Network Working Group
Request for Comments: 3296
Category: Standards Track

K. Zeilenga
OpenLDAP Foundation

July 2002

# Named Subordinate References in Lightweight Directory Access Protocol (LDAP) Directories

## Status of this Memo



## Copyright Notice



## Abstract

This document details schema and protocol elements for representing and managing named subordinate references in Lightweight Directory Access Protocol (LDAP) Directories.

## Conventions

Schema definitions are provided using LDAPv3 description formats [RFC2252]. Definitions provided here are formatted (line wrapped) for readability.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" used in this document are to be interpreted as described in BCP 14 [RFC2119].

## 1. Background and Intended Usage

The broadening of interest in LDAP (Lightweight Directory Access Protocol) [RFC2251] directories beyond their use as front ends to X.500 [X.500] directories has created a need to represent knowledge information in a more general way. Knowledge information is information about one or more servers maintained in another server, used to link servers and services together.

This document details schema and protocol elements for representing and manipulating named subordinate references in LDAP directories. A referral object is used to hold subordinate reference information in

the directory. These referral objects hold one or more URIs [RFC2396] contained in values of the ref attribute type and are used to generate protocol referrals and continuations.

A control, ManageDsaIT, is defined to allow manipulation of referral and other special objects as normal objects. As the name of control implies, it is intended to be analogous to the ManageDsaIT service option described in X.511(97) [X.511].

Other forms of knowledge information are not detailed by this document. These forms may be described in subsequent documents.

This document details subordinate referral processing requirements for servers. This document does not describe protocol syntax and semantics. This is detailed in RFC 2251 [RFC2251].

This document does not detail use of subordinate knowledge references to support replicated environments nor distributed operations (e.g., chaining of operations from one server to other servers).

## 2. Schema

### 2.1. The referral Object Class

A referral object is a directory entry whose structural object class is (or is derived from) the referral object class.

```
( 2.16.840.1.113730.3.2.6
    NAME 'referral'
    DESC 'named subordinate reference object'
    STRUCTURAL
    MUST ref )
```

The referral object class is a structural object class used to represent a subordinate reference in the directory. The referral object class SHOULD be used in conjunction with the extensibleObject object class to support the naming attributes used in the entry's Distinguished Name (DN) [RFC2253].

Referral objects are normally instantiated at DSEs immediately subordinate to object entries within a naming context held by the DSA. Referral objects are analogous to X.500 subordinate knowledge (subr) DSEs [X.501].

In the presence of a ManageDsaIT control, referral objects are treated as normal entries as described in section 3. Note that the ref attribute is operational and will only be returned in a search entry response when requested.

In the absence of a ManageDsaIT control, the content of referral objects are used to construct referrals and search references as described in Section 4 and, as such, the referral entries are not themselves visible to clients.

## 2.2 The ref Attribute Type

```
( 2.16.840.1.113730.3.1.34
    NAME 'ref'
    DESC 'named reference - a labeledURI'
    EQUALITY caseExactMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.15
    USAGE distributedOperation )
```

The ref attribute type has directoryString syntax and is case sensitive. The ref attribute is multi-valued. Values placed in the attribute MUST conform to the specification given for the labeledURI attribute [RFC2079]. The labeledURI specification defines a format that is a URI, optionally followed by whitespace and a label. This document does not make use of the label portion of the syntax. Future documents MAY enable new functionality by imposing additional structure on the label portion of the syntax as it appears in the ref attribute.

If the URI contained in a ref attribute value refers to a LDAP [RFC2251] server, it MUST be in the form of a LDAP URL [RFC2255]. The LDAP URL SHOULD NOT contain an explicit scope specifier, filter, attribute description list, or any extensions. The LDAP URL SHOULD contain a non-empty DN. The handling of LDAP URLs with absent or empty DN parts or with explicit scope specifier is not defined by this specification.

Other URI schemes MAY be used so long as all operations returning referrals based upon the value could be performed. This document does not detail use of non-LDAP URIs. This is left to future specifications.

The referential integrity of the URI SHOULD NOT be validated by the server holding or returning the URI (whether as a value of the attribute or as part of a referral result or search reference response).





When returning a referral result or search continuation, the server MUST NOT return the separator or label portions of the attribute values as part of the reference. When the attribute contains multiple values, the URI part of each value is used to construct the referral result or search continuation.

The ref attribute values SHOULD NOT be used as a relative name-component of an entry's DN [RFC2253].

This document uses the ref attribute in conjunction with the referral object class to represent subordinate references. The ref attribute may be used for other purposes as defined by other documents.

## 3. The ManageDsaIT Control

The client may provide the ManageDsaIT control with an operation to indicate that the operation is intended to manage objects within the DSA (server) Information Tree. The control causes Directory-specific entries (DSEs), regardless of type, to be treated as normal entries allowing clients to interrogate and update these entries using LDAP operations.

A client MAY specify the following control when issuing an add, compare, delete, modify, modifyDN, search request or an extended operation for which the control is defined.

The control type is 2.16.840.1.113730.3.4.2. The control criticality may be TRUE or, if FALSE, absent. The control value is absent.

When the control is present in the request, the server SHALL NOT generate a referral or continuation reference based upon information held in referral objects and instead SHALL treat the referral object as a normal entry. The server, however, is still free to return referrals for other reasons. When not present, referral objects SHALL be handled as described above.

The control MAY cause other objects to be treated as normal entries as defined by subsequent documents.

## 4. Named Subordinate References

A named subordinate reference is constructed by instantiating a referral object in the referencing server with ref attribute values which point to the corresponding subtree maintained in the referenced server. In general, the name of the referral object is the same as the referenced object and this referenced object is a context prefix [X.501].





That is, if server A holds "DC=example,DC=net" and server B holds "DC=sub,DC=example,DC=net", server A may contain a referral object named "DC=sub,DC=example,DC=net" which contains a ref attribute with value of "ldap://B/DC=sub,DC=example,DC=net".

```
dn: DC=sub,DC=example,DC=net
dc: sub
ref: ldap://B/DC=sub,DC=example,DC=net
objectClass: referral
objectClass: extensibleObject
```

Typically the DN of the referral object and the DN of the object in the referenced server are the same.

If the ref attribute has multiple values, all the DNs contained within the LDAP URLs SHOULD be equivalent. Administrators SHOULD avoid configuring naming loops using referrals.

Named references MUST be treated as normal entries if the request includes the ManageDsaIT control as described in section 3.

## 5. Scenarios

The following sections contain specifications of how referral objects should be used in different scenarios followed by examples that illustrate that usage. The scenarios described here consist of referral object handling when finding target of a non-search operation, when finding the base of a search operation, and when generating search references. Lastly, other operation processing considerations are presented.

It is to be noted that, in this document, a search operation is conceptually divided into two distinct, sequential phases: (1) finding the base object where the search is to begin, and (2) performing the search itself. The first phase is similar to, but not the same as, finding the target of a non-search operation.

It should also be noted that the ref attribute may have multiple values and, where these sections refer to a single ref attribute value, multiple ref attribute values may be substituted and SHOULD be processed and returned (in any order) as a group in a referral or search reference in the same way as described for a single ref attribute value.

Search references returned for a given request may be returned in any order.





## 5.1. Example Configuration

For example, suppose the contacted server (hosta) holds the entry "O=MNN,C=WW" and the entry "CN=Manager,O=MNN,C=WW" and the following referral objects:

```
dn: OU=People,O=MNN,C=WW
ou: People
ref: ldap://hostb/OU=People,O=MNN,C=US
ref: ldap://hostc/OU=People,O=MNN,C=US
objectClass: referral
objectClass: extensibleObject

dn: OU=Roles,O=MNN,C=WW
ou: Roles
```

```
ref: ldap://hostd/OU=Roles,O=MNN,C=WW
objectClass: referral
objectClass: extensibleObject
```

The first referral object provides the server with the knowledge that subtree "OU=People,O=MNN,C=WW" is held by hostb and hostc (e.g., one is the master and the other a shadow). The second referral object provides the server with the knowledge that the subtree "OU=Roles,O=MNN,C=WW" is held by hostd.

Also, in the context of this document, the "nearest naming context" means the deepest context which the object is within. That is, if the object is within multiple naming contexts, the nearest naming context is the one which is subordinate to all other naming contexts the object is within.

## 5.2. Target Object Considerations

This section details referral handling for add, compare, delete, modify, and modify DN operations. If the client requests any of these operations, there are four cases that the server must handle with respect to the target object.

The DN part MUST be modified such that it refers to the appropriate target in the referenced server (as detailed below). Even where the DN to be returned is the same as the target DN, the DN part SHOULD NOT be trimmed.

In cases where the URI to be returned is a LDAP URL, the server SHOULD trim any present scope, filter, or attribute list from the URI before returning it. Critical extensions MUST NOT be trimmed or modified.





Case 1: The target object is not held by the server and is not within or subordinate to any naming context nor subordinate to any referral object held by the server.

The server SHOULD process the request normally as appropriate for a non-existent base which is not within any naming context of the server (generally return noSuchObject or a referral based upon superior knowledge reference information). This document does not

detail management or processing of superior knowledge reference information.

Case 2: The target object is held by the server and is a referral object.

The server SHOULD return the URI value contained in the ref attribute of the referral object appropriately modified as described above.

Example: If the client issues a modify request for the target object of "OU=People,O=MNN,c=WW", the server will return:

```
ModifyResponse (referral) {
    ldap://hostb/OU=People,O=MNN,C=WW
    ldap://hostc/OU=People,O=MNN,C=WW
}
```

Case 3: The target object is not held by the server, but the nearest naming context contains no referral object which the target object is subordinate to.

If the nearest naming context contains no referral object which the target is subordinate to, the server SHOULD process the request as appropriate for a nonexistent target (generally return noSuchObject).

Case 4: The target object is not held by the server, but the nearest naming context contains a referral object which the target object is subordinate to.

If a client requests an operation for which the target object is not held by the server and the nearest naming context contains a referral object which the target object is subordinate to, the server SHOULD return a referral response constructed from the URI portion of the ref value of the referral object.





Example: If the client issues an add request where the target object

has a DN of "CN=Manager,OU=Roles,O=MNN,C=WW", the server will return:

```
AddResponse (referral) {
    ldap://hostd/CN=Manager,OU=Roles,O=MNN,C=WW"
}
```

Note that the DN part of the LDAP URL is modified such that it refers to the appropriate entry in the referenced server.

## 5.3. Base Object Considerations

This section details referral handling for base object processing within search operations. Like target object considerations for non-search operations, there are the four cases.

In cases where the URI to be returned is a LDAP URL, the server MUST provide an explicit scope specifier from the LDAP URL prior to returning it. In addition, the DN part MUST be modified such that it refers to the appropriate target in the referenced server (as detailed below).

If aliasing dereferencing was necessary in finding the referral object, the DN part of the URI MUST be replaced with the base DN as modified by the alias dereferencing such that the return URL refers to the new target object per [RFC2251, 4.1.11].

Critical extensions MUST NOT be trimmed nor modified.

Case 1: The base object is not held by the server and is not within nor subordinate to any naming context held by the server.

The server SHOULD process the request normally as appropriate for a non-existent base which not within any naming context of the server (generally return a superior referral or noSuchObject). This document does not detail management or processing of superior knowledge references.

Case 2: The base object is held by the server and is a referral object.

The server SHOULD return the URI value contained in the ref attribute of the referral object appropriately modified as described above.

Example: If the client issues a subtree search in which the base object is "OU=Roles,O=MNN,C=WW", the server will return

```
SearchResultDone (referral) {
    ldap://hostd/OU=Roles,O=MNN,C=WW??sub
}
```

If the client were to issue a base or oneLevel search instead of subtree, the returned LDAP URL would explicitly specify "base" or "one", respectively, instead of "sub".

Case 3: The base object is not held by the server, but the nearest naming context contains no referral object which the base object is subordinate to.

If the nearest naming context contains no referral object which the base is subordinate to, the request SHOULD be processed normally as appropriate for a nonexistent base (generally return noSuchObject).

Case 4: The base object is not held by the server, but the nearest naming context contains a referral object which the base object is subordinate to.

If a client requests an operation for which the target object is not held by the server and the nearest naming context contains a referral object which the target object is subordinate to, the server SHOULD return a referral response which is constructed from the URI portion of the ref value of the referral object.

Example: If the client issues a base search request for "CN=Manager,OU=Roles,O=MNN,C=WW", the server will return

```
SearchResultDone (referral) {
    ldap://hostd/CN=Manager,OU=Roles,O=MNN,C=WW??base"
}
```

If the client were to issue a subtree or oneLevel search instead of subtree, the returned LDAP URL would explicitly specify "sub" or "one", respectively, instead of "base".

Note that the DN part of the LDAP URL is modified such that it refers to the appropriate entry in the referenced server.

## 5.4. Search Continuation Considerations

For search operations, once the base object has been found and determined not to be a referral object, the search may progress. Any entry matching the filter and scope of the search which is not a referral object is returned to the client normally as described in [RFC2251].

For each referral object within the requested scope, regardless of the search filter, the server SHOULD return a SearchResultReference which is constructed from the URI component of values of the ref attribute. If the URI component is not a LDAP URL, it should be returned as is. If the LDAP URL's DN part is absent or empty, the DN part must be modified to contain the DN of the referral object. If the URI component is a LDAP URL, the URI SHOULD be modified to add an explicit scope specifier.

Subtree Example:

If a client requests a subtree search of "O=MNN,C=WW", then in addition to any entries within scope which match the filter, hosta will also return two search references as the two referral objects are within scope. One possible response might be:

```
SearchEntry for O=MNN,C=WW
SearchResultReference {
    ldap://hostb/OU=People,O=MNN,C=WW??sub
    ldap://hostc/OU=People,O=MNN,C=WW??sub
}
SearchEntry for CN=Manager,O=MNN,C=WW
SearchResultReference {
    ldap://hostd/OU=Roles,O=MNN,C=WW??sub
}
SearchResultDone (success)
```

One Level Example:

If a client requests a one level search of "O=MNN,C=WW" then, in addition to any entries one level below the "O=MNN,C=WW" entry

matching the filter, the server will also return two search references as the two referral objects are within scope. One possible sequence is shown:





```
SearchResultReference {
    ldap://hostb/OU=People,O=MNN,C=WW??base
    ldap://hostc/OU=People,O=MNN,C=WW??base
}
SearchEntry for CN=Manager,O=MNN,C=WW
SearchResultReference {
    ldap://hostd/OU=Roles,O=MNN,C=WW??base
}
SearchResultDone (success)
```

Note: Unlike the examples in Section 4.5.3.1 of RFC 2251, the LDAP URLs returned with the SearchResultReference messages contain, as required by this specification, an explicit scope specifier.

## 5.6. Other Considerations

This section details processing considerations for other operations.

### 5.6.1 Bind

Servers SHOULD NOT return referral result code if the bind name (or authentication identity or authorization identity) is (or is subordinate to) a referral object but MAY use the knowledge information to process the bind request (such as in support a future distributed operation specification). Where the server makes no use of the knowledge information, the server processes the request normally as appropriate for a non-existent authentication or authorization identity (e.g., return invalidCredentials).

### 5.6.2 Modify DN

If the newSuperior is a referral object or is subordinate to a

referral object, the server SHOULD return affectsMultipleDSAs. If the newRDN already exists but is a referral object, the server SHOULD return affectsMultipleDSAs instead of entryAlreadyExists.

## 6. Security Considerations

This document defines mechanisms that can be used to tie LDAP (and other) servers together. The information used to tie services together should be protected from unauthorized modification. If the server topology information is not public information, it should be protected from unauthorized disclosure as well.





## 7. Acknowledgments

This document borrows heavily from previous work by IETF LDAPext Working Group. In particular, this document is based upon "Named Referral in LDAP Directories" (an expired Internet Draft) by Christopher Lukas, Tim Howes, Michael Roszkowski, Mark C. Smith, and Mark Wahl.

## 8. Normative References

[RFC2079] Smith, M., "Definition of an X.500 Attribute Type and an Object Class to Hold Uniform Resource Identifiers (URIs)", RFC 2079, January 1997.

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

[RFC2251] Wahl, M., Howes, T. and S. Kille, "Lightweight Directory Access Protocol (v3)", RFC 2251, December 1997.

[RFC2252] Wahl, M., Coulbeck, A., Howes, T. and S. Kille, "Lightweight Directory Access Protocol (v3): Attribute Syntax Definitions", RFC 2252, December 1997.

[RFC2253] Wahl, M., Kille, S. and T. Howes, "Lightweight Directory

Access Protocol (v3): UTF-8 String Representation of Distinguished Names", RFC 2253, December 1997.

[RFC2255] Howes, T. and M. Smith, "The LDAP URL Format", RFC 2255, December, 1997.

[RFC2396] Berners-Lee, T., Fielding, R. and L. Masinter, "Uniform Resource Identifiers (URI): Generic Syntax", RFC 2396, August 1998.

[X.501] ITU-T, "The Directory: Models", X.501, 1993.

## 9. Informative References

[X.500] ITU-T, "The Directory: Overview of Concepts, Models, and Services", X.500, 1993.

[X.511] ITU-T, "The Directory: Abstract Service Definition", X.500, 1997.





## 10. Author's Address

Kurt D. Zeilenga
OpenLDAP Foundation

EMail: Kurt@OpenLDAP.org

## 11. Full Copyright Statement

## Acknowledgement

Funding for the RFC Editor function is currently provided by the Internet Society.