1. Objectives

Aliases may be used within LDAP to reference entries anywhere within the directory tree. Conceptually, an alias is simply a pointer to the DIT entry it represents. It does not contain additional information about that entry; only the location of the entry.

The behavior of the alias object within LDAP is not well-defined, both in creation of the alias object and the behavior when dereferencing the alias.

For successful interoperability, the expected behavior of servers when encountering alias objects SHOULD be consistent.
Additionally, it MUST be possible to use aliases without changing the LDAPv3 schema as defined in [Wahl] and without adding server-dependent data.

2. Schema Definition

2.1 Schema Expansion

The alias objectclass definitions presented in the LDAPv3 Schema [Wahl] are the basis for aliasing within ldap. The alias objectclass is a Structural objectclass with a single required attribute; the single valued aliasObjectName.

This definition of the alias objectclass does not allow for any attribute other than ‘aliasedObjectName’ to be used as the naming attribute within the RDN. The resulting dn for the alias object must therefore be of the form "aliasedObjectName=<dn>, <rdn>, <rdn>..." This is not a user-friendly name for a directory entry, and quite possibly corrupts the naming hierarchy within the directory tree.

In order to remain true the concept of an alias; that it is merely a pointer to another entry, an entry of objectclass alias SHOULD NOT be combined with any other objectclass. If multiple objectclasses are combined, it becomes possible to add information to the alias entry without violating the schema rules.

While not explicitly specified as either a ‘required’ or ‘may’, any naming attribute MUST be allowed to form the RDN of the alias. Restricting the possible naming attributes would potentially corrupt the hierarchy. For example, it would be impossible to distinguish between a person alias and an organisation alias.

2.2 AliasObject Objectclass

In order to create an alias object which can be appropriately named to that which it represents, the definition of the alias object MUST be expanded. A new objectclass must be defined which inherits from the current definition of alias but extends the attributes allowed within the RDN.

( 1.3.6.1.4.1.42.2.27.1.2.1
 NAME  ‘aliasObject’
 DESC  objectclass for all alias objects
 SUP   ’ALIAS’
 MAY   *
 )

The ‘*’ allows any naming attribute to be used in forming the RDN of the object.

For example, the following is a correct LDIF:

dn: cn=John Doe, ou=myOrg, c=US
objectclass: alias
objectclass: aliasObject
aliasedObjectName: cn=President, ou=myOrg, c=US
To prevent the alias from containing extra information about the object, the naming attribute SHOULD contain only a single value.

For example, the following is not a correct LDIF:

dn: cn=John Doe, ou=myOrg, c=US
objectclass: alias
objectclass: aliasObject
aliasedObjectName: cn=President, ou=myOrg, c=US
cn: John Doe
cn: Doe

Similarly, the following would not be a correct LDIF file because it adds extra information to the alias object.

dn: cn=John Doe, ou=myOrg, c=US
objectclass: alias
objectclass: aliasObject
aliasedObjectName: cn=President, ou=myOrg, c=US
cn: John Doe
title: President

The naming attribute used to form the RDN of the object SHOULD reflect the naming attribute of the referenced object. However, there are some cases where the naming attribute MAY be different.

Within the X.501 [ITU-T], the attribute used to described the aliased object is 'aliasedEntryName'. Since the OID for 'aliasedEntryName' and 'aliasedObjectName' are the same for both X.500 and LDAP, LDAP servers SHOULD treat the 'aliasedEntryName' as a synonym for 'aliasedObjectName'.

3. Alias Behavior

In general alias objects SHOULD NOT be dereferenced during any operation other than search unless requested to do so by the client.

Since an alias points to another section of the tree, it MUST NOT be possible to add an object under an alias object; alias objects MUST always be leaf nodes.

During the dereferencing of aliases, a loop is detected if the server visits the same alias entry more than once. In this case a data integrity error has occurred and the server MUST return an error of 'aliasProblem'.

If an alias is dereferenced, and the resulting directory entry does not exists, a data integrity problem has occurred, and the server MUST return an error code of 'aliasDereferencingProblem'.

If the base entry for an ldapsearch is an alias, and alias dereferencing is set to either derefFindBaseObj, or derefAlways, the base entry MUST be dereferenced before the search is performed. The new base for the search will become the entry to which the alias resolves. The search is then
If multiple aliases are chained, the alias for the first object MUST resolve to the last entry in the chain. For example, A, B, and C are alias objects. If A points to B which points to C which points to D, A resolves to D when dereferencing the alias.

If an alias is dereferenced as part of a search, the alias entry itself SHOULD NOT be returned as part of the search.

If an alias matches the search filter, and dereferencing is set to ‘searching’ or ‘always’, the dereferenced object SHOULD be returned, even if it does not match the filter.

If the alias is not dereferenced during the search, and it matches the filter, then it SHOULD be returned within the search result.

Each directory object matching a filter SHOULD be returned only once during a search. If an entry is found twice because of aliases pointing to a part of the tree already searched, the entry SHOULD NOT be returned to the client a second time.

4. Scenarios

Using the following LDIF, the scenarios would return the expected information as follows:

```
 dn: c=myCountry
c: myCountry
objectclass: country

dn: ou=Area1, c=myCountry
ou: Area1
aliasedObjectName: o=myCorporation, c=myCountry
objectclass: alias
objectclass: aliasObject

dn: o=myCorporation, c=myCountry
ou: myCorporation
objectclass: organization

dn: cn=President, o=myCorporation, c=myCountry
cn: President
aliasedObjectName: cn=John Doe, o=myCorporation, c=myCountry
objectclass: alias
objectclass: aliasObject

dn: cn=John Doe, o=myCorporation, c=myCountry
cn: John Doe
objectclass: person
```

```
c = myCountry
/
ou = Area1 ----> o = myCorporation
   |
   cn=President ---> cn = John Doe
```

Performing a base search of ‘ou = Area1, c=myCountry’ with a
filter of ‘objectclass=aliasObject’
NeverDerefAlias would return ‘ou=Area1, c=myCountry’
DerefFinding would return ‘cn=President, o=myCorporation, c=myCountry’
DerefSearching would return ‘o=myCorporation, c=myCountry’
DerefAlways would return ‘cn=John Doe, o=myCorporation, c=myCountry’

Performing a one level search of ‘c=myCountry’ with a filter of ‘ou = * ’
NeverDerefAlias would return ‘ou=Area1, c=myCountry’
DerefFinding would return ‘ou=Area1, c=myCountry’
DerefSearching would return ‘o=myCorporation, c=myCountry’
DerefAlways would return ‘ o=myCorporation, c=myCountry’

Performing a full tree search of ‘c=myCountry’ with a filter of ‘ cn = President ’
NeverDerefAlias would return ‘cn=President, o=myCorporation, c=myCountry’
DerefFinding would return ‘cn=President, o=myCorporation, c=myCountry’
DerefSearching would return ‘cn=John Doe, o=myCorporation, c=myCountry’
DerefAlways would return ‘cn=John Doe, o=myCorporation, c=myCountry’

6. Security Considerations
Permissions to dereferencing an alias, adding, deleting or returning alias entries are decided by the directory server’s ACL administration policy.

7. References


AUTHOR(S) ADDRESS

Debbie Byrne
IBM
11400 Burnet Rd
Austin, TX 78758
USA
mail-to: djbyrne@us.ibm.com
phone: +1 512 838 1930
Ludovic Poitou
Sun Microsystems
32, Chemin du vieux Chene
38240 Meylan
France
Phone: +33.(0)4.76.41.42.12
email: ludovic.poitou@france.sun.com

Ellen Stokes
IBM
11400 Burnet Rd
Austin, TX 78758
USA
mail-to: stokes@austin.ibm.com
phone: +1 512 838 3725