

OCLC LDAP Referral Investigation

Harold Cheney

June 20, 2002

Members of IPIG have raised questions concerning the differences between *referrals*, where a directory server refers a client to another server, and *chaining*, where one server retrieves information from another server on behalf of a client.

Overview of LDAP Directory and Referrals

A directory entry is a collection of attributes that have *names*, *types*, and *values*. Entries are organized in a hierarchical *Directory Information Tree* (DIT). Each entry is uniquely identified by its *distinguished name* (DN), which specifies its position in the DIT. Note that DNs read from leaf to root, for example `cn=Bob Jones, o=ibm, c=us`.

A search request includes a *search base*, which is the DN of highest node of the DIT to search from. Each server is configured with one or more *suffixes*, which are the DNs of the highest node of each branch of the DIT contained on that server. The server may be configured with a *default referral*. If the server receives a request with a search base that does not have a suffix configured on that server, it refers the client to the server specified in the default referral. In addition, any node of the DIT may contain a referral to a server that contains entries for the portion of the DIT under that node.

Configuration and Testing

OCLC installed and configured three LDAP servers using IBM Directory Server 4.1. A simplified representation of the server configuration is given in figure 1.

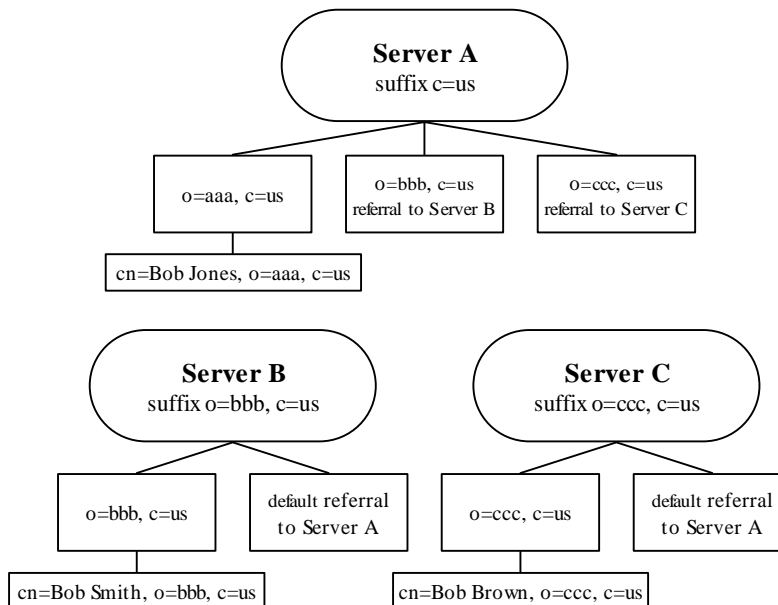


Figure 1

OCLC implemented a simple LDAP search client using the Java 2 Platform, Standard Edition, SDK version 1.4.0, and on each server performed identical searches that would match entries on all three servers. Each search returned the correct entries from all three servers, indicating that referrals were followed as expected. For example, a request with search base `c=us` and filter `(cn=bob*)` would result in the entries for Bob Jones, Bob Smith, and Bob Brown, regardless of which server the request was sent to.

Questions and Conclusions

- **Do referrals affect a user's experience?**
Not if the user's client software handles referrals in a transparent manner.
- **Is it difficult for client software to handle referrals?**
Not if it is implemented using a standard client library that has the ability to follow referrals automatically. For an example, see the client source at the end of this document.
- **Do servers need to be configured with referrals to every other server?**
Not necessarily. For example, in figure 1, Servers B and C have no direct knowledge of each other—they find each other through their default referrals to Server A. However, there is nothing preventing Server A or B from being configured with a direct referral to the other, if its administrator decides that the volume of traffic justifies the effort.
- **Can referrals degrade client performance?**
Referrals generate the same number of messages as server-side chaining, but with referrals, the client participates in every message, so if the client's network connection is slow, performance might suffer. However, the search request and referral messages are typically not large. The search result message may be large, but in either case it is sent to the client only once, so there shouldn't be much of a difference.
- **Can referrals complicate authentication?**
Yes, if the client is required to authenticate on each server it is referred to, but referrals are normally used for searching, which can be performed by anonymous access. For operations that require authenticated access, such as updating, referrals should not be necessary, since users typically will have authorization to update only on their own server.

Client Source

```
// File: Client.java

import javax.naming.Context;
import javax.naming.NamingEnumeration;
import javax.naming.directory.Attribute;
import javax.naming.directory.DirContext;
import javax.naming.directory.InitialDirContext;
import javax.naming.directory.SearchControls;
import javax.naming.directory.SearchResult;
import java.util.Hashtable;

/**
 * Simple Java LDAP search client.
 */
public class Client
{
    public static void main(String[] args) throws Exception
    {
        if (args.length != 3)
        {
            System.out.println("args: host searchbase filter");
            System.out.println("example: ldap://xyzyzy.com:389 c=us (cn=bob*");
            System.exit(1);
        }

        String host = args[0];
        String base = args[1];
        String filter = args[2];

        Hashtable env = new Hashtable();
        env.put(Context.INITIAL_CONTEXT_FACTORY, "com.sun.jndi.ldap.LdapCtxFactory");
        env.put(Context.PROVIDER_URL, host);
        env.put(Context.REFERRAL, "follow"); // automatically follow referrals

        DirContext context = new InitialDirContext(env);
        SearchControls controls = new SearchControls(
            SearchControls.SUBTREE_SCOPE, // search entire subtree
            0, // no result count limit
            0, // no time limit
            null, // return all attributes
            true, // return objects
            true); // dereference links

        NamingEnumeration results = context.search(base, filter, controls);
        while (results != null && results.hasMore())
        {
            SearchResult entry = (SearchResult)results.next();
            System.out.println("-----");
            System.out.println("dn: " + entry.getName());

            NamingEnumeration attributes = entry.getAttributes().getAll();
            while (attributes != null && attributes.hasMore())
            {
                Attribute attr = (Attribute)attributes.next();
                String id = attr.getID();
                NamingEnumeration values = attr.getAll();
                while (values != null && values.hasMore())
                {
                    System.out.println(id + ": " + values.next());
                }
            }
        }
    }
}

// End of Client.java
```