Putting Directories to Work

LDAP

System Administration

O'REILLY®

Gerald Carter

LDAP System Administration

LDAP System Administration

Gerald Carter

O'REILLY®

Beijing • Cambridge • Farnham • Köln • Paris • Sebastopol • Taipei • Tokyo

CHAPTER 7

Email and LDAP

One of the most important applications of a directory is storing email addresses and other contact information. Although many ad hoc solutions to this problem have been implemented over the years, LDAP provides a natural online publishing service for this type of data. This chapter explores the ins and outs of integrating email clients (MUAs) and mail servers (MTAs) with an LDAP directory. It covers the configuration details of some of the more popular email clients, including Mozilla Mail, Pine, Microsoft Outlook, and Eudora. We'll also discuss the schema required to support these clients and the types of LDAP searches to expect when the application attempts to locate a user in the directory.

On the server side, we'll discuss three popular email servers—Sendmail, Postfix, and Exim—all of which can use a directory. We will cover the level of LDAP support within each MTA, the schema needed to support this integration, and the configuration process for integrating an LDAP directory into a production email environment. This discussion assumes that you are familiar with basic MTA administration and the interaction between SMTP servers.

Representing Users

The server you will build combines the white pages server you created in Chapter 4 and the server for administrative databases you created in Chapter 6 as a replacement for NIS. You already have a head start on integrating user account information because both servers used the `ou=people` container for storing user account information. With only a few modifications to your directory, the `posixAccount` and `inetOrgPerson` object classes can be used to store a single user entry for both authentication and contact information.

Here's an entry for "Kristi Carter," which is similar to those presented in Chapter 4:

```
dn: cn=Kristi W. Carter,ou=people,dc=plainjoe,dc=org
objectClass: inetOrgPerson
cn: Kristi W. Carter
```

```
sn: Carter
mail: kcarter@plainjoe.org
labeledURI: http://www.plainjoe.org/~kristi
roomNumber: 102 Ramsey Hall
telephoneNumber: 222-555-2356
```

In Chapter 6, this same user might have been presented as:

```
dn: uid=kristi,ou=people,dc=plainjoe,dc=org
uid: kristi
cn: Kristi Carter
objectClass: account
objectClass: posixAccount
userPassword: {crypt}LnMJ/n2rQsR.c
loginShell: /bin/bash
uidNumber: 781
gidNumber: 100
homeDirectory: /home/kristi
gecos: Kristi Carter
```

Looking at both examples side by side, some differences can be noted. The first is that the RDN used for each entry is different. It doesn't really matter whether you choose `cn=Kristi W. Carter` or `uid=kristi`. Since Unix accounts must already possess a unique login name, the `uid` attribute is a good choice to prevent name conflicts in `ou=people`.

The second issue is more serious and shows why the initial directory design should not be rushed. Both the `account` and `inetOrgPerson` object classes are structural object classes. Remember that an entry cannot have more than one structural object class and that once an entry is created, its structural class cannot be changed. Some LDAP servers may allow you to reassign an entry's object classes at will, but do not rely on this behavior.

To solve this dilemma, initially create each entry with the `inetOrgPerson` class and then extend it using the `posixAccount` auxiliary class. The means that the account entry will have to filtered from the output of PADL's migration scripts—a simple task using *grep*:

```
$ ./migrate_passwd.pl /etc/passwd | \
  grep -iv "objectclass: account" > passwd.ldif
```

The combined user entry now appears as:

```
dn: uid=kristi,ou=people,dc=plainjoe,dc=org
objectClass: inetOrgPerson
objectClass: posixAccount
cn: Kristi Carter
cn: Kristi W. Carter
sn: Carter
mail: kcarter@plainjoe.org
labeledURI: http://www.plainjoe.org/~kristi
roomNumber: 102 Ramsey Hall
telephoneNumber: 222-555-2356
```

```
uid: kristi
userPassword: {crypt}LnMJ/n2rQsR.c
loginShell: /bin/bash
uidNumber: 781
gidNumber: 100
homeDirectory: /home/kristi
gecos: Kristi Carter
```

One final note before we begin looking at specifics of email integration: the `mail` attribute is optional in the `inetOrgPerson` schema definition. However, it's clearly mandatory when you're trying to support mail clients and servers.

Email Clients and LDAP

When planning a strategy for supporting an application with a directory, you always start by examining the application and determining what schema has the ability to support it. Using a standard schema is vastly preferable to building your own. Of course, with email you don't have the ability to specify what client users will use: at your site, many different clients are probably in use, and you won't make friends by asking users to change. In this section, we'll look at four clients, all of which are in common use: Mozilla Mail, Pine from the University of Washington, Qualcomm's Eudora, and Microsoft's Outlook Express. Fortunately, the `inetOrgPerson` schema supports all of the information items we are concerned with using in this section.

The following parameters are common to all clients:

- The LDAP server is *ldap.plainjoe.org*.
- The base search suffix is `ou=people,dc=plainjoe,dc=org`.

Beyond the basic LDAP search parameters and supporting schema, it is imperative to know what version of LDAP the clients will use. Table 7-1 reveals that 3 out of the 4 mail clients listed use LDAPv2 to bind to the directory server. This means that you must explicitly add support for these connections as OpenLDAP 2.1 rejects LDAPv2 binds in default configurations. Add the following line to the global section of *slapd.conf*:

```
## Allow LDAPv2 binds from clients needed by several mail client packages.
allow      bind_v2
```

then restart the OpenLDAP server to make it recognize the change.

Table 7-1. LDAP versions used by various mail clients

Mail client	LDAPv2 bind	LDAPv3 bind
Mozilla Mail	✓	
Pine 4	✓	
Eudora	✓	
Outlook Express		✓

Mozilla Mail

In 1998, Netscape Communications announced that it would give away the source code to the next generation of Netscape Communicator browser suites. In the Fall of 2002, the 1.0 release of Mozilla finally saw the light of day. Today, this code base is still alive, well, and growing at *http://www.mozilla.org/*. Versions of the browser suite are available for various flavors of Unix, Windows, and Mac OS.

When configuring Mozilla's address book client to access a directory, you must keep two questions in mind:

- Should users be required to authenticate themselves, or should they be able to access directory information anonymously?
- Should the information sent to and retrieved from the LDAP server be sent in clear-text form (i.e., LDAP), or should it be transmitted over SSL (i.e., LDAPS)?

The simplest method of adding a new directory server profile in Mozilla is through the Address Book application shown in Figure 7-1. Select File → New → LDAP Directory to launch the Directory Server Properties dialog shown in Figure 7-2.

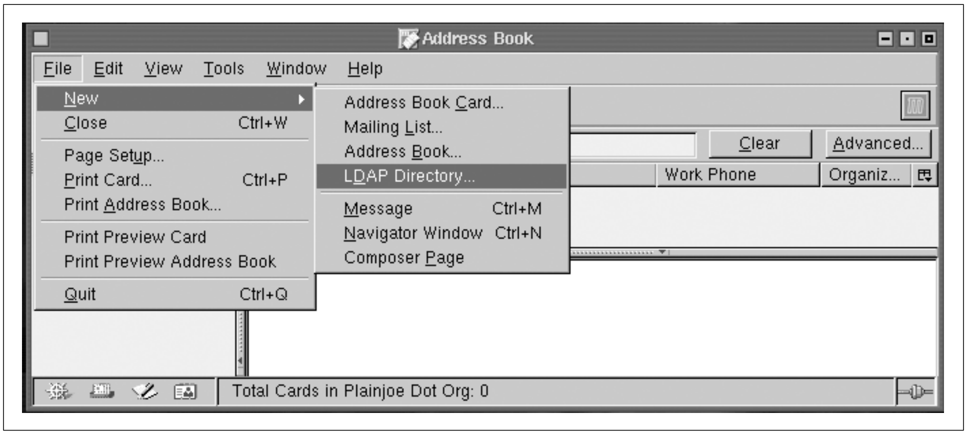
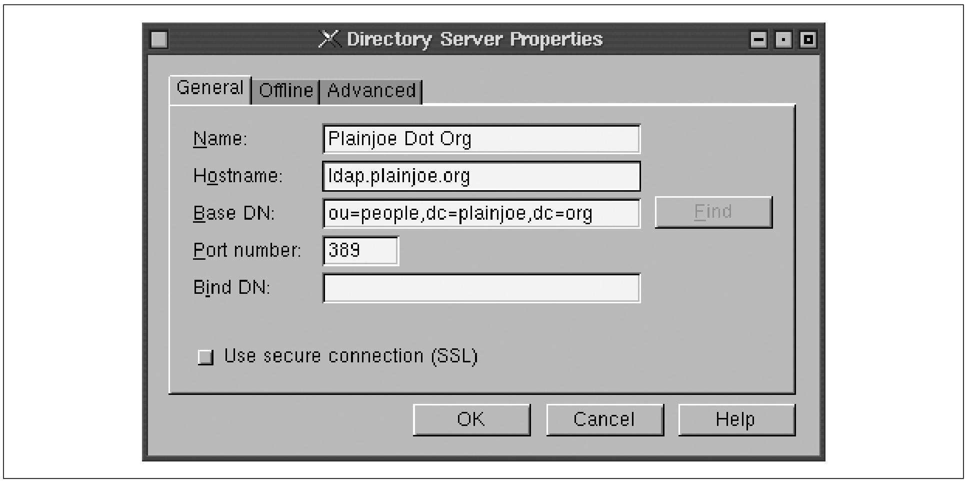


Figure 7-1. Adding a new LDAP directory to the Mozilla Address Book

The Name field lets you provide a descriptive title name for the directory (Plainjoe Dot Org); this name is used in the address book display window, but has no other effect on directory lookups. Put the hostname or IP address of the directory server in the Hostname field (*ldap.plainjoe.org*). Set the Base DN to the base search suffix used when querying the server (`ou=people,dc=plainjoe,dc=org`). The Port number, which defaults to port tcp/389 for non-SSL directories, should be set to the port on which the LDAP server is listening. By default, Mozilla uses an anonymous bind when searching the server. If a simple bind is preferred, you can define the Bind DN to use



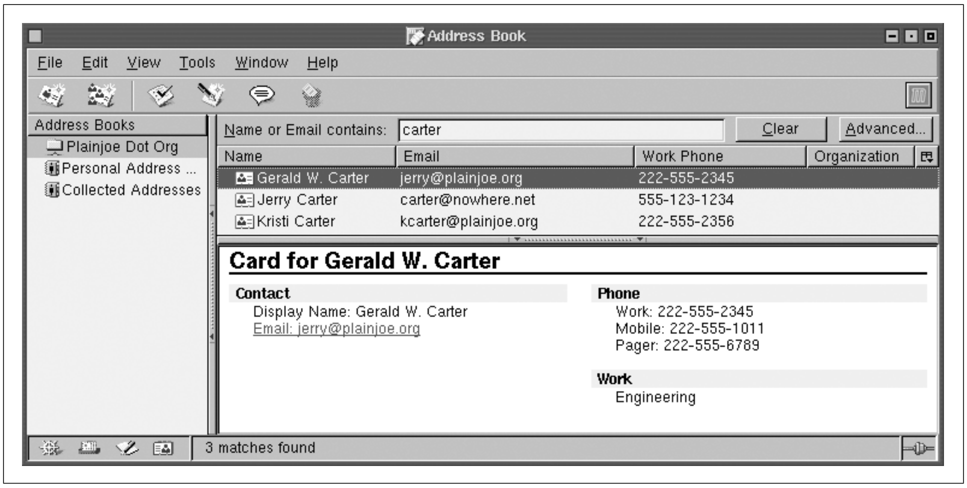
Name: Plainjoe Dot Org
Hostname: ldap.plainjoe.org
Base DN: ou=people,dc=plainjoe,dc=org
Port number: 389
Bind DN:
Use secure connection (SSL)

Figure 7-2. Setting directory search parameters

for authentication. Mozilla will prompt you for the corresponding password before it actually performs a search.

Once the directory has been added to the list of address books, you can query the directory by entering a substring to search for, or by using some of the more advanced search dialogs. Figure 7-3 shows the basic substring search test entry box. Any text you enter in this box is used to query the `cn`, `sn`, `givenName`, and `mail` attributes using a subtree search scope. For example, if you enter the text “carter”, the client uses this search filter:

```
(|(mail=*carter*)(cn=*carter*)(givenName=*carter*)(sn=*carter*))
```



Name or Email contains: carter

Name	Email	Work Phone	Organization
Gerald W. Carter	jerry@plainjoe.org	222-555-2345	
Jerry Carter	carter@nowhere.net	555-123-1234	
Kristi Carter	kcarter@plainjoe.org	222-555-2356	

Card for Gerald W. Carter

Contact
Display Name: Gerald W. Carter
Email: jerry@plainjoe.org

Phone
Work: 222-555-2345
Mobile: 222-555-1011
Pager: 222-555-6789

Work
Engineering

3 matches found

Figure 7-3. Basic search screen in the Mozilla Address Book

A simple way to determine the search filters used by any LDAP client is to enable connection logging on your directory server. OpenLDAP uses log level 256 for this purpose. Another possibility is to use a network traffic–monitoring tool such as Ethereal that can decode LDAP requests and replies. More information about Ethereal can be found at *http://www.ethereal.com/*.

The advanced search dialog box (Figure 7-4) allows you to create more elaborate searches. The string entered as the search characteristic is the text for which you're searching; for example, the text "jerry" entered as part of the email address would result in the search filter of `(mail=*jerry*)`.

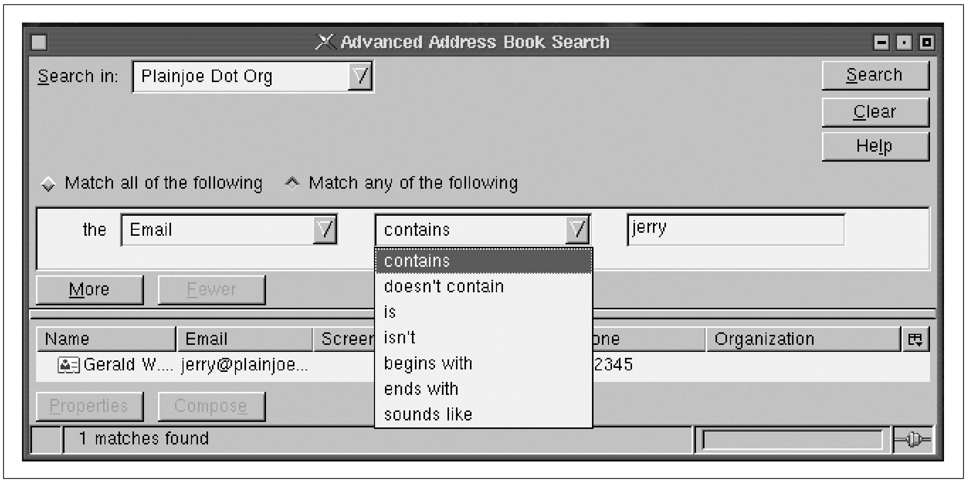


Figure 7-4. Advanced searching in the Mozilla Address Book

If the user selects the "Use secure connection (SSL)" checkbox on the directory properties window displayed in Figure 7-2, Netscape automatically changes the configured port number to tcp/636 (LDAPS). All traffic between the address book client and the directory server will be encrypted. Do not confuse this secure connection with the StartTLS LDAPv3 extension. Mozilla supports LDAPS only for secure communication with the directory server.

For this configuration to work, the LDAP server must be configured to support LDAPS and be listening on the port specified by the Server Port entry field. In addition, the Mozilla client must trust the certificate used by the LDAP server.

The procedure for configuring a directory server to support LDAPS varies from vendor to vendor; refer to your server documentation for details. In the case of OpenLDAP 2, you must generate a certificate and key file for *slapd* (refer to Chapter 3 for details on this) and then instruct the daemon to associate LDAPS with the correct port. The *-h* command-line option tells *slapd* which protocols to support. The command below starts *slapd* with support for LDAP and LDAPS on the default tcp ports 389 (for LDAP) and 636 (for LDAPS):

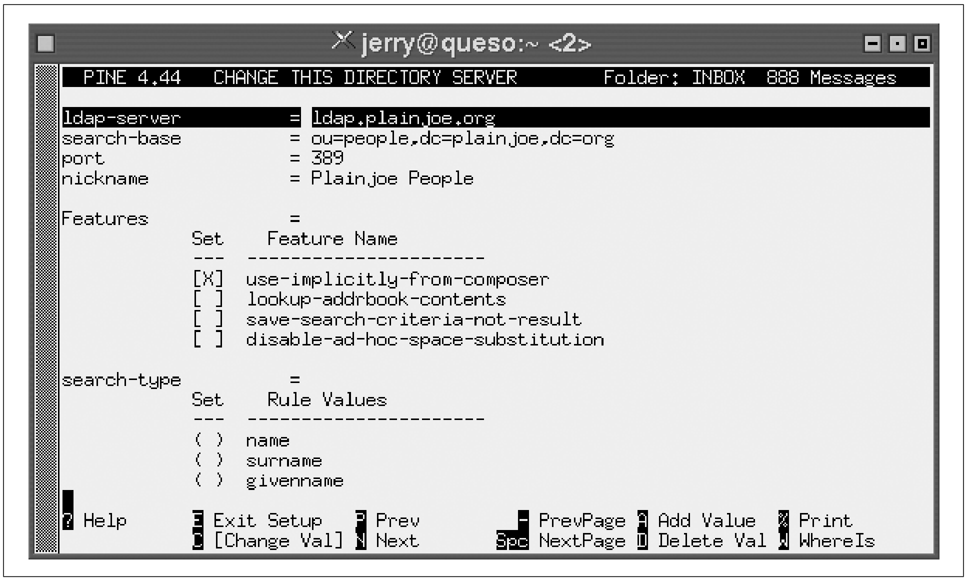
```
root# /usr/local/libexec/slapd -h "ldap:/// ldaps:///"
```

Unless the LDAP server's certificate can be verified by a certificate authority (CA), Mozilla will ask you whether it should trust the server before continuing the connection. If you decide that this will be too much trouble (or confusing) for your users, avoid self-signed certificates (or set up your own CA).

Pine 4

Pine is a popular, console-based email client developed by the University of Washington. The source distribution of the mail client is available at *http://www.washington.edu/pine/*; precompiled versions are available for most modern Unix and Unix-like systems. Support for retrieving addresses from an LDAP directory was introduced in Pine 4.00. A Windows version of Pine, known as PC-Pine, offers similar features to the Unix version, including LDAP support. PC-Pine is available from the University of Washington at *http://www.washington.edu/pine/pc-pine/*.

If you are using a precompiled version of Pine, you must ensure that LDAP support was enabled when the package was built. LDAP-enabled versions of Pine should allow you to configure a new directory from the Setup menu, as shown in Figure 7-5.



```
jerry@queso:~ <2>
  PINE 4.44   CHANGE THIS DIRECTORY SERVER       Folder: INBOX  888 Messages

ldap-server          = ldap.plainjoe.org
search-base          = ou=people,dc=plainjoe,dc=org
port                 = 389
nickname             = Plainjoe People

Features             =
            Set    Feature Name
            ---  ----------------------
            [X]  use-implicitly-from-composer
            [ ]  lookup-addrbook-contents
            [ ]  save-search-criteria-not-result
            [ ]  disable-ad-hoc-space-substitution

search-type          =
            Set    Rule Values
            ---  ----------------------
            ( )  name
            ( )  surname
            ( )  givenname

? Help      E Exit Setup   P Prev       - PrevPage A Add Value  % Print
            C [Change Val] N Next     Spc NextPage D Delete Val W WhereIs
```

Figure 7-5. Setup menu for Pine 4

Pine's directory configuration screen supports the common LDAP search parameters, including:

`ldap-server`
: The IP address or hostname of the directory server (*ldap.plainjoe.org*)

`search-base`
: The base search suffix (`ou=people,dc=plainjoe,dc=org`)

`port`
: The tcp port on which the server is listening (389)

`nickname`
: A descriptive display name (Plainjoe People)

Pine allows users to construct searches using up to four search attributes. By default, it uses the name (`cn`), surname (`sn`), given name (`givenName`), and email (`mail`) attributes. The default search filters only append the wildcard onto the end of the user's search string. For example, by default, Pine converts an email search for the string "kristi" into the search filter:

```
(|(cn=kristi*)(mail=kristi*)(sn=kristi*)(givenName=kristi*))
```

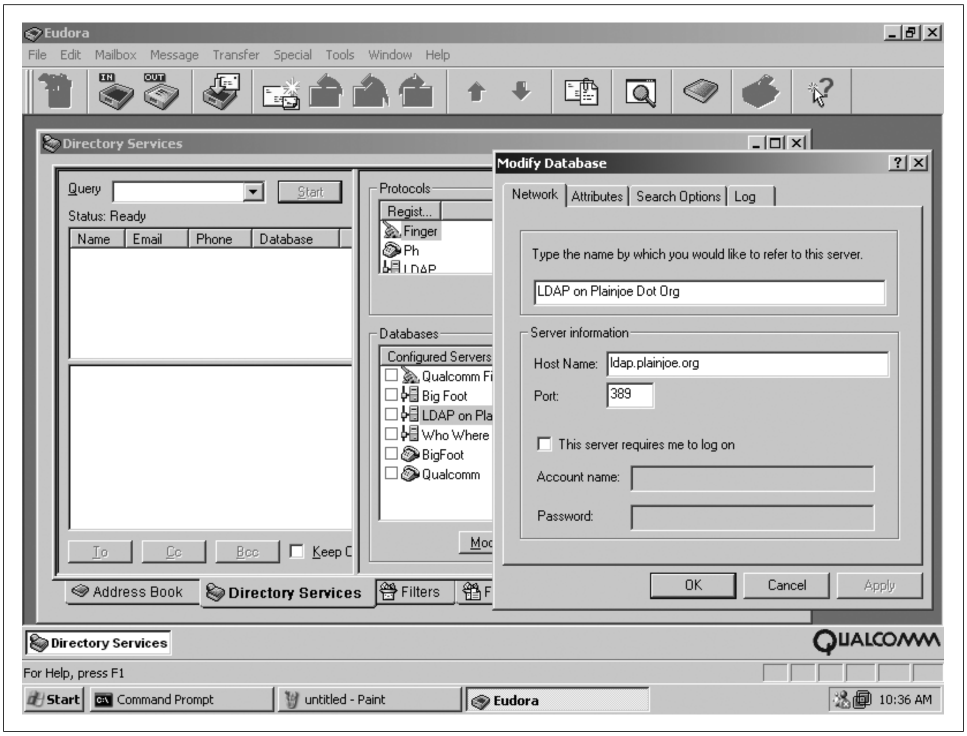
However, Pine offers a fair amount of flexibility for more adventurous users; you can change the kind of substring match (e.g., exact match, match at the beginning, match at the end, match anywhere), change the attributes Pine uses in the search, or specify a custom search filter. All of these settings can be accessed from Pine's directory configuration screen.

One shortcoming of Pine's LDAP implementation is the lack of support for LDAPS (and the fact that it uses LDAPv2). Although Pine supports SSL, it supports SSL only for POP or IMAP access to mailboxes. Pine cannot use SSL to access an LDAP directory.

Eudora

Qualcomm's Eudora has become a popular mail client on both Windows and Mac OS platforms. Configuring access to an LDAP directory in Eudora is similar to configuring LDAP for Mozilla. To start, go to the Modify Database window shown in Figure 7-6 by selecting the LDAP protocol from the directory window (Tools → Directory Services) and clicking the New Database button.

The Network tab allows you to specify connection information for the directory server. If you configure Eudora to use a login name and password to search the directory, the username must be in the form of a DN. The Search Options view shows that Eudora, by default, performs a substring match on the `cn` attribute; you can customize the search using standard search filter syntax. The Attributes dialog shown in Figure 7-6 provides a way to select which attributes are shown in response to a directory query. Eudora displays all the attributes for each entry that the search returns.



Type the name by which you would like to refer to this server.
LDAP on Plainjoe Dot Org

Server information
Host Name: ldap.plainjoe.org
Port: 389
☐ This server requires me to log on
Account name:
Password:

Figure 7-6. Modify Database dialog views used by Eudora 5.1 for Windows

You can define more descriptive names for attribute types using the Attributes window; for example, you can display the `cn` attribute type as "Real Name."

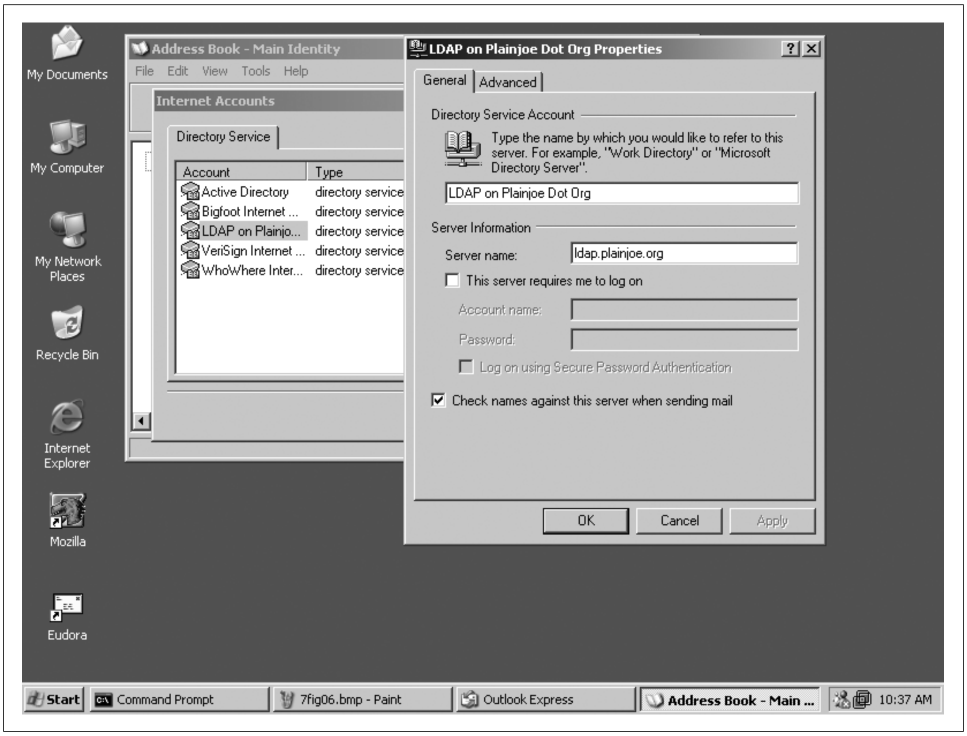
Eudora 5.1 does not support LDAPS when searching directories.

Microsoft Outlook Express

Microsoft Outlook in its various incarnations has become one of the most important mail clients on Windows networks. This section examines the version of Outlook Express that is included with Microsoft's Internet Explorer.

To configure Outlook Express to use a directory, start with the Directory Services configuration dialog, shown in Figure 7-7. To get to this dialog, select Tools → Accounts. The General tab contains settings for the directory server host information, such as the server's hostname and display name. The Advanced tab provides a means to define the port on which the server is listening and the base search suffix.

To search for someone in a directory, Outlook users go to the Find People dialog shown in Figure 7-8. Outlook Express uses a combination of the `cn`, `sn`, `mail`, and `givenName` attributes to generate the search filter. A search for the user "carter" is



Account	Type
Active Directory	directory service
Bigfoot Internet ...	directory service
LDAP on Plainjo...	directory service
VeriSign Internet ...	directory service
WhoWhere Inter...	directory service

LDAP on Plainjoe Dot Org Properties

General | Advanced

Directory Service Account

Type the name by which you would like to refer to this server. For example, "Work Directory" or "Microsoft Directory Server".

LDAP on Plainjoe Dot Org

Server Information

Server name: ldap.plainjoe.org

☐ This server requires me to log on

Account name:

Password:

☐ Log on using Secure Password Authentication

☑ Check names against this server when sending mail

Figure 7-7. The Directory Services configuration window from Outlook Express 5.5

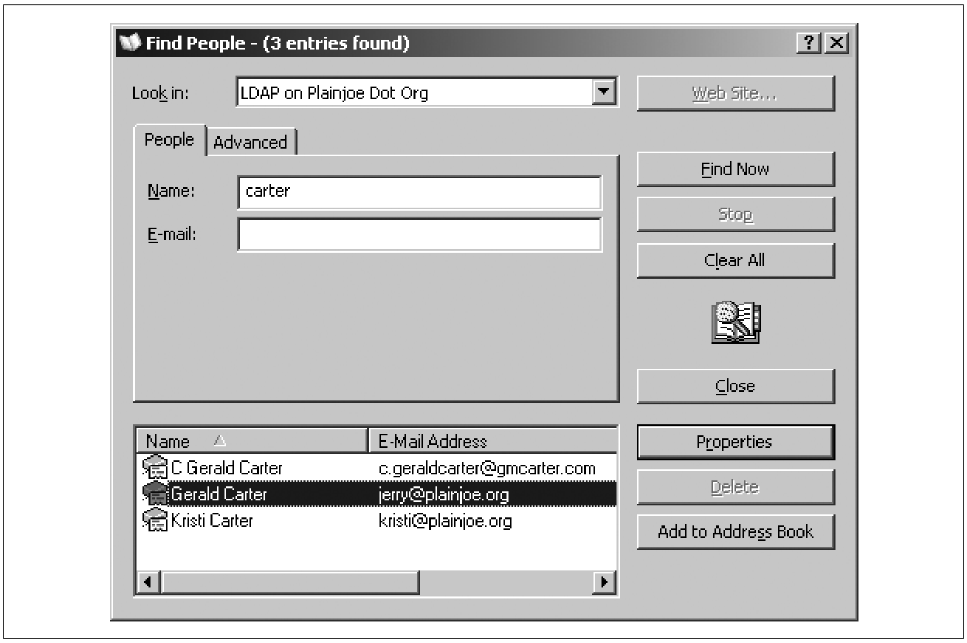
translated into a search filter very similar to the one used by both Mozilla Mail and Pine:

```
(|(mail=carter*)(cn=carter*)(sn=carter*)(givenname=carter*))
```

Searches can be customized using the Advanced tab of the Find People dialog in Figure 7-8.

Like both Mozilla and Eudora, Outlook Express can perform authenticated binds when searching a directory. If you check "This server requires me to logon," Outlook asks for a user ID and password, which it uses when binding to the server. Unlike Eudora, which expected a DN only for the username, Outlook Express supports two different styles of usernames. When using a simple bind, Outlook expects the login name to be the DN used in the bind request. However, if the "Logon using Secure Password Authentication" box is checked, Outlook negotiates with the directory server to use the NTLMv1 challenge/response authentication model (or possibly the GSSAPI SASL mechanism).

Refer to your directory server's documentation to determine whether it supports NTLM authentication. OpenLDAP does not currently support this feature.



LDAP on Plainjoe Dot Org

Name: carter

E-mail:

Name	E-Mail Address
C Gerald Carter	c.geraldcarter@gmcarter.com
Gerald Carter	jerry@plainjoe.org
Kristi Carter	kristi@plainjoe.org

Figure 7-8. Using the Find People dialog to search an LDAP directory

Outlook Express is a little more friendly than Mozilla when it comes to using LDAPS to connect to a directory server. On Windows 98, simply indicating that the directory server should be accessed using SSL on port 636 is enough. It is not necessary to tell Internet Explorer or Outlook to trust the self-signed certificate used by an OpenLDAP server. I'll let you decide whether this is a good thing; Mozilla was less trusting, requiring you to tell it to trust the LDAP server's certificate.

Mail Transfer Agents (MTAs)

The remainder of this chapter discusses LDAP support within several popular MTAs. You can skim this material if you want an overview of various mail servers, or you can focus on the details regarding your specific MTA and skip the others. In either case, I assume that you have some familiarity with the Simple Mail Transport Protocol (SMTP) and mail servers in general.

Before we begin, Table 7-2 provides a summary of the LDAP versions used by the mail servers presented in this section. The same rule for enabling LDAPv2 binds described in the beginning of this chapter still holds true for two out of the three mail servers listed.

Table 7-2. LDAP versions used by various mail servers

Mail transfer agent	LDAPv2 bind	LDAPv3 bind
Sendmail	✓	
Postfix		✓
Exim	✓	

Sendmail

Sendmail is the default MTA on most current versions of Unix. A number of alternatives have appeared in the past few years (such as Postfix, Qmail, and Exim), but if you work with Unix or Linux systems, chances are you'll deal with Sendmail. Sendmail introduced support for retrieving information from an LDAP directory in Version 8.9. However, this support didn't really stabilize until later versions (this discussion focuses on Version 8.12). It by no means attempts to give comprehensive coverage of Sendmail. For information on the details of configuring and running a Sendmail server, refer to *Sendmail,* by Bryan Costales and Eric Allman (O'Reilly).

Sendmail's LDAP integration falls into four categories: support for retrieving mail aliases from a directory, support for accessing generic Sendmail maps using LDAP queries, expansion of Sendmail classes at startup using information obtained from a directory, and support for retrieving specific mail-routing information from an LDAP directory. We will return to the specifics of these features in later sections.

In order to support any of these functions, Sendmail must be compiled with the *LDAPMAP* option enabled. Here's how to modify *site.config.m4* to compile LDAP against the client libraries installed by OpenLDAP 2:

```
dnl .../devtools/Site/site.config.m4
dnl Enable LDAP features in Sendmail during compilation
dnl
APPENDDEF (`confMAPDEF´, `-DLDAPMAP´)dnl
APPENDDEF(`confLIBS´, `-lldap -llber´)dnl
dnl
dnl The following two entries are needed so
dnl that make can find the the ldap header files
dnl and libraries
APPENDDEF(`confINCDIRS´, `-I/opt/ldap/include´)dnl
APPENDDEF(`confLIBDIRS´, `-L/opt/ldap/lib´)dnl
```

Refer to the *sendmail/README* file for any relevant details about your server's operating system, particularly if you're using LDAP libraries other than OpenLDAP. The previous example was used to build Sendmail 8.12.6 linked against OpenLDAP client libraries on a Linux system. The resulting *sendmail* binary should be checked to ensure that the *LDAPMAP* option was properly enabled. Here we have a Linux host named *garion* that includes several compile-time options that were enabled by default. The only one to be concerned with is the *LDAPMAP* flag that you specified in *site.config.m4*:

```
$ cd sendmail-8.12.6/obj.Linux.2.4.19.i686/sendmail
$ echo | ./sendmail -bt -d0
Version 8.12.6
 Compiled with: DNSMAP LDAPMAP LOG MATCHGECOS MIME7TO8
                MIME8TO7 NAMED_BIND NETINET NETUnix NEWDB
                PIPELINING SCANF USERDB USE_LDAP_INIT XDEBUG

============ SYSTEM IDENTITY (after readcf) ============
      (short domain name) $w = garion
  (canonical domain name) $j = garion.plainjoe.org
         (subdomain name) $m = plainjoe.org
              (node name) $k = garion
```

Sendmail 8.12 includes what developers have described as an experimental schema file for OpenLDAP 2. The attributes and object classes are not defined in any Internet-Draft or RFC and may change in future releases. Because we are mainly concerned with exploring the specifics of sendmail, this is of little risk to us. However, if other applications made use of it, changes to the directory schema would require tight control so no dependencies are broken. Figure 7-9 displays the six object classes defined Sendmail's schema file. All of the attributes are defined as strings (either as a Directory String or an IA5String).

There is a syntax error in the *sendmail.schema* file included with Sendmail 8.12.6. The `sendmailMTAAliasGrouping` attribute uses an invalid combination with String matching rules. This can be fixed by changing the SYNTAX for this attribute to use the Directory String OID (1.3.6.1.4.1.1466.115.121.1.15).

To install the Sendmail objects and attributes, simply copy *sendmail.schema* to the *schema/* directory:

```
root# cp cf/sendmail.schema /usr/local/etc/openldap/schema
```

and then include it in the global section of *slapd.conf*:

```
## Add support for Sendmail 8.12 objects and atrributes.
include  /usr/local/etc/openldap/schema/sendmail.schema
```

As usual, *slapd* will need to be restarted to recognize the new items.

Maps

To access information quickly, Sendmail uses a number of maps in which it retrieves dates by searching for a unique key value. These maps can take various forms; some of the more common ones are the Berkeley DBM or YP/NIS maps. Within the *sendmail.cf* file, an LDAP map is designated by the `ldap` keyword. If you're hacking

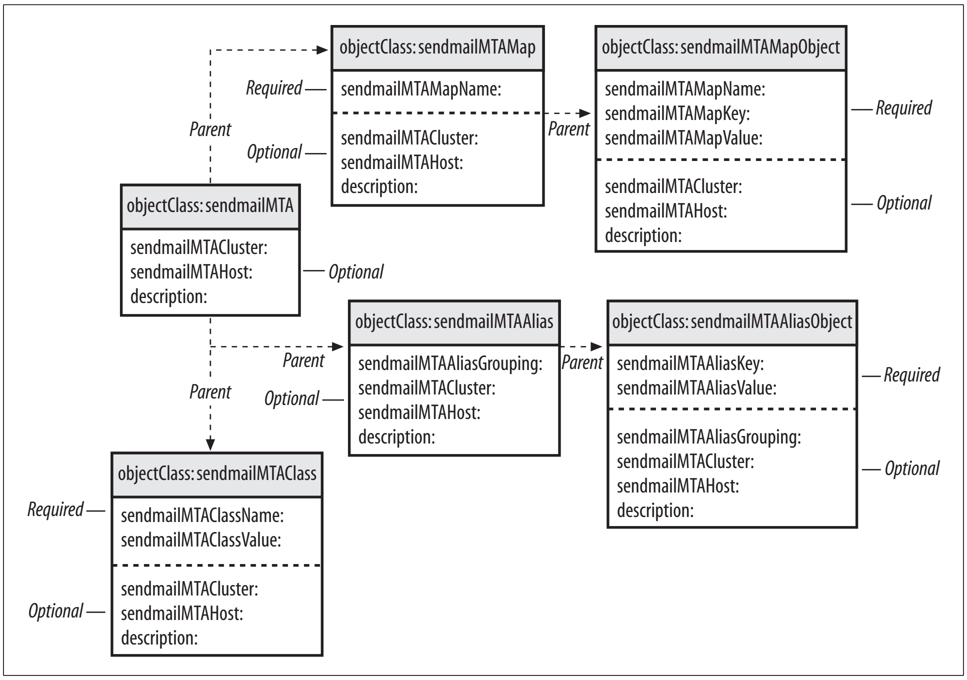


Figure 7-9. Object classes defined in sendmail.schema

the Sendmail configuration directly, you can use LDAP for any of the maps, but that's beyond the scope of this book. We will work only with Sendmail's m4 configuration generator.

Sendmail provides support for several frequently used maps as `FEATURE( )`s. Any of these features that accept an optional argument to refine the search can also accept the keyword `LDAP` to specify that the lookup should be performed using directory calls. The most basic definition of an access_db table using LDAP would look like:

```
FEATURE(`access_db´, `LDAP´)
```

The default *sendmail.cf* entry generated for this m4 macro is:

```
Kaccess ldap -k (&(objectClass=sendmailMTAMapObject)
                  (sendmailMTAMapName=access)
                  (|(sendmailMTACluster=${sendmailMTACluster})
                    (sendmailMTAHost=$j))
                  (sendmailMTAKey=%0))
             -1 -v sendmailMTAMapValue
```

The *-k* option defines the search, and the *-v* parameter specifies the name of the attribute's value to return. *-1* indicates that the search must return only one value or else it will be considered a failure. Table 7-3 contains a complete list of LDAP-specific options. It is best to refer to the Sendmail documentation (*doc/op/op.ps*) for a complete list of all lookup parameters.

Table 7-3. Configuration options for use with LDAP maps

Switch	Description
-1	The search must return a single value or else it is considered to be a failed lookup.
-b suffix	The DN to use as the base search suffix.
-d binddn	Defines a DN to use when binding to the directory.
-h hostname	The LDAP server hostname.
-k filter	Defines the search filter.
-l time *-Z size*	Define the time and size limits for a given search. The time limit is given in seconds.
-M method	The method of authentication to use when binding to the LDAP server: `LDAP_AUTH_NONE`, `LDAP_AUTH_SIMPLE`, or `LDAP_AUTH_KRBV4`.
-n	Retrieves attribute names only, not values. This is the same as the `attrsOnly` Boolean flag used during `ldap_search( )`.
-P pwfile	The file containing the credentials for *-d binddn*.
-p port	The port to use when connecting to the LDAP server.
-R	Does not automatically chase referrals.
-r deref	Controls how Sendmail should dereference aliases when searching: `never`, `always`, `search`, or `find`.
-s	The search scope (`base`, `one`, or `sub`).
-V sep	Retrieves both the attribute name and value separated by the *sep* character.
-v	Defines the attribute type that contains the value of the search result.

It is possible to define your own defaults for LDAP searches using the `confLDAP_DEFAULT_SPEC` variable in your m4 source file. This is a common place to set the hostname of the LDAP server and the base suffix used in searches. We will see an example of this later when we discuss aliases.

Table 7-4 lists all of the the Sendmail m4 features that support LDAP queries.

Table 7-4. Sendmail features that can be defined to use LDAP searches

Feature	sendmailMTAMapName	Description
`access_db`	`access`	List of hosts or networks that should be allowed to relay mail
`authinfo`	`authinfo`	Provides a separate map for storing client authentication information
`bitdomain`	`bitdomain`	A table for mapping bitnet hosts to Internet addresses
`domaintable`	`domain`	Makes use of a table that maps domain names to new domain names
`genericstable`	`generics`	Utilizes a table that contains rules for rewriting sender addresses in outgoing mail
`mailertable`	`mailer`	Includes support for a mailer table that contains rules for routing mail to specific domains
`uucpdomain`	`uucpdomain`	A table for mapping UUCP hosts to Internet addresses
`virtusertable`	`virtuser`	Includes support for a domain-specific version of aliasing

Aliases

Before implementing mail alias lookups via LDAP, let's begin with a simple *sendmail.mc* configuration file for a central mail hub for the plainjoe.org domain. Figure 7-10 illustrates how this host fits into the plainjoe.org network. Clients on the network spool messages to the mail hub, which ensures that all outgoing messages have a send in the form of *user@plainjoe.org*:

```
divert(-1)
#######################################################
 Sendmail m4 file for plainjoe.org mail hub on local
 network.
#######################################################
divert(0)
OSTYPE(`linux´)dnl
FEATURE(`use_cw_file´)dnl
dnl
dnl Masquerading settings
dnl
EXPOSED_USER(`root´)dnl
MASQUERADE_AS(`plainjoe.org´)dnl
MASQUERADE_DOMAIN(`plainjoe.org´)dnl
FEATURE(`masquerade_envelope´)dnl
FEATURE(`masquerade_entire_domain´)dnl
dnl
FEATURE(`relay_entire_domain´)dnl
FEATURE(`local_procmail´)dnl
define(`PROCMAIL_MAILER_PATH´, `/usr/bin/procmail´)dnl
define(`STATUS_FILE´, `/var/log/mail.stats´)dnl
dnl
dnl Mailer settings
dnl
MAILER(`smtp´)dnl
MAILER(`procmail´)dnl
```

The *ALIAS_FILE* m4 option (*AliasFile* in *sendmail.cf*) allows an administrator to define the location of the aliases file (even within an LDAP directory). A very basic */etc/mail/aliases* might appear as:

```
postmaster:         root, mailadmin@plainjoe.org
nobody:             /dev/null
```

Here, the postmaster alias maps to the *root* account and the address *mailadmin@plainjoe.org*. Any mail addressed to *nobody@plainjoe.org* is sent to the bit bucket (*/dev/null*).

To use Sendmail's default LDAP search parameters for aliases, simply add:

```
define(`ALIAS_FILE´, `ldap:´)dnl
```

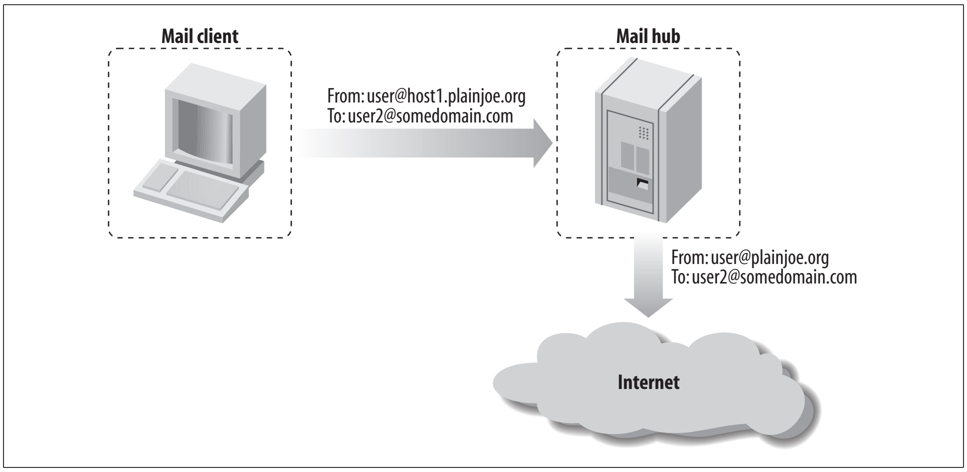


Figure 7-10. Utilizing a simple mail hub for the plainjoe.org domain

to the source m4 configuration file. This will generate a search similar to the one shown for the access_db lookup. However, this default search does not restrict the returned results to a single value (i.e., there is no *-1* option specified).

```
ldap -k (&(objectClass=sendmailMTAAliasObject)
          (sendmailMTAAliasGrouping=aliases)
          (|(sendmailMTACluster=${sendmailMTACluster})
            (sendmailMTAHost=$j))
          (sendmailMTAKey=%0))
     -v sendmailMTAAliasValue
```

You could integrate the mail aliases into the existing ou=people organizational unit within your directory. There is one main problem with this, however: all of the object classes defined in *sendmail.schema* are defined as structural. The user accounts with ou=people cannot have a second structural class. You should therefore create a new ou to store aliases for Sendmail. Other applications may arise in the future that also require a portion of the directory for storing data. In preparation, the naming scheme ou=*servicename*,ou=services,dc=plainjoe,dc=org has been chosen to organize subtrees. Figure 7-11 shows your new directory namespace.

The LDIF needed to create these three new ous should be very familiar by now:

```
dn: ou=services,dc=plainjoe,dc=org
objectClass: organizationalUnit
ou: services

dn: ou=sendmail,ou=services,dc=plainjoe,dc=org
ou: sendmail
objectClass: organizationalUnit

dn: ou=aliases,ou=sendmail,ou=services,dc=plainjoe,dc=org
objectClass: organizationalUnit
ou: aliases
```

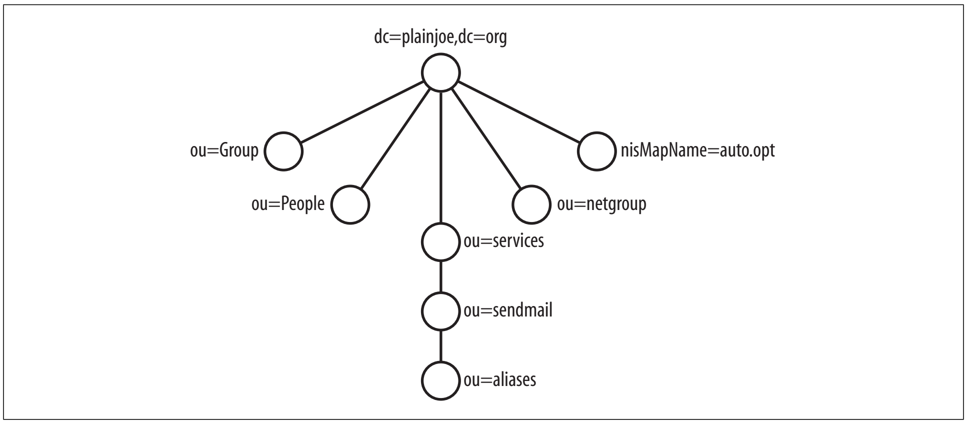


Figure 7-11. New ou=aliases for use by Sendmail

Next, you will create the directory entries corresponding to the */etc/mail/aliases* entries for `postmaster` and `nobody`:

```
dn: sendmailMTAKey=postmaster,ou=aliases,ou=sendmail,
 ou=services,dc=plainjoe,dc=org
objectClass: sendmailMTAAliasObject
sendmailMTAAliasValue: root
sendmailMTAAliasValue: mailadmin@plainjoe.org
sendmailMTAKey: postmaster

dn: sendmailMTAKey=postmaster,ou=aliases,ou=sendmail,
 ou=services,dc=plainjoe,dc=org
objectClass: sendmailMTAAliasObject
sendmailMTAAliasValue: /dev/null
sendmailMTAKey: nobody
```

The final step is to configure the actual lookup in *sendmail.mc*. Because you expect to use additional LDAP searches in Sendmail, it is best to define any global defaults using `confLDAP_DEFAULT_SPEC`. Specify that all LDAP requests should be sent to the host *ldap.plainjoe.org*:

```
define(confLDAP_DEFAULT_SPEC, `-h ldap.plainjoe.org´)dnl
```

The `ALIAS_FILE` definition will contain the base suffix, search filter, and requested attribute values. By default, Sendmail uses a subtree scope, which is fine for the alias searches:

```
define(`ALIAS_FILE´, `ldap:-k
(&(objectClass=sendmailMTAAliasObject)(sendmailMTAKey=%0)) -v sendmailMTAAliasValue
-b "ou=aliases,ou=sendmail,ou=services,dc=plainjoe,dc=org"´)dnl
```

After generating and installing the new *sendmail.cf* file:

```
$ cd sendmail-8.12.6/cf/cf
$ sh Build sendmail.cf
$ /bin/su -c "cp sendmail.cf /etc/mail/sendmail.cf"
```

you can test aliases using Sendmail's verify mode (*sendmail -bv*):

```
$ sendmail -bv postmaster@plainjoe.org
root... deliverable: mailer local, user root
mailadmin@plainjoe.org... deliverable: mailer local, user mailadmin
```

Before continuing on to Sendmail's `ldap_routing` feature, you may be wondering what advantage was achieved by storing Sendmail's aliases in LDAP. After all, you did create a new subtree within the directory, and it certainly does seem that some information, such as usernames, will end up being duplicated from the organizational unit. How did you reduce the duplication of data?

By shifting your focus from account management to service management, you can see that your directory provides a means of sharing basic Sendmail configuration data among multiple servers. This means that you no longer have to manage duplicate */etc/mail/aliases* files on each of your Sendmail hosts. Remember that the default Sendmail LDAP queries include:

```
(|(sendmailMTACluster=${sendmailMTACluster})(sendmailMTAHost=$j))
```

as part of the search filter. A Sendmail installation can be defined as a member of a cluster using the `confLDAP_CLUSTER` variable:

```
define(`confLDAP_CLUSTER´, `MailCluster´)
```

This provides a means of associating directory entries with individual hosts (`$j`) or groups of servers (`${sendmailMTACluster}`).

Mail routing using LDAP

Sendmail's LDAP mail-routing functionality can be described as an LDAP `virtusertable`. This is a domain-specific form of aliasing that supports the handling of virtual domains. It provides rules for rewriting recipient addresses or rerouting a message to the appropriate host. The following virtual user table entry would route messages that are addressed to *joe@foo.com* to the host *somehost.foo.com*:

```
joe@foo.com     somehost.foo.com
```

Under its default configuration, Sendmail's `ldap_routing` uses the `inetLocalMailRecipient` auxiliary object class defined in the expired Internet-Draft *draft-lachman-laser-ldap-mail-routing-xx.txt*. A version of this draft is included with the OpenLDAP source distribution. There are no required attributes in this object class, as you can see in Figure 7-12.

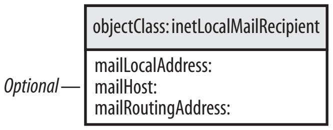


Figure 7-12. inetLocalMailRecipient object class used by Sendmail's FEATURE('ldap_routing')

The three optional attributes in `inetLocalMailRecipient` are:

`mailLocalAddress`
: The RFC 822–compliant mail address of the message recipient

`mailHost`
: The DNS name specifying the host to which the message should be relayed

`mailRoutingAddress`
: The RFC 822–compliant mail address to which the original recipient address should be rewritten

OpenLDAP includes a definition for the `inetLocalMailRecipient` object and associated attributes in *misc.schema*. You must include this file in *slapd.conf* and restart OpenLDAP before you can support Sendmail's `ldap_routing` feature:

```
## Support the inetLocalMailRecipient object.
include   /usr/local/etc/openldap/schema/misc.schema
```

To enable LDAP mail routing, add the following feature definition to *sendmail.mc*:

```
FEATURE(`ldap_routing´)
```

We must also inform Sendmail which mail domains should be routed. Without control over which email domains Sendmail should attempt to look up in the directory, each incoming message triggers a lookup, resulting in severely degraded performance on high-traffic sites.

To define a single routable domain, Sendmail provides the `LDAPROUTE_DOMAIN` m4 macro. The configuration for your server requires you to add this line to your *sendmail.mc* source file:

```
LDAPROUTE_DOMAIN(`plainjoe.org´)
```

A list of LDAP-routable domains can be read from a file defined by the `LDAPROUTE_DOMAIN_FILE` macro. Sendmail 8.12 introduced support for retrieving such class values from a directory using the syntax `LDAPROUTE_DOMAIN_FILE(`@LDAP´)`. More information on file class macros can be found in the documentation included with Sendmail.

As mentioned previously, `ldap_routing` uses the `inetLocalMailRecipient` object class. It is possible to use an alternative schema by defining the `ldap_routing` feature as:

```
FEATURE(`ldap_routing´,mailHost,mailRoutingAddress,bounce,detail)
```

The *`mailHost`* and *`mailRoutingAddress`* entries are just LDAP map configuration lines; they default to:

```
ldap -1 -T TMPF -v mailHost
  -k (&(objectClass=inetLocalMailRecipient)
    (mailLocalAddress=%0))
```

They also default to:

```
ldap -1 -T TMPF -v mailRoutingAddress
  -k (&(objectClass=inetLocalMailRecipient)
    (mailLocalAddress=%0))
```

The search filters and the resulting attributes can be redefined to better suit your directory, if required. Both the *bounce* and *detail* parameters specify actions to take if a lookup does not return any routing information. The default behavior is to accept addresses not located by the LDAP search. Sendmail's *cf/README* file has more details on changing this if you are interested.

The default searches used by ldap_routing do not define an LDAP server, nor do they include a search suffix. The confLDAP_DEFAULT_SPEC option can be used to specify defaults for all of Sendmail's LDAP queries (maps, aliases, classes, and mail routing):

```
define(`confLDAP_DEFAULT_SPEC´, `-h ldap.plainjoe.org -b
  ou=people,dc=plainjoe,dc=org´)dnl
```

This is fully compatible with the configuration used to retrieve mail aliases from the directory. The ALIAS_FILE option used its own base suffix (*-b*) which overrode any matching default set by confLDAP_DEFAULT_SPEC.

With three optional attributes in the inetLocalMailRecipient object class, Sendmail must consider six unique routing cases. Note that if the mailLocalAddress attribute is absent, Sendmail will ignore the entry altogether. The possible results are described in Table 7-5.

Table 7-5. Possible results from an ldap_routing search

mailHost value	mailRoutingAddress value	Result
A local host	Exists	The recipient is rewritten to mailRoutingAddress and delivered to the local host.
A local host	Does not exist	The mail is delivered to the original address on the local host.
A remote host	Exists	The mail is relayed to the mailRoutingAddress at the mailHost.
A remote host	Does not exist	The mail is relayed to the original address at mailHost.
Does not exist	Exists	The recipient is rewritten to mailRoutingAddress and delivered to the local host.
Does not exist	Does not exist	The mail is delivered locally to the original address or possibly bounced as a unknown user.

The following LDIF listings help explain the entries in Table 7-5. Here, you extend the original user entries in the ou=people subtree. You could have created a new organizational unit beneath ou=sendmail. However, adding the mail-routing information to a user's entry means that when a user's account is deleted, the mail-routing information is removed as well.

In the first listing, the `mailLocalAddress` and `mailHost` attributes cause mail addressed to *kcarter@plainjoe.org* to be relayed to the host designated by *mail.engr.plainjoe.org*'s DNS MX record for local delivery:

```
dn: uid=kristi,ou=people,dc=plainjoe,dc=org
objectclass: inetOrgPerson
objectclass: posixAccount
objectclass: inetLocalMailRecipient
cn: Kristi Carter
sn: Carter
mail: kcarter@plainjoe.org
mailLocalAddress: kcarter@plainjoe.org
mailHost: mail.engr.plainjoe.org
<...remaining attributes not shown...>
```

The following example adds the `mailRoutingAddress` attribute. With this attribute, all mail addressed to *kcarter@plainjoe.org* is relayed to the host named by the MX record for *mail.engr.plainjoe.org*, but only after the recipient address has been rewritten to *kristi@engr.plainjoe.org*:

```
dn: uid=kristi,ou=people,dc=plainjoe,dc=org
objectclass: inetOrgPerson
objectclass: posixAccount
objectclass: inetLocalMailRecipient
cn: Kristi Carter
sn: Carter
mail: kcarter@plainjoe.org
mailLocalAddress: kcarter@plainjoe.org
mailHost: mail.engr.plainjoe.org
mailRoutingAddress: kristi@engr.plainjoe.org
<...remaining attributes not shown...>
```

These rewrites can be verified using Sendmail's rule set–testing mode:

```
$ /usr/sbin/sendmail -bt
> /parse kcarter@plainjoe.org
  <...intervening ruleset output deleted...>
mailer relay, host mail.engr.plainjoe.org,
     user kristi@engr.plainjoe.org
```

This output shows that mail received for *kcarter@plainjoe.org* will be forwarded to the host *mail.engr.plainjoe.org* after rewriting the recipient address to *kristi@engr.plainjoe.org*.

The `mailLocalAddress` attribute can also be used to specify that all mail for a domain should be relayed to another host. The following LDIF entry relays all mail addressed to the *@plainjoe.org* domain to the host denoted by *hq.plainjoe.org*'s MX record:

```
dn: o=plainjoe.org,ou=people,dc=plainjoe,dc=org
objectclass: organization
objectclass: inetLocalMailRecipient
o: plainjoe.org
description: plainjoe.org mail domain
mailLocalAddress: @plainjoe.org
mailHost: hq.plainjoe.org
```

It should be noted that Sendmail gives exact matches for the `mailLocalAddress` precedence over entries returned by matching the @*domain* syntax.

Postfix

Our next stop during this tour of MTAs is to examine Wietse Venema's Postfix mailer. This MTA is a popular replacement for Sendmail because it has:

- Feature and interface compatibility with Sendmail
- A simpler configuration
- A history of fewer security holes

This section focuses on Postfix's ability to integrate with an LDAP directory. I assume that the terminology and configuration files are familiar to Postfix administrators. If this is your first exposure to Venema's MTA, the Postfix web site (*http://www.postfix.org/*) offers several good documents on the software's design philosophy and architecture. It may also be helpful to refer to *Postfix*, by Richard Blum (Sams Publishing) for case studies of working installations.

After the gory details of configuring LDAP queries in Sendmail, Postfix's configuration files are a welcome relief. In comparison to Sendmail, Postfix's configuration is much more intuitive.

We will begin by ensuring that the proper features are enabled when you compile Postfix. The source distribution for Postfix can be downloaded from *http://www.postfix.org/*. Assuming that the OpenLDAP 2 client libraries have been installed in the directory */usr/local/lib/*, the following commands clear all remaining intermediate files from a previous build (just to be safe), and then create the necessary Makefiles to enable LDAP client support:

```
$ cd postfix-1.1.2/
$ make tidy
$ make makefiles CCARGS="-I/usr/local/include -DHAS_LDAP" \
> AUXLIBS="-L/usr/local/lib -lldap -llber"
$ make
$ /bin/su -c "make install"
```

Refer to the *LDAP_README* file included with the Postfix distribution for details about building the software on your server platform.

Postfix 1.1.2 will not compile when using the OpenLDAP 2.1 client libraries. You must use the most recent OpenLDAP 2.0 libraries in this case (or libraries from some other vendor described in the *README_FILES/LDAP_README* document). Note that this does not affect communications with an OpenLDAP 2.1 server.

Once you have built and installed Postfix, verify that LDAP support has been included. To do so, use the *postconf* utility installed with the Postfix server. The *-m* switch informs *postconf* to display the list of supported storage mediums for tables. The output should look something like this:

```
$ /usr/sbin/postconf -m
static
nis
regexp
environ
ldap
btree
unix
hash
```

The exact list will vary, depending on how you've built Postfix. Your immediate concern is to verify that *ldap* is listed as a supported storage medium. However, it's important to understand what's going on. Postfix maintains six tables, any of which may be stored on any of the media reported by *postconf -m*. Table 7-6 introduces each of the tables and shows which core program acts as the table's main client.

Table 7-6. Postfix tables and associated core programs

Table	Description	Core program
Access	Provides information about which messages to accept or reject based on sender, host, network , etc.	*smtpd*
Aliases	Provides information on redirecting mail received for local users.	*local*
Canonical	Provides information on local and nonlocal addresses.	*cleanup*
Relocated	Provides information on "user has moved to a new location" bounce messages.	*qmgr*
Transport	Provides information on delivery methods and relay hosts for the domain.	*trivial-rewrite*
Virtual	Provides information used in redirecting local and nonlocal users or domains.	*cleanup*

The remainder of this section shows how to configure a Postfix server to retrieve local aliases via LDAP queries. The following configuration file, *main.cf*, is the starting point for our discussion:

```
## /etc/postfix/main.cf
## Postfix configuration file for the plainjoe.org SMTP server.
##   Written by <jerry@plainjoe.org>

## Host/domain information
myhostname = garion.plainjoe.org
mydomain = plainjoe.org
myorigin = plainjoe.org

## Who is local?
mydestination = localhost $myhostname

## Who do we accept mail relaying from?
mynetworks = 192.168.1.0/24 127.0.0.0/8

## Program locations
command_directory = /usr/sbin
```

```
daemon_directory = /usr/libexec/postfix
queue_directory = /var/spool/postfix
mail_owner = postfix

## Sendmail-compatible mail spool directory
mail_spool_directory = /var/spool/mail
```

As before, an alias entry maps a local username to an email address; this address can be either another local user or a user on a remote system. In your LDAP schema, a local user is represented by the `uid` attribute of the `posixAccount` object class. The aliased entry is represented by the `mail` attribute of the `inetOrgPerson` object class. Note that you do not use the `sendmailMTA` and related schema objects presented in the previous section, but rely on the original object classes and attributes used by the mail clients presented in the first half of this chapter.

This schema does not address the case of mapping one local user to another for email delivery. Nor does it allow the use of external files to list the addresses that should be used as expansions for aliases; this feature is useful for supporting a local mailing list. This limitation is a result of the attributes chosen and not of Postfix's LDAP implementation.

Here's a typical LDIF entry for a user account that has an email alias. Mail for this account (a guest account) is forwarded to *jerry@plainjoe.org*:

```
## User account including a mail alias
dn: uid=guest1,ou=People,dc=plainjoe,dc=org
uid: guest1
cn: Guest Account
objectClass: posixAccount
objectClass: inetOrgPerson
userPassword: {CRYPT}Fd8nE1RtCh5G6
loginShell: /bin/bash
uidNumber: 783
gidNumber: 1000
homeDirectory: /home/guest1
gecos: Guest Account
sn: Account
mail: jerry@plainjoe.org
```

To inform the Postfix daemons that they should read the alias map from an LDAP directory, add the following entry to the server's *main.cf*:

```
alias_maps = ldap:ldapalias
```

The `ldap` keyword denotes the type of lookup table; the `ldapalias` string is the name of the table. This name is used as a prefix for parameter names; it identifies which settings are associated with this table.

After specifying that Postfix should look up alias information from the directory, you have to define several parameters that tell Postfix how to search the directory. These should be familiar by now. The most common settings include the LDAP server name (`server_host`), the search base (`search_base`), the search scope (`scope`), the search filter (`query_filter`), and the resulting attribute value to return (`result_attribute`). Each of these parameters is prefaced by the LDAP table name (`ldapalias_`). Add these definitions to *main.cf*:

```
## Parameters for LDAP alias map
ldapalias_server_host = localhost
ldapalias_search_base = ou=people,dc=plainjoe,dc=org
ldapalias_scope = sub
ldapalias_query_filter = (uid=%s)
ldapalias_result_attribute = mail
```

You can test the alias table lookup using the *postmap(1)* utility to verify that mail to the user *guest1* will be forwarded to the mail account at *jerry@plainjoe.org*:

```
$ postmap -q guest1 ldap:ldapalias
jerry@plainjoe.org
```

After starting the Postfix daemons (*/usr/sbin/postfix start*), you can test your configuration further by sending a test message to *guest1@garion*. This *slapd* log entry (which comes from a logging level of 256) proves that Postfix did query the server using the filter `"(uid=guest1)"`:

```
Aug 15 10:53:37 ldap slapd[6728]: conn=24 op=1 SRCH
base="ou=people,dc=plainjoe,dc=org" scope=2 filter="(uid=guest1)"
```

The following excerpt from the header of the delivered message shows that the message was delivered to *guest1@garion.plainjoe.org*. However, the message was then forwarded to *jerry@plainjoe.org*, as specified by the value of the `mail` attribute for the *guest1* account:

```
Return-Path: <root@plainjoe.org>
Delivered-To: jerry@plainjoe.org
Received: from XXX.XXX.XXX.XXX ([ XXX.XXX.XXX.XXX ] helo=garion.plainjoe.org)
        by gamma.jumpserver.net with esmtp (Exim 3.36 #1)
        id 18M1Sc-0003tj-00
        for jerry@plainjoe.org; Wed, 11 Dec 2002 01:39:14 -0600
Received: by garion.plainjoe.org (Postfix)
        id 15CA23FB62; Tue, 10 Dec 2002 11:40:23 -0600 (CST)
Delivered-To: guest1@garion.plainjoe.org
Received: by garion.plainjoe.org (Postfix, from userid 0)
        id F042E3FB69; Tue, 10 Dec 2002 12:40:22 -0500 (EST)
To: guest1@garion.plainjoe.org
Subject: testing Postfix/LDAP lookups
Message-Id: <20021210174022.F042E3FB69@garion.plainjoe.org>
Date: Tue, 10 Dec 2002 12:40:22 -0500 (EST)
From: root@plainjoe.org (root)
```

There are many possibilities beyond the simple example presented here. Your *query_filter* used only a single attribute, but nothing prevents the use of more complex filters that match on multiple attributes. Furthermore, many additional LDAP parameters allow you to fine-tune the way Postfix interacts with the directory. Table 7-7 gives a complete listing of all LDAP-related Postfix parameters as well as the default setting for each one.

Table 7-7. Postfix LDAP parameters

Parameter	Default	Description
`bind`	`yes`	Defines whether an LDAP bind request should be issued prior to performing the query. This value must be `yes` or `no`.
`bind_dn`	`""`	The DN used when binding to the LDAP directory.
`bind_pw`	`""`	The clear-text password used when binding to the directory using the `bind_dn` value.
`cache`	`no`	Determines whether to enable client-side caching of LDAP search results, as described in the *ldap_enable_cache(3)* manpage.
`cache_expiry`	30 seconds	Defines the cache expiration timeout when `cache=yes`.
`cache_size`	32 KB	Specifies the size of the LDAP cache when `cache=yes`.
`dereference`	`0`	Controls whether Postfix should dereference aliases when searching the directory. Possible values are `0` (never), `1` (when searching), `2` (when locating the base object for the search), and `3` (always).
`domain`	`none`	A list (possibly a table lookup) of domain names that restricts when a query is made. This means that a local "user" (with the `@`...) will not be queried, nor will any email address that does not match one of the domains listed. For example: `ltable_domain = plainjoe.org, hash:/etc/postfix/moredomains`
`query_filter`	`(mailacceptinggeneralid=%s)`	The RFC 2254–style LDAP search filter.
`result_attribute`	`maildrop`	The attribute value that should be read as a result of the `query_filter`.
`scope`	`sub`	The scope of the directory search; must be one of `sub`, `base`, or `one`.
`search_base`	`none`	The DN that acts as the base search suffix for the query.
`server_host`	`localhost`	The hostname of the LDAP server to which queries should be submitted. The value is of the form `*hostname*[:*port*][,*hostname*[:*port*], ...]`.

Table 7-7. Postfix LDAP parameters (continued)

Parameter	Default	Description
server_port	389	The port on which the server_host is listening (unless overridden by the *hostname*:*port* syntax).
special_result_attribute	none	Allows administrators to define an attribute that returns DNs from an LDAP search. If this value is present in the entry returned by a successful search, another query is issued using the returned DN as the search_base.
timeout	10 seconds	The maximum amount of time, in seconds, that can elapse before the search is abandoned.

Exim

The Exim MTA is another Sendmail alternative. It was first developed in 1995 by Dr. Philip Hazel while at Cambridge University. For the full details on configuring Exim, Philip Hazel's book, *Exim*: *The Main Transfer Agent* (O'Reilly), provides an excellent tutorial on the various configuration details.* If you are not familiar with Exim, it is a good idea to visit *http://www.exim.org/* to obtain an overview of Exim's mail architecture. We will be looking at Exim 4.10.

Like Sendmail and Postfix, Exim supports various types of file and database lookups, such as mySQL, Berkeley DBM, and LDAP. In its default form, the Exim Makefile supports only linear searches in files (*lsearch*) and database lookups (*dbm*). To enable LDAP lookups, a handful of Makefile variables must be set. These variables are presented in Table 7-8.

Table 7-8. Exim LDAP-related Makefile variables

Variable	Description
LOOKUP_LDAP	This variable must be set to yes to include LDAP lookup support in the *exim* binary.

LOOKUP_INCLUDE
LOOKUP_LIBS
	These variables provide a means of supplementing the existing CFLAGS and LDFLAGS variables when building Exim. To support LDAP lookups, they must specify the locations of include files and LDAP libraries. For example:
	`LOOKUP_INCLUDE=-I/opt/ldap/include`
	`LOOKUP_LIBS=-L/opt/ldap/lib -llldap -llber`

LDAP_LIB_TYPE	This variable defines which LDAP client libraries will be used. Possible values are UMICHIGAN, OPENLDAP1, OPENLDAP2, NETSCAPE, and SOLARIS.

* At the time of this writing, Hazel's book covers Exim 3. The current release discussed in this chapter is Exim 4. There have been some substantial changes between the two versions.

Build the mail server with the following LDAP settings in Exim's *Local/Makefile*:

```
## Included in Exim's Local/Makefile to enable LDAP lookup support
LOOKUP_LDAP=yes
LOOKUP_INCLUDE=-I /usr/local/include
LOOKUP_LIBS=-L/usr/local/lib -lldap -llber
LDAP_LIB_TYPE=OPENLDAP2
```

It is a good idea to verify that the OpenLDAP libraries have been linked to the *exim* binary using some type of tool, such as *ldd(1)*, to view linking dependencies:

```
$ ldd /usr/exim/bin/exim
    libresolv.so.2 => /lib/libresolv.so.2 (0x40026000)
    libnsl.so.1 => /lib/libnsl.so.1 (0x40037000)
    libcrypt.so.1 => /lib/libcrypt.so.1 (0x4004b000)
    libdb-4.0.so => /lib/libdb-4.0.so (0x40078000)
    libldap.so.2 => /usr/local/lib/libldap.so.2 (0x4010f000)
    liblber.so.2 => /usr/local/lib/liblber.so.2 (0x40146000)
    libc.so.6 => /lib/libc.so.6 (0x40153000)
    libdl.so.2 => /lib/libdl.so.2 (0x4027b000)
    libsasl2.so.2 => /usr/lib/libsasl2.so.2 (0x4027e000)
    libssl.so.2 => /lib/libssl.so.2 (0x40290000)
    libcrypto.so.2 => /lib/libcrypto.so.2 (0x402bd000)
    /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Once the Exim binaries have been built and installed (we'll assume that the default location of */usr/exim/* is the installation directory), the next step is decide what data should be retrieved from the directory. Our discussion of Postfix defined a useful schema for retrieving mail aliases for local users; let's see how this same schema applies to Exim. The schema makes use of the uid attribute type (posixAccount) as the key and the mail attribute type (inetOrgPerson) as the resulting value. For completeness's sake, here's the LDIF entry for a user account with a mail alias; it's the same entry you used for the Postfix server. All mail that would be delivered to the local user named *guest1* should be forwarded to the address *jerry@plainjoe.org*.

```
## User account including a mail alias
dn: uid=guest1,ou=People,dc=plainjoe,dc=org
uid: guest1
cn: Guest Account
objectClass: posixAccount
objectClass: inetOrgPerson
userPassword: {CRYPT}Fd8nE1RtCh5G6
loginShell: /bin/bash
uidNumber: 783
gidNumber: 1000
homeDirectory: /home/guest1
gecos: Guest Account
sn: Account
mail: jerry@plainjoe.org
```

Exim searches are defined using the data keyword. The general syntax for a table lookup is:

```
data = ${lookup db_type {db_search_parameters}}
```

LDAP queries can use a *db_type* of:

`ldap`
: Indicates that the search will return a single value and that Exim should interpret multiple values as a failure

`ldapdn`
: Specifies that the search will match one entry in the directory and that the returned value is the DN of that entry

`ldapm`
: Defines searches that may return multiple values

To inform Exim that local `alias` data should be retrieved from an LDAP directory, you must configure an appropriate *redirect* router. To do so, you create an `ldap_aliases` entry in */usr/exim/configure*:

```
## Alias Director, which retrieves data from an LDAP director. The name
## "ldap_aliases" has been arbitrarily chosen.
ldap_aliases:
  driver = redirect
  data = ${lookup ldap \
    { ldap://ldap.plainjoe.org/\
      ou=people,dc=plainjoe,dc=org\
      ?mail?sub?(uid=${local_part})} }
```

The `driver` keyword is used to define the type of router being implemented. In contrast to both Sendmail and Postfix, Exim uses an LDAP URL to define the LDAP host, port, search base, retrieved attribute values, scope, and filter. The line continuation character (\) has been used to make the line more readable. The variable `${local_part}` is the username extracted from the local recipient's mail address (for example, the `${local_part}` of *jdoe@garion.plainjoe.org* would be *jdoe*). So you can read the query specification as: "Using the LDAP server at *ldap.plainjoe.org* (on the default port of tcp/389), perform a substring search of the `uid` attribute, searching for the local part of the email address, and returning the value of the `mail` attribute. Perform the search with a search base of `ou=people,dc=plainjoe,dc=org`."

It is possible to define multiple servers for LDAP queries with the `ldap_default_servers` parameter in Exim's *configure* file. This option accepts a colon-separated list of servers and ports that are tried one by one until a server is successfully contacted. This setting would utilize two directory servers, *ldap1* and *ldap2,* for fault tolerance purposes:

```
ldap_default_servers = ldap1::389:ldap2::389
```

Using Exim's address-testing mode, you can verify that mail sent to *guest1@garion* will indeed be forwarded to *jerry@plainjoe.org*:

```
root# exim -v -bt guest1@garion
jerry@plainjoe.org
    <-- guest1@garion.plainjoe.org
  deliver to jerry@plainjoe.org
```

```
    router = dnslookup, transport = remote_smtp
    host plainjoe.org [xxx.xxx.xxx.xxx]
```

The log file for *slapd* (`loglevel 256`) shows that the lookup for `(uid=guest1)` was performed as expected:

```
Aug 16 17:05:09 ldap slapd[3574]: conn=36 op=1 SRCH
base="ou=people,dc=plainjoe,dc=org" scope=2 filter="(uid=guest1)"
```

The LDAP URL format does not allow any space for defining credentials to be used when binding to the server. The default behavior is to perform an anonymous bind and not request any limits on search results. Exim can request a simple bind using credentials specified by the *user* and *pass* options. Table 7-9 lists several parameters that can be included in the LDAP query as *`option`*`=`*`value`* to specify authentication information as well as search limits.

Table 7-9. Additional Exim LDAP query parameters

Parameter	Description
`user`	The DN used when binding to the directory server
`pass`	The clear-text password used when binding to the directory server with a non-empty `user`
`size`	The upper limit on the number of entries returned from the lookup
`time`	The upper limit, in seconds, on the time to wait for a response to a lookup

To use these additional parameters when performing an LDAP lookup, they must preceed the URL in the *data* string. For example, to bind to the LDAP server as the user `cn=Mail Admin,dc=plainjoe,dc=org` using the password `secret`, you would define the following query:

```
data = ${lookup ldap \
   { user="cn=Mail Admin,dc=plainjoe,dc=org"\
     pass=secret \
     ldaps://ldap.plainjoe.org/\
     ou=people,dc=plainjoe,dc=org\
     ?mail?sub?(uid=${local_part})}  }
```

Because Exim uses LDAPv2 binds, it cannot take advantage of SASL authentication or the StartTLS LDAv3 extension. However, it can understand URLs that use *ldaps://*. This is important when sending a DN and password to the directory server in clear text.

It should also be mentioned that because the `pass` value is stored in clear text, it is preferable to preface the data line with the `hide` keyword (i.e., `hide data =` ...) directive so that the line cannot be displayed by ordinary users using the *exim -bP* command.