Ecartis
Modular Mailing List Manager
http://www.ecartis.org/

November 20, 2002

Version History

2002-04-29 Rewrite from Listar docs to Ecartis docs; added paragraph to Introduction about name change from Listar to Ecartis

Notes About This Document

The Ecartis manual is most definitely a work in progress. As is common with many software projects, development of the software has far exceeded development of the documentation to explain it; this is a shortcoming we are attempting to address.

Discussion of this documentation should be directed to the mailing list ecartis-doc@ecartis.org. Subscription information for the list is available at `http://www.ecartis.org`. Anyone who has submissions they would like added to the documentation, or has suggestions for rewording, changes, etc. to the existing documentation should direct their comments to this list.

For purposes of portability, this documentation is currently maintained in LaTeX, a very simple, yet powerful text markup language that provides us the ability to easily generate versions of this documentation formatted as PostScript, PDF, HTML, or raw text. Those wishing to make direct changes to the documentation source should first familiarize themselves with LaTeX 2_ε, and should keep the ideal of portability in mind when making decisions regarding format.

Contents

Chapter 1

Introduction

1.1 What is a Mailing List Manager?

A mailing list manager is a piece of computer software which accepts a piece of e-mail from a single source and distributes it to a number of recipients. Possible uses for such a piece of software include a monthly newsletter for customers of a business, a way to distribute information to users of a particular software package — for example, notification of a security fix — or a way for people who share a common interest to communicate with each other.

How people choose to implement mailing lists can vary widely. The simplest method involves setting up a single address on an e-mail domain you control, and forwarding it to a number of people. This has a number of disadvantages, however; users have no way to add themselves to the list, or remove themselves from it, there are no ways to restrict who can send mail to a distribution list, and other such headaches.

Most people, therefore, choose to use a piece of software to manage such lists for them; hence the term mailing list manager, or MLM for short. Many people also refer to such packages as 'list-servers,' as they are server software for managing lists. There are an ever-increasing number of MLMs available out there, but almost all share certain common traits; the ability for users to subscribe or unsubscribe themselves from a distribution list, the ability for an administrator to manually remove a user, the ability to restrict posting to a small number of individuals, and so on. In addition, some MLMs support many additional advanced features, such as the ability to filter out unsolicited commercial e-mail (UCE, also known as 'spam').

1.2 A Brief History of MLMs

Back in the mid-1980's, the system of interconnected computers we know as the Internet was not yet around. While in the United States, there was some interconnection between colleges and the government's ARPAnet, the only way

any other machines — such as those at different universities — communicated was over a system called BITNET. BITNET machines let messages for each other pile up, and then would call each other over the phone lines and send the messages.

BITNET had a central control post, a Network Information Center (or NIC) called 'BITNIC.' BITNIC kept a number of distribution lists for BITNET users. However, the BITNIC's lists were set up in the primitive way mentioned in Section 1.1; a single address with no way for users to add themselves or remove themselves. If you wished to be on a mailing list, you had to contact the BITNIC staff and have them add you by hand.

Unfortunately, as BITNET grew larger, managing the lists by hand was no longer feasible. Additionally, since all mail for BITNIC was affected by the traffic of the lists — which by now were quite large — even private BITNET e-mail was affected. It may be hard for users of today's Internet to imagine, but try to picture things becoming so slow that when someone sent you an e-mail, it took over a week to arrive in your mailbox. Clearly, something needed to be done.

Since the source of the problem was the traffic on BITNIC's mailing lists, a computer science student named Eric Thomas decided to write a piece of software to replace the manually-managed mailing lists. It also used a number of alternate paths to send e-mail to the list recipients, to keep it from clogging BITNET's mail pathways, but what was more important to the future of MLMs was the fact that this software allowed users to add and remove themselves from the lists, instead of relying on a BITNIC system administrator to do it for them. When it went online in July 1986, a piece of software was managing a mailing list for the first time — Eric Thomas had created the first MLM. Soon thereafter, others created similar packages for other systems, such as the LISTSERV imitation for UNIX, Listproc.

As time went on, Eric Thomas' LISTSERV developed into a commercial product beyond BITNET and was ported to other systems, and is still widely used on the Internet today. However, as the days of BITNET faded into the past and the Internet became a reality, students who wished to run their own lists and did not have access to the funds necessary to purchase a license for LISTSERV began to look at developing their own MLMs to meet their particular needs.

Perhaps one of the most popular is Majordomo, which has been worked on by a variety of people over the years. Majordomo is written in the Perl scripting language, which is perhaps its greatest failing as it makes Majordomo rather inefficient. However, Perl is very powerful for text processing, and thus Majordomo is readily extendible by those who know Perl and are willing to learn Majordomo's source code.

A good — if somewhat biased — summary of the history of MLMs is available online from Lyris Technologies (who are themselves the authors of a commercial MLM called Lyris, which is targeted specifically at business users) at `http://www.lyristechnologies.com/historyls.html`

1.3 History of Ecartis

Ecartis was born as a simple little project called 'uList' (microList) in October 1997. The original author, Rachel Blackman, had been using the Majordomo mailing list package but wished for one that was more efficient and did not require any special system permissions to run. Additionally, Rachel wanted a server that would allow the individual subscribers to change their subscription options — one of the more desirable features of LISTSERV.

The original design for uList was simple enough; users needed to be able to set a few simple options on themselves as well as perform all the standard operations that most MLMs provide. However, Rachel got tired of having to constantly rewrite the core processing code as new functionality was added, and changed uList over to a modular system, where almost all of the system was added to a changeable table of information. Suddenly, the system could be easily changed; a new command could be added with only a few lines of code, or a new step in processing a message to be sent. The small uList program suddenly had more potential. And to go with the new design, the software got a new name: Listar.

Listar developed into a more stable piece of software, and a mailing list was set up using it called listar-dev, for those who were interested in the ongoing development of the project. On January 12, 1998, Joseph (JT) Traub began to work on the project as well. JT's first contribution to the project was the development of a 'dynamic module' system. Since Listar was already based entirely around a dynamic model, this allowed new Listar plugin modules (or LPMs) to be installed and immediately provide new commands, subscription flags, or functionality.

The first public release of Listar came in February of 1998, and it was used only by a few curious parties. However, many provided good feedback on features and functionality they wished to see in such a project, and Listar grew rapidly into a more mature program.

Then, tragedy struck. In October of 1998, the machine that Listar's main development resources were housed on was cracked into by a malicious individual, and the mailing lists were destroyed. The Listar source code remained safe in backup copies, but the lists themselves were no longer available. The recovery from this event took a while, and development on Listar was slow again afterwards at first, until users discovered the site was back and resubscribed to the lists.

Once past the recovery, however, 1999 proved to be a year full of rapid development for Listar. It became even further fleshed out, and began to be used by some large organizations, such as the Internet Software Consortium. The developers eagerly accepted suggestions and created new LPMs to add custom functionality, while folding additional functionality into the core module.

Listar encountered one other setback late in 2000 (again in October) when Rachel received a Cease & Desist letter from the company holding a trademark for ListSTAR, a defunct MLM for Apple computers. Despite the obvious differences in the names (Listar being "List" in Spanish; ListSTAR being List +

Star) no agreement could be reached, and in mid-2001 a name change for the project was announced: Ecartis.

As of this writing, Ecartis approaches the fifth anniversary of its birth as uList, and has a growing user base as well as a budding external developer community. Some of the features that appeared first in Ecartis are now being borrowed for other MLMs. Ecartis is an Open Source project, so users are encouraged to join in creating code for it.

Chapter 2

Installation

2.1 Introduction

There are several ways that Ecartis can be obtained, and thus several methods of installation. Perhaps the most common is as a collection of source code; which will compile on any number of UNIX-type platforms, as well as Microsoft Windows 95/98 or NT.

The package can also be obtained as an RPM (Redhat Package Manager) installation, either from the Ecartis FTP site, or from RedHat's PowerTools collection or the RedHat Contrib—Net. It can also be obtained as a .deb file for Debian GNU/Linux from the Ecartis FTP site or a local Debian mirror. Or lastly, for those more inclined towards living on the edge of development, the source repository for Ecartis is available for read via CVS.

This document makes the assumption that if you choose to download either the .rpm or .deb versions of Ecartis to install, you already know how to use that package manager to install it, and will thus only cover compiling from source tarball, and compiling under Windows.

2.2 Source Tarball Installation

The Ecartis source tarball will unpack into a directory called ecartis-version, where version is the version of the tarball you are unpacking. Fairly straightforward. Feel free to explore the directory tree before moving along to the installation instructions for Unix (Section 2.4) or Windows (Section 2.5).

2.3 CVS Installation

CVS stands for 'Concurrent Versions System,' and is a method of allowing multiple programmers to work on source code simultaneously, from a single repository of source code. The Ecartis installation is designed to be able to

work from a CVS installation, to make it easy to keep an installation up-to-date. To access the Ecartis CVS tree, you need a CVS client capable of CVS-pserver access.

Find the location that you wish to have your 'ecartis' tree, and enter the following command:

```
cvs -d :pserver:guest@cvs.ecartis.org:/usr/cvsroot login
```

When it prompts you for a password, enter guest. Then type:

```
cvs -d :pserver:guest@cvs.ecartis.org:/usr/cvsroot checkout ecartis
```

You will see a list of filenames scroll by as CVS retrieves the current Ecartis source tree, and then you will have a source tree as if you'd unpacked it from the source tarball, except that it has a 'CVS' subdirectory in each directory; this allows your installation to remain synched with CVS. Now, any time you wish to get the latest version, go into the Ecartis directory and do:

```
cvs update -P -d
```

For information on CVS and CVS servers, visit Cyclic Software, the authors of CVS, at `http://www.cyclic.com/` — there are graphical CVS clients available for several operating systems as well as binaries for various systems.

2.4 Compiling Under UNIX

To compile under a UNIX-style system (such as Linux, FreeBSD, SunOS, etc.), Ecartis needs GCC or EGCS. While it is technically possible that Ecartis would compile under the GCC for Windows that Cygnus provides, it is unlikely. Ecartis has a specific Windows port; see Section 2.5 for more information.

The first step under UNIX is to go into the 'src' directory under where you unpacked the tarball (or did a CVS checkout to). Copy Makefile.dist to Makefile. Now you'll want to pull Makefile up in your favorite text editor. There are a variety of comments about what you'll need to tune for specific operating systems; tweak the file as appropriate for your operating system and then save it. Now type 'make'.

Ecartis uses the GNU Make program to handle the build process. Some operating systems, like Linux, provide this as the default 'make' program, while others, like FreeBSD, have it available as 'gmake'. If you have BSD Make, when you run 'make' in the Ecartis src directory, you will get a string of errors that look like:

```
"Makefile", line 114: Need an operator
"Makefile", line 116: Need an operator
"Makefile", line 118: Need an operator
"Makefile", line 129: Need an operator
```

If this is the case, you will need to run 'gmake' instead of 'make'. If the build process is going correctly, you will see output like this:

```
gmake[1]: Entering directory `/home/loki/src/ecartis/src/modules/bounc
er'
gcc -fPIC -DDYNMOD -Wall -Werror -I../../inc -I. -DGNU_STRFTIME   -c
bouncer.c
gcc -shared   -o bouncer.lpm bouncer.o
cp bouncer.lpm ../../build
gmake[1]: Leaving directory `/home/loki/src/ecartis/src/modules/bounce
r'
[Build: bouncer] Built module successfully.
gmake[1]: Entering directory `/home/loki/src/ecartis/src/modules/lista
rchive'
gcc -fPIC -DDYNMOD -Wall -Werror -I../../inc -I. -DGNU_STRFTIME   -c
archive.c
gcc -shared   -o ecartischive.lpm archive.o
cp ecartischive.lpm ../../build
gmake[1]: Leaving directory `/home/loki/src/ecartis/src/modules/ecartis
chive'
[Build: ecartis] Built module successfully.
```

...and so on. The actual text of your output may vary slightly depending on the system. Assuming no build errors occur, you should end up with a final line that looks something like this:

```
gcc   -o ecartis alias.o command.o user.o parse.o list.o core.o forms.
o smtp.o io.o regerror.o regsub.o regexp.o flag.o cookie.o file.o mod
ule.o fileapi.o variables.o internal.o cmdarg.o modes.o dynmod.o unmi
me.o codes.o hooks.o tolist.o mystring.o lma.o userstat.o snprintf.o
moderate.o mysignal.o unhtml.o liscript.o submodes.o lcgi.o upgrade.o
```

If there are no linker errors, you now have all the binaries you need. Type 'make install' (or 'gmake install') and it will place all the Ecartis binaries into the appropriate places. If you wish to run Ecartis directly from this location, it is now ready to go. Otherwise, you need to copy things elsewhere. If you are using the dynamic module mode (Ecartis's recommended configuration), there is a directory under where you unpacked Ecartis which is called 'modules', and contains a number of files with an .LPM extension. And in the directory where you unpacked Ecartis is a binary executable called ecartis. The ecartis binary should be placed wherever you intend to have your Ecartis installation, along with the ecartis.cfg file. The modules should be placed into a modules directory, which by default is a subdirectory called modules under wherever the ecartis binary is. It is possible to split up Ecartis into a number of separate directories, however, as the Debian installation does. At this point, you should have enough information to move along to Section 3, Getting Started.

2.5 Compiling Under Windows

Ecartis can be compiled under Windows, but there are a few caveats. First, it still functions as a mail filter, so you need a way to feed the mail to it; many Windows mail servers do not function in this way. Secondly, it requires Microsoft Visual C++ 5.0 or higher to compile.

If these still meet your criteria, Ecartis is fairly easily compiled. When you unpack the source tree, or do a CVS checkout, you will want to go to the top-level directory, and load the ecartis.dsw file. Then you can simply go and do a batch build of all the targets. In the end, you will have a debug and a release ecartis.exe, in `src\debug\Ecartis.exe` and `src\release\Ecartis.exe`. You will also have `src\modules\debug` and `src\modules\release`, which will contain debug and release builds of all the LPM modules. From here on out, Ecartis is configured like any of the UNIX installations.

There is also a commercially supported mailing list package for Windows which is based on Ecartis. It is called SLList and is sold and supported by Seattle Lab, at `http://www.seattlelab.com/`.

Chapter 3

Getting Started

3.1 Introduction

Most MLMs function as a mail filter — in other words, a program that is invoked when a piece of mail arrives for a specific address, and which is fed the mail that arrived. How individual mail servers function with this varies; the widely-used package sendmail uses a file called aliases to determine things like this, while other packages such as qmail handle it differently. We'll cover a few generic pieces, and then a few specific mail servers.

3.2 Ecartis and Mail Filters

A list running on Ecartis has several addresses associated with it. The list address itself (for example, foolist@domain.com), the list administrators (for example, foolist-admins@domain.com), and so on. In addition, Ecartis has an address for itself (ecartis@mydomain.com). Each of these addresses is represented by an alias. The ecartis@mydomain.com address calls the Ecartis program without any parameters, while the individual list ones have various parameters.

Hence, when you set up the Ecartis aliases for your system, you'll probably have a number of things looking like this:

```
# Ecartis address
ecartis:           |/home/ecartis/ecartis
# Testlist aliases
testlist:         |"/home/ecartis/ecartis -s testlist"
testlist-request: |"/home/ecartis/ecartis -r testlist"
testlist-admins:  |"/home/ecartis/ecartis -admins testlist"
testlist-repost:  |"/home/ecartis/ecartis -a testlist"
testlist-bounce:  |"/home/ecartis/ecartis -bounce testlist"
```

This means that when some e-mail comes in for the 'ecartis' address, the mailserver would run the program /home/ecartis/ecartis and pass the contents of that e-mail to the program. The 'testlist' address, for example, will also run

the program, but it will have some additional parameters. If the aliases look a bit confusing, don't worry; Ecartis will generate them for you when you make a new list, but if your mailserver isn't sendmail, you may have to edit the way the aliases look slightly. Most mailservers use an alias format similar — or identical — to the sendmail format, so for most servers you will be able to use the aliases that Ecartis generates directly.

3.2.1 Sendmail

Sendmail (`http://www.sendmail.org`) is perhaps the widest used mail server on the Internet, and comes preinstalled on most UNIX-type systems. Sendmail is extremely configurable — almost too configurable, as it is possible to get lost in the configuration options — and is more than capable of using Ecartis as a mail filter.

There are several ways to handle setting up Ecartis under Sendmail. Perhaps the easiest is simply to copy and paste the aliases that you get from a Ecartis newlist.pl script into the Sendmail /etc/aliases file, and run the newaliases program. Others may prefer to create a separate aliases file for Ecartis; the method to do this depends on how you're setting up your sendmail config file. The direct method is to go into the /etc/sendmail.cf file, find the line referring to /etc/aliases, and add AliasFile line below it for the Ecartis aliases file.

Perhaps the most common pitfall of people setting up new Ecartis installations — or indeed, any new mail filter under Sendmail — is the fact that many Sendmail installations come with a program called SMRSH already enabled. SMRSH, or the SendMail Restricted SHell, is a program that increases system security by making sure that Sendmail only allows programs in a certain directory to be run as mail filters. Where this directory is varies depending on your Sendmail installation, but a common one is /etc/smrsh.

If any attempt to mail Ecartis meets with a bounce message like the following, you are using SMRSH.

```
----- The following addresses had permanent fatal errors -----

"|/home/ecartis/ecartis -s mylist"

     (expanded from: <mylist@mydomain.com>)


    ----- Transcript of session follows -----

sh: ecartis not available for sendmail programs

554 "|/home/ecartis/ecartis -s mylist"... Service unavailable
```

The solution is to locate the directory that SMRSH is using for its programs, and write a shell script like this, placing it there.

```
#!/bin/sh
/home/ecartis/ecartis $@
```

Of course, /home/ecartis should be the path where you installed the Ecartis binary. It is also vitally important to make sure that this wrapper script is set executable so that smrsh can run it. Also make sure that the directory leading up to the Ecartis binary has execute permissions for whatever user or group your mail server runs as.

Then you'd want to go and change the aliases so that they had the path to the shell script wrapper. For example, changing ...

```
mylist:          |"/home/ecartis/ecartis -s mylist"
```

...to ...

```
mylist:          |"/etc/smrsh/ecartis -s mylist"
```

...then Sendmail should allow Ecartis to be used as a mail filter.

3.2.2 Exim

Exim is another of the mail servers out there, though it doesn't come pre-installed on many — if any — systems. Information on it can be found at `http://www.exim.org/`. Exim was developed at the University of Cambridge over in England. To make Ecartis work with Exim, you could take the simple approach like with Sendmail and paste the Ecartis aliases into the existing Exim aliases file, or you can go into exim.conf and add the following lines:

```
ecartis_aliases:
  driver = aliasfile
  file_transport = address_file
  pipe_transport = address_pipe
  file = /usr/lib/ecartis/aliases
  search_type = lsearch
  user = ecartis
  group = ecartis
```

Other than that, Ecartis should function correctly with Exim out-of-the-box.

3.2.3 Postfix

Postfix (`http://www.postfix.org`) is another mail server, designed with the goal of creating a package as secure and fast as sendmail, while still providing as much backwards compatibility as possible. For Postfix installations, you can once again take the simple route and paste the Ecartis aliases into the default aliases file and then run the postaliases program any time you change it.

If you wish to have a separate Ecartis aliases file, you will need to pull /etc/postfix/main.cf up in your favorite text editor. Find alias_maps line, and add the ecartis aliases file to it. The resulting line will look something like:

```
alias_maps = hash:/etc/aliases, hash:/etc/mail/ecartis.aliases
```

Then run postalias /etc/mail/ecartis.aliases every time you add aliases to the Ecartis file. Of course, /etc/mail/ecartis.aliases should be replaced by the path to where you keep your Ecartis aliases file.

3.2.4 qmail

The qmail (`http://www.qmail.org`) program is perhaps the second most widely used UNIX mail server on the Internet, being considered to be small, fast and secure. However, the method of setting up qmail aliases is far different from setting up aliases under any other mail package. Instead of a single file that contains multiple aliases mapping through to Ecartis, you need to create what are called dot-qmail files in the home directory of the qmail system user (often 'alias').

If the line in a normal aliases file would be ...

```
mylist:          |"/home/ecartis/ecartis -s mylist"
```

... then you would need to create a file in the home directory of the qmail system user. This file would be named .qmail-mylist and which will contain the text `|"/home/ecartis/ecartis -s mylist"`

You could create such a file by going to the appropriate directory and typing:

```
echo "|/home/ecartis/ecartis -s mylist" > .qmail-mylist
```

In other words, for a line like ...

```
address:         command
```

... you want ...

```
echo "command" > .qmail-address
```

Unlike most mail servers, you do not need to run a program to rebuild aliases under qmail.

Ecartis's internal newlist command can create dot-qmail files for a given list; to set up your Ecartis installation to do so by default, pull ecartis.cfg up in your favorite text editor, and add the following line:

```
newlist-qmail = yes
```

Then, when you create a list, it will have a subdirectory called qmail-aliases, and in that directory will be all the dot-qmail files you need to copy over to your qmail global aliases directory.

You can force this feature on by adding the command-line argument -qmail when creating a new list.

It is also worth noting that there is an optional add-on package for qmail by Dan Bernstein called FastForward (`http://www.pobox.com/~djb/fastfoward.html`) which allows qmail to use /etc/aliases.

Chapter 4

Configuring the Server

4.1 Primary (Global) Config File

Chapter 5

Setting Up a List

5.1 Creating the List

Ecartis contains a set of commands internally to make it easier to create a list. This entire process can be invoked from the command line and easily controlled, and the various installed modules are responsible for generating the configuration file and documenting it, as well as any other custom handling for new list creation.

To begin this process, you will want to run the Ecartis binary with the command-line argument -newlist; this takes one parameter, the name of the new list. In addition, if you're creating a list for a virtual host, simply put the virtual host config file at the beginning of the line. Ecartis determines what path to put in the aliases by what path you use when invoking it, so in this case be sure to run it with a full path name. Here are two examples:

```
/home/ecartis/ecartis -newlist testlist

/etc/smrsh/ecartis -c virthost1.cfg -newlist testlist2
```

Ecartis will prompt you for the e-mail address of the list administrator; this should be an address from which mail can be sent as well as received. Do not use a forwarding address for the primary administrator of the list. Once you've entered this information, the script will create the basic list configuration and basic list directory. Then it will give you a block of information that looks something like this:

```
# Aliases for list 'testlist'
testlist: "|/home/ecartis/ecartis -s testlist"
testlist-request: "|/home/ecartis/ecartis -r testlist"
testlist-repost: "|/home/ecartis/ecartis -a testlist"
testlist-admins: "|/home/ecartis/ecartis -admins testlist"
testlist-moderators: "|/home/ecartis/ecartis -moderators testlist"
testlist-bounce: "|/home/ecartis/ecartis -bounce testlist"
testlist-owner: joe@mydomain.com
```

You will want to put this information into your mail server's alias file, and you may need to run a program to rebuild the aliases database (for example, Sendmail users will need to run the newaliases program).

As a side note for advanced users, all the output Ecartis gives in this mode OTHER than the aliases is on stderr, while the aliases are printed to stdout, thus making it easy to simply redirect output and append it to your aliases file.

If you have Ecartis set up to use qmail instead (see Section 3.2.4), you will not get the block of aliases, but will instead find that in the directory of your new mailing list is a subdirectory called qmail-aliases; in this directory are the dot-qmail files you will want to copy over to your qmail global aliases directory.

5.2 Editing the List Configuration

Now, you doubtless want to go tune a few of the default settings in the list configuration file. Go to the directory where you have your Ecartis lists stored, and go into the directory with the name of the list you just made. Bring the config file up in your favorite editor, and edit to taste. There is an appendix of configuration variables at the end of this document.

This section needs more detail.

Once you have things configured to your taste, you're ready to move on to your first interaction with Ecartis!

Chapter 6

Interacting with Ecartis

6.1 Concepts

Unlike many MLMs, Ecartis has something called a 'list context.' What this means is that, much like someone in a conversation, Ecartis remembers what the previous command in a session referred to. If you tell it to do a command on list1 and then don't give it a list name for the next command, it will assume you are still talking about list1. If, however, you tell it to refer to list2 in the second command, it will do so. How this works will become clearer as we move along.

Ecartis also has what is called a 'secure' mode. Certain commands — such as administrative functions — must be performed through a method that prevents people from pretending to be you. Some MLMs only check the address a message comes from before allowing administrative commands, leaving them open to easy hacking by a mail-spoofer. Other MLMs use passwords, which could be discovered. Ecartis uses a unique method which secures it in any situation but that where either the machine that Ecartis runs on has been compromised, or the e-mail account of an administrator has (something out of the realm of control of an MLM). This method is referred to as using 'cookies.'

Ecartis's cookies are used for more than just administrative modes, as well. When a cookie is "baked," (created by Ecartis) it has a specific "flavor." A cookie of one flavor cannot be used for a task requiring a different flavor — for example, a cookie for a user to confirm their subscription could not be used to authorize administrative commands. If a cookie is used, it is "eaten" and cannot be used again. If a cookie remains unused, eventually it goes stale and is removed from the cookie jar. And lastly, a cookie can be baked for a specific person, and then only that person can use it.

It is the cookie system that allows Ecartis to have much of the security and power it does.

6.2 Normal Mode

Normal mode is how the majority of users interact with Ecartis. This is where you send e-mail to the Ecartis server itself with a set of commands. For example, a user might send a message like this ...

```
Date: Tue, 28 Sep 1999 23:26:32 -0700 (PDT)
From: Joe Q. User <joe@someisp.com>
To: ecartis@mydomain.com
Subject:

lists
which
end
```

And Ecartis would send back a message like ...

```
Date: Tue, 28 Sep 1999 23:27:27 -0700 (PDT)
From: Ecartis <ecartis@mydomain.com>
To: joe@someisp.com
Subject: Ecartis command results: lists

>> lists
Ecartis lists available on this machine:

testlist
          A test list.

>> which
Retrieving list subscriptions.

joe@someisp.com is subscribed to the following lists:
      testlist

>> end
Command set concluded.  No further commands will be processed.

---
Ecartis v0.126a - job execution complete.
```

In other words, each line of the message is parsed by Ecartis, which tries to find a valid command in it. If there is a valid command in the subject line, it will use that instead of parsing the body (unless configured to ignore the subject line). What commands are available vary depending on the Ecartis installation, since an LPM can add new commands.

Additionally, if you enclose what is called a "job/eoj wrapper," Ecartis will ignore everything except what is between the beginning and ending of the wrapper. (Multiple job/eoj wrappers can be used per message.) This is useful for administrative functions, as you will see later.

```
// job
commands
// eoj
```

When a command takes a list as one of the parameters, it turns that into the list 'context,' and if you omit a list from the next command, it will assume you mean the same list as before. When a command finally has another list given, it will change the context for following commands. Whenever a command changes the list context, you will be told in the results list:

```
List context changed to 'testlist' by following command.
>> who testlist
Membership of list 'testlist':
        joe@mydomain.com (ADMIN)
        jane@mydomain.com (ADMIN)
        joe@someisp.com

>> stats
Current account flags for 'joe@mydomain.com' on 'testlist':

        REPORTS
        CCERRORS
        ECHOPOST
        MODERATOR

List context changed to 'testlist2' by following command.
>> stats testlist2
Current account flags for 'joe@mydomain.com' on 'testlist2':

        ADMIN
        SUPERADMIN
        ECHOPOST
        REPORTS
        CCERRORS

>> who
Membership of list 'testlist2:
        joe@mydomain.com (ADMIN)
        joe@someisp.com
```

In the above example, the user (joe@mydomain.com) sent the commands:

```
who testlist
stats
stats testlist2
who
```

Now you know how to issue commands to Ecartis, but what commands can you issue? Well, there are a great many of them, and they vary from installation to installation because of customization. Luckily, there is a way to query Ecartis for what commands it supports. If you send Ecartis the command commands, it will send back a list of what commands it supports on a given installation. Some commands are only ever useful in messages Ecartis generates, such as for secured-mode messages.

6.3 Secure (Administrator) Mode

Obviously, there needs to be a way to issue administrative commands to Ecartis; it isn't feasible to expect every user to have access to the files on disk that control a list they might potentially be an administrator on. But it is equally obvious that it's undesirable to have random unauthorized users able to issue administrative commands. Clearly, some sort of authentication is needed. Ecartis achieves this authentication with cookies, as mentioned before.

There are two ways to handle administrative commands, but they have the same end result. The first method is to send Ecartis the command admin list. If you are a valid administrator for list, Ecartis will send you what is called an admin wrapper. This is something that looks like:

```
// job
adminvfy testlist 37F1C6FC:58BB.1:ybxvznvfbabgnxharg
adminend
// eoj
```

You want to fill out any administrative commands (things like getconf, or setfor, which require administrative permissions) between the adminvfy and adminend lines, and send the wrapper back to Ecartis.

The second method was added in a later version of Ecartis, and is more convenient in most cases. Instead of sending the command admin list, send the command admin2 list, and then give it the commands you want in the wrapper followed by adminend2. That will send back the wrapper already filled out, and you can simply hit 'reply' to approve it. This is also where job/eoj blocks come in handy! Picture receiving a note from a user (say, joe@someisp.com) which says that they need to be unsubscribed and can't remember how. You could simply reply and carbon-copy Ecartis, saying something like:

```
> Hey, sorry... I know I should know how to unsubscribe myself
> But...
> Could you unsubscribe me for me?

Sure!

// job
admin2 testlist
unsubscribe joe@someisp.com
adminend2
// eoj
```

Then you would receive back an already filled-out administrative wrapper such as:

```
// job
adminvfy testlist 37F1C6FC:58BB.1:ybxvznvfbabgnxharg
unsubscribe joe@someisp.com
adminend
// eoj
```

Ecartis can parse reply-formatted messages in many cases (the exceptions being the putconf command and moderated messages), so when you use the admin2 command, you can simply hit reply to the wrapper it generates.

Any commands that are listed with an (ADMIN) after them in the commands list that Ecartis will generate will only work between an adminvfy line and an adminend line. Normal commands will also work between those lines, but some have different uses; for example, the stats command normally functions as stats list, to allow a user to view statistics on themselves for a list. But in admin mode, you are locked to a specific list - the list you validated administrator permissions for - so the command you would enter instead is stats user, to allow an administrator to view stats for a specific user on that list.

6.4 Web Interface

Information on LSG/2 needs to go here (after LSG/2 is finished).

Chapter 7

List Moderation

Information on moderating mailing lists.

Chapter 8

Modules

Will give a listing of the default Ecartis modules (administrivia, etc.) and what they provide.

Chapter 9

Example List Configurations

Some useful list configurations, like an announcement list or a connected set of lists like the nwcpp-discuss and nwcpp-announce lists, or the ecartis-support/ecartis-dev/ecartis-announce setup.

Chapter 10

Writing Modules

Instructions on how to write a Ecartis module.

Appendix A

Config Variables