Internet Draft
draft-good-ldap-changelog-04.txt
Intended Category: Informational
Expires: September 2003

Gordon Good
Loudcloud
Ludovic Poitou
Sun Microsystems

1 March 2003

Definition of an Object Class to Hold LDAP Change Records

Status of this Memo

1. Abstract

In order to support more flexible replication methods, it is desirable to specify some manner in which an LDAP client may retrieve a set of changes that have been applied to an LDAP server's database. The client, who may be another LDAP server, may then choose to update its own replicated copy of the data. This document specifies an object class that may be used to represent changes applied to an LDAP server. It also specifies a method for discovering the location of the container object that holds these

change records, so that clients and servers have a common rendezvous point for this information.

2. Background and Intended Usage

This document describes an object class that can be used to represent changes that have been applied to a single directory server.

This object class is not intended to be used in a multi-mastered replication environment, where changes can occur on more than one master server.

This document also suggests a common location for a container to hold these objects.

A client may update its local copy of directory information by reading the entries within this container, and applying the changes to its local database.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", and "MAY" in this document are to be interpreted as described in RFC 2119 [3].

3. New Attribute Types Used in the changeLogEntry Object Class

```
( 2.16.840.1.113730.3.1.5
    NAME 'changeNumber'
    DESC 'a number which uniquely identifies a change made to a
          directory entry'
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.27
    EQUALITY integerMatch
    ORDERING integerOrderingMatch
    SINGLE-VALUE
)

( 2.16.840.1.113730.3.1.6
    NAME 'targetDN'
    DESC 'the DN of the entry which was modified'
    EQUALITY distinguishedNameMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
    SINGLE-VALUE
 )

( 2.16.840.1.113730.3.1.7
    NAME 'changeType'
    DESC 'the type of change made to an entry'
    EQUALITY caseIgnoreMatch
```

SYNTAX 1.3.6.1.4.1.1466.115.121.1.15

```
        SINGLE-VALUE
    )

    ( 2.16.840.1.113730.3.1.8
        NAME 'changes'
        DESC 'a set of changes to apply to an entry'
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.40
    )

    ( 2.16.840.1.113730.3.1.9
        NAME 'newRDN'
        DESC 'the new RDN of an entry which is the target of a
              modrdn operation'
        EQUALITY distinguishedNameMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
        SINGLE-VALUE
    )

    ( 2.16.840.1.113730.3.1.10
        NAME 'deleteOldRDN'
        DESC 'a flag which indicates if the old RDN should be retained
              as an attribute of the entry'
        EQUALITY booleanMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.7
        SINGLE-VALUE
     )

    ( 2.16.840.1.113730.3.1.11
        NAME 'newSuperior'
        DESC 'the new parent of an entry which is the target of a
              moddn operation'
        EQUALITY distinguishedNameMatch
        SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
        SINGLE-VALUE
    )
```

4. Schema Definition of the changeLogEntry Object Class

```
    ( 2.16.840.1.113730.3.2.1
        NAME 'changeLogEntry'
        SUP top
        STRUCTURAL
        MUST ( changeNumber $ targetDN $ changeType )
        MAY ( changes $ newRDN $ deleteOldRDN $ newSuperior )
    )
```

5. Discussion of changeLogEntry Attributes:

changeNumber: the change number, as assigned by the supplier. This integer MUST strictly increase as new entries are added, and must always be unique within a given server.
Syntax: INTEGER

targetdn: the distinguished name of the entry which was added, modified or deleted. In the case of a modrdn operation, the targetdn gives the DN of the entry before it was modified.
Syntax: DN

changeType: the type of change. One of: "add", "delete", "modify", "modrdn". Later RFCs may define additional values for changeType.
Syntax: DirectoryString

changes: the changes which were made to the directory server. These changes are in LDIF format, which is described in [1].
Syntax: OctetString

newRDN: the new RDN (Relative Distinguished Name) of the entry, if the changeType is "modrdn". If the changeType attribute does not have the value "modrdn", then there should be no values contained in the newRDN attribute.
Syntax: DN

deleteOldRDN: a flag which tells whether the old RDN of the entry should be retained as a distinguished attribute of the entry, or should be deleted. A value of "FALSE" indicates that the RDN should be retained as a distinguished attribute, and a value of "TRUE" indicates that it should not be retained as a distinguished attribute of the entry. If any value other than "TRUE" or "FALSE" is contained in the deleteOldRDN, the RDN should be retained as a distinguished attribute (that is, "FALSE" is the default if no values are present, or if illegal values are present).
Syntax: BOOLEAN

newSuperior: if present, gives the name of the entry which becomes the immediate superior of the existing entry. This optional attribute reflects LDAPv3 functionality, and MUST NOT be generated by LDAPv2 servers [2].
Syntax: DN

6. Discussion of the changeLogEntry object class

The changeLogEntry object class is used to represent changes made to a directory server. The set of changes made to a directory server, then, is given by the ordered set of all entries within the changelog container, ordered by changeNumber. As a changeNumber is unique, it is recommended that the changeNumber attribute be used to name changeLogEntry entries.

A client may synchronize its local copy of a remote directory server's contents by searching the remote server's changelog container for any entries where the changenumber is greater than or equal to the last change previously retrieved from that server. If the entry with the changenumber matching the last change retrieved is not returned in the search results, then the server's changelog has been trimmed. The client must then fall back to reading the entire directory to bring its copy in sync with the server's.

Assuming that the client has successfully retrieved one or more changelog entries from the server, it can then use the information contained in each entry to update the corresponding entry (named by the targetDN attribute) in its local copy of the database.

Note that, due to access control restrictions, the client is not guaranteed read access to the "changes" attribute. If the client discovers that the "changes" attribute has no values, then it must read the entry given by the targetDN attribute, possibly only retrieving attributes it deems "interesting". However, in the case of delete and modrdn operations, there is never a "changes" attribute, so it is never necessary to read the target entry in these cases.

Servers implementing this document MUST trim the changeLogEntry entries only in changeNumber order.

7. Examples of the changeLogEntry object class

In each example below, the "changes" attribute is shown in plain text, with embedded new-line characters. This is done for clarity. It is intended that new-line characters be stored in the entry literally, not encoded in any way. If a "changes" attribute value is stored in an LDIF file, it must base-64 encoded.

Example 1: A changeLogEntry representing the addition of a new entry to the directory

```
dn: changenumber=1923, cn=changelog
changenumber: 1923
targetdn: cn=Barbara Jensen, ou=Accounting, o=Ace Industry, c=US
changetype: add
changes: cn: Barbara Jensen\ncn: Babs Jensen\nsn: Jensen\n
 givenname: Barbara\ntelephonenumber: +1 212 555-1212\nmail:
 babs@ace.com\nobjectclass: top\nobjectclass: person\nobjectclass:
 organizationalPerson\nobjectclass: inetOrgPerson
```

Example 2: A changeLogEntry representing the deletion of an entry from the directory

```
dn: changenumber=2933, cn=changelog
```

```
changenumber: 2933
targetdn: cn=Gern Jensen, ou=Product Testing, o=Ace Industry, c=US
changetype: delete
```

Example 3: A changeLogEntry representing the modification of an entry in the directory

```
dn: changenumber=5883, cn=changelog
changenumber: 5883
targetdn: cn=Bjorn Jensen, ou=Product Development, o=Ace Industry,
c=US
changetype: modify
changes: delete: telephonenumber\ntelephonenumber: 1212\n-\n
 add: telephonenumber\ntelephonenumber: +1 212 555 1212\n-
```

Example 4: A changeLogEntry representing a modrdn operation performed on an entry in the directory

```
dn: changenumber=10042, cn=changelog
changenumber: 10042
targetdn: cn=Bjorn Jensen, ou=Product Development, o=Ace Industry,
c=US
changetype: modrdn
newrdn: cn=Bjorn J Jensen
deleteoldrdn: FALSE
```

8. Location of the container containing changeLogEntry objects

For LDAPv3 servers, the location of the container that holds changeLogEntry objects is obtained by reading the "changeLog" attribute of a server's root DSE. For example, if the container's root is "cn=changelog", then the root DSE must have an attribute named "changeLog" with the value "cn=changelog".

The "changelog" attribute is defined as follows:

```
(   2.16.840.1.113730.3.1.35
    NAME 'changelog'
    DESC 'the distinguished name of the entry which contains
          the set of entries comprising this server's changelog'
    EQUALITY distinguishedNameMatch
    SYNTAX 1.3.6.1.4.1.1466.115.121.1.12
)
```

For LDAPv2 servers, the name of the changelog container must be "cn=changelog".

9. Interoperability between LDAPv2 and LDAPv3 implementations

Implementors are discouraged from developing implementations in which an LDAPv2 server is synchronized from an LDAPv3 server using the changelog method described in this document. Problems can arise when an LDAPv2 server reads a "moddn" changelog entry which gives a new superior. Since LDAPv2 does not support such an operation, there is no way for the v2 server to perform the moddn operation atomically. It could, of course, delete all the entries under the old superior and add them under the new superior entry, but such an operation would either not be atomic, or require extensive server-side support on the LDAPv2 server to make the operation appear as if it were atomic.

It is recommended that servers who only implement LDAPv2 should refuse to synchronize from LDAPv3 servers. Before beginning synchronization, the LDAPv2 server should attempt to read the root DSE of the supplier server. If the root DSE is present, and the supportedldapversion attribute contained in the root DSE contains the value "3", then the LDAPv2 server should immediately disconnect and proceed no further with synchronization.

10. Security Considerations

Servers implementing this scheme MUST NOT allow the "changes" attribute to be generally readable. The "changes" attribute contains all modifications made to an entry, and some changes may contain sensitive data, e.g. Passwords.

If a server does allow read access on the "changes: attribute to a particular bound DN, then that DN should be trusted. For example, two cooperating servers may exchange the password for some DN that is granted read access to the "changes" attribute of the changeLog. This would allow one server to retrieve changes and apply them directly to its database.

In situations where the "changes" attribute is not readable by a client, that client may still use the entries in the changeLog to construct a list of entry DNs which are known to have changed since the last time the client synchronized. The client may then read each of those entries, subject to whatever access control is in effect on the server, and update its local copy of each entry.

Servers implementing this scheme should disallow write access to the changelog container object and all entries contained within.

11.IANA Considerations

The OIDs used in this document have been assigned by Netscape Communications Corp, under its ANSI-assigned private enterprise

allocation, for use in this specification.

12. Acknowledgements

This material is based in part upon work supported by the National Science Foundation under Grant No. NCR-9416667.

13. Normative References

[1] Good, G., "The LDAP Data Interchange Format (LDIF) - Technical Specification", RFC 2849, June 2000.

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

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

14. Authors's Address

Gordon Good
Loudcloud, Inc.
599 N. Mathilda Avenue
Sunnyvale, CA 94085
USA
Phone: +1 408 744-7300
EMail: ggood@loudcloud.com

Ludovic Poitou
Sun Microsystems Inc.
180 Avenue de l'Europe
Zirst de Montbonnot
38334 Saint Ismier cedex
France
Phone: +33 476 188 212
Email: ludovic.poitou@Sun.com

15. Full Copyright Statement

document itself may not be modified in any way, such as by removing