Virtual Mailserver HOWTO

Dave Dribin

dave@dribin.org

Keith Garner

kgarner@kgarner.com

This guide describes how to set up a mail server that supports multiple domains with virtual and local users using Postfix, OpenLDAP, Courier-IMAP, and Jamm. It also covers many design decisions behind the technology and implementation choices.

Table of Contents

1. Introduction

This guide provides instructions on how to set up an integrated mail server using Postfix, OpenLDAP, Courier-IMAP, and Jamm. We also examine the design decisions that went into what we finally set up. We also want to document two to three weeks worth of research into one place.

Our intention isn't to write a revolutionary document. Our goal is explain how to set up a fairly flexible and robust mail solution using existing open source software. Hopefully, we'll have saved at least one person the trouble of some of the research we had to do.

1.1. Why write this document?

Our exploration into setting up this mail server started with a simple question from our friend JD. He wanted to know if it was possible to set up a mail server where <`jd@domain1.example`> and <`jd@domain2.example`> were separate users and separate from <`jd@mailserver.example`>.

Of course we knew it was possible, but we didn't know exactly how to go about it. We didn't know what technology we would use nor how it would work. This started us down the path to find this information.

In our investigations to find documentation, we found bits of information here and there, but not one document that described a setup like this. We're hoping to fill that void with this paper.

2. Requirements

Here is our list of requrements. Some of these are brought to us through the power of 20/20 hindsight, some of these we knew from the get go.

The Requirements List

- Support for delivery to local users, i.e. users that have a shell account.
- Local users shall be able to use **procmail**.
- Ease of integration into existing mail systems.
- Access to mail through IMAP.
- Support for virtual IMAP accounts, i.e. users that have an IMAP login and password, but do not have a shell acount.
- Access to mail through maildir folders on the local filesystem for shell users.
- Forwarding (alias) entries for virtual domains.
- An interface to read email via the web.
- The ability to host multiple domains from a single server with one IP address.
- Liberal use of encryption such that passwords never go over the network in clear text.

For Keith, the important item was fitting into how he already did his e-mail (using procmail and mutt with maildir mailboxes.) For Dave, the important items were IMAP and procmail support. Both of us thought web access would be a huge plus for when we couldn't use our mail reader of choice. Both had other domains they were asked to host for friends and family, but they didn't want to give out shell access.

A goal we had given ourselves was *Universal Mail Availability*. With local support, IMAP support, and web support we were sure to reach that goal.

3. System Architecture

This sections describes which servers were used and how they all fit together at a system level.

3.1. Software Selection

Choosing one piece of software over another can often become the place of holy wars on the scale of **emacs** vs. **vi**. One could write a paper debating the merits of each of the SMTP servers out there, for example. This section briefly describes our choices for an SMTP server, an IMAP server, an LDAP server, and webmail software.

3.1.1. Postfix

One of the most important aspects of our MTA/MDA was that it would easily support virtual users. The other one was that it would allow local users to continue to use **procmail**[17] for their filtering to keep their mailboxes in order. We also wanted to keep the configuration easy for the administrator. Postfix[14] fits the bill quite nicely.

> Wietse Venema [13]
>
> Postfix is my attempt to provide an alternative to the widely-used Sendmail program. Postfix attempts to be fast, easy to administer, and hopefully secure, while at the same time being Sendmail compatible enough to not upset your users.

Postfix is designed modularly with many small programs working together to fill the mail service role. Most of Postfix runs as a non-root user and most parts of Postfix can run in a chroot environment. While still a relatively new mailer, it has a design that lends itself to long term security. Its configuration is straight-forward and very human readable. All of the parameters are of the "*key = value*" variety.

Starting with Postfix 1.1.x a virtual user mail delivery agent has been included. This makes delivery to virtual mailboxes very easy with only a few lines added to the config. It's also very easy to integrate LDAP into Postfix. Another benefit is the fact that it was made to be a drop in replacement for Sendmail. So, if a program is designed to work with Sendmail, it'll most likely work with Postfix.

It should also be noted that Sendmail [22], Qmail [18], or Exim [2] may work in this situation as well. We just had a greater familiarity and confidence with Postfix for this task. A lot of the rest of what is described in this document will apply to the other MTAs at a high level, but your mileage may vary when it comes to implementation.

3.1.2. Courier-IMAP

There are numerous IMAP servers for Unix available, the most popular being UW IMAP from the University of Washington [24]. UW IMAP, however, does not support the Maildir format nor does it support virtual users. Thus, we decided to use Courier-IMAP [1], a popular alternative IMAP server that works *only* with Maildir format.

Native support for Maildir was the original attraction to Courier, but it also has good support for virtual users, LDAP, and IMAP over SSL. It also integrates nicely with **procmail** since **procmail** can deliver to Maildir directly, too.

I should point out that **procmail** can only work for local users and not virtual users. In practice this works out fine because virtual users have no way of editing the `.procmailrc` file since they do not have shell access. If you want **procmail** for both local and virtual users, you may have to choose a different IMAP server or a different method of integrating the pieces.

3.1.3. OpenLDAP

There's really only once choice for Open Source LDAP servers and that's OpenLDAP [12]. It is more than capable of handling everything we need, including support for LDAP over SSL and advanced authentication encryption algorithms such as salted SHA and salted MD5. There have been some doubts as to the scalability and stability of OpenLDAP for huge systems, but we have not experienced any problems. You may consider using a commercial

LDAP server such as iPlanet's Directory Server [9], Novell's eDirctory [11], or OctetString's VDE Directory Suite [25], all of which run on Linux.

3.1.4. Jamm

There wasn't really any software out there that did what we wanted in terms of easily allowing people to admin their own domains, so we rolled our own. Jamm [4] is a web application to manage virtual email account information stored in an LDAP directory. It is meant as an administration tool for sites using mail servers that store virtual user information in LDAP. Jamm is essentially an application specific LDAP editor and does not manage mail server specific configuration files. This makes Jamm quite portable to any SMTP, POP, or IMAP server so long as it supports LDAP. It also, however, means that some adminstration needs to take place outside of Jamm, such as cleaning up disk space of deleted accounts.

3.1.5. SquirrelMail

There are many choices of webmail systems. Some read directly from the file system, while others use IMAP or POP. We were recommended SquirrelMail [23], a PHP-based IMAP client, and it has worked out very well. Another webmail client that comes with good recommendations is IMP [7]. Be sure to check Freshmeat [3] for more options.

3.2. The Big Picture

Figure 1 shows how Postfix, Courier, Procmail, OpenLDAP and Jamm interact with each other. Postfix accepts incoming mail from SMTP and delivers it to the maildir mailboxes on the file system. It needs to have a list of valid users so it can bounce email for unknown users. It also needs to know where each user's mailbox is located on the file system so it can properly deliver messages. Postfix delivers the mail itself for virtual users and uses **procmail** as the MDA for local users.

Courier provides remote access to the maildir mailboxes via the IMAP and POP protocols. It needs to have a list of valid users and some means to authenticate them so users can log into their account. It, too, needs to know where each users' mailbox is located on the file system so it can read their messages.

Local user information can be accessed from the standard account database. A list of valid users can be obtained from `/etc/passwd`. The users' home directory, which can also be obtained from `/etc/passwd`, provides the location of the mailbox. Authentication can be handled by standard Unix mechanisms, such as pluggable authentication modules (PAM).

Virtual user information is stored in an LDAP directory since LDAP provides a mechanism for searching for valid users, getting user information, and authenticating users. It is possible to avoid an LDAP directory, but it will be more difficult to administer the virtual user information. For example, Postfix and Courier both support virtual users without LDAP using configuration files, but they have different file formats. It is possible to write scripts to keep these two sets of configuration files in sync, but it can be a chore and is error prone. Also, these file formats are not standard and require custom code to access them. LDAP is a much cleaner solution since it provides a standard interface for accessing the directory and many languages provided LDAP APIs. Jamm speaks LDAP and can manipulate the mail alias and account information stored in LDAP.

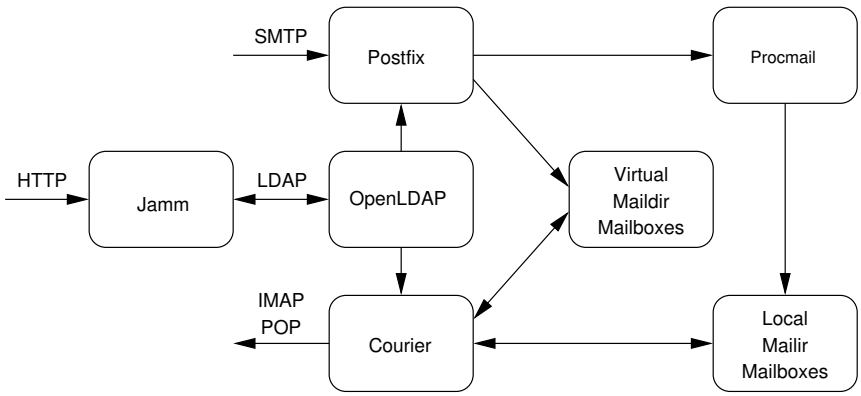

Figure 1. Overall Design

3.3. Mailbox Location

Local users' mail is stored in their home directory at `${HOME}/Maildir/` in the maildir format. It is standard practice for maildir delivery to go into the users' home directory rather than `/var/`. Both Postfix and Courier work with this standard behavior out of the box.

Unlike local users, there is no standard location for virtual user's email. We chose to create a single Unix account, called `vmail`, that holds all the virtual mailboxes. [1] Postfix and Courier support looking up the directory and Unix account information out of LDAP. This document only describes how to setup this system with one user, though you are free to choose a different policy. Each virtual domain has a subdirectory within the `~vmail/domains/` directory. For example, if there is a user <`john@example.com`>, mail would be stored in `~vmail/domains/example.com/john/` in maildir format.

4. Implementation

This section describes how to implement a virtual mail solution. Not every little detail is covered, just what is needed above and beyond the "standard" installations.

4.1. Prerequisites

Here is the list of software, with their version numbers, that we have tested this configuration with:

Software

- Red Hat Linux 6.2, 7.1, 7.2, 7.3
- Postfix 1.1.3 or greater
- OpenLDAP 2.0.21 or greater
- Courier-IMAP 1.4.1 or greater
- Procmail v3.22
- Jamm v0.9.1

4.2. Preparing the Unix System

To prepare the Unix system there are a few tasks you'll need to accomplish: create the `vmail` user and decide where you're going to store the virtual users email.

Creating the `vmail` user is just like creating any other system account. You'll want to have a UID and a GID that is used alone for `vmail`. You may also want to set its home directory to the location you've selected for the storage area of the virtual users' email.

In our system we used `vmail` as the user and group name. We also decided to store our virtual users email in `/home/vmail`.

The following example would work on a RedHat Linux distribution and result in a `vmail` user being created and an empty mail storage directory being created.

```
% useradd -k -m -r -d /home/vmail vmail
% mkdir ~vmail/domains
% chown vmail.vmail ~vmail/domains
```

You will also need to create an account and two groups for postfix, but that is covered in Postfix's `INSTALL` documentation.

4.3. OpenLDAP

You do not need to follow any special instructions for compiling and installing OpenLDAP, so please consult its documentation for full instructions. For a production environment, you should read up on how to run OpenLDAP as a non-root user, setup a chroot environment, and replication. This section describes how to configure **slapd** for a single server and how to create the base tree structure. Please consult Figure 2 as this is the LDAP tree we will aim to setup.

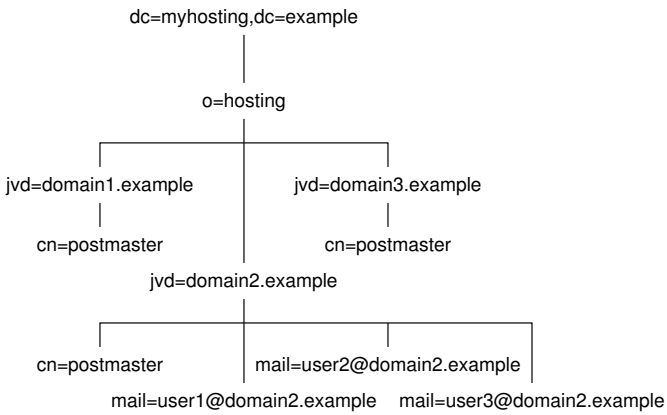

Figure 2. Sample LDAP Tree for a Web Hosting Company

4.3.1. Configuring slapd

All **slapd** configuration is in `slapd.conf`.

4.3.1.1. Adding Schemas

You need to make Jamm's schema file available, so copy the `jamm.schema` file in the Jamm distribution to `/usr/local/etc/openldap/schema/jamm.schema`. `jamm.schema` depends on `cosine.schema` and `nis.schema`. Add these lines to `slapd.conf`:

```
include         /usr/local/etc/openldap/schema/cosine.schema
include         /usr/local/etc/openldap/schema/nis.schema
include         /usr/local/etc/openldap/schema/jamm.schema
```

4.3.1.2. Adding a Database Definition

Next, you need to set up a database definition. You can do this with the following lines:

```
database      ldbm
directory     /usr/local/var/openldap-ldbm
suffix        "dc=myhosting,dc=example"
```

The `database` directive specifies the back-end type to use. You should use LDBM as the back-end database. The `directory` directive specifies the path to the LDBM database. The `suffix` directive specifies the root suffix for this database.

4.3.1.3. Creating the root User

The next few lines set up the "super user" or "root" account:

```
rootdn        "cn=Manager,dc=myhosting,dc=example"
rootpw        {SSHA}ra0sD47QP32ASAlaAhF8kgi+8Aflbgr7
```

The `rootdn` entry has complete access to the database, which is why the password is stored outside the actual database. The password in `rootpw` should always be stored in hashed format. Do not store the password in clear text. To convert the clear text password `secret` to a hashed format, use the **slappasswd** command:

```
% slappasswd
New password: secret
Re-enter new password: secret
{SSHA}ra0sD47QP32ASAlaAhF8kgi+8Aflbgr7
```

Take the output from **slappasswd**, and copy that into `slapd.conf`, as we did above.

4.3.1.4. Defining Indexes

To speed up searches, you should create indexes for commonly searched attributes. In OpenLDAP, you can not only choose which attributes to index, but you can choose which types of searches to index on. For example, if you index the field `mail`, you have the option of creating indexes for presence, equality, approximate, and/or substring searches. We want to create the following index policy:

- Create presence and equality indexes on `objectClass`.
- Create equality and substring indexes on `mail` and `cn`.

To implement this policy, add these lines to `slapd.conf`:

```
index   objectClass  pres,eq
index   mail,cn      eq,sub
```

4.3.1.5. Setting up Access Control

The last part in `slapd.conf` is the access control. You can define your own policy, be here’s the one we’ve adopted and Jamm follows:

- The user can change any of their own attributes.
- Anyone in the `postmaster` group of the domain may change any user’s attributes in their domain, including the password. This allows the `postmaster` to reset a users password if they forget it.
- Anonymous (non-authenticated) users may read all information, except the password attribute.

Access control statements are evaluated in order, so they should be defined from most specific to most general. Access to the password attribute, `userPassword`, is the most specific in our case, and hence it's specified first:

```
access to dn=".*,jvd=([^,]+),o=hosting,dc=myhosting,dc=example"
          attr=userPassword
        by self write
        by group/jammPostmaster/roleOccupant=\
            "cn=postmaster,jvd=$1,o=hosting,dc=myhosting,dc=example" write
        by anonymous auth
        by * none
```

> **Note:** Please note that the backslash in the above lines means that the line with the backslash and the following line are actually one. The lines are broken up here due to space concerns.

The `access to` line specifies what entries and attributes to which the following rules apply. The `dn` regular expression matches any entry in a domain of our hosting tree, and `attr` limits these rules to the `userPassword` attribute. Write access is granted to the user itself and anyone in the `postmaster` group. Anonymous users may only access this field when trying to authenticate. For all other cases, access is denied.

Next, all other attributes to entries in a domain's tree are specified:

```
access to dn=".*jvd=([^,]+),o=hosting,dc=myhosting,dc=example"
        by self write
        by group/jammPostmaster/roleOccupant=\
            "cn=postmaster,jvd=$1,o=hosting,dc=myhosting,dc=example" write
        by * read
```

> **Note:** Please note that the backslash in the above lines means that the line with the backslash and the following line are actually one. The lines are broken up here due to space concerns.

This `access to` line is very similar the previous one, except that there is no `attr` specification. Hence, this matches all other attributes other than `userPassword`. Again, write access is granted to the user and anyone in the `postmaster` group. Everyone is granted read access.

Finally, we provide read access to all other elements in the database:

```
access to *
        by * read
```

4.3.2. Creating the Directory Tree

Now that **slapd** is configured, it's time to start adding data to the LDAP directory. We will use the command line tools that come with OpenLDAP and create LDIF files to modify the directory.

4.3.2.1. Creating the Base Directory

The first step is to create a base tree structure with our root node, the hosting organization, and an entry for the `rootdn`. Create a file called `base.ldif` with the following contents:

```
dn: dc=myhosting, dc=example
objectClass: top
objectClass: domain
domainComponent: myhosting
```

```
dn: cn=Manager, dc=myhosting, dc=example
objectClass: top
objectClass: organizationalRole
cn: Manager

dn: o=hosting, dc=myhosting, dc=example
objectClass: top
objectClass: organization
o: hosting
```

Now use **ldapadd**, binding as the root user, to add this LDIF:

```
$ ldapadd -x -D "cn=Manager,dc=myhosting,dc=example" -w secret -f base.ldif
adding new entry "dc=myhosting, dc=example"
adding new entry "cn=Manager, dc=myhosting, dc=example"
adding new entry "o=hosting, dc=myhosting, dc=example"
```

4.4. Postfix

We'll only cover the sections of Postfix that pertain to the mail hosting. To deal with other parts of Postfix setup, please visit the Postfix web page[14]. However, we do recommend that you set up as much of Postfix as you can to run in a chroot environment. In our experience, this means all of the Postfix daemons except `pipe`, `local`, and `virtual`.

4.4.1. Compiling Postfix with LDAP

This is covered fully in `README_FILES/LDAP_README`[15] that comes with the Postfix source. We'll just cover it briefly here.

Download the Postfix source and untar it. You need to rebuild the Postfix `Makefile`s to be aware of LDAP and link against it. To do this, execute the following command.

```
% make makefiles CCARGS="-I/usr/local/include -DHAS_LDAP" \
AUXLIBS="-L/usr/local/lib -lldap -L/usr/local/lib -llber"
```

At this point, follow the normal Postfix compiling and installing instructions as documented in its `INSTALL` file.

4.4.2. Configuring Postfix

While configuring Postfix for this task, we'll be mostly concerned with `/etc/postfix/main.cf`.

For most of the Postfix configuration, you will configure this in a way that makes the most sense for your site and you can follow the documentation contained in the Postfix source or on the Postfix web page[14]. In this document, we'll talk about the settings that are unique to and/or affected by this setup. If any of the configuration examples shown below aren't explicitly attributed to a specific file, assume they would be found in `main.cf`.

4.4.2.1. Procmail

Having Postfix use **procmail** for delivery is easy. All you need to do is define the `mailbox_command` parameter in `main.cf`.

```
mailbox_command = /usr/bin/procmail
```

4.4.2.2. Configuring LDAP sources

You can easily define multiple LDAP sources. LDAP source parameters are documented in `README_FILES/LDAP_README`[15]. The parameter names follow the pattern of `ldapsource_parameter`. The LDAP source name is defined when it is first used. In `main.cf`, you'll need one LDAP source definition per each lookup.

4.4.2.2.1. Transports

```
transport_server_host = localhost
transport_search_base = o=hosting,dc=myhosting,dc=example
transport_query_filter = (&(jvd=%s)(objectClass=JammVirtualDomain)(accountActive=TRUE)(delete=FALSE)
transport_result_attribute = postfixTransport
transport_cache = yes
transport_bind = no
transport_scope = one
```

This first LDAP source definition is for defining the transport for domains. Postfix uses the transport to determine what to do next with the mail. For our purposes it will always be the **virtual** delivery agent that comes with Postfix. However, we want to be able to dynamically look up new domains, so we store the transport in the JammVirtualDomain entry in LDAP.

We've named this LDAP source `transport`. In our configuration, as specified by the `server_host` line, our LDAP server is running on localhost. Our search base is the top of the hosting subtree we defined in our LDAP server, and according to `scope` we only want to search the directory level right under the base. We're quering for items where the `jvd` element matches the domain of the e-mail recipient as well as items that are of the `jammVirtualDomain` object class. The `transport` agent is defined in the `postfixTransport` attribute. We do not want to bind to the LDAP server, we just want to do an anonymous lookup. Also, to help us over performance humps, we'll cache the results. The default cache has a life time of 30 seconds and has a size of 32K.

4.4.2.2.2. Aliases

```
aliases_server_host = localhost
aliases_search_base = o=hosting,dc=myhosting,dc=example
aliases_query_filter = (&(objectClass=JammMailAlias)(mail=%s)(accountActive=TRUE))
aliases_result_attribute = maildrop
aliases_bind = no
aliases_cache = yes
```

This LDAP source definition is for virtual aliases. We've named this LDAP source `aliases`. We're querying for items where the `mail` elements matches the email recipient as well as items that are of the `jammMailAlias` object class. The destination of the alias is the `maildrop` attribute. Because we have not specified a `scope` in our ldap definition, it will perform the default search of the entire subtree under the base.

4.4.2.2.3. Accounts

```
accounts_server_host = localhost
accounts_search_base = o=hosting,dc=myhosting,dc=example
accounts_query_filter = (&(objectClass=JammMailAccount)(mail=%s)(accountActive=TRUE)(delete=FALSE))
accounts_result_attribute = mailbox
accounts_cache = yes
accounts_bind = no
```

The `accounts` source is very similar to our `aliases` source. The big difference here is that we're looking for entries that have an object class of `jammMailAccount` and we're interested in the `mailbox` attribute of the resulting match.

One of the side effects of the modular nature of Postfix is that a second ldap source for accounts is needed to help Postfix determine if an address is valid before Postfix tries the catchall address.

```
accountsmap_server_host = localhost
accountsmap_search_base = o=hosting,dc=myhosting,dc=example
accountsmap_query_filter = (&(objectClass=JammMailAccount)(mail=%s)(accountActive=TRUE)(delete=FALSE
accountsmap_result_attribute = mail
accountsmap_cache = yes
accountsmap_bind = no
```

This lookup is exactly the same as the first account lookup except that its named `accountsmap` and the returned attribute is the `mail` attribute.

4.4.2.3. The transport map

Now that the `transport` LDAP source is defined, we need to make Postfix aware of it. This is taken care of using the `transport_maps` and `mydestionation` parameters in `main.cf`. In your setup `mydestination` is probably already defined, so you'll just want to append `$transport_maps` to the end of the line.

```
transport_maps = ldap:transport
mydestination = $myhostname, localhost.$mydomain, $mydomain, mail.$mydomain,
                www.$mydomain, ftp.$mydomain, $transport_maps
```

4.4.2.4. The virtual alias maps

Now that the `aliases` and `accountsmap` LDAP sources are defined, we need to let Postfix know to use them. This is taken care of using the `virtual_maps` parameter in `main.cf`

```
virtual_maps = ldap:aliases, ldap:accountsmap
```

4.4.2.5. The virtual accounts

Telling Postfix about the virtual accounts is a bit trickier than the aliases. This is due to the fact that we need to define a lot of extra information about the virtual mail storage.

For this example, we assume that there is a `vmail` Unix account created that has a UID of 101, a GID of 101, and its home directory is `/home/vmail`. We will use the home directory of the vmail user as the place where we store our virtual mail repository.

```
virtual_mailbox_base = /home/vmail/domains
virtual_mailbox_maps = ldap:accounts
virtual_minimum_uid = 101
virtual_uid_maps = static:101
virtual_gid_maps = static:101
```

Most of the above is pretty straight forward, except for `virtual_minimum_uid`, `virtual_uid_maps`, and `virtual_gid_maps`. The Postfix documentation states "[`virtual_minimum_uid`] specifies a minimum UID that will be accepted as a return from a `virtual_uid_maps` lookup. Returned values less than this will be rejected, and the message will be deferred."[16] Since we have decided that all mail for virtual accounts will be stored using the `vmail` Unix account, we set the `virtual_minimum_uid` to be the UID of `vmail`. Also, we set the `virtual_uid_maps` and `virtual_gid_maps` to a special static map and hard code it to the UID and GID of the `vmail` user. All of the parameters shown here are fully documented in `README_FILES/VIRTUAL_README`[16] that comes with the Postfix source.

We also need to edit the `local_recipient_maps` parameter to look at the `virtual_mailbox_maps` so Postfix knows who is a user our mail server supports and who is not. The main reason we do this is so Postfix can reject mail for unknown users.

```
local_recipient_maps = $alias_maps unix:passwd.byname $virtual_mailbox_maps
```

4.5. Courier

You do not need to follow any special instructions for installing Courier, so please see its documentation for full instructions. It should auto-detect LDAP and build it in. You should seriously consider passing the `--enable-workarounds-for-imap-client-bugs` option to **./configure**, otherwise Netscape mail users may have trouble interacting with your server. Yeah, this bends the IMAP protocol a little bit, but it's better to have happy users than a perfect protocol with unhappy users.

4.5.1. Configuring the Authentication Daemon

Courier uses an authentication daemon to keep authentication separate from the other parts of the system. We need to configure it so that a valid email user is either found in LDAP or in PAM. You specify this in `authdaemonrc` using the `authmodulelist` parameter:

```
authmodulelist="authldap authpam"
```

4.5.2. Configuring LDAP

All LDAP parameters are in `authldaprc`. Most parameters are self explanatory. You also need to map all virtual users to the `vmail` account. Here is a summary of the updates you need to make to `authldaprc`:

```
LDAP_GLOB_UID           vmail
LDAP_GLOB_GID           vmail
LDAP_HOMEDIR            homeDirectory
LDAP_MAILDIR            mailbox
LDAP_CRYPTPW            userPassword
LDAP_FILTER             (objectClass=JammMailAccount)(accountActive=TRUE)(delete=FALSE)
```

Three other settings we must concern ourselves with is `LDAP_AUTHBIND`, `LDAP_BINDDN`, and `LDAP_BINDPW`. These settings relate to authenicating the user. `LDAP_AUTHBIND` is mutually exclusive with `LDAP_BINDDN` and `LDAP_BINDPW`. We recommend using `LDAP_AUTHBIND`. A comment in `authldaprc` mentions a memory leak in OpenLDAP when using `LDAP_AUTHBIND`, but it has been fixed in OpenLDAP version 2.0.19.

If you use `LDAP_BINDDN` and `LDAP_BINDPW`, you are limited to the crypt, MD5, and SHA algorithms for passwords. SMD5 and SSHA are not available. Also, you also must put the root LDAP password in clear text in `authldaprc` when defining `LDAP_BINDPW`. There are security issues with putting the root LDAP password in clear text, so you should defintely use `LDAP_AUTHBIND` if you can.

The last change you need to do is to enable the IMAP server by setting the `IMAPDSTART` parameter to `YES`. You should now be able to use the `courier-imap.sysvinit` startup script to start and stop the IMAP daemon.

4.5.3. Setting up IMAP over SSL

If you had OpenSSL installed when compiling Courier, you will have support for IMAP over SSL. Courier can either do SSL using the `STARTTLS` extension or use a separate port for SSL connections. We prefer not to use `STARTTLS`, and have separate ports for SSL and non-SSL connections. You also need to give Courier the full path to your certificate file. And finally, you need to enable the SSL daemon, as it is off by default. Here is how we modified our `imapd-ssl`:

```
IMAPDSSLSTART=YES
IMAPDSTARTTLS=NO
TLS_CERTFILE=/usr/local/share/imapd.pem
```

4.6. Jamm

Installing and configuring a web servlet container like Tomcat or Resin is outside the scope of this document. However, once you have a working servlet container, installing and configuring Jamm is a snap.

Drop the jamm.war file into the webapp deployment directory. Make a directory named `jamm`, cd into that directory and then un(j|w)ar the file into the `jamm` directory. **cd** to the WEB-INF directory. Copy `jamm.properties.dist` to `jamm.properties`, and edit jamm.properties as apppropriate.

```
% mkdir jamm
% cd jamm
% jar -xf ../jamm-0.9.1.war
% cd WEB-INF
% cp jamm.properties.dist jamm.properties
```

Now you need to edit `jamm.properties` as appropriate. To continue to follow our examples for `dc=myhosting,dc=example`, we've edited the following lines in `jamm.properties`.

```
jamm.ldap.search_base = o=hosting,dc=myhosting,dc=example
jamm.ldap.root_dn = cn=Manager,dc=myhosting,dc=example
```

> **Note:** None of the values in `jamm.properties` should have quotes around them. This will cause problems at run time as Jamm is not expecting them. This has bitten people in the past when they copied their `rootdn` from `slapd.conf`.

4.7. SquirrelMail

Since SquirrelMail works directly against IMAP it does exactly what we want it to out of the box. Consider it to be the same as any other IMAP client. There is nothing special about SquirrelMail's setup in this configuration, just follow its install documentation.

We recommend that you set up SquirrelMail on a webserver that is running SSL. This will allow you to make sure that passwords are not going across the network in the clear.

5. Administration

Jamm allows for three levels of access: the site admin, the domain admin, and the user. The site admin controls the entire site and has access to every option all the time, very much like root on a unix system. The domain admin can add, remove, and modify accounts and aliases for his domain as well as assign other people to be a domain admin. The user can only effect his settings.

5.1. Site Admin

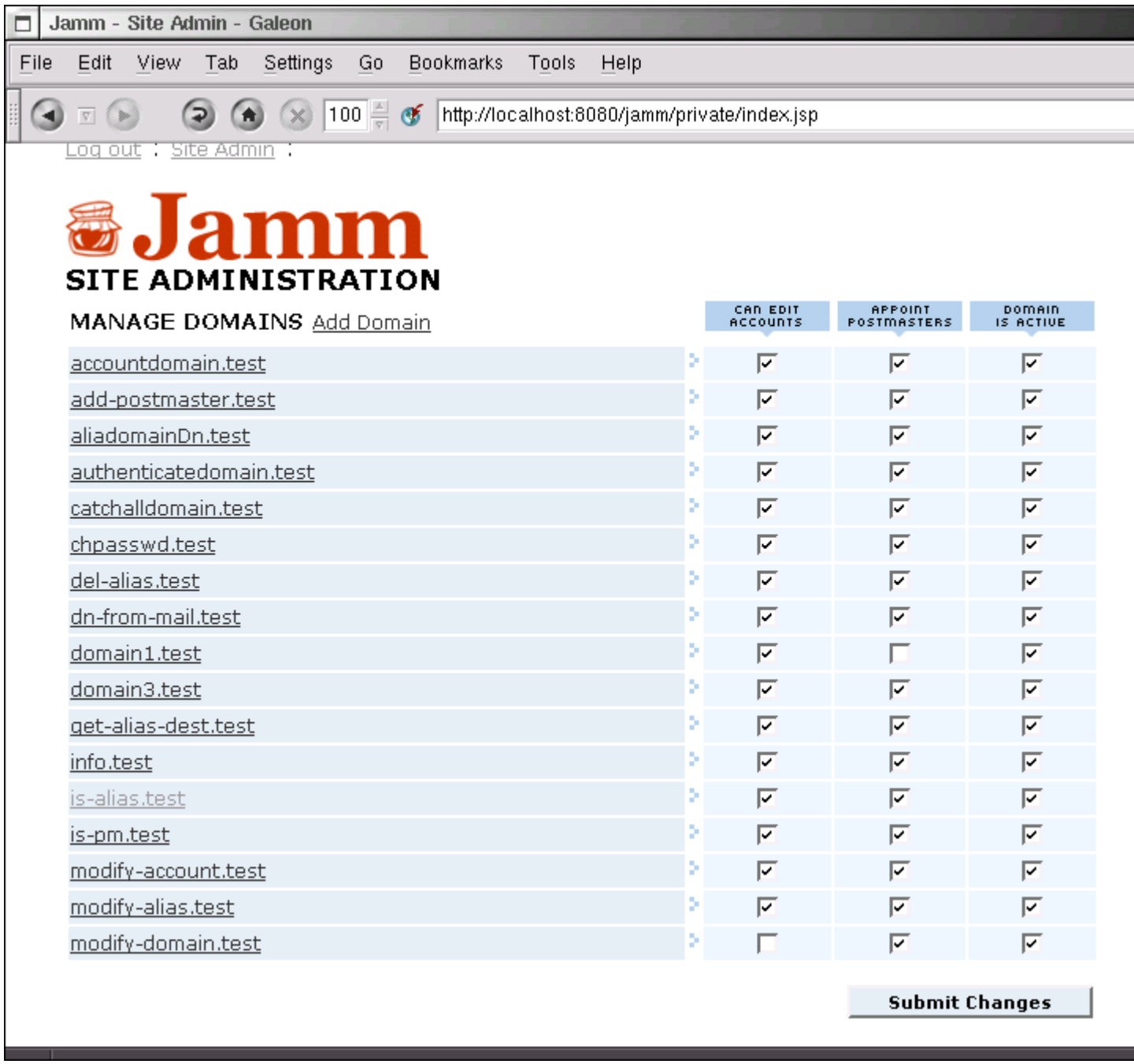

Figure 3. Site Admin Page Screenshot

When a site admin logs in, they are presented with a list of domains. They can click on the domain to drill down to that domain admin page or manipulate the capabilities of the domain admin.

Can Edit Accounts controls the ability for a domain admin to add and remove virtual accounts. When this is switched off, the domain admin can still modify the attributes of existing accounts such as the password.

Appoint Postmasters controls the ability for a domain admin to grant the powers of domain admin to other accounts in the domain. With this turned off, only the site admin can give users domain admin access.

Domain Is Active turns on or off the "active" flag on the domain in ldap. If your mail server or imap server are configured to pay attention to this flag, one can turn on or off domains temporarily without removing them from ldap.

5.2. Domain Admin

Figure 4. Domain Admin Page Screenshot

When a domain admin logs in, they are presented with a list of accounts and aliases for their domain. They can click on a user to drill down to that user admin page, add or delete accounts or aliases, appoint other admins/postmasters, and activate and deactivate accounts. Some of this options may not be present depending on how the site admin has configured the domain's capabilities.

Delete Account does pretty much what it says it will.

Account Is Active actives or deactives an account without deleting it. Much like **Domain Is Active**, your mail server and imap servers must be configured to pay attention to this flag inside ldap.

Postmaster gives or removes the ability for that user to act as a domain admin.

5.3. User Admin

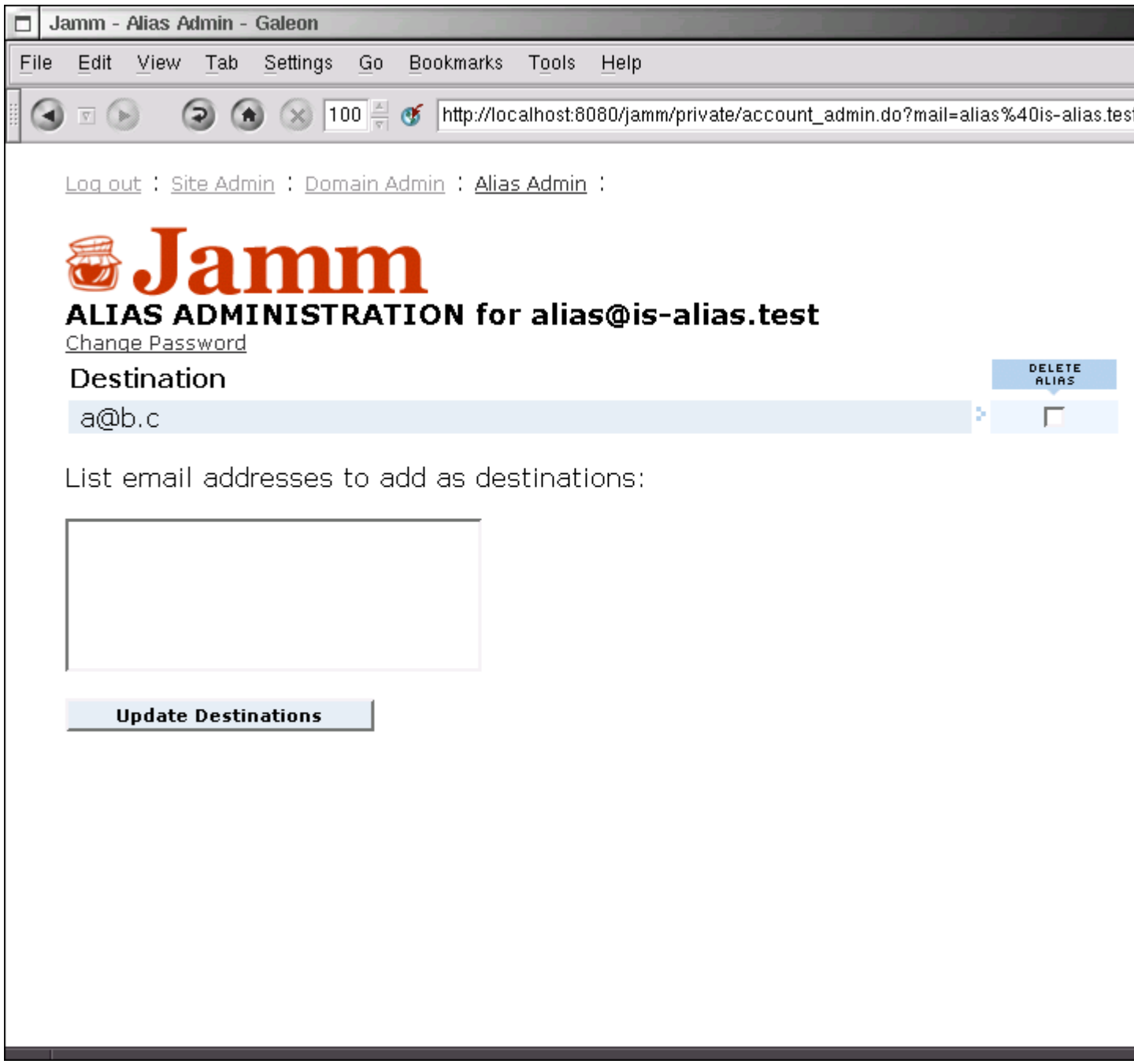

Figure 5. User Admin Page Screenshot

When a user logs in, they are presented with a user screen appropriate to whether they have an account or an alias. Currently, all that a user with an account can do is change their password. An alias user is a bit more intereting, they can edit their destination(s).

To add destinations to an alias, the user only needs to add them in the text area in either a comma seperated list or one per line. To delete destinations, just check the box next to the destination to be deleted.

5.4. Account creation notes

When you create an account or an alias inside the LDAP database it will instantly become active as far as the mail system is concerned. For virtual accounts, it should be noted that the Unix directory in `~vmail` is not created at this time. However, we can work around this because Postfix's `virtual` delivery agent will create the necessary directories the first time it has to deliver mail. Due to this fact, we recommend sending a *welcome* e-mail as soon as you create the account.

5.5. Account deletion notes

When you delete an account or an alias in the LDAP database, it will instantly become inactive. For virtual accounts, it should be noted that the Unix file system isn't cleaned up, i.e. the data remains on disk until a sysadmin can remove it. This will allow you to keep the data from dead accounts around for a grace period in case the account was deleted in error. However, if another account is created with the same name with the same mail path, the data will be available to the new user. This could be considered a privacy violation for the previous user.

6. Thanks

The authors would like to thank their friend JD for asking the innocent question that started all of this off. They'd also like to thank the National Assocation of REALTORS, specifically the Center for REALTOR Technology, for giving them the time during the work day to work on this paper. It should go without saying, but we'd also like to thank all the developers who have worked on, and continue to work on, the Open Source software that we love to use everyday.

7. About the authors

7.1. Dave Dribin

Dave has been using Unix since 1991 and Linux since 1993. He has been professionally developing software for or on Unix since 1995. He is currently an independent consultant specializing in custom software ranging from embedded systems in C and assembly language to enterprise systems in Java.

7.2. Keith Garner

Keith has been using Linux since January 1994. He originally installed it to learn Unix better and ended up becoming a believer and advocate of the Open Source movement. He is currently employed by the National Association of REALTORS as a Strategic Architect in the Center for REALTOR Technology. He lives in Roselle, IL, with his wife, their cat, and 6 computers.

Bibliography

[1] *Courier-IMAP Web Site*, http://www.inter7.com/courierimap/ .

[2] *Exim Web Site*, http://www.exim.org/ .

[3] *Freshmeat Web Site*, http://freshmeat.net/ .

[4] *JAMM Web Site*, http://jamm.sourceforge.net (http://jamm.sourceforge.net/) .

[5] *GQ Web Site*, http://biot.com/gq/ .

[6] *Keith's GQ Patches*, http://www.kgarner.com/code/ .

[7] *IMP Web Site*, http://www.horde.org/imp/ .

[8] *Internet Assigned Numbers Authority*, http://www.iana.org/ .

[9] *iPlanet Directory Server*, http://www.iplanet.com/products/iplanet_directory/home_directory.html .

[10] *iPlanet Directory Server 5.1 Deployment Guide*, http://docs.sun.com/source/816-5609-10/index.html .

[11] *Novell eDirectory*, http://www.novell.com/products/edirectory/ .

[12] *OpenLDAP Web Site*, http://www.openldap.org/ .

[13] *Postfix Overview - Introduction*, http://www.postfix.org/motivation.html .

[14] *Postfix Web Site*, http://www.postfix.org/ .

[15] *Postfix LDAP_README*, README_FILES/LDAP_README from the Postfix 1.1.3 source. .

[16] *Postfix VIRTUAL_README*, README_FILES/VIRTUAL_README from the Postfix 1.1.3 source. .

[17] *Procmail Web Site*, http://www.procmail.org/ .

[18] *Qmail Web Site*, http://www.qmail.org/ .

[19] *Qmail-LDAP Web Site*, http://www.nrg4u.com/ .

[20] *QmailAdmin Web Site*, http://www.inter7.com/qmailadmin/ .

[21] *vpopmail Web Site*, http://www.inter7.com/vpopmail/ .

[22] *Sendmail Web Site*, http://www.sendmail.org/ .

[23] *SquirrelMail Web Site*, http://www.squirrelmail.org/ .

[24] *University of Washington IMAP Web Site*, http://www.washington.edu/imap/ .

[25] *OctetString Web Site*, http://octetstring.com/ .

Notes

1. You may spread virtual users across multiple Unix accounts, too, for example, creating a Unix user for each virtual domain.