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"

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:

```plaintext
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:

\[
\text{instance(emails.type eq "work" and emails.value co "@example.com") or instance(emails.type eq "home" and emails.primary eq true)}
\]

\[
\text{instance(addresses.state eq "CA" and instance(addresses.rooms.type eq "bedroom" and addresses.room.number gt 2))}
\]

- 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

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

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

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

```plaintext
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](http://mql.freebaseapps.com/ch03.html)
## Comparison

<table>
<thead>
<tr>
<th></th>
<th>Complete?</th>
<th>Concise?</th>
<th>Usable in URL?</th>
<th>Clauses can be separated?</th>
<th>Handles Multiple Instances &amp; Nesting</th>
<th>Precedent?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. Filter dot Notation</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>2. Instance( ) operator</td>
<td>Y</td>
<td>Y</td>
<td>Y</td>
<td>N</td>
<td>Y</td>
<td>Y</td>
</tr>
<tr>
<td>3. Back-reference variable %n</td>
<td>Y</td>
<td>Y</td>
<td>w/escape</td>
<td>Y</td>
<td>Y</td>
<td>e.g. RegEx</td>
</tr>
<tr>
<td>4. JSON Structure</td>
<td>Y</td>
<td>No</td>
<td>No</td>
<td>Y</td>
<td></td>
<td>e.g. MetaWeb, Jaql</td>
</tr>
</tbody>
</table>
Questions and Discussion