

Build an LDAP-based address book

Presented by developerWorks, your source for great tutorials

ibm.com/developerWorks

Table of Contents

If you're viewing this document online, you can click any of the topics below to link directly to that section.

1. About this tutorial	2
2. A primer on LDAP	4
3. Installing and configuring OpenLDAP	9
4. Populating your LDAP directory	13
5. Creating an LDAP address book with Rolodap	17
6. Configuring an LDAP-aware e-mail client	20
7. Summary, resources, and feedback	22

Section 1. About this tutorial

What does this tutorial cover?

This tutorial shows you how to create an LDAP-based backend to store contact information that multiple applications can share easily. Along the way, we give you an overview of LDAP fundamentals and introduce you to a pre-built contact management tool that will help you get started using this open technology.

Almost all e-mail clients on the market today (both open source and proprietary) suffer from at least one fatal, nagging flaw -- the address books they provide for storing contact information make it easy to store data, and notoriously difficult to get that data back out in a form that can be used by other programs.

The material in this tutorial is a direct result of my own personal frustration in easily sharing contact information between e-mail clients, other programs, operating systems, and computers. The solution presented is to use a centrally located LDAP directory service to store e-mail addresses and contact details, and access this information for a wide range of applications as needed.

One caveat up front: To utilize the methods described here, your application(s) must support LDAP access in one form or another. LDAP support is becoming commonplace in most popular e-mail clients and a growing number of other data-driven applications. But be sure to check the documentation provided with your applications *before* making any dramatic transitions. Even if your program of choice doesn't currently support LDAP access, chances are it will in the near future; by reading through this tutorial, you'll be prepared when that day comes.

On completion of this tutorial, you will:

- Understand the technology behind an LDAP directory, how its database structures are organized, and how an LDAP store is accessed by a client.
- Know how to install and configure an OpenLDAP server.
- Be able to add content to the LDAP directory.
- Understand the general procedures required to allow an LDAP-enabled e-mail client to look up contact information stored in an LDAP directory.

Who should take this tutorial?

Have you ever tried to share your contact list with someone who uses a different data format, or tried to migrate your address book to another application? If so, you know it can be a headache, and this tutorial is for you.

Readers should have a basic working knowledge of common administrative tasks and the concepts behind them. This includes tasks such as setting file permissions, managing user

accounts, moving and copying files, creating symlinks, etc.

Prerequisites

Given the diverse needs of users, the variety of software implementations, and the flexibility of open source and open standards, it's virtually impossible to cover all potential system configurations in the space provided here. Some kind of "entry point" is a must. To follow along with the examples in this tutorial, you'll need access to a correctly installed and configured Linux system and the following software:

- Red Hat Linux 7.3. Operating system specific instructions are based on Red Hat Linux 7.3 (see [Resources](#) on page 22). Red Hat Linux was chosen because of its popularity, and because most administrators/users have at least a passing acquaintance with its system layout and conventions.
- OpenLDAP. OpenLDAP is used as an LDAP directory server (see [Resources](#) on page 22). OpenLDAP is open source, based on open standards, and available as a free download. For the most part, however, the structures, layout, and administrative tasks discussed are readily transferable to commercially available directory servers such as IBM's SecureWay Server and Netscape's Directory Server offering.

About the author

Tom Syroid is a contract writer for [Studio B Productions](#), a literary agency based in Indianapolis, IN, specializing in computer-oriented publications. His specialties include *NIX system security, Samba, Apache, and Web database applications based on PHP and MySQL. He has experience administering and maintaining a diverse range of operating systems including Linux (Red Hat, OpenLinux, Mandrake, Slackware, Gentoo), Windows (95, 98, NT, 2000, and XP), and AIX (4.3.3 and 5.1). He is also the co-author of *Outlook 2000 in a Nutshell* (O'Reilly & Associates) and *OpenLinux Secrets* (Hungry Minds). Tom lives in Saskatoon, Saskatchewan, with his wife and two children. Hobbies include breaking perfectly good computer installations and then figuring out how to fix them, along with gardening, reading, and building complex structures out of Legos with his kids.

Questions, comments, and errata submissions are welcome; you can either e-mail the author directly (dwcomments@syroidmanor.com) or use the [Your feedback](#) on page 24 form at the end of the tutorial.

Section 2. A primer on LDAP

Broad strokes

The acronym *LDAP* stands for Lightweight Directory Access Protocol. Unlike some computer terms, the term LDAP is amazingly self-descriptive.

- LDAP is an open standard based in part on the X.500 directory standard, but simpler, less cumbersome, and more extensible -- it's *lightweight* in contrast to some other communication protocols. The specifications for the LDAP standard are laid out in a series of RFCs (or Request For Comment). For more information on LDAP-related RFCs, see the LDAPman RFC page listed in [Resources](#) on page 22 .
- Information is stored on a centrally located server in an LDAP *directory*. An LDAP directory is a type of database; it is not, however, a *relational* database. The structure of this directory or database closely resembles a UNIX filesystem: data is stored hierarchically; it has a "root" or "base DN" (Distinguished Name); the directory is further subdivided into Organization Units (or OUs); within these OUs are entries containing data. This *tree-leaf* structure makes LDAP not only extensible, but faster than a traditional relational database when it comes to simple searches or lookups.
- Using the LDAP protocol, clients send a query to the LDAP server (technically speaking, LDAP has no "read" function; clients "read" a directory entry by sending a *search* request to the server). The server checks for client authorization (that is, can the client *access* the database? Read the requested tree? Write information to the database? Delete an entry?), and returns the requested information. Almost all modern programming languages have an LDAP API, which means almost any piece of software can be made "LDAP aware."

Unfortunately, the term LDAP is often used either without context or in an inappropriate context. And given the variety of ways the term can be used (the LDAP protocol, an LDAP server, or an LDAP client), anyone new to the world of LDAP can easily become disoriented. For the purposes of this tutorial, every effort has been made to ensure context is stated or clearly obvious from the surrounding discussion.

Why LDAP?

Over the last two or three years, LDAP implementations have gone from relative obscurity to "hot topic du jour." The reason for this sudden increase in popularity can be loosely summed up in two words: Extensibility and flexibility.

The LDAP protocol is both cross-platform and standards-based. This means almost any application, running on almost any computer platform, can obtain information from an LDAP directory. In addition, the server operating system, file system, or platform is of no significance whatsoever to the client.

An LDAP directory can store an almost infinite range of data: e-mail addresses, DNS information, NIS maps, security keys, contact lists, computer names, etc. If specialized

organizational units or entries are required, the rules that control what type of information a given field can hold (called *schemas*; more on this shortly) can be customized to the implementation.

Most LDAP servers are relatively simple to install and configure, run for years with little or no maintenance, and can be readily optimized for specific types of access.

LDAP directories can be easily configured to replicate some or all of the directory tree (using push or pull methods). This eliminates any single point of failure concerns for the system administrator.

Access to the directory can be controlled via ACLs (Access Control Lists). For example, an administrator can restrict who can see what based on membership in a given group or location, or give particular users the ability to modify selected fields within their own records. ACLs provide for extremely fine-grained access control, and they tie that control to the LDAP installation, not the client requesting the information. Furthermore, LDAP can be easily integrated with most existing security layers and/or authentication systems (for example, SSL, Kerberos, PAM, etc.)

LDAP directory structure: The base DN

So far, we've discussed what LDAP is, how it works (from a high level), the general structure of an LDAP directory, and why LDAP implementations are so popular. Now it's time to dig down a little deeper into the various components of an entry itself. As noted in a previous panel, the "root" or top of an LDAP directory tree is the *base DN*. A base DN typically takes one of two forms: an *organization=* for (for example, `o=syroidmanor.com`) or a DN derived from the organization's DNS domain components (`dc=syroidmanor,dc=com`).

For most installations, the latter is the preferred format simply because it separates out the two distinct components. If, down the road, your company decides to merge with another company, the existing structure does not have to be revised; the base DN then becomes `dc=com`, and the two companies branches of the `com` tree (this, of course, doesn't help much if `syroidmanor.com` merges with `syroidmanor.com`).

All of which brings us to two very important points: (1) 99 percent of a successful LDAP implementation lies in pre-planning an extensible, workable structure for your organization, and (2) every LDAP installation is, to some extent, unique; in other words, there are no "cast in stone" rules for structural layout.

The figure below shows an example of a "dc" (versus "organization") LDAP directory structure.

LDAP directory structure: Organizational units

Underneath the directory base DN are *containers* or *organizational units* (OUs) that logically

separate or group your data. Choices here are usually determined by the organizational structure of your business or installation. In addition, second-tier OUs can be used to further segregate data. For example, an international business might use the following structure:

```
dc=foobar,dc=com
  ou=customers
    ou=northamerica
    ou=southamerica
    ou=asia
    ou=europe
  ou=employees
  ou=group
  ou=projects
  ou=accounting
  ou=resource
  ou=service
```

A general rule of thumb is to keep your organizational structure as simple as possible, without compromising future extensibility. Also keep in mind that the deeper you nest your structural containers, the longer it's going to take to return a query.

LDAP directory structure: Individual entries

Under the organization units of an LDAP structure are the actual entries. Below is an example, shown in LDIF format (more on the LDIF format in [Populating your LDAP directory](#) on page 13).

```
dn: cn=Tom Syroid, ou=employee, dc=syroidmanor, dc=com
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: Tom Syroid
cn: Thomas Syroid
sn: Syroid
givenName: Tom
initials: TMS
title: Project Manager
uid: tsyroid
mail: tom@syroidmanor.com
telephoneNumber: 306 555 1212
mobile: 306 555 1999
roomNumber: 115
employeeNumber: 33
employeeType: full time
```

First of all, note the following:

- An entry is simply a place to store *attributes*.
- An attribute is a container that may be used to store a single type of information within your directory.
- Each attribute has a *type* and one or more *values*.

So `cn: Tom Syroid` is one attribute of the entry, `cn` is the type, and the values associated with that type are "Tom Syroid" and "Thomas Syroid".

Now let's dissect the attributes of the above entry. The first line contains the *DN*, or "Distinguished Name" of the entry. Every entry in an LDAP directory has a unique DN, and each DN is composed of two parts -- the "Relative Distinguished Name" (or RDN) and reference to the location within the LDAP directory structure where the record is stored. Almost all data stored in an LDAP directory has a unique name, which is typically stored in the `cn` attribute. In the example above, the `cn` value used to uniquely identify or distinguish the record is "Tom Syroid". Note that "Thomas Syroid" could be used as well, as long as the `dn` supplied was unique to all other entries in the database. The `ou` and `dc` values that follow point to a record's location within the directory structure.

Object classes

Object classes are used by an LDAP directory to define which attributes are allowed for objects of a given type. An object class also defines what attributes an entry *must* have, and what attributes an entry *can* have. All object classes inherit the requirements from their parent object class, then add their own. Below is an example of an object class:

```
objectclass ( 2.5.6.6 NAME 'person' SUP top STRUCTURAL
             MUST ( sn $ cn )
             MAY ( userPassword $ telephoneNumber $ seeAlso $ description ) )
```

An object class has five components: an OID (object identifier), a unique name, a parent (*SUP*), any required attributes (*MUST*), and a list of allowed attributes (*MAY*). OIDs are numeric identifiers used by an LDAP directory's internal database mechanism. They're similar in concept to an IP address in that each object class must have a unique number. And like the relationship between DNS and IPs, OIDs are registered and "owned" by the individual who created them. For more information on registering an OID, see the Internet Assigned Numbers Authority (or IANA). A link is available in the [Resources](#) on page 22 section.

Do you need an OID number to create your own object class? The answer depends on your directory server software -- some allow it, and others do not. Check your LDAP documentation for details. For more information on defined object classes and their attributes, check out the LDAPman Schema reference page or the extremely useful LDAP Schema Viewer Web site (see [Resources](#) on page 22 for links).

What's a *schema*, you ask? A schema is simply a collection of object classes grouped by similarity. For example, the `inetOrgPerson` schema contains object classes for `departmentNumber`, `employeeType`, `givenName`, `homePhone`, `manager`, etc. The `inetOrgPerson` schema also inherits numerous object classes from other "parent" schemas.

Wrap up

This section has covered a lot of ground in a small amount of space. And while the material presented here is not required prior to installing and configuring an LDAP server -- the topic of the next section -- having at least a passing grasp of the rudimentary concepts behind an LDAP directory service helps the administrator better understand the whys behind the how-tos. For more information on LDAP structures, object classes, and attributes, check OpenLDAP Administrator's Guide and numerous FAQs available from www.openldap.org (see [Resources](#) on page 22 for links). Another good resource is *Introduction to LDAP* written by one of the staff at *Plugged In Software* (see [Resources](#) on page 22).

Section 3. Installing and configuring OpenLDAP

Required packages

Installing and configuring OpenLDAP is a relatively painless process on most package-based systems (for example, on RPM-based distributions such as Red Hat, Mandrake, and SuSE). The first step is to determine which, if any, OpenLDAP components were installed as part of your initial Linux setup.

From a console window or the command line, type:

```
[root@thor root]# rpm -qa | grep openldap
openldap-devel-2.0.23-4
openldap-2.0.23-4
openldap-servers-2.0.23-4
openldap-clients-2.0.23-4
[root@thor root]#
```

You should see output similar to the above. Note that Red Hat distributions install the OpenLDAP client software, but *do not* install the `openldap-servers` package even if you chose a server configuration. To install an RPM package, locate the required file on the distribution media and type:

```
rpm -ivh packagename
```

Other distributions vary in how programs are packaged and named; consult the documentation for details.

Configuring the OpenLDAP server

With the requisite software installed, the next step is to configure the server. First, *back up the original configuration file* for future reference (`cp /etc/openldap/slapd.conf /etc/openldap/slapd.conf.orig`). Now open the `/etc/openldap/slapd.conf` file in your favorite text editor and take a few minutes to read through the comments. Most of the default settings in `slapd.conf` are adequate with the exception of several entries that will define the type of directory database, the `suffix`, the `rootdn`, and the location to store the directory database.

```
database          ldbm
suffix            "dc=syroidmanor,dc=com"
rootdn            "cn=root,dc=syroidmanor.com,dc=com"
rootpw            {CRYPT}05T/JKDW00SuI
directory         /var/lib/ldap
index             objectClass,uid,uidNumber,gidNumber,memberUid    eq
index             cn,mail,surname,givenname                       eq,subinitial
```

Securing the `rootdn`

The `rootdn` entry controls who can write to the directory database, and the password they must supply to do so. Be sure to read the comments at the end of the Access Control section:

```
# if no access controls are present, the default is:
#     Allow read by all
#
# rootdn can always write!
```

"rootdn can always write!" means what it says. Whatever entry you filled in for the `cn=` portion of the `rootdn` entry is the user who has full read/write access to the database. In addition, the default configuration file uses a password of "secret", which is transmitted in plain text. If your LDAP server is only accessible from an intranet firewalled from the rest of the world, and you're confident that none of the users who will access the LDAP server know anything about packet sniffing, you're probably safe transmitting the `rootdn` password in plain text (just be sure to change the password to something a tad less obvious than "secret"). But if any of the data you intend to store in the directory is even remotely sensitive, *hash the password*. This can be done with the `slappasswd` utility like so:

```
[root@thor root]# slappasswd -h {crypt}
```

You will be asked for a password, and `slappasswd` will then provide a *crypt* string that corresponds to the entry provided. Cut and paste this string into `slapd.conf` as shown in the preceding panel. Other hash options include SSHA (the default), SMD5, MD5, and SHA. Type `man slappasswd` for more information.

Test the server

Now is a good time to test the server. The configuration at this point is relatively simple and easy to troubleshoot should problems arise. On a Red Hat Linux system, the command is:

```
[root@thor root]# service ldap start
```

Next, test your ability to access the directory:

```
[root@thor root]# ldapsearch -x -b '' -s base '(objectclass=*)' namingContexts
```

If the server has been configured correctly, you should see output similar to the following (with different `dc`s of course):

```
version: 2
#
# filter: (objectclass=*)
```

```
# requesting: namingContexts
#
#
dn:
namingContexts: dc=syroidmanor,dc=com

# search result
search: 2
result: 0 Success

# numResponses: 2
# numEntries: 1
```

If you get errors or output vastly different, go back and check your configuration. To have the LDAP service automatically start on reboot, enter the following:

```
[root@thor root]# chkconfig ldap on
```

Again, the above is specific to Red Hat distributions.

Configuring ACLs

The final step in configuring your LDAP server is to set up some basic access controls. Doing so ensures that users will only be able to access entries they need to access.

There are two methods for setting ACLs (Access Control Lists) under OpenLDAP: you can put an `include` line at the top of the `/etc/openldap/slapd.conf` pointing to a separate file (for example, `include /etc/openldap/slapd.access.conf`); or you can add the ACLs directly to `slapd.conf`. The choice is entirely up to you -- Mandrake typically uses an `include` line; Red Hat adds ACLs to the configuration file.

You'll find a set of example ACLs and an explanation of what they do on the next panel.

An ACL example

```
# Define ACLs -- access control definitions

access to dn=".*,dc=syroidmanor,dc=com" attr=userPassword
    by dn="cn=root,dc=syroidmanor,dc=com" write
    by self write
    by * auth

access to dn=".*,dc=syroidmanor,dc=com" attr=mail
    by dn="cn=root,dc=syroidmanor,dc=com" write
    by self write
    by * read

access to dn=".*,ou=people,dc=syroidmanor,dc=com"
```

```
by * read  
  
access to dn=".*,dc=syroidmanor,dc=com"  
by self write  
by * read
```

The above configuration allows only the owner of the `userPassword` attribute to modify the entry, but only when the owner supplies his or her prior password. In all other cases, the entry can only be accessed for authentication purposes, and cannot be viewed. The second `access to...` entry allows users to modify their own e-mail address (`attr=mail`). The third entry specifies that any DN in `ou=people,dc=syroidmanor,dc=com` is read-only to everyone but the `rootdn`. This prevents users from changing their username, uid, gid, home directory, etc. Finally, the last entry is a safety "catch all" for anything else not covered in the preceding access controls. It would, for example, allow users to change entries in their own address books.

Before the server can use the new ACLs, a restart is required: **`service ldap restart`**.

With the base configuration complete, it's time to populate the database with some useful entries.

Section 4. Populating your LDAP directory

Getting from here to there

At this stage of the game you should have a general handle on LDAP's internal mechanisms and structures, and have a functioning OpenLDAP server. The next step is to populate your directory with contact data, which will subsequently be used by an e-mail application to look up e-mail addresses. Unfortunately, this is where things get a bit tricky.

There are three basic ways to get contact information into the directory tree: Manually from the command line, importing via an LDIF (LDAP Database Interchange File), or by using a script. The tricky part is in choosing a method that works, and getting the data into the database without error. The reward at the end of the rainbow is that, once accomplished, you'll never have to go through the process again -- providing, of course, you continue to use LDAP-aware applications.

Populating your database manually is the most straightforward of the three approaches (although, as the word "manually" implies, it's also the most labor intensive), so we'll tackle this procedure first.

Manual data entry

To begin, from a console window or the command line, enter the following:

```
[root@thor root]# ldapadd -D "cn=root" -h
server
password: *****
dn: uid=juser,ou=people,dc=syroidmanor,dc=com
uid: juser
cn: Joe User
givenname: Joe
sn: User
mail: juser@syroidmanor.com
objectClass: top
objectClass: mailRecipient
objectClass: person
objectClass: inetOrgPerson
^D

adding new entry uid=juser,ou=people,dc=syroidmanor,dc=com
[root@thor root]#
```

The process outlined above uses three basic LDAP operations: a binding operation, an update operation, and an implicit unbind operation. You must *bind*, or connect, to the LDAP server as a privileged user in order to modify the directory. The example shown uses `cn=root` as that's how our OpenLDAP server is configured. If you used another name for your `rootdn` entry in `slapd.conf`, substitute it as applicable.

After the password prompt, enter a DN, followed by the data you want associated with the DN (RDN entries), followed by the object classes that contain the attributes for the type/value pairs. The CTRL-D at the end of the procedure sends the data to the server and implicitly unbinds the server. The LDAP server then responds with either a message that the data has been successfully entered (shown), or an error. Common errors are trying to add a type/value without specifying the correct object class, adding a user or RDN that already exists, forgetting a "MUST" entry (for example, the object class person requires both `givenname` and `sn`), etc.

Manual data entry, continued

Also, with manual data entry, be aware that:

- You must know which object class holds the attributes for the type/value RDN data you're adding.
- The process is relatively labor intensive.
- It's easy to type an entry incorrectly, which leaves you with inaccurate information in your directory tree.
- Generally speaking, it's important to be able to both visualize your directory tree layout and be familiar with the schemas the LDAP server is configured to use.

The last point is common across all data entry methods, which was the purpose behind the [A primer on LDAP](#) on page 4 section. Familiarity with LDAP's structures and a clear vision of what it is you're trying to accomplish go a long way toward removing the frustration and inevitable errors associated with populating an LDAP database.

The LDIF approach

The second approach to inserting data into an LDAP directory is to use an LDIF file. An LDIF file is simply a plain text document containing the data you want to insert, laid out in a specific syntax. You're already familiar with the syntax: `dn:` followed by location in the tree to store the entry, followed by one or more RDN entries (the type/value pairs containing the data), followed by the requisite object classes. To create an LDIF, use a plain text editor and enter the data you want to add to the directory. Using our previous example:

```
dn: uid=juser,ou=people,dc=syroidmanor,dc=com
uid: juser
cn: Joe User
givenname: Joe
sn: User
mail: juser@syroidmanor.com
objectClass: top
objectClass: mailRecipient
objectClass: person
objectClass: inetOrgPerson
```

Save the file (say, `example.ldif`), and in a console window or at the command line, type:

```
[root@thor root]# ldapadd -x -D "cn=root,dc=syroidmanor,dc=com" -W -f sample.ldif
```

You'll be prompted for the `rootdn` password, and upon authentication, the data contained in the LDIF will be written to the LDAP database.

Advantages and disadvantages of the LDIF method

There are advantages and disadvantages to the LDIF method. On the plus side:

- You can check your spelling and syntax before importing the file into the database.
- You can create an LDIF file with a large number of entries and add them to the directory in one operation.
- If the import fails, simply open the LDIF file, find the error(s), and try re-importing it.
- LDIF files are an open standard and can be imported into just about any directory server.

On the negative side of things:

- The process is still relatively labor intensive -- all entries in the LDIF must be typed in, and correct syntax observed.
- The LDAP server is not always kind when it encounters errors in the import file. While you might get a "syntax error" message, it won't tell you *where* -- in a potentially very large LDIF file -- the error is.

The bottom line is this: LDIF files have certain distinct advantages over manually entering your data from the command line. But you still have to type your contact information into a file, follow correct syntax, and import it into the directory. Is there not a more automated method to populate an LDAP database? Well, yes and no -- read on.

The script approach

Scripts are available (typically written in Perl or PHP) that purport to take your data and "automagically" place it into an LDAP directory. This approach has two problems. First and foremost, any of the scripts I've personally tried are notoriously buggy, and in some worst-case scenarios, either damage your data during import or corrupt the directory tree itself. Second, using a script to import data assumes the data already exists in some form. This is all well and good when it comes to importing user passwords and group information from `/etc/passwd` and `/etc/groups` respectively, but chances are your contact information is not in a commonly recognized format. After all, the primary purpose of this tutorial is to put contact information someplace that is not subject to proprietary formats.

What if you export your contact data to a plain text, comma-delimited file and find a script that is capable of importing the data to an LDAP directory? If you can find such a script, and if it works as advertised, all the power to you. Keep in mind that your e-mail client might have its

own rendition of what exactly a "comma-separated file" is. Are carriage returns added to the end of each line? How does the exporter deal with spaces embedded in a field? LDAP import scripts are typically hacked together by someone who wants to transfer his or her data from Application A (which has been exported in format X), into an LDAP directory. If their application and export format meshes with yours, try it. Just make sure you back up your directory database first so you can return to a "known good" point if the import fails.

GUI directory management

While most administrators prefer to work with LDIF files and OpenLDAP's command line interface, there are several GUI directory management programs available for those inclined to try them. Two examples:

- Directory Manager (see [Resources](#) on page 22) is an open source graphical tool for managing LDAP directory data. It includes a schema-independent editor, a schema viewer, and a number of different ways to view your data. In addition, Directory Manager has provisions for creating your own custom views. Directory Manager requires an LDAP v3 server. Despite the early version number (0.91), the product is very usable, and based on personal experience, safe to use in a production environment.
- LDAP Browser/Editor (see [Resources](#) on page 22), developed by Jarek Gawor, is another directory management utility. It is more mature and feature-rich than some similar tools (the current version is 2.8.1) that allow the user to browse, search, and edit directory entries. It is a paid-for product, however, and requires the user to purchase a licence. Here is a screenshot of Jarek's LDAP Browser/Editor:

Section 5. Creating an LDAP address book with Rolodap

Introducing Rolodap

For all those individuals who aren't keen on populating an LDAP directory manually or with scripts, there is an alternative: Rolodap (see [Resources](#) on page 22 for information and links). Rolodap is an open source project that uses an LDAP directory to store contact information. In addition, Rolodap provides a PHP-driven Web interface that allows you to maintain your address book (multiple address books are permitted as well) from any Web browser.

What's the catch? None really, except that in order to use the Web interface, you'll need to install and configure three additional packages -- Apache, PHP, and PHP-LDAP. This section details how to install and configure Rolodap.

Package requirements

Rolodap requires the following packages:

- `openldap-servers`
- `openldap`
- `openldap-devel`
- `openldap-clients`
- `apache`
- `php`
- `php-ldap`

Version numbers are intentionally missing from the above list, as both Apache and PHP will both likely have newer releases by the time you read this. Check the Red Hat site for the latest stable packages (see [Resources](#) on page 22 for a link).

Once you have everything installed, check to ensure that both Apache and PHP are working correctly before going any further.

The following configuration steps assume:

- Apache's DocumentRoot is `/home/httpd/html`.
- Your OpenLDAP configuration files are in `/etc/openldap`.
- You are logged in as root.
- Your OpenLDAP server will only hold data entries from Rolodap.
- You are only using Apache to run the Rolodap interface and not serving public Web pages.

This last issue is important -- running an LDAP interface on a public Web server demands security provisions beyond the scope of this tutorial.

Download Rolodap

The current release of Rolodap is version 1.0 (see [Resources](#) on page 22 for download link). Download the gzipped tarball (plus the accompanying *schema*, *ldif*, and *conf* files) to a directory of your choosing, move the file to Apache's DocumentRoot, unpack it, and rename the directory created as shown below:

```
mv /temp/rolodap-v1.tar.gz /home/httpd/html
cd /home/httpd/html
tar xvzf rolodap-v1.tar.gz
mv rolodap-v1 rolodap
```

Edit rolodap.conf

The next step is to configure Rolodap. Copy the `rolodap.conf` file to `/etc` (using the example paths provided in the previous panel, `cp /home/httpd/html/rolodap/config/rolodap.conf /etc`). Open this file with your favorite text editor (`vi /etc/rolodap.conf`), read through the comments, and edit the entries as required for your implementation.

See the Rolodap README (`/home/httpd/html/rolodap`) for details on the database layout.

Configure the OpenLDAP server

Rolodap, at its current revision level, is designed to be the only data stored on your OpenLDAP server. If this matches your scenario, copy `core.schema` and `rolodap.schema` from `/home/httpd/html/rolodap/` to `/etc/openldap/schema`. Rolodap also includes a sample `slapd.conf` file located in the `/home/httpd/html/rolodap/config` directory. Either copy this file to `/etc/openldap` (make a backup of your original FIRST), or note the required changes and edit your original `slapd.conf` as necessary. Again, the README file bundled with Rolodap contains details on the required changes.

If you are using OpenLDAP to store other data in addition to Rolodap entries, you will probably have to tweak the two schema files and the config file bundled with Rolodap to make everything work as advertised.

Next, ensure the file permissions on `/etc/openldap/slapd.conf` are correct. The OpenLDAP server bundled with Red Hat 7.3 runs as user `ldap`, so you must make sure this user can read the file:

```
chgrp ldap /etc/openldap/slapd.conf
chmod g+rw /etc/openldap/slapd.conf
```

Finally, restart the OpenLDAP server to effect the changes made: `service ldap restart`.

Populating the LDAP directory

The next step is to configure and populate an initial Rolodap database. Edit the `sample.ldif` file located in `/home/httpd/html/extras` to accommodate your desired configuration. Specifically you'll need to change the DN entries, the admin information, UIDs to suit your system configuration, etc. Remember, `dn: o=contacts.company.net` is just an alternative to `dn: ou=contacts,dc=company,dc=net`. If you're more comfortable with the latter convention, edit `sample.ldif` as appropriate. When you're satisfied, enter the following command to enter the information contained in `sample.ldif` into the OpenLDAP directory:

```
ldapadd -f sample.ldif -xv -h yourldapservers  
-D"cn=admin,dc=yourdomain,dc=com" -w password
```

Replace "yourldapservers" with the server name, "yourdomain" with your domain or organization, and "password" with the password entry you used for the admin user in `sample.ldif`. If you receive errors during the update process, carefully check the spelling and syntax of all entries in your `sample.ldif`.

Finally, fire up your browser...

The final step is to try out Rolodap by perusing your new OpenLDAP directory. Direct your browser at `http://localhost/rolodap`. A login page should appear. Type the username and password you entered in `sample.ldif` (for the regular user, not the admin user) and spend some time checking out Rolodap's various features, add some new contacts, or create a new address book. The screenshots below show the interface for a simple search (top) and a user's "home page" (bottom).

If you experience difficulty accessing or logging into Rolodap, check your Apache log file for any clues as to the source of the problem.

So, can you use a Rolodap-based LDAP directory as an address book for your favorite e-mail client? You betcha... read on; the next section has all the details.

Section 6. Configuring an LDAP-aware e-mail client

Configuration overview

This final section details the final step in accessing contact data stored in an LDAP directory -- configuring your e-mail client to search the LDAP directory rather than the application's default address book. It would be unrealistic to try to provide specific details for every e-mail client on the market, but there are several distinct pieces of information all LDAP-aware applications require before they can connect to a directory service:

- The name of the server
- A location, or *search base*, to begin searching for the requested information
- The port number the LDAP server is "listening" on (the default is 389)
- A user name and password
- And for some applications, a *search rule* (this typically defaults to "*", meaning all records)

The next few panels outline the necessary procedures for three commonly used e-mail clients: Pine, Mulberry, and Outlook Express. As you'll see, each program has its own quirks, but they all follow the same general process.

Pine

Many UNIX diehards refer to Pine as the "One True mail program." Pine is a console/text-based client and is included in most major Linux distributions. The following instructions are based on Pine version 4.44 running on Red Hat 7.3.

Save your changes. Now, when you type part of an e-mail address when composing a message, Pine should find a match for the entry and complete it. For entries with more than one match, a popup will appear allowing you to select which address to use.

Mulberry

Mulberry (see [Resources](#) on page 22 for a link) has been around for years, runs on a wide range of platforms, and is one of those "hidden gems" -- few people have heard of it, but those who use it, swear by it. To configure Mulberry to use an LDAP directory as its default address book:

1. From any open window, select File --> Preferences.
2. Click the Accounts tab, and from the Accounts drop-down list, choose New.
3. Select New, type in an account name, and from the Account Type drop-down, click on LDAP Address Search.
4. On the Authentication tab, choose a Method.
5. On the Options tab, check Use when Searching, and if you so desire, Use with Address Expansion.

6. Under Attributes 1 and 2, fill in any field appropriate to your installation. The one field you'll need is Root. Using the example shown throughout this section:
cn=people,dc=syroidmanor,dc=com.

Outlook Express

To migrate your Outlook Express address book to this open standard, follow these steps:

1. Go to the Tools menu and select Address Book.
2. From the Address Book window, go to the Tools menu and click Accounts.
3. On the right side of Internet Accounts dialog, select the Add button.
4. The Internet Connection Wizard will appear. You'll need to enter the name of your Internet directory (LDAP) server and select "My LDAP server requires me to logon. On the next dialog, select "Yes" under "Do you want to check addresses using this directory service". Finally, click Finish to complete the Wizard.
5. Before closing the Internet Accounts dialog, click the Directory Services tab and ensure the entry just created is at the top of the list shown.

Section 7. Summary, resources, and feedback

Summary

An LDAP directory service is a powerful solution for anyone with a need or desire to store specific types of information in a centralized location. LDAP's tree/leaf structure is especially well-suited to fast searches on data that does not change frequently, such as user authentication, DNS lookups, and user/group information. This tutorial has focused on implementing an LDAP-based address book that could then be used by an e-mail client to search for contact details.

The following topics were discussed:

- The technology behind LDAP, directory structures, object classes, schemas, etc.
- Installing and configuring OpenLDAP
- Configuring an OpenLDAP server
- Populating an OpenLDAP server with data
- Accessing and searching an LDAP database
- Configuring an e-mail client to look up e-mail addresses

As you've discovered, setting up a functional LDAP directory server is a large, diverse topic, highly dependent on the needs of the implementation. The next two panels contain a multitude of additional resources, both introductory and advanced, that provide a small snapshot of the almost limitless capabilities of an LDAP directory service.

Resources

Due to the current popularity of LDAP, there are literally thousands of online resources on the topic ranging from basic introductions to the protocol to topic-specific articles detailing schemas and object classes. Depending on your experience and background, one or more of the following resources may help you find more information on the technology behind, or the implementation of, LDAP.

Tutorial-specific resources:

These resources are directly related to topics covered in the tutorial:

- The [OpenLDAP 2.1 Administrator's Guide](#) is a comprehensive introduction to installing and configuring an OpenLDAP server, available online from [OpenLDAP.org](#). The document provides background on OpenLDAP's history and relationship to other directory service offerings, how to build and install OpenLDAP from source, the structure and options available in the `slapd.conf` configuration file, replication, and SASL/TLS authentication.
- For more information on LDAP-related RFCs, see the [LDAPman RFC page](#).
- For more information on registering an OID, check with the [Internet Assigned Numbers](#)

Authority (IANA).

- For more information on defined object classes and their attributes, check out the [LDAPman Schema reference page](#) or the extremely useful [LDAP Schema Viewer](#) Web site.
- Another good resource is the [Introduction to LDAP](#) written by one of the staff at *Plugged In Software*.
- The [Red Hat site](#) provides more information on Red Hat Linux.

Introductory resources:

- Michael Donnelly's excellent article [An Introduction to LDAP](#) discusses "Why use LDAP," LDAP directory structures, LDAP records, customizing object classes, and LDAP replication.
- On the O'Reilly Network you'll find an introductory piece, [Getting Started with LDAP](#), by Luke A. Kanies. Kanies discusses LDAP structures and object classes in the context of implementing a directory database that consolidates access, authentication, and authorization (AAA, or Triple-A) information.
- The IBM Redbook [Understanding LDAP](#) targets readers who need to understand the basic principles and concepts of LDAP. The Redbook avoids discussion of vendor-specific products, sticking instead to the technology behind the LDAP protocol and how that technology applies to directory services.
- LDAP is not just for contacts. Read about how an IBM team used [LDAP to create a robust Linux authentication server](#).
- If you're using Windows services, take this tutorial to learn how to [configure LDAP for user authentication for a Samba primary domain controller](#).

Advanced resources:

- For more information on schemas, see Tom Thornton's [Directory schema notes](#) page.
- For more information on OIDs (object identifiers), check out [Harald Avelstrand's OID page](#).

Address book-related LDAP projects and products:

- On the [Rolodap Project home page](#), you'll find an overview of the project, a link to the project's discussion forums, a download link, and a public demo of Rolodap.
- The [current release](#) of Rolodap is version 1.0, available at the SourceForge site.
- Check the [Red Hat site](#) for the latest stable packages.
- [LDAP-abook](#) is an LDAP-based address book similar to Rolodap. It too uses a Web browser interface to maintain a contact database, but while Rolodap is driven by PHP, LDAP-abook uses Perl as the scripting language.
- For a good example of the flexibility of LDAP, check out the [ldap2dns](#) project page. Here you'll find instructions for storing DNS information in an LDAP directory.
- [Directory Manager](#) is an open source graphical tool for managing LDAP directory data.
- [Directory Manager](#) is an open source graphical tool for managing LDAP directory data.
- [LDAP Browser/Editor](#) is a mature and feature-rich directory management utility; you must purchase a licence to use it.

- The [Mulberry e-mail client](#) has been around for years and runs on a wide range of platforms.

Books:

For those who prefer paper-based reference material, check out any of the following titles:

- *Implementing LDAP*, by Mark Wilcox (Wrox Press); ISBN: 1861002211.
- *Ldap: Programming Directory-Enabled Applications With Lightweight Directory Access Protocol* (MacMillan Technology Series), by Tim Howes and Mark Smith; ISBN: 1578700000.
- *Internet Directories: How to Build and Manage Applications for LDAP, DNS, and Other Directories* (Prentice Hall PTR), by Bruce Greenblatt; ISBN: 0139744525.
- *Windows 2000 Active Directory Service* (O'Reilly and Associates), by Alistair G. Lowe-Norris; ISBN: 1565926382.

Your feedback

Please send us your feedback on this tutorial. We look forward to hearing from you! You may also contact the author directly at dwcomments@syroidmanor.com.

Colophon

This tutorial was written entirely in XML, using the developerWorks Toot-O-Matic tutorial generator. The open source Toot-O-Matic tool is an XSLT stylesheet and several XSLT extension functions that convert an XML file into a number of HTML pages, a zip file, JPEG heading graphics, and two PDF files. Our ability to generate multiple text and binary formats from a single source file illustrates the power and flexibility of XML. (It also saves our production team a great deal of time and effort.)

You can get the source code for the Toot-O-Matic at www6.software.ibm.com/dl/devworks/dw-tootomatic-p. The tutorial [Building tutorials with the Toot-O-Matic](#) demonstrates how to use the Toot-O-Matic to create your own tutorials. developerWorks also hosts a forum devoted to the Toot-O-Matic; it's available at www-105.ibm.com/developerworks/xml_df.nsf/AllViewTemplate?OpenForm&RestrictToCategory=11. We'd love to know what you think about the tool.