Golden Apple Enterprises Ltd.

What is LDAP and why should I care?

A Rough Guide to LDAP Directory Services

http://www.gaeltd.com

LDAP - The History

In the beginning, there were networks. These networks had users, and roles, and login details, and email addresses, and home directories, and access rights - basically, there was a mess.

The nice chaps at the ITU and ISO came up with X.500, which is a series of protocols to provide electronic directory services. By defining how you can store electronic records in a central directory, and access them, publish them, update them, etc. an attempt was made to tame this mess and make all this electronic information manageable again.

Unfortunately X.500 was both a) quite a complex stack of protocols, and b) was tied to the OSI networking stack. The only really successful implementation that I can remember is Novell's Netware Directory Services (NDS), which appeared in Netware 4 and totally and utterly kicked arse. NDS design handbooks are still the best thing around when designing a large directory service. Even NDS didn't use the OSI stack though.

DAP - the Directory Access Protocol - was the protocol to actually access an X.500 directory, and it was this that was tied to the OSI stack.

Anyway, along came 'open systems' and the TCP/IP stack, and people wanted to access their directories without all that OSI hassle.

So they came up with the Lightweight Directory Access Protocol - LDAP.

What is an LDAP directory?

Think of it as an electronic phonebook - it's basically a collection of objects organised by some sort of hierarchy. For a phonebook, it's alphabetical listings. For an LDAP directory, it's usually some sort of organisational or geographic hierarchy.

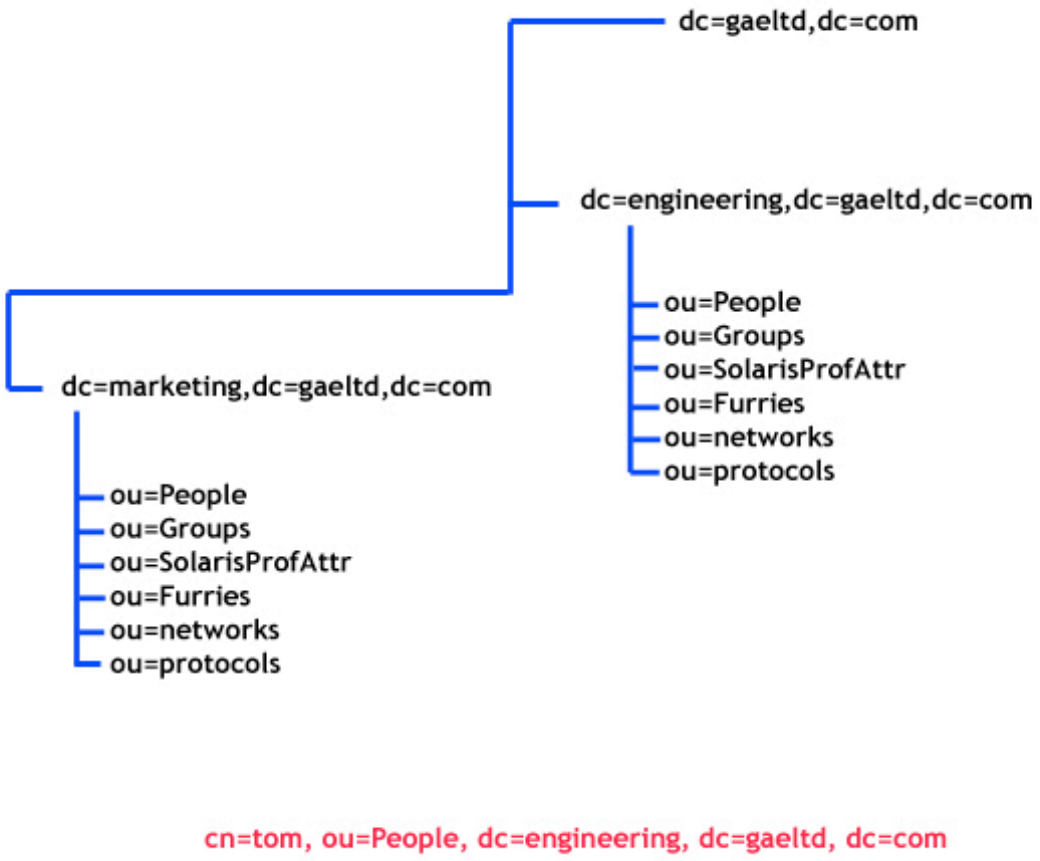
The directory structure is pretty straightforward:

- A directory is a tree of directory entries.

- An entry consists of a set of attributes.

- An attribute has a name (an attribute type or attribute description) and one or more values. The attributes are defined in a schema.

- Each entry has a unique identifier: its Distinguished Name (DN).



LDAP is a binary protocol, and the directory server uses a bunch of binary databases which are optimised for very very fast reads, at the expense of very slow writes.

The LDAP protocol can support encryption - either by using the LDAPS protocol or by using the STARTLS functionality (much like a web browser using SSL).

So what can we store in LDAP?

Pretty much anything. User login details, desk numbers, addresses, phone numbers, access rights, printer details, UNIX credentials, Windows credentials - you name it, you can store it. A directory server relies on it's schema to define the information it stores, and if you update or modify the schema, you can store what you like in there.

Common uses are things like username and passwords for UNIX logins, Solaris RBAC roles and privileges, and application configuration. Microsoft's Active Directory is actually an LDAP server wrapped up in the Kerberos authentication protocol - that gives you some idea of how flexible it can be. Some companies have their phone system plugged into a central LDAP directory, as well as their corporate email and meeting booking systems.

Entries can be added, edited, and view by using the LDAP Data Interchange Format (LDIF). LDIF is a way of representing directory data in text format, making it easy to wrap up in scripts to manage the data.

The key thing to remember is that LDAP and the directory server are optimised for very fast reads - to quickly retrieve information - but not for writes. If you want to put data into a LDAP directory server which needs to be constantly updated, it's not going to work that well.

Why are there so many different versions?

This gets a bit complex. Back in the day, a team at the University of Michigan came up with a reference implementation of LDAP, called the slapd project. Netscape hired the team and came up with the Netscape Directory Server, pushing it and LDAP as a good backend for the emerging web services infrastructure.

Then Netscape went under, and were asset stripped by AOL, with Sun buying the intellectual property rights to most of their software stack. RedHat bought the ownership rights to the Netscape Directory Server not long afterwards. At roughly the same time, the open source community started working on OpenLDAP, an open source implementation of the slapd reference server project.

This was also the time when Sun's marketing team spent far too much time at the crack pipe, and essentially went mental. They kept on renaming their Directory Server - first it was iPlanet, then Sun ONE Directory Server, then Sun Java System Directory Server. It's now been rewritten and open-sourced by Sun, as OpenDS.

Companies like IBM also had reseller deals with Netscape (and then Sun, who owned the IP) and so have essentially the same code base which they sell as their own LDAP solution.

RedHat renamed Netscape Directory Server to Fedora Directory Server, and then open sourced it.

It's all very confusing and is a hallmark of the rapid explosion of the Internet back in the mid-90s. And too many marketing teams smoking crack and trying to differentiate their product offerings.

One of the worst side effects of this mess (apart from 15 different names for the same product from Sun) is that a lot of the terminology is confusing. LDAP can refer to the protocol, or the protocol and the data. An LDAP directory could mean the directory server, the data held in the directory, or the way you access the data.

Does it scale?

Yes. LDAP directories implement a very efficient replication protocol for sharing changes to data across servers. If you think about your organisational hierarchy, you'll soon realise that it's a natural fit for splitting things up into sub-trees of directory entries. Replication only looks at changes, and only copies across those changes.

You can make things even more efficient by dedicating some directory servers as Masters (where records can be updated) and some as Slaves (which have updates replication down to them, and essentially provide a read-only view to their part of the directory hierarchy).

Obviously, it's important to understand how your organisation is going to use LDAP, and the resulting designs can be very very complex. Dropping in a central LDAP directory server to manage user accounts across 40 UNIX servers is easy. Implementing an LDAP solution for a global corporation, managing phone, email, and user accounts, is slightly more of a challenge.

Arguably most of the development and innovation in development of the directory has come from Sun. Despite the constant and pointless renaming exercises, Sun's LDAP directory server is very powerful and feature rich, although it has a (frankly) crappy JAVA based GUI.

NDS lives on as Novell's eDirectory, and it still kicks arse as much as it did when it appeared in Netware 4. It's fully cross platform and you should give it serious considering when looking at a large scale directory service.

Why should I care?

Infrastructure in general is complex and expensive to manage. If you can plug in a central system to handle common tasks like user authentication, email address management and phone number lookups, you can make it easier to manage your infrastructure. That in turn makes it cheaper, and that is a Good Thing. Do you want to have to add a new user account to 500 machines, or add it to one central directory server?

Frankly, when it comes to designing or managing infrastructure, LDAP is awesome. Ever since NDS hit the network in Netware 4, implementations of LDAP and directory servers have been getting more flexible and much more powerful. Once you've experienced the power and simplicity of centrally managing so many aspects of your infrastructure, you'll wonder how you ever managed without it.