Internet-Draft
draft-moats-ldap-dereference-match-02.txt
Expires in six months
Category: Experimental Track

Ryan Moats
Jerry Maziarski
AT&T
John Strassner
cisco Systems
December 1999

Extensible Match Rule to Dereference Pointers

Filename: draft-moats-ldap-dereference-match-02.txt

Status of this Memo

Abstract

This document defines a LDAPv3 extensible matching rule that allows a server to dereference pointers stored in an object's attribute and apply a LDAPv3 search filter to the resulting objects. This rule allows schema definitions to capture richer association models without requiring extra protocol exchanges or special client code.

1. Introduction

When mapping rich information models, it sometimes becomes necessary to use DN pointers to show relationships between objects in the schema. An example is the information model and resulting core schema that is currently being proposed by the policy working group (see [1]).

To maintain client efficiency, it is desirable to define an

extensible match rule that follows DN pointers as part of a query.

2. dereferencingMatch

A server will advertise support for this matching rule by having the following rule definition present in its root subschema subentry.

```
( <oid-m1> NAME "dereferencingMatch"
  DESC "Extended match that dereferences before searching"
  SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
)
```

This extensibleMatch filter is used by a client presenting <oid-m1> as the matchingRule, any attribute with DN syntax as the type and a string representation of a LDAP search filter as the value. The server first collects the objects that the attribute points to and the client has permission to read and then applies the specified LDAP filter to them, returning the objects that were matched.

For example, a client that presented the following filter:

```
(targetDN:<oid-m1>:=(&(objectClass=cimActiveConnection)(trafficType=2)))
```

The server would apply the filter specified to all objects referenced to by the values of the targetDN attribute of the current object. It would then return the requested attributes of the objects that matched the specified filter.

If the LDAP filter itself contains a dereferencingMatch rule, it is possible to do double dereferencing. The following filter causes the server to first apply the embedded filter (trafficType=2) to objects pointed to by the cimActiveConnectionRefs attribute of the base object. Requested attributes of all objects pointed to by the cimProtocolEndpointsRef attribute of objects that passed the first filter are then returned to the client.

```
(cimProtocolEndpointsRef:<oid-m1>:=
 (cimActiveConnectionRefs:<oid-m1>:=(trafficType=2)))
```

In cases where a client does not have permission to access an object pointed to by a reference, that object is not placed on the list.

3. Considerations

3.1 Advertisement

A server implementing this extended match rule MUST include the OID of this matching rule in the subschema entry of the root DSE, per RFCs 2251 [2].

3.2 Non-supporting Server Reply

Servers that do not support this extension MUST follow section 7.8 of X.511 [3]. This is consistent with RFC 2251.

3.3 Interaction with sizelimit

During the application of this search filter, intermediate collections of entries will result. If the number of entries in these intermediate collections exceed the server's size limit, the server MUST respond with size limit exceeded error.

3.4 Search scoping considerations

Because of the possibility of size limit overload and the loss of relationship between the returned objects and the source object during one-level or subtree searches, it is strongly recommended that this match rule only be used with base level searches.

3.5 Combining with other search filters

Because this extensible matching rule is intended to result in objects and not a true/false value, rules for combination with other filters are necessary. When combining filters, all non-dereferencing match filters at the level of the dereferencing-match are applied to the current object first. Then, any remaining objects have the dereferencing match applied to it.

This rule means that it is illegal to have two dereferencing match filters at the same level. The only legal combination is for a dereferencing match filter to "include" another dereferencing match filter. This would allow a multi-step pointer dereferencing.

Also, it is illegal for a boolean filter to be at a level "above" the dereferencing match, because of scope conflicts.

To help clarify this rule, the following examples are provided:

```
  &(targetDN:<oid-m1>:=(&(objectClass=cimActiveConnection)
                          (trafficType=2)))
   (sourceDN:<oid-m1>:=(objectClass=foobar)
```

This filter is illegal because it specifies two dereferencing match rules at the same level.

```
  (targetDN:<oid-m1>:=(sourceDN:<oid-m1>:=(objectClass=*)))
```

This filter is legal and returns all objects pointed to by the sourceDN attribute of all objects pointed to by the targetDN

attribute of all objects in the search scope.

```
  &(objectClass=fancyconnectiontype)
   (targetDN:<oid-m1>:=(&(objectClass=cimActiveConnection)
                         (trafficType=2)))
```

This filter is legal, and would return all objects pointed to by the targetDN attribute of objects whose objectClass attribute was equal to fancyconnectiontype that had their objectClass attribute equal to cimActiveConnection and their trafficType attribute equal to 2.

```
  |(foo=bar)
   (foo=barshelf)
   (&(objectClass=fancyconnectiontype)
     (targetDN:<oid-m1>:=(&(objectClass=cimActiveConnection)
                           (trafficType=2))))
```

This filter is illegal because the first two filters are at a "higher" level than the dereferencing match filter.

3.6 How to handle referrals

For a query that encounters referrals, the solution set returns answers based on the local data set only. Referrals should be returned using a rewritten query contained in a LDAP URL so that the client may submit the rewritten query to the referred machine. Without query rewriting, it would be impossible for the client to know at what stage of pointer dereferencing the referral occurred.

4. Examples of Use

We present two (admittedly simple) examples for how this rule could be used.

4.1 White Pages Example

In our first example, a directory holds white pages information, including building managers for buildings. In their schema, buildings have their own object class (objectClass: building), which contains an attribute manager that has multi-valued DN syntax. These DNs point to objects of object class person (objectClass: person) and include attributes phoneNumber and building.

To find the phone number and building of all managers of any building the following query could be used:

```
   query type: subtree
   filter: (manager:<oid-m1>:(objectClass=person))
```

```
   returned attributes: (phoneNumber, building)
```

To find the phone numbers of managers of building 12, the following query could be used:

```
   query type: subtree
   filter: (manager:<oid-m1>:((objectClass=person)&(building=12)))
   returned attributes: phoneNumber
```

4.2 Policy Example

As a more complex example, we consider the case of managing QoS policy rule at a router. We assume that the routers are modeled in the schema as objects with their own object class (objectClass: router) that include the attributes ipAddress and qosPolicyRules (a multi-valued DN attribute). The attribute qosPolicyRules points to objects with objectClass set to qosPolicyRule and include attributes QosPolicyDirection (integer) and policyRulesAuxContainedSet (multi-valued DN). This latter attribute points, in turn, to policy rules (objectClass: policyRule) with multiple attributes.

To find policy rules for outbound traffic (QosPolicyDirection = 2) for routers with IP address 192.128.170.x, the following search could be used

```
   query type: subtree
   filter: (policyRulesAuxContainedSet:<oid-m1>:
            &(ipAddress="192.128.170.*")
             (qosPolicyRules:<oid-m1>:(QosPolicyDirection=2)))
   returned attributes: all
```

4.3 Referral Example

As an example of how query-rewriting occurs during a referral, let us consider the example from section 4.2 above. It could be possible that policy rules for the routers are stored under a different portion of the directory tree that is contained in a different server (for example referral.foo.bar on port 389). In that case, the referral would point to object <dn> and the following URL would be returned:

```
   ldap://referral.foo.bar/<dn>?sub?(qosPolicyRules:<oid-m1>
       :(QosPolicyDirection=2))
```

The client could then follow this URL to continue the search.

5. Why use an extensible matching rule rather than a control?

An alternative implementation of this procedure would be to use a new control, rather than an extensible matching rule. We have considered this, but have failed to find a method for supporting multiple levels of dereferencing, which can be supported when using an extensible matching rule.

6. Security considerations

An improperly formed query can create a denial of service attack by using up excessive resources. Therefore, servers that support queries of this type should implement specific time limits that cannot be overriddent to ensure that other clients can continue to make use of the directory.

Although this type of query allows a client to request that the server collect objects before applying the search filter, it creates no additional security issues above what needs to be considered when allowing a subtree search.

7. References

Request For Comments (RFC) and Internet Draft documents are available from numerous mirror sites.

[1] J. Strassner, E. Ellesson, B. Moore, R. Moats, "Policy Framework Core Information Model," Internet Draft (work in progress), November 1999.

[2] M. Wahl, T. Howes, S. Kille, "Lightweight Directory Access Protocol (v3)," RFC 2251, December 1997.

[3] ITU-T Rec. X.511, "The Directory: Abstract Service Definition," 1993.

7. Author's Addresses

Ryan Moats
15621 Drexel Circle
Omaha, NE 68135
USA
E-mail: jayhawk@att.com

Jerry Maziarski
Room C3-3Z01
200 S. Laurel Ave.
Middletown, NJ 07748
USA
E-mail: gfm@qsun.att.com

John Strassner
Cisco Systems, Bldg 1
170 West Tasman Drive
San Jose, CA 95134
E-mail: johns@cisco.com