,
        "caseExact" : false,
        "subAttributes" : [
          {
            "name" : "managerId",
            "type" : "string",
            "multiValued" : false,
            "description" : "The id of the SCIM resource representing the User's
  manager.  REQUIRED.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "$ref",
            "type" : "string",
            "multiValued" : false,
            "description" : "The URI of the SCIM resource representing the User'
s manager.  REQUIRED.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "displayName",
```

```
            "type" : "string",
            "multiValued" : false,
            "description" : "The displayName of the User's manager.  OPTIONAL an
d READ-ONLY.",
            "required" : false,
            "caseExact" : false,
            "mutability" : "readOnly",
            "returned" : "default",
            "uniqueness" : "none"
          }
        ],
        "mutability" : "readWrite",
        "returned" : "default",
        "uniqueness" : "none"
      }
    ],
    "meta" : {
      "resourceType" : "Schema",
      "created" : "2010-01-23T04:56:22Z",
      "lastModified" : "2014-02-04T00:00:00Z",
      "version" : "W/\"3694e05e9dff596\"",
      "location" : "https://example.com/v2/Schemas/urn:scim:schemas:extension:en
terprise:2.0:User"
    }
  }
]}
```


13. Security Considerations

The SCIM Core schema contains personally identifiable information as well as other sensitive data. Aside from prohibiting password values in a SCIM response this specification does not provide any means or guarantee of confidentiality.

14. References

14.1. Normative References

[ISO3166] "ISO 3166:1988 (E/F) - Codes for the representation of names of countries - The International Organization for Standardization, 3rd edition", 08 1988.

[ISO639-2] ISO 639.2 Registration Authority, "ISO639-2: Codes for the Representation of Names of Languages", July 2013.

[Olson-TZ] "Sources for Time Zone and Daylight Saving Time Data", .

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC3966] Schulzrinne, H., "The tel URI for Telephone Numbers", RFC 3966, December 2004.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, January 2005.

[RFC4627] Crockford, D., "The application/json Media Type for JavaScript Object Notation (JSON)", RFC 4627, July 2006.

[XML-Schema] Biron, P. and A. Malhotra, "XML Schema Part 2: Datatypes Second Edition", October 2004.

14.2. Informative References

[PortableContacts] Smarr, J., "Portable Contacts 1.0 Draft C - Schema Only", August 2008.

[RFC4512] Zeilenga, K., "Lightweight Directory Access Protocol (LDAP): Directory Information Models", RFC 4512, June 2006.

Appendix A. Acknowledgements

The editors would like to acknowledge the contribution and work of the past draft editors:

Chuck Mortimore, Salesforce

Patrick Harding, Ping

Paul Madsen, Ping

Trey Drake, UnboundID

The SCIM Community would like to thank the following people for the work they've done in the research, formulation, drafting, editing, and support of this specification.

Morteza Ansari (morteza.ansari@cisco.com)

Sidharth Choudhury (schoudhury@salesforce.com)

Samuel Erdtman (samuel@erdtman.se)

Kelly Grizzle (kelly.grizzle@sailpoint.com)

Chris Phillips (cjphillips@gmail.com)

Erik Wahlstroem (erik.wahlstrom@nexussafe.com)

Phil Hunt (phil.hunt@yahoo.com)

Special thanks to Joeseph Smarr, who's excellent work on the Portable Contacts Specification [PortableContacts] provided a basis for the SCIM schema structure and text.

Appendix B. Change Log

[[This section to be removed prior to publication as an RFC]]

Draft 02 - KG - Addition of schema extensibility

Draft 03 - PH - Revisions based on following tickets:

09 - Attribute uniquenes

10 - Returnability of attributes

35 - Attribute mutability (replaces readOnly)

52 - Minor textual changes

53 - Standard use of term client (some was consumer)

56 - Make manager attribute consistent with other $ref attrs

58 - Add optional id to ResourceType objects for consistency

59 - Fix capitalization per IETF editor practices

60 - Changed <eref> tags to normal <xref> and <reference> tags

Authors' Addresses

Kelly Grizzle
SailPoint

Email: kelly.grizzle@sailpoint.com

Phil Hunt (editor)
Oracle Corporation

Email: phil.hunt@yahoo.com

Erik Wahlstroem
Technology Nexus

Email: erik.wahlstrom@nexussafe.com

Chuck Mortimore
Salesforce.com

Email: cmortimore@salesforce.com

SCIM WG
Internet-Draft
Intended status: Informational
Expires: September 5, 2014

P. Hunt
Oracle
B. Khasnabish
ZTE USA,Inc.
A. Nadalin
Microsoft
K. Li
Huawei
Z. Zeltsan
Individual

March 4, 2014

SCIM Use Cases
draft-ietf-scim-use-cases-01

Abstract

This document lists the user scenarios and use cases of System for Cross-domain Identity Management (SCIM).

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 September 5, 2014.

Copyright Notice

Table of Contents

1. Introduction

This document describes the SCIM scenarios and use cases. It also provides a list of the requirements derived from the use cases. The document's objective is to help with understanding of the design and applicability of SCIM schema [I-D.ietf-scim-core-schema] and SCIM protocol [I-D.ietf-scim-api].

The following section provides the abbreviated descriptions of the scenarios and use cases.

1.1. Terminology

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] when they appear in ALL CAPS. These words may also appear in this document in lower case as plain English words, absent their normative meanings.

Here is a list of acronyms and abbreviations used in this document:

- COI: Community Of Interest
- CRM: Customer Relationship Management
- CRUD: Create Read Update Delete
- CSP: Cloud Service Provider
- CSU: Cloud Service User
- ECS: Enterprise Cloud Subscriber
- IaaS: Infrastructure as a Service
- JIT: Just In Time
- PaaS: Platform as a Service
- SaaS: Software as a Service
- SAML: Security Assertion Markup Language
- SCIM: System for Cross-domain Identity Management
- SSO: Single-Sign On

2. SCIM User Scenarios

2.1. Background & Context

The System for Cross-domain Identity Management (SCIM) specification is designed to make managing user identity in cloud based applications and services easier. The specification suite seeks to build upon experience with existing schemas and deployments, placing specific emphasis on simplicity of development and integration, while applying existing authentication, authorization, and privacy models. It's intent is to reduce the cost and complexity of user management operations by providing a common user schema and extension model, as well as binding documents to provide patterns for exchanging this schema using standard protocols. In essence, make it fast, cheap, and easy to move users in to, out of, and around the cloud.

The SCIM scenarios are overview user stories designed to help clarify the intended scope of the SCIM effort.

2.2. Model Concepts

2.2.1. Triggers

Quite simply, triggers are actions or activities that start SCIM flows. Triggers may not be relevant at the protocol or the schema, they really serve to help identity the type or activity that resulted in a SCIM protocol exchange. Triggers make use of the traditional provisioning C.R.U.D (Create Read Update & Delete) operations but add additional use case contexts like "SSO" as it is designed to capture a class of use case that makes sense to the actor requesting it rather than to describe a protocol operation.

- Create SCIM Identity Resource - Service On-boarding Trigger: A create SCIM resource trigger is a service on-boarding activity in which a business action such as a new hire or new service subscription is initiated by one of the SCIM Actors. In the protocol itself, service on-boarding may well be implemented via the same resource PUT method as a service change. This is particular to the implementation not to the use cases that drive that implementation.

- Update SCIM Identity Resource - Service Change Trigger: An Update SCIM resource trigger is a service change activity as a result of an identity moving or changing its service level. An Update Identity trigger might be the result of a change in a service subscription level or a change to key identity data used to denote a service subscription level. Password changes are specifically

called out from other more general identity attribute changes as they are considered to have specific use case differences.

- Delete SCIM Identity Resource - Service Termination Trigger: A delete SCIM resource trigger represents a specific and deliberate action to remove an identity from a given SCIM service point. At this stage it is unclear if the SCIM protocol needs to identify separate protocol exchange for a service suspension actions. This may be relevant as target services usually differentiate between these result and may require separate resource representations as a result.

- Single-Sign On (SSO) Trigger - Real-time Service Access Request: A SSO trigger is a special class of activity in which a Create or Update trigger is initiated during an SSO operational flow. The implication here is that as the result of a real-time service access request by the end user (SSO), defined SCIM protocol exchanges can be used to initiate SCIM resource CRUD somewhere in the service cloud.

2.2.2. Actors

Actors are the operating parties that take part in both sides of a SCIM protocol exchange, and help identify the source of a given Trigger. So far, we have identified the following SCIM Actors:

- Cloud Service Provider (CSP): A CSP is the entity operating a given cloud service. In a SaaS scenario this is simply the application provider. In an IaaS or PaaS scenario, the CSP may be the underlying IaaS/PaaS infrastructure provider or the owner of the application running on that platform. In all cases, the CSP is the thing that holds the identity information being operated upon. Put another way, the CSP really is the service that the end-end user interacts with.

- Enterprise Cloud Subscriber (ECS): An ECS represents a middle-tier of aggregation for related identity records. In one of our sample enterprise SaaS scenarios, the ECS is "FooBar.Inc" that subscribes to a cloud based CRM service service "SaaS-CRM.Inc" (the CSP) for all of its sales staff. The actual Cloud Service Users (CSUs) are the FooBar.Inc. sales staff. The ECS actor is identified to help capture use cases in which a single entitle is given administrative responsibility for other identity accounts. SCIM may not address the configuration and setup of an ECS within the CSP, but it does address use cases in which SCIM identity resources are grouped together and administers as part of some broader agreement or operational exchange.

o Cloud Service User (CSU): A CSU represents the real cloud service end-end user - the "person logging into and using the cloud service". As described above, and ECS will typically own or manage multiple CSU identities where as the CSU represents the FooBar.Inc. employee using the cloud service to manage their CRM process.

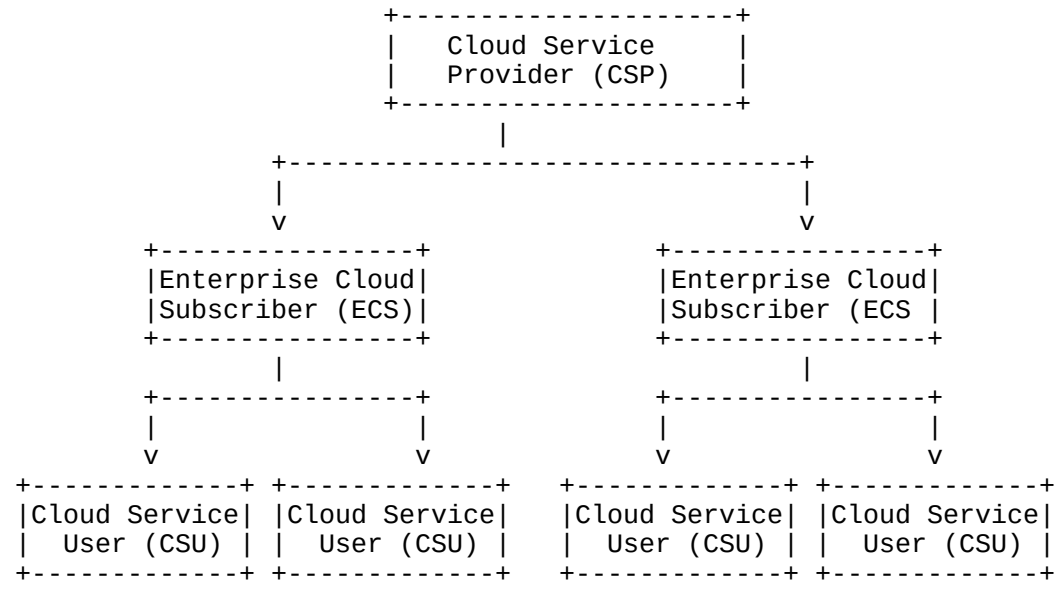

Figure 1: SCIM Actors

2.2.3. Modes & Flows

Modes identify the functional intent of a data-flow initiated in a SCIM scenario. The modes identified so far are 'push' and 'pull' referring to the fact of pushing data to, or pulling data from an authoritative identity data store.

In the SCIM scenarios, Modes are often used in the context of a flow between two Actors. For example, one might refer to a Cloud-to-Cloud Pull exchange. Here one Cloud Service Provider (CSP) is pulling identity information from another CSP. Commonly referenced flows are:

o Cloud Service Provider to Cloud Service Provider (CSP->CSP)

o Enterprise Cloud Subscriber to Cloud Service Provider (ECS-CSP)

Modes & flows simply help us understand what is taking place; they are likely to be technically meaningless at the protocol level, but again they help the reader follow the SCIM scenarios and apply them to real work use cases.

2.2.4. Bulk & Batch Operational Semantics

It is assumed that each of the triggers action outlined in this document may be part of the larger bulk or batch operation. Individual SCIM actions should be able to be collected together to create single protocol exchanges.

The initial focus of SCIM scenarios is on identifying base flows and single operations. The specific complexity of full bulk and batch operations is left to a later version of the scenarios or to the main specification.

2.3. Cloud Service Provider to Cloud Service Provider Flows (CSP->CSP)

These scenarios represent flows between two Cloud Service Providers (CSPs). It is assumed that each CSP maintains an Identity Data Store for its Cloud Service Users (CSUs). These scenarios address various joiner, mover, leaver and JIT triggers, resulting in push and pull data exchanges between the CSPs.

2.3.1. CSP->CSP - Create Identity (Push)

In this scenario two CSPs (CSP-1 & CSP-2) have a shared service agreement in place that requires the exchange of Cloud Service User (CSU) accounts. CSP-1 receives a Create Identity trigger action from its Enterprise Cloud Subscriber (ECS-1). CSP-1 creates a local user account for the new CSU. CSP-1 then pushes the new CSU joiner push request down-stream to CSU-2 and gets confirmation that the account was successfully created. After receiving the confirmation from CSP-2, CSP-1 sends an acknowledgement to the requesting ECS.

2.3.2. CSP->CSP - Update Identity (Push)

In this scenario two CSPs (CSP-1 & CSP-2) have a shared service agreement in place that requires the exchange of Cloud Service User (CSU) accounts. The Enterprise Cloud Subscriber (ECS-1) has already created an account with CSP-1 and supplied a critical attribute "department" that is used by CSP-1 to drive service options. CSP-1 then receives an Update Identity trigger action from its Enterprise Cloud Subscriber (ECS). CSP-1 updates its local directory account with the new department value. CSP-1 then initiates a separate SCIM protocol exchange to push the mover change request down-stream to

CSP-2. After receiving the confirmation from CSP-2, CSP-1 sends an acknowledgment to ECS-1.

2.3.3. CSP->CSP - Delete Identity (Push)

In this scenario two CSPs (CSP-1 & CSP-2) have a shared service agreement in place that requires the exchange of Cloud Service User (CSU) accounts. CSP-1 receives a Delete Identity trigger action from its Enterprise Cloud Subscriber (ECS-1). CSP-1 suspends the local directory account for the specified CSU account. CSP-1 then pushes a termination request for the specified CSU account down-stream to CSP-2 and gets confirmation that the account was successfully removed. After receiving the confirmation from CSP-2, CSP-1 sends an acknowledgment to the requesting ECS.

This use case highlights how different CSPs may implement different operational semantics behind the same SCIM operation. Note CSP-1 suspends the account representation for its service where as CPS-2 implements a true delete operation.

2.3.4. CSP->CSP - SSO Trigger (Push)

In this scenario two CSPs (CSP-1 & CSP-2) have a shared service agreement in place that requires the exchange of Cloud Service User (CSU) accounts. However, rather than pre-provisioning accounts from CSP-1 to CSP-2, CSP-1 waits for a service access request from the end Cloud Service User (CSU-1) before issuing account creation details to CSP-2. When the CSU completes a SSO transaction from CSP-1 to CSP-2, CSP-2 then creates an account for the CSU based on information pushed to it from CSP-1.

At the protocol level, this class of scenarios may result in the use of common protocol exchange patters between CSP-1 & CSP-2.

2.3.5. CSP->CSP - SSO Trigger (Pull)

In this scenario two CSPs (CSP-1 & CSP-2) have a shared service agreement in place that requires the exchange of Cloud Service User (CSU) accounts. However, rather than pre-provisioning accounts from CSP-1 to CSP-2, CSP-2 waits for a service access request from the Cloud Service User (CSU-1) before initiating a Pull request to gather information about the CSU sufficient to create a local account.

At the protocol level, this class of scenarios may result in the use of common protocol exchange patterns between CSP-2 & CSP-1.

2.3.6. CSP->CSP - Password Reset (Push)

In this scenario two CSPs (CSP-1 & CSP-2) have a shared service agreement in place that requires the exchange of Cloud Service User (CSU) accounts. CSP-1 wants to change the password for a specific Cloud Service User (CSU-1). CSP-1 sends a request to CSP-2 to reset the password value for CSU-1.

At the protocol level, this scenario may result in the same protocol exchange as any other attribute change request.

2.4. Enterprise Cloud Subscriber to Cloud Service Provider Flows(ECS->CSP)

These scenarios represent flows between an Enterprise Cloud Subscriber (ECS) and a Cloud Service Providers (CSP). It is assumed that both the ECS and the CSP maintains an LDAP service for the relevant Cloud Service Users (CSUs). These scenarios address various joiner, mover, leaver and JIT triggers, resulting in push and pull data exchanges between the ECS and the CSP.

Many of these scenarios are very similar to those defined in the Cloud Service Provider to Cloud Service Provider section above. They are identified separately here so that we may explore any differences and might emerge.

2.4.1. ECS->CSP - Create Identity (Push)

In this scenario an Enterprise Cloud Subscriber (ECS-1) maintains a service with a Cloud Service Provider (CSP-1) that requires the sharing of various Cloud Service User (CSU) accounts. A new user joins ECS-1 and so ECS-1 pushes an account creation request to CSP-1, supplying all required base SCIM schema attribute values and additional extended SCIM schema values as required.

2.4.2. ECS ->CSP - Update Identity (Push)

In this scenario an Enterprise Cloud Subscriber (ECS-1) maintains a service with Cloud Service Provider (CSP-1) that drives service definition from a key account schema attribute called Department. ECS-1 wishes to move a given CSU from Department A to Department B and so it pushes an attribute update request to the CSP.

2.4.3. ECS ->CSP - Delete Identity (Push)

In this scenario an Enterprise Cloud Subscriber (ECS-1) maintains a service with a Cloud Service Provider (CSP-1). Upon termination of one of its employees' employment agreement, ECS-1 sends a suspend

account request to CSP-1 (Figure 1.4.3-1). One week later the ECS wishes to complete the process by fully removing the Cloud Service User (CSU) account and so it sends a terminate account request to CSP-1.

2.4.4. ECS ->CSP - SSO Pull

In this scenario an Enterprise Cloud Subscriber (ECS-1) maintains a service with a Cloud Service Provider (CSP-1). No accounts are created or exchanged in advance. However, rather than pre-provisioning accounts from ECS-1 to CSP-1, CSP-1 waits for a service access request from the Cloud Service User (CSU-1) under the control domain of ECS-1, before issuing an account Pull request to CSP-1.

3. SCIM use cases

This section lists the SCIM use cases.

3.1. Change of the ownership of a file

Description:

Bob - an employee of the company SomeEnterprise - creates a file, which is located at the cloud provided by SomeCSP. After Bob leaves SomeEnterprise, SomeCSP on a request from SomeEnterprise terminates Bob’s rights to the file and transfers his former rights to Bill - another employee of SomeEnterprise.

Pre-conditions:

- SomeCSP is a cloud service provider for SomeEnterprise
- With permission of SomeEnterprise, Bob had created a file at the cloud provided by SomeCSP
- Bob has left SomeEnterprise
- SomeEnterprise terminates Bob’s rights to the file and, possibly, decommissions Bob’s identity
- SomeEnterprise communicates the changes to Bob’s rights to SomeCSP
- SomeCSP enforces the changes made by SomeEnterprise
- SomeEnterprise requests SomeCSP to transfer Bob’s former rights to Bill

Post-conditions:

- Bob does not have the rights to the file at the cloud provided by SomeCSP
- Bill has the rights to the file that Bob had had

Requirements:

- SomeEnterprise can securely communicate to SomeCSP all changes regarding its employee’s identity
- SomeCSP can enforce the requested changes
- SomeCSP shall be able to log all changes regarding a SomeEnterprise employee’s identity
- The logs should be secure and available for auditing

3.2. Migration of the identities

Description:

A company SomeEnterprise runs an application ManageThem that relies on the identity information about its employees (e.g., identifiers, attributes). The identity information is stored at the cloud provided by SomeCSP. SomeEnterprise has decided to move identity information to the cloud of a different provider - AnotherCSP. In addition, SomeEnterprise has purchased a second application ManageThemMore, which also relies on the identity information. SomeEnterprise is able to move identity information to AnotherCSP without changing the format of identity information. The application ManageThemMore is able to use the identity information.

Pre-conditions:

- SomeCSP is a cloud service provider for SomeEnterprise
- SomeCSP has a known attribute name and value for the Enterprise used for managing and transferring data
- AnotherCSP is a new cloud service provider for SomeEnterprise
- All involved cloud service providers and applications support the same standard specifying the format for and actions on the user (e.g., employee) identity information

Post-conditions:

- SomeEnterprise has moved its employees' identity information from SomeCSP to AnotherCSP without making any changes to representation of identity information
- Application ManageThemMore is able to use the identity information

Requirements:

- SomeEnterprise, the applications ManageThem and ManageThemMore, the providers SomeCSP and AnotherCSP support a common standard for identity information, which specifies the following:
  - Format (or schema) for representing user identity information
  - Interfaces and protocol for managing user identity information
- Cloud providers shall be able to log all actions related to SomeEnterprise employees' identities
- The logs should be secure and available for auditing

3.3. Single Sign-On (SSO) Service

Description:

Bob has an account with application hosted by a cloud service provider SomeCSP. SomeCSP has federated its user identities with a cloud service provider AnotherCSP. Bob requests a service from an application running on AnotherCSP. The application running on AnotherCSP, relying on Bob's authentication by SomeCSP and using identity information provided by SomeCSP, serves Bob's request.

Pre-conditions:

- Bob's identity information is stored on SomeCSP
- SomeCSP and AnotherCSP have established trust and federated their user identities
- SomeCSP is able to authenticate Bob
- SomeCSP is able to securely provide the authentication results to AnotherCSP
- SomeCSP is able to securely provide Bob's identity information (e.g., attributes) to AnotherCSP
- AnotherCSP is able to verify information provided by SomeCSP

- SomeCSP is able to process the identity information received from AnotherCSP

Post-conditions:

Bob has received the requested service from an application running on AnotherCSP without having to authenticate to that application explicitly.

Requirements:

- Bob must have an account with SomeCSP
- SomeCSP and AnotherCSP must establish trust and federate their user identities
- SomeCSP must be able to authenticate Bob
- SomeCSP must be able to securely provide the authentication results to AnotherCSP
- SomeCSP must be able to securely provide Bob’s identity information (e.g., attributes) to AnotherCSP
- AnotherCSP must be able to verify the identity information provided by SomeCSP
- SomeCSP must be able to process the identity information received from AnotherCSP
- SomeCSP and AnotherCSP must log information generated by Bob’s actions according to their policies and the trust agreement between them

3.4. Provisioning of the user accounts for a Community of Interest (CoI)

Description:

Organization YourHR provides Human Resources (HR) services to a Community of Interest (CoI) YourCoI. The HR services are offered as Software-as-a-Service (SaaS) on public and private clouds. YourCoI’s offices are located all over the world. Their Information Technology (IT) systems may be composed of the combinations of the applications running on Private and Public clouds along with the traditional IT systems. The local YourCoI offices are responsible for establishing personal information and (i.e., setting the user identities and attributes). YourHR services provide means for provisioning and

distributing the employee identity information across all YourCoI offices. YourHR also enables the individual users (e.g., employees) to manage their personal information that they are responsible for (e.g., update of an address or a telephone number).

Pre-conditions:

- YourCoI has a complex infrastructure composed of the large number of local offices that rely on the diverse IT systems
- YourCoI has contracted YourHR to provide the HR services
- Each local office has a right to establish a personal account for an employee

Post-conditions:

- All personal accounts are globally available to any authorized user or application across the YourCoI system through the services provided by YourHR
- The employees have ability to manage the part of personal information that is in their responsibility

Requirements:

- YourHR must ensure that the personal information generated by the local offices is timely available in a globally-accessible database
- Identity management of the personal data must be secure
- All operation with identity data must be securely logged
- The logs should be available for auditing

3.5. Transfer of attributes to a relying party web site

Description:

An end user has an account in a directory service A with one or more attributes. That user then visits relying party web site B, and the user authorizes the transfer of data via authorization protocols (e.g. OAuth, SAML), so selected attributes of the user are transferred from the user’s account in directory service A to the web site B at the time of the user’s first visit to that site.

Pre-conditions:

- User has an account in a directory service A
- User has one or more attributes
- User visits web site of a relying party B

Post-conditions:

Selected attributes of the user are transferred from the user's account in directory service A to the web site B at the time of the user's first visit to that site.

Requirements:

Relying parties have to be aware of changes to their cached copy, as these would potentially cause a state change in other relying parties.

3.6. Change notification

Description:

An end user has an account in a directory service A with one or more attributes. That user then visits relying party web site B. Relying party web site B queries directory service A for attributes associated with that user, and related resources.

The attributes of the user change later in directory service A. For example, the attributes might change if the user changes their name, has their account disabled, or terminates their relationship with directory service A. Furthermore, other resources and their attributes might also change. The directory service A then wishes to notify relying party web site B of these changes, as relying party B might (or might not) have a cache of those attributes, and if the relying party B were aware of these changes to their cached copy, would potentially cause a state change in relying party B.

The volume of changes, however, might be substantial, and only some of the changes may be of interest to relying party B, so directory service A does not wish to "push" all the changes to B. Instead, directory service A wishes to notify B that there are changes potentially of interest, such that B can at an appropriate time subsequently contact directory service A and retrieve just the subset of changes of interest to B.

Note that the user must authorize the directory A service to transfer data to the web site, and the user must authorize the directory A service to notify the web site.

Pre-conditions:

- User has an account in a directory service A
- User has one or more attributes
- User visits relying party web site B
- The resource being updated is at the web site

Post-conditions:

Service A is able to notify B that there are changes potentially of interest.

Requirements:

B must be able at an appropriate time to subsequently contact directory service A and retrieve just the subset of changes of interest to B.

4. Security considerations

Authorization and authentication must be guaranteed for the SCIM operations.

5. IANA considerations

This Internet Draft includes no request to IANA.

6. Acknowledgements

Authors would like to thank Ray Counterman, Richard Fiekowsky and Bert Greevenbosch for their reviews and comments.

Also thanks to Darran Rolls and Patrick Harding, the SCIM user scenarios section is taken from them.

7. References

7.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

7.2. Informative References

[I-D.ietf-scim-api]
Grizzle, K., Hunt, P., Ansari, M., Wahlstroem, E., and C. Mortimore, "System for Cross-Domain Identity Management:Protocol", draft-ietf-scim-api-03 (work in progress), February 2014.

[I-D.ietf-scim-core-schema]
Grizzle, K., Hunt, P., Wahlstroem, E., and C. Mortimore, "System for Cross-Domain Identity Management: Core Schema", draft-ietf-scim-core-schema-03 (work in progress), February 2014.

Authors' Addresses

Phil Hunt
Oracle

Email: phil.hunt@oracle.com

Bhumip Khasnabish
ZTE USA,Inc.

Phone: +001-781-752-8003
Email: vumip1@gmail.com, bhumip.khasnabish@zteusa.com

Anthony Nadalin
Microsoft

Email: tonynad@microsoft.com

Kepeng LI
Huawei
Bantian
Shenzhen, Guangdong 518129
China

Email: likepeng@huawei.com

Zachary Zeltsan
Individual

Email: Zachary.Zeltsan@gmail.com

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

M. Wahl
Microsoft

October 18, 2013

SCIM Profile For Enhancing Just-In-Time Provisioning
draft-wahl-scim-jit-profile-01.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.

- 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 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.

- 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.

- 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.

- 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 false, 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 false.

- 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:

- 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)

- 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

- 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 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

{
  "totalResults":1,
  "schemas":["urn:scim:schemas:core:2.0:ListResponse"],
  "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 structure with a "schemas" key, and one or more keys such as "userName", "displayName", "active" or "name". 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: 72

{
  "schemas": ["urn:scim:schemas:core:2.0:User"],
  "displayName": "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] Drake, T., Mortimore, C., Ansari, M., Grizzle, K., Wahlstroem, E., "System for Cross-Domain Identity Management:Protocol", draft-ietf-scim-api-02, August 2013.

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

[3] Mortimore, C., Harding, P., Madsen, P., Drake, T., "System for Cross-Domain Identity Management:Core Schema", draft-ietf-scim-core-schema-02, August 2013.

[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-00, August 2013.

[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_0-28.html , July 2013.

[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