SCIM Ticket #50
“Filter semantics for complex plural attributes”

Summary for IETF-88

Vancouver, November 2013

Issue #50

How to write a search filter like to find users whose primary email address contains “@example.com”

```
filter=emails co "_@example.com"
and
emails.primary eq "true"
```

```
<emails>
     <email>
          <value> bjensen@jensen.com</value>
          <type>work</type>
          <primary>true</primary>
     </email>
     <email>
          <value> babs@example.com</value>
          <type>home</type>
     </email>
</emails>
```

When:

- Email is a multi-valued attribute
- The value “@example.com” and “primary” are different sub-attributes
- Thus the “emails.” value instance matched may be different in the two clauses, leading to unexpected results.

- Realization: SCIM filters need a syntax for indicating that two or more criteria apply to the same instance of a multi-valued attribute

Options to Consider

1. Use a filter dot notation
2. Use a filter operator
3. Use a back reference
4. Use structured JSON filter definitions
5. Something else…

Each has both pros and cons

1. Use a Filter dot Notation

- Proposed by Jeff Moore
- Use the attribute name plus [ ] with . to group clauses that refer to the same attribute instance:

```
emails[.type eq "work" and .value co "@example.com"]

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

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

- If the dot "." is missing - then it would be an absolute reference.
- Could also have ".." to move scope up one.

2. Use the instance() Filter Operator

- Proposed by ?
- Use a grouping operator for clauses that refer to the same attribute instance:

```
instance(emails.type eq "work" and emails.value co “@example.com“)
or
instance(emails.type eq “home" and emails.primary eq true)

instance(addresses.state eq "CA" and
  instance(addresses.rooms.type eq "bedroom" and addresses.room.number gt 2)
)
```

- Nesting is possible
- Is it ok to refer to other attributes within the instance( ) ?

3. Use a Back-Reference

- Proposed by Bjorn Aannestad
- Use the attribute name plus a “%1” to indicate the same instance

```
emails%1.type eq "work" and emails%1.value co “@example.com”

emails%1.type eq "work"
and emails%1.value co “@example.com” or emails%2.primary eq true

emails%1.type eq “home”
and emails%2.ims.type eq "xmpp" and emails%2 co @example.com

addresses%1.state eq "CA" and addresses%1.rooms%2.type eq
"bedroom" and rooms%2.number gt 2
```

- Supports criteria involving multiple, different, instances of an attribute
- Terms in the filter do not have to be adjacent

4. Use a JSON Filter Structure

- Many examples (e.g. MetaWeb Query, Jaql)
- http://mql.freebaseapps.com/ch03.html

Query	Result
{ "type" : "/music/album", "artist": "The Police", "name" : "Synchronicity", "id" : null }	{ "type": "/music/album", "artist": "The Police", "name": "Synchronicity", "id": "/guid/9202a8c04000641f8 }

Query	Result
{ "type" : "/music/artist", "name" : "The Police", "album" : { "name" : "Synchronicity", "track" : [] } }	{ "type": "/music/artist", "name": "The Police", "album": { "name": "Synchronicity", "track": ["Synchronicity II", "Every Breath You Take", "King of Pain", "Wrapped Around Your Finger", "Tea in the Sahara", "Walking in Your Footsteps", "Miss Gradenko", "Murder by Numbers", "O My God", "Synchronicity I", "Mother"] } }

Comparison

	Complete?	Concise?	Usable in URL?	Clauses can be separated?	Handles Multiple Instances & Nesting	Precedent?
1. Filter dot Notation	Y	Y	Y	N	Y	
2. Instance( ) operator	Y	Y	Y	N	Y	
3. Back-reference variable %n	Y	Y	w/escape	Y	Y	e.g. RegEx
4. JSON Structure	Y	No	No		Y	e.g. MetaWeb, Jaql

Questions and Discussion