EZFAQ 0.32 - ezmlm-idx and ezmlm FAQ

Fred Lindberg, lindberg@id.wustl.edu & Fred B. Ringel, fredr@rivertown.net 22-MAR-1999

This document is a collection of frequently asked questions about ezmlm-idx. Where applicable, ezmlm itself is also covered. This FAQ presumes familiarity with Unix, and with the basic concepts of E-mail and mailing lists. This FAQ is updated for ezmlm-0.53 and ezmlm-idx-0.32.

Contents

1 General Information

1.1 Acknowledgements

Many ezmlm users have contributed to improvements in ezmlm-idx. These are listed in the **README.idx** file in the ezmlm-idx distribution. Others have through questions and suggestions inspired parts in this FAQ, or pointed out errors or omissions. Thanks! Direct contributions are attributed to the respective authors in the text. Thanks again!

1.2 What is this document?

This FAQ contains answers to many questions that arise while installing ezmlm, ezmlm-idx, and while setting up and managing ezmlm mailing lists. See 1.4 () for a brief summary of what is ezmlm and what is ezmlm-idx.

Many aspects of ezmlm are covered in several places in this FAQ. The early sections explain how ezmlm works. Later sections discuss how to deal with possible errors/problems. Subsequent sections discuss details of customization and list setup in a *HOWTO* form. Finally, there are sections on information philosophy for moderated lists and on security aspects on ezmlm lists.

This is an evolving document. If you find any errors, or wish to comment, please do so to the authors. This FAQ is currently aimed at system administrators and knowledgeable users, and heavily weighted towards questions specific to the ezmlm-idx add-on.

If you have problems with the ezmlm-idx package, please start by reading the "man" pages which come with each program, then this document and other ezmlm documentation which is identified here. If you have exhausted these resources, try the ezmlm and qmail mailing lists and their respective mailing list archives. If you have solved a problem not in the documentation, write it up as a proposed section of a FAQ and send it to the authors. This way, it can be added to the next version of this FAQ.

1.3 Terminology

This document uses a number of terms. Here are the meanings ascribed to them by the authors.

DIR

The base directory of the list.

SENDER

The envelope sender of the message, as passed to ezmlm by qmail via the $SENDER environment variable.

LOCAL

The local part of the envelope recipient. For `list-get-1@host`, it is usually *list-get-1*. If host is a virtual domain, controlled by *user-sub*, then local would be *user-sub-list-get-1*.

moddir

Base directory for moderators. Moderator E-mail addresses are stored in a hashed database in **moddir/subscribers/**. By default, "moddir" is **DIR/mod/**.

To add or remove moderators:

```
% ezmlm-sub DIR/moddir moderator@host.domain
% ezmlm-unsub DIR/moddir moderator@host.domain
```

dotdir

The second argument of ezmlm-make is the main .qmail file for the list. dotdir is the directory in which this "dot file" resides, i.e. the directory part of the "dot" argument. This is usually the home directory of the user controlling the list (but NOT necessarily of the one creating the list). Thus, *dotdir* is **~alias/** if "root" creates a list:

```
# ezmlm-make ~alias/list ~alias/.qmail-list ...
```

dotdir is where the **.ezmlmrc** file is expected when the ezmlm-make(1) "-c" switch is used (see 7 (Customizing ezmlm-make operation)).

ezmlm binary directory

The directory where the ezmlm-binaries are normally stored, as defined at compile time in **conf-bin**. This is compiled into the programs and does not change just because you have moved the program.

ezmlm-get(1)

This is a reference to the ezmlm-get.1 man page. Access it with one of the following:

```
% man ezmlm-get
% man 1 ezmlm-get
```

or if you have not yet installed ezmlm-idx (replace "xxx" with the version number):

```
% cd ezmlm-idx-0.xxx
% man ./ezmlm-get.1
```

basedir

The list directory when referencing the list subscriber address database. For E-mail addresses stored in a set of files within **DIR/subscribers/**, the "basedir" is "DIR".

address database

A collection of E-mail addresses stored in a set of files within the "subscribers" subdirectory of the basedir, **DIR/subscribers/**.

message moderator

An address to which moderation requests for posts to the list are sent. The moderation requests are formatted with "From:"-"reject" and a "To:"-"accept" default headers for moderator replies. A reply to the "reject" address leads to the rejection of the post. A reply to the "accept" address leads to the acceptance of the post. Any E-mail address can be a moderator E-mail address. Any number of moderator E-mail addresses can be used. If a post is sent from a moderator E-mail address, the moderation request is sent to that E-mail address only. If a post is sent from an E-mail address that is not a moderator, a moderation request is sent to all moderators.

The first reply to the moderation request determines the fate of the message. Further requests for the action already taken are silently ignored, while a request for the contrary action results in an error message stating the actual fate of the message. Thus, if you want to "accept" the message and it has already been accepted, you receive no reply, but if you attempt to "reject" it, you will receive an error message stating that the message already has been accepted.

Most lists are not message moderated. If they are, the owner is usually a "message moderator", sometimes together with a few other trusted users.

For an announcement list, it is common to make all the "official announcers" "message moderators". This way, they can post securely and "accept" their own posts, while posts from other users will be sent to this set of "official announcers" for approval.

subscription moderator

An E-mail address where subscription moderation requests are sent. A subscription moderation request is sent after a user has confirmed her intention to subscribe. The subscription moderation request is sent to all moderators. As soon as a reply to this message is received, the user is subscribed and notified. Any E-mail address can be a subscription moderator and any number of subscription moderators can be used.

Unsubscribe requests are never moderated (except when the ezmlm-manage(1) "-U" flag is used and the sender attempts to remove an address other than the one s/he is sending from). It is hard to imagine a legitimate mailing list that would want to prevent unsubscriptions.

remote administrator

When a remote administrator subscribes or unsubscribes a list member, the "confirm" request is sent back to the remote administrator, rather than to the subscriber's E-mail address. This allows the remote administrator to (un)subscribe any list member without the cooperation of the subscriber at that address. Any E-mail address can be a remote administrator and any number of E-mail addresses can be remote administrators.

The set of E-mail addresses that are "remote administrators" and "subscription moderators" are always the same. This set of E-mail addresses can be "remote administrators", "subscription moderators" or both.

For most lists, the owner would be the "remote administrator", if s/he wishes to moderate messages, the owner would be the "message moderator" and if s/he wishes to moderate subscriptions the owner would also be the "subscription moderator".

The list's "message moderator(s)" can be the same, but can also be set up to be completely different.

Changing list "ownership"

Within this FAQ there are references to the need to check or change the list "ownership." This is not a reference to the individual user who is the "list-owner", but a reference to the ownership of the files by your operating system which make up the list and reside in **DIR**/.

To change the ownership of **DIR**/ and everything within:

```
% chown -R user DIR
% chgrp -R group DIR
```

Depending on your system/shell, it may be possible to combine these commands into either:

```
% chown -R user.group DIR
% chown -R user:group DIR
```

1.4 What is the difference between ezmlm and ezmlm-idx?

ezmlm-0.53 is a qmail-based mailing list manager written by Dan J. Bernstein. It has all the basic functionality of a mailing list manager, such as subscriber address management including automated bounce handling as well as message distribution and archiving.

ezmlm-idx is an add-on to ezmlm. It adds multi-message threaded message retrieval from the archive, digests, message and subscription moderation, and a number of remote administration function. It modifies the configuration program ezmlm-make(1) so that it uses a text file template rather than compiled-in texts in list creation. In this manner, ezmlm-idx allows easy setup of lists in different languages and customization of default list setup. ezmlm-idx also adds MIME handling, and other support to streamline use with languages other than English. As an ezmlm add-on, ezmlm-idx does not work without ezmlm and tries to be compatible with ezmlm as much as possible. ezmlm-idx also modifies the ezmlm subscriber database to be case insensitive to avoid many unsubscribe problems.

ezmlm-idx-0.32 adds improved handling of very large lists with optimized bounce handling, ezmlm-split(1) for forwarding (un)subscribe requests to sublists to allow sublisting transparent to the subscriber, and SQL support to allow sublisting with improved message authentication and monitoring of list function, as well as dynamic addition/removal/reconfiguration of sublists. Also, subscriber "From:" lines are logged with support for finding a subscription address from a name. The qmail DEFAULT variable is used, if present. Together, these additions eliminate the most common problems making ezmlm use and administration even easier.

This document is a FAQ for ezmlm-idx. However, many of the basic items that are discussed also apply to ezmlm per se. Referring to the two paragraphs above, it should be relatively easy to figure out which features require ezmlm-idx.

1.5 Where can I get all of the ezmlm-related programs?

We have now registered `ezmlm.org` to make access to ezmlm-idx and related programs/documentation easier. `www.ezmlm.org` is currently an alias for Fred B. Ringel's `www.rivertown.net/~ezmlm/` and `ftp.ezmlm.org` an alias for Fred Lindberg's `ftp.id.wustl.edu`.

Dan J. Bernstein's ezmlm-0.53

- `<ftp://cr.yp.to/pub/software/ezmlm-0.53.tar.gz>`
- `<ftp://ftp.ezmlm.org/pub/qmail/ezmlm-0.53.tar.gz>`
- `<ftp://ftp.ntnu.no/pub/unix/mail/qmail/ezmlm-0.53.tar.gz>`
- `<ftp://ftp.pipex.net/mirrors/qmail/ezmlm-0.53.tar.gz>`
- `<ftp://ftp.jp.qmail.org/qmail/ezmlm-0.53.tar.gz>`
- `<ftp://ftp.rifkin.technion.ac.il/pub/qmail/ezmlm-0.53.tar.gz>`
- `<ftp://ftp.mira.net.au/unix/mail/qmail/ezmlm-0.53.tar.gz>`
- `<http://www.qmail.org/>`

The latest version of ezmlm-idx

ezmlm-idx releases are numbered "ezmlm-idx-0.xy[z]". Versions with the same "x" are backwards compatible. A change in "x" signifies major changes, some of which *may* require list changes (see UPGRADE.idx). However, backwards compatibility with ezmlm-0.53 list will be maintained. Thus, this is an issue only if you are already using an older version of ezmlm-idx.

Addition of "z" are bug fixes only. Thus, ezmlm-idx-0.301 is ezmlm-idx-0.30 with known bugs fixed (but no other significant changes). When available, patches are named "filename-0.xy[z].diff", where "0.xy[z]" corresponds to the release to which they apply. When a number of bugs (or a significant bug) are found a bug-fix release is made incorporating all the patches for the previous version.

To get the latest features, look for the highest number ("e.g. ezmlm-idx-0.32"). Any bugs in versions with new features are expected to be limited to the new features.

To get the most solid version, get the highest 3-digit number, i.e. a bug fix. If you already run a version in that series and a new bug fix is released, see CHANGES.idx to determine if it is worthwhile to upgrade. Most bugs so far have been relevant only when using lists in very unusual ways or with rarely used options.

- `<ftp://ftp.ezmlm.org/pub/patches/>`
- `<ftp://gd.tuwien.ac.at/infosys/mail/qmail/ezmlm-patches/>` ftp mirror in Austria.
- `<http://gd.tuwien.ac.at/infosys/mail/qmail/ezmlm-patches/>` http access to the same mirror.
- `<ftp://ftp.win.or.jp/pub/network/mail/qmail/ezmlm-idx/>` ftp mirror in Japan.

ezmlmrc(5) files for different languages

The latest versions at the time of release of a package are included in that package. Thus, this directory will have a file labeled with the current ezmlm-idx version number only if it has been updated later than the package. ezmlmrc(5) files are updated and new ones are added all the time, also with bug fix releases. Therefore, always look at the latest package.

ezmlmrc(5) files contain some release-specific configurations. Do not use a later file (other than from bug fix releases) with an earlier version of the programs. It is usually OK to use a version from an earlier package (see UPGRADE.idx), but some new functionality may nor be available.

To contribute an ezmlmrc(5) file in a new language, start with the en_US version from the latest package, and send the gzipped file to `lindberg@id.wustl.edu`. Please leave comments intact and in English and do not change the order of items in the file. This will facilitate maintenance.

- `<ftp://ftp.ezmlm.org/pub/patches/ezmlmrc/>`
- `<ftp://gd.tuwien.ac.at/infosys/mail/qmail/ezmlm-patches/ezmlmrc/>`
- `<http://gd.tuwien.ac.at/infosys/mail/qmail/ezmlm-patches/ezmlmrc/>`
- `<ftp://ftp.win.or.jp/pub/network/mail/qmail/ezmlm-idx/ezmlmrc/>`

ezmlm-issub-0.05

- `<ftp://ftp.ezmlm.org/pub/patches/ezmlm-issub-0.05.tar.gz>`. Use ezmlm-issub only if you do not use ezmlm-idx. The same functionality is available in ezmlm-idx and the packages are not compatible.
- Also via mirrors mentioned above.

RPMs and SRPMS of qmail, ezmlm and ezmlm-idx

- `<ftp://ftp.ezmlm.org/pub/patches/>`
- `<ftp://summersoft.fay.ar.us/pub/qmail/>`

1.6 Where can I find documentation for ezmlm and patches?

man pages

All ezmlm component programs come with their own man pages. Thus, for info on *ezmlm-send*, type:

```
% man ezmlm-send
```

or if you have unpacked ezmlm, but not made it or installed it:

```
% cd ezmlm-0.53
% man ./ezmlm-send.1
```

ezmlm(5)

General info on ezmlm and list directories is in **ezmlm.5**:

```
% man ezmlm
```

or

```
% cd ezmlm-0.53
% man ./ezmlm.5
```

NOTE: Installation of the ezmlm-idx package updates some existing man pages to reflect changes made by the patch (e.g. ezmlm-send(1), ezmlm(5)).

Text files in the distribution

ezmlm comes with a **README** file with general instructions, an **INSTALL** file with installation instructions, an **UPGRADE** file for upgrading from a previous version and a **CHANGES** file with information on changes from previous versions. ezmlm-idx comes with similar files suffixed with "**.idx**". Most other patches or add-ons contain similar files and man pages and should contain identifying suffixes (.iss for ezmlm-issub, for example). For a discussion of the authors' understanding of ezmlm security, see 21 (Ezmlm-idx security).

"Ezman", an ezmlm/idx manual

The ezmlm manual is a brief manual that is meant for list subscribers, list moderators and remote administrators, and as an introduction for list owners. It is useful even if you do not use ezmlm-idx. Features requiring ezmlm-idx are marked as such. The manual is available as a set of html files, as a text file, and in a "letter" and "A4" postscript version:

- *ezman for download* <ftp://ftp.ezmlm.org/pub/patches/ezman/>
- *An on-line html version* <http://www.ezmlm.org/ezman>

This FAQ

This FAQ is built from a sgml source. It is available in the following formats:

- *A text file* <ftp://ftp.ezmlm.org/pub/patches/ezfaq.txt.gz>
- *An on-line html version* <http://www.ezmlm.org/>
- *Html for download* <ftp://ftp.ezmlm.org/pub/patches/ezfaq.html.tar.gz>
- *A postscript (letter) version* <ftp://ftp.ezmlm.org/pub/patches/ezfaq.ps.gz>
- *A postscript (A4) version* <ftp://ftp.ezmlm.org/pub/patches/ezfaq.ps4.gz>
- Via mirrors mentioned for the ezmlm-idx package.

- An up-to-date text version,**FAQ.idx**, included with the ezmlm-idx package.

WWW resources

An on-line version of this FAQ

`<http://www.ezmlm.org/>`The main site with an up-to-date mirror list. `<http://www.de.ezmlm.org/>`German mirror. `<http://www.pl.ezmlm.org/www.ezmlm.org/>`Polish mirror. `<http://www.jp.ezmlm.org/>`Japanese mirror. `<http://www.pt.ezmlm.org/>`Portuguese mirror. `<http://www.at.ezmlm.org/>`Austrian mirror. `<http://www.ca.ezmlm.org/ezmlm/>`Canadian mirror.

General qmail and ezmlm info

- *Dan J. Bernstein's qmail page* `<http://www.pobox.com/~djb/qmail.html>`
- *Dan J. Bernstein's ezmlm page* `<http://www.pobox.com/~djb/ezmlm.html>`
- *Russell Nelson's qmail page* `<http://www.qmail.org>`
- *Mirrors of www.qmail.org* `<http://www.ISO.qmail.org>`. Substitute your two-letter country abbreviation for "ISO".

The qmail mailing list archive

- `<http://www.ornl.gov/cts/archives/mailing-lists/qmail/>`

The ezmlm mailing list archive

- `<http://sunsite.auc.dk/mhonarc-archives/ezmlm/>` `<http://www.ezmlm.org/archive/>` This archive of the ezmlm list is searchable from 11/97-present.

Mailing lists

Please read other documentation and mailing list archives before posting questions to the lists. It's also useful to "lurk" on the list for a few days, (i.e. to subscribe and read without posting) before asking your questions on the list.

To subscribe, send mail to the E-mail addresses listed:

- Dan Bernstein's ezmlm list: *ezmlm-subscribe@list.cr.yp.to*
- A digest version of the ezmlm list *fredr-ezmlm-digest-subscribe@rivertown.net*
- Dan Bernstein's qmail list: *qmail-subscribe@list.cr.yp.to*
- The Japanese ezmlm list: *ezmlm-subscribe@jp.qmail.org*
- The Japanese qmail list: *qmail-subscribe@jp.qmail.org*
- A ezmlm/idx digest list of djb-qmail: *qmail-digest-subscribe@id.wustl.edu*
- A ezmlm/idx sublist of djb-qmail (you can test ezmlm-idx commands): *qmail-index@id.wustl.edu*

1.7 Where do I send comments on this document?

To the authors via E-mail:

- Fred Lindberg, *lindberg@id.wustl.edu*
- Fred B. Ringel, *fredr@rivertown.net*

1.8 How to experiment with new versions of ezmlm-idx.

ezmlm-idx>=0.23 writes **DIR/config** in a standard format. If ezmlm-make(1) is invoked with the "-e" or "-+" switch and the "DIR" argument only, ezmlm-make(1) will read other arguments from this file. The difference between the switches is that with "-e" the options used are the ones specified on the command line, whereas with "-+" they are the ones currently active for the list, as overridden by any command line options. Thus, with just:

```
% ezmlm-make -+ DIR
```

you can rebuild the list, without affecting any archives, list state variables, etc. You will *lose manual customizations to some of your **DIR/text/** files.* **DIR/headeradd**, **DIR/text/info**, **DIR/text/faq**, **DIR/text/mod-sub**, and **DIR/text/sub-ok** are protected against being overwritten, so that your manual customizations of these files are retained.

To test a new version of ezmlm-idx or to run several version, make the new version as per **INSTALL.idx** (if you haven't used ezmlm-idx before) or **UPGRADE.idx** (if you've got a previous version of ezmlm-idx installed), setting **conf-bin** to a new directory. You can use either the current directory or any other directory. If not using the current dir, you also have to:

```
% make setup
```

If you now edit the list using the new ezmlm-make program, the list will automatically be configured to use the new binaries. To change back to the "default" installation, just edit the list again, this time with the old ezmlm-make(1).

If your system has an **/etc/ezmlmrc** file, you may need to temporarily place the **ezmlmrc(5)** file for the ezmlm version you want to test in **dotdir** of the list and use the ezmlm-make(1) "-c" switch (see 1.3 (Terminology: dotdir)).

ezmlm-idx>=0.314 comes with ezmlm-test(1), a program that tests most functions of ezmlm+idx and can be used before installation.

2 Quick start

1. Create a use "eztest" for testing. If you use another name, add the switch "-u another_name" to the ezmlm-test(1) line below. (The space between the switch and the argument is required.)
2. Unpack the ezmlm-0.53 distribution.
3. Unpack the ezmlm-idx distribution.
4. Move the ezmlm-idx files to the ezmlm-0.53 directory.
5. Edit **conf-bin** and **conf-man** to reflect the target directories.
6. build and install:

```
% cd ezmlm-0.53
% patch < idx.patch
% make; make man
```

```
% su
# su eztest
% ./ezmlm-test
% exit
# make setup
# exit
```

7. Make a list and digest list

```
% ezmlm-make -rdugm -5 me@host ~/list ~/.qmail-list me-list host
% ezmlm-sub ~/list me@host
% ezmlm-sub ~/list/digest me@host
% ezmlm-sub ~/list/mod me@host
```

   where "me" is your user name and "host" the host your list is on.

Now, you are the owner, remote administrator, and subscriber of both `list@host` and the accompanying digest list `list-digest@host`. Only subscribers are allowed to access the archive and to post. To post to the list, mail to `list@host`. For a user to subscribe, s/he should mail to `list-subscribe@host` and for help to `list-help@host`.

When a non-subscriber posts, you will be asked to approve, reject, or ignore the request. If you want to subscriber `joe@joehost.dom`, mail `list-subscribe-joe=joehost.dom@host`.

Digests are generated about every two days, when 30 messages have arrived since the last digest, or when more than 64 kbytes of message body has arrived. To manage the digest list, use the same commands as the main list, but replace "list" with "list-digest".

The sender restriction on posting used in this setup works, but is not secure. For more info, read the man pages (start with ezmlm(5) and ezmlm-make(1)), this FAQ (**FAQ.idx** in the distribution), **README**/**README.idx**, **INSTALL**/**INSTALL.idx**, and **UPGRADE.idx**.

3 Overview of mailing list management and mailing list managers

(To be written. Until then, please consult the `<http://www.ezmlm.org/ezman/>` manual for ezmlm and ezmlm-idx related material.)

4 Overview of ezmlm function

4.1 The basic setup.

In designing ezmlm, *Dan J. Bernstein* has used the unix philosophy of small component programs with limited and well defined functions. Requests for specific functions can then be met by the addition of new programs.

Thanks to the program execution mechanism Dan built into qmail, it is easy to execute several small programs per delivery in a defined sequence. It is also very easy to add shell scripts for further customization.

4.2 Inventions in ezmlm.

Dan J. Bernstein has written ezmlm in C. It is written for speed and reliability even in the face of power loss and NFS. These features are augmented to a large extent by the ruggedness of the qmail (also by Dan) delivery mechanism (see qmail-command(8)).

ezmlm uses some routines and techniques that still are not frequently seen in many mailing list managers. For example, subscriber E-mail addresses are stored in a hash so that searches require reading only, at most, 2% of the E-mail addresses. ezmlm has a optional message archive, where messages are stored 100 per directory, again to allow more efficient storage and retrieval. Important files are written under a new name and, only when safely written, moved in place, to assure that crashes do not leave the list in an undefined state.

In addition, ezmlm has a number of new inventions. One of these is bounce detection, which generates an automatic warning containing information identifying the messages which have bounced, followed by a probe message to the E-mail addresses for which mail has bounced. If the probe bounces, the address is unsubscribed. Thus, the system won't remove E-mail addresses due to temporary bounces: it takes 12 days after the first bounce before a warning is sent, and another 12 days of bounces after the warning bounce before the probe message is set.

Another Dan J. Bernstein invention is the use of cryptographic cookies based on a timestamp, address, and action. These are used to assure that the user sending a request to subscribe or unsubscribe really controls the target address. It is also used to prevent forgery of warning or probe messages to make it exceedingly difficult to subvert the bounce detection mechanism to unsubscribe another user.

4.3 The qmail delivery mechanism.

See qmail(7), qmail-local(8), qmail-command(8), envelopes(5), and dot-qmail(5). Briefly, qmail having resolved the delivery address delivers it via the **.qmail** file that most completely matches the address. This file may be a link to another file, as is the case in ezmlm lists. qmail then delivers the message according to successive lines in this file forwarding it to an address, storing it, or piping it to a program. In the latter case, the program is expected to exit 0 leading delivery to proceed to the next line in the **.qmail** file, or 99 leading to success without delivery to succeeding lines. An exit code of 100 is a permanent error leading to an error message to the SENDER. An exit code of 111 is used for temporary errors, leading to re-delivery until successful or until the queue lifetime of the message has been exceeded.

Delivery granularity is the **.qmail** file and re-deliveries start at the top. Thus, if the message fails temporarily at a later line, the delivery according to an earlier line will be repeated. Similarly, qmail may have made deliveries successfully according to most of the **.qmail** file and then fail permanently. The SENDER is informed that the delivery failed, but not about at which point.

ezmlm takes advantage of these basic mechanisms to build a fast, efficient, and very configurable mailing list manager from a set of small independent programs.

4.4 What the different programs do.

See ezmlm(5) and the man pages for the different programs (listed in ezmlm(5)).

4.5 What the different files in the list directory do.

See ezmlm(5).

4.6 The paper path for posts.

Messages to the list are delivered to a **.qmail** file, usually **~/.qmail-listname** which is linked to **DIR/editor**. Here, the message is first delivered to ezmlm-reject(1) which can reject messages based on subject line contents, MIME content-type, and message body length. It also by default rejects all messages that do not have the list address in the "To:" or "Cc:" header. This eliminates most bulk spam. If the list is set up for restrictions based on envelope SENDER, the next delivery is to one or more instances of ezmlm-issubn(1). If the messages passed this check, it is usually delivered to ezmlm-send(1) for distribution. If the list is message moderated, it is instead delivered to ezmlm-store(1) which queues the message and sends out a moderation request. ezmlm-gate(1) is used by some other setups. It will for message moderated lists invoke ezmlm-send(1) directly if the message is from a specific set of SENDERs, and in other cases ezmlm-store(1) to send the message out for moderation.

If the list is configured for digests, **DIR/editor** also contains an ezmlm-tstdig(1) line followed by an ezmlm-get(1) line. If ezmlm-tstdig(1) determines that the criteria are met for digest generation, it exits with an exit code of 0, causing the ezmlm-get(1) line to be executed leading to a digest mailing. Otherwise, ezmlm-tstdig(1) exits 99, resulting in the remainder of the **DIR/editor** file to be ignored too long. The digest is not related to the message being delivered, but the delivery is used to trigger execution of the relevant programs.

In addition, **DIR/editor** contains a number of house-keeping functions. These are invocations of ezmlm-warn(1) to send out bounce warnings and and (if the list is moderated) ezmlm-clean(1) to clean the moderation queue of messages that have been ignored. Again, these functions are not related to the specific message delivered, but the delivery itself is used as a convenient "trigger" for processing.

4.7 The ezmlm path for moderation messages.

Replies to moderation requests are channeled to **DIR/moderator**. This file contains an invocation of ezmlm-moderate(1) which invokes ezmlm-send(1) for accepted messages and sends out a rejection notice for rejected messages. It also sends error messages if the message is not found or already accepted/rejected *contrary* to the moderation message. Thus, if you accept a message already accepted, no error message is sent. ezmlm-clean(1) is also invoked from **DIR/moderator** for house keeping.

4.8 The ezmlm path for administrative messages.

Administrative requests for both list and digest lists are captured by **~/.qmail-listname-default** linked to **DIR/manager**. Here they are delivered first to ezmlm-get(1) which processed archive retrieval requests, exiting 99 after successful completion which causes the rest of the delivery lines to be ignored. If the request is not for ezmlm-get(1) it rapidly exits 0. This leads to invocation of ezmlm-manage(1) which handles subscriber database functions, help messages, and (if configured) editing of **DIR/text/** files. Again, ezmlm-warn(1) lines are included for bounce directory processing.

If configured, an ezmlm-request(1) line is present. This program constructs valid ezmlm requests from command in the subject lines of messages sent to `listname-request@host` and exits 99. These requests are mailed and will then return to be processed by one of the other programs.

4.9 The ezmlm path for bounces.

Bounces to the list are handled by **DIR/bouncer**. For the digest list this is **DIR/digest/bouncer**. The two were combined in previous versions, which is still supported. As this leads to problems with list names ending in "digest", the functions are separate with lists set up or edited with ezmlm-idx>=0.32. The bounce is first delivery is to ezmlm-weed(1) which removes delivery delay notification and other junk. The second to ezmlm-return(1) which analyzes valid bounces storing the information in **DIR/bounce/** for the list and **DIR/digest/bounce/** for the digest. This is the information that ezmlm-warn(1) (invoked from **DIR/editor** and **DIR/manager**) uses and processes for automatic bounce handling. ezmlm-return(1) will also unsubscribe a subscriber from whom a probe message has bounced.

4.10 Messages to list-owner and list-digest-owner.

These are processed by **DIR/owner** and delivered to **DIR/mailbox** by default. It is better to put the real owner address in this location. This can be done manually, via editing of **ezmlmrc(5)**, or via the ezmlm-make(1) -5 switch. Again, some house-keeping functions are also executed.

4.11 Structure of subscriber databases.

ezmlm subscriber E-mail addresses are stored within **DIR/subscribers/** as a hashed set of 53 files. The hash calculated from the address determines which of the 53 files and address is stored in. Thus, to find out if an address is a subscriber, ezmlm has to read at most about 2% of the E-mail addresses. The hash function insures that E-mail addresses are reasonably evenly distributed among the 53 files.

Addresses in the files in **DIR/subscribers/** are stored as strings starting with "T", followed by the address, followed by a zero byte. This is the same format as taken by qmail-queue(8) on file descriptor 1. Thus, subscriber lists can be directly copied to qmail without any further processing.

With ezmlm-idx>=0.32 you can use an SQL server for the subscriber databases. Please see the SQL section (5 (ezmlm support for SQL datbases)).

4.12 Local case in E-mail addresses.

rfc822 states that the host part of an address is case insensitive, but that case of the local part should be respected and the interpretation of it is the prerogative of the machine where the mailbox exists. Thus, ezmlm preserves the case of the local part, but converts the host part to lower case. ezmlm proper also bases the hash on the case of the local part, so that `USER@host` and `user@host` are not (usually) stored in the same file.

Locally, deliveries are most often case insensitive, i.e. mail to `USER@host` and `user@host` are delivered to the same mail box. A consequence of this is that many users use E-mail addresses with different case interchangeably. The problem is that when `USER@host` is subscribed, ezmlm will not find that address in response to an unsubscribe request from `user@host`. This is even more problematic when E-mail addresses have been added by hand to e.g. moderator lists.

ezmlm-idx>=0.22 changes address storage to make comparisons case insensitive and store E-mail addresses based on the hash of the all lower case address. Case is maintained for the local part. Thus, if `USER@host` is subscribed, mail is set to `USER@host`, but `user@host` is recognized as a subscriber and an unsubscribe request from `user@host` will remove `USER@host` from the subscriber list.

To maintain backwards compatibility with old subscriber lists, a second lookup is made for partially upper case E-mail addresses in some cases. This will find `USER@host` subscribed with a case sensitive hash as well.

If may be useful to move all old mixed case E-mail addresses to the "new" positions. Without this, `USER@host` subscribed with the old system will be able to unsubscribe as `USER@host`, but not as `user@host`. After the repositioning, s/he will be successfully able to use any case in an unsubscribe request, e.g. `UsEr@host`. To do this:

```
% ezmlm-list DIR | grep -G '[A-Z]' > tmp.tmp
% xargs ezmlm-sub DIR < tmp.tmp
```

This works, because subscribing an address, even if it already exists, will assure that it is stored with a case insensitive hash. On some systems, the grep "-G" switch need/should not be used.

4.13 Testing SENDER to allow posts only from list subscribers.

This mode of operation is automatically set up if you specify the ezmlm-make(1) "-u" switch. Since there may be some addresses that should be allowed to post, but are not subscribers of list or list-digest, ezmlm-make(1) sets up an additional address database in **DIR/allow/**. Use ezmlm-sub(1), ezmlm-unsub(1), and ezmlm-list(1) to manipulate these addresses. If the list is configured for remote administration (see 4.17 (How remote administration works)), you can add/remove addresses from the **DIR/allow/** database by mailing `list-allow-subscribe@listhost` and `list-allow-unsubscribe@listhost`, respectively. Other commands that access subscriber databases work in the same manner.

To similarly restrict archive access, use the ezmlm-make(1) "-g" switch.

Since SENDER is under the control of a potential attacker, it is not secure to use tests of SENDER for anything important. However, when replies are always sent to SENDER (such as for archive access), a check of SENDER can prevent the sending of information to E-mail addresses not in the database.

To test sender, use the program ezmlm-issubn(1). It will return 0 (true for the shell, success for qmail deliveries) if SENDER is in at least one of a set of subscriber databases. If not, it will return 99 (false for the shell: success, but skip remainder of **.qmail** file for qmail deliveries). The basedirs of the subscriber lists (i.e. the directories in which the "subscriber" dirs are located) are given as arguments. ezmlm-issubn(1) can take any number of arguments.

Thus, to permit an action if SENDER is a subscriber to the list in any of **DIR/**, **DIR/digest/**, or **DIR/allow/** and exit silently, put the following into the relevant **.qmail** file:

```
|/usr/local/bin/ezmlm/ezmlm-issubn DIR DIR/digest DIR/allow [...]
|/path/action_program
```

Restricting your list to posts from your subscribers is as easy as that. If your ezmlm binaries are in a different directory, you may have to modify the ezmlm-issubn(1) path.

ezmlm-issubn(1) has a "-n" switch which "negates/reverses" the exit code. To do an action if SENDER is *NOT* a subscriber of any of the lists:

```
|/usr/local/bin/ezmlm-ezmlm-issubn -n DIR/deny [dir2 ...]
|/path/other_program
```

To automatically configure the list with a blacklist address database in **DIR/deny**, use the ezmlm-make(1) "-k" switch. If the list is configured for remote administration (see 4.17 (How remote administration works)) and if you are a remote administrator, you can manipulate the "deny" database remotely by sending mail to `list-deny-subscribe-user=userhost@listhost`, etc.

4.14 How cookies work.

Each ezmlm list has it's own "key" created by ezmlm-make at setup time. This key is stored in **DIR/key**, and you can improve it by adding garbage of your own to it. However, changing the key will make all outstanding cookies invalid, so this should be done when the list is established.

When ezmlm receives an action request, such as "subscribe", it constructs a cookie as a function of:

- the request,
- the time,
- and the target address.

The cookie and these items are then assembled into a address that is sent out as the "Reply-To:" address in the confirmation request sent to the subscriber. When the subscriber replies, ezmlm first checks if the timestamp is more than 1,000,000 seconds old (approx 11.6 days) and rejects the request if it is. Next, ezmlm recalculates the cookie from the items. If the cookies match, the request is valid and will be completed. Depending on the circumstances, ezmlm generates an error message or a new cookie based on the current time and sends the target a new confirmation request.

Dan has based these cookies on cryptographic functions that make it very unlikely that a change in any part of the cookie or the items will result in a valid combination. Thus, it is virtually impossible to forge a request even for someone who has a number of valid requests to analyze. Since the algorithm ezmlm uses is available, the security rests on the key (and the correctness of the algorithm). Anyone who knows the key for your lists can easily construct valid requests.

As ezmlm-make(1) doesn't use a truly random process to generate the key, it is theoretically possible that someone with sufficient knowledge about your system can guess your key. In practice, this is very unlikely, and the safety of the system is orders of magnitude higher than that of other mechanisms that you may rely on in your list management and mail transport (exclusive of strong encryption, such as *PGP*).

4.15 How moderator E-mail addresses are stored.

Moderator E-mail addresses are stored just like ezmlm subscriber addresses, in a set of up to 53 files within the **subscribers** subdirectory of the list's **basedir/**. For subscribers, the **basedir/** is the list directory itself, i.e. **DIR/**. For moderators, the default is **DIR/mod/**, which can be overridden by placing a **basedir** name (starting with a "/") in **DIR/modsub**, **DIR/remote**, or **DIR/modpost** for subscription moderation, remote administration, and message moderation, respectively. This permits the use of one moderator database for multiple lists. *Note: Subscription moderators and remote administrators are always the same addresses. If both **DIR/modsub** and **DIR/remote** contain paths, only the **DIR/modsub** path is used.*

4.16 How subscription moderation works.

Subscription moderation is a simple extension of the ezmlm subscribe mechanism. Once the user has confirmed the subscribe request, a new request is constructed with a *different action code*. This is sent out to the moderator(s). When a moderator replies with a valid request and cookie combination, the user is subscribed. The user is then also welcomed to the list. Other moderators won't know that the request has already been approved. If other moderators reply to the request, no notification of the duplicate action is sent to the subscriber of the duplicate action. Ezmlm knows that this is a repeat request since the target address is already a subscriber.

The moderators are not informed about the result, unless there was an error (subscribing a target that is already a subscriber is not considered an error). This cuts down the number of messages a moderator receives. Any list moderator knows (or *should* know) the qmail/ezmlm/unix paradigm: *if you're not told otherwise, your command was carried out successfully*. This may be counterintuitive to those used to some other operating systems, but in our experience it doesn't take long to get used to the reliability and efficiency of U*ix/qmail/ezmlm.

Subscription moderation is enabled by creating **DIR/modsub** and adding the subscription moderator to **DIR/mod/**:

```
% ezmlm-sub DIR/mod moderator@host
```

To use an alternative basedir for subscription moderators, place that directory name with a leading "/" in **DIR/modsub**.

4.17 How remote administration works.

The term "remote administration" is used to denote the ability of a list administrator by E-mail to add or remove any E-mail address from the subscriber list without the cooperation of the user. Normally, when `user@userhost` sends a message to `list-subscribe-other=otherhost@listhost` to subscribe `other@otherhost`, the confirmation request goes to `other@otherhost`. However, if remote administration is enabled and `user@userhost` is a moderator, a confirmation request (with a different action code) is sent back to `user@userhost` instead. The reply from the administrator is suppressed in the welcome message sent to the new subscriber (`other@otherhost`). This protects the identity of the remote administrator.

Remote administration is enabled by creating **DIR/remote** and adding the remote administrator E-mail address(es) to **DIR/mod/**:

```
% ezmlm-sub DIR/mod remoteadm@host
```

To use an alternative basedir for remote administrators, place that directory name with a leading "/" in **DIR/modsub**. Remote administrators and subscription moderators databases always consist of the same E-mail addresses. If both are enabled and one of **DIR/modsub** and **DIR/remote** contains an alternative basedir name, this basedir is used for both functions. If both **DIR/modsub** and **DIR/remote** contain directory names, the one in **DIR/modsub** is used for both functions.

Remote administrators can add and remove addresses to the digest list, the "allow" list (user aliases for lists using SENDER restrictions on posting and archive access), and if used the "deny" list containing addresses that are denied posting rights to the list. The latter is easy to circumvent and intended to block errant mail robots, rather than human users.

4.18 How message moderation works.

ezmlm-store(1), invoked in **DIR/editor**, receives messages for message moderated lists. If **DIR/modpost** does not exist, ezmlm-store(1) just calls ezmlm-send(1) and the message is posted to the list as if it were not moderated. If **DIR/modpost** exists, ezmlm-store(1) places the message in **DIR/mod/pending/**. It also sends a moderation request to all the moderators. Included with this request is a copy of the message. The "From:" and "Reply-To:" E-mail addresses contain codes for "reject" and "accept", together with a unique message name (derived from the message timestamp and process id) and a cookie based on these items. When a moderator replies, ezmlm-moderate(1) is invoked via **DIR/moderator**. ezmlm-moderate(1) validates the request, and if the request is valid and the message is found in **DIR/mod/pending/**, it carries out the requested action.

If the request is "reject" the post is returned to SENDER with an explanation and an optional moderator comment. If the request is "accept" the message is posted to the list via ezmlm-send(1). As the request is processed, a stub for the message is created in **DIR/mod/rejected/** or **DIR/mod/accepted/** for "reject" and "accept" requests, respectively.

If a valid reply is received but the message is no longer in **DIR/mod/pending/**, ezmlm-moderate(1) looks for the corresponding stub in **DIR/mod/rejected/** and **DIR/mod/accepted/**. If the stub is found and the fate of the message was the one dictated by the new request, no further action is taken. If, however, no stub is found or the request and the actual message fate do not match, a notification is sent to the moderator. This scheme was chosen to impart a maximum of information with a minimum of messages. Also, it is the least demoralizing setup for multiple moderator lists, where it is important not to notify subsequent moderators that their work was in vain since the action of the first responding moderator has already resulted in processing of the message.

If a message is not "rejected" or "accepted" it remains in **DIR/mod/pending/** until it times out. Cleanup of both messages and stubs is accomplished by ezmlm-clean(1) which is invoked through both **DIR/editor** and **DIR/moderator** for message moderated lists. ezmlm-clean(1) looks at the timestamp used to generate the message/stub name. If it is older than 120 hours (configurable in a range of 24-240 hours, by placing the value in **DIR/modtime**) it is removed. Unless suppressed with the ezmlm-clean(1) "-R" switch, the SENDER of the message is notified.

By default, the E-mail addresses of message moderators are stored as a subscriber list with a basedir of **DIR/mod/**. This can be changed to any other **basedir** by placing the name of that directory with a leading "/" in **DIR/modpost**. Although the default basedirs for message moderation and subscription moderation/remote administration are the same, both the functions and actors are entirely independent.

4.19 How messages are stored in the archive.

The structure of the ezmlm list archive is described in the ezmlm(5) manual page. Basically, the message is stored in **DIR/archive/n/m**, where "n" is the message number divided by 100 and "m" the remainder (2 digits). The first message is stored in **DIR/archive/0/01**.

4.20 How the message index works.

The ezmlm-idx(1) adds the option (default) of a message index to ezmlm. The "From:" line, the subject, the author's E-mail address and name and the time of receipt are logged for each message as it is received. The subject is "normalized" by concatenating split lines and removing reply-indicators such as "Re:". A hash of the normalized subject with all white space removed is also stored. The hash for any message within a thread is almost always the same and is used together with the order of receipt to connect a set of messages into a "thread". A hash is needed due to the inconsistent handling by MUAs of white space in rfc2047-encoded subject headers.

The message index is stored as **DIR/archive/n/index**, where "n" is the message number mod 100. Thus, the directory **DIR/archive/52/** stores messages 5200 through 5299 and the file "index" which contains the index for those messages.

The message index can be retrieved with the -index command (see ezmlm-get(1)). You can also retrieve a range of messages, a specific thread, or generate a message digest (see ezmlm-get(1)). Each of these commands can be disabled or restricted as desired by the list owner.

The ezmlm-idx(1) can be used at any time to either reconstruct an existing index or create one an index for an existing message archive. without one.

4.21 How threading works.

A ezmlm thread is just a message number-ordered set of messages with identical "normalized" subject entries. This is a very reliable method for threading messages. It does not rely on any variably present "In-Reply-To:" or "References:" headers. If the subject changes, the continuation becomes a separate thread very close to the original thread in a digest. ezmlm uses this mechanism to return message sets threaded and with a thread and author index, unless specifically told not to do so with the "n" format specifier. Naturally, lists set up without a message index (using the ezmlm-make "-I" switch) do not maintain thread information.

4.22 How digests work.

A "digest" is just an ordered collection of messages from a list, usually sent out regularly depending on the time and traffic volume since the last digest. Digest subscribers thus can read messages as "threads" once daily, rather than receiving a constant trickle of messages.

As a major change in ezmlm-idx-0.30, the digest is no longer a totally separate ezmlm-list, but a part of the main list. This has security advantages, makes setup and administration easier, saves space, and allows a consistent way for subscribers of both "list" and "list-digest" to retrieve missed messages from a single archive.

The digest of the list "list" is always called "list-digest". To set up a list with a digest, simply use the ezmlm-make(1) "-d" switch. You subscribe to and unsubscribe from a digest the same way as for the main list, except that the request is sent to e.g. `list-digest-subscribe@host` rather than to `list-subscribe@host`.

Any option such as remote admin or subscription moderation that is active for the list applies also to the digest list. Any restrictions in posts or archive retrieval set up for the list, automatically accept both subscribers of the main list and of the digest list.

The changes in ezmlm-idx>=0.30 allow all programs to service both list and list-digest functions. All digest-specific files are stored in **DIR/digest/**. Digest list subscriber addresses in **DIR/digest/subscribers/** and digest list bounce information in **DIR/digest/bounce/**. Text files are shared between list and digest. To get the local part of the list or list-digest name in a context sensitive manner, use "<#l#>" (lower case "L") in the text file.

In order to generate digest, the list needs to be archived and indexed (both default). You can retrieve sets of messages from the message archive. Such sets are always returned to the SENDER of the request. "Digests" are a special form of such a set/request. First, there are no restrictions on the number of messages that can be in a digest (which is balanced by the requirement for a "digest code" that needs to be specified in order to create a digest based on a mailed request). Second, special files (**DIR/digissue** and **DIR/dignum**) keep track of the digest issue and the message number, amount, and time when the last digest was created. Thus, the system is adapted to make it easy to create the regular collections of messages commonly referred to as "digests".

Digest can be generated in several different ways:

Command line

ezmlm-get can be invoked on the command line, or via a script from e.g. crond(8):

```
% ezmlm-get DIR
```

If for some reason the digest should be disseminated via a separate list, the digest can be redirected to a "target address" with the ezmlm-get(1) "-t" switch. This may be useful if a non-standard digest list name is required. In this case, the list disseminating the digest must be set up as a sublist of the main list (see 4.24 (How sublists work)).

from DIR/editor

This is the default and does not require and additional setup. It works well with most lists. The only possible advantage is for very low traffic lists and for lists where it is important that a digest be sent out at a specific time (as **DIR/editor** digests are triggered only when messages are received).

In **DIR/editor**, ezmlm-get(1) needs to be combined with ezmlm-tstdig(1) so that digests are generated only if certain criteria are met (in this case, more than 30 messages, 64 kbytes of message body or 48 hours since the latest digest). Add these lines after the ezmlm-send line in **DIR/editor**:

```
|/usr/local/bin/ezmlm/ezmlm-tstdig -t48 -m30 -k64 DIR || exit 99
|/usr/local/bin/ezmlm/ezmlm-get diglist@host DIR || exit 0
```

To set this up automatically when you create the list:

```
% ezmlm-make -d DIR dot local host [code]
```

Again, the ezmlm-get(1) "-t" switch can be used for non-standard arrangements to redirect the digest. The ezmlm-make(1) "-4" switch can be used to specify alternative ezmlm-tstdig(1) parameters.

from DIR/manager

This is useful only if you want digests at specific times, and you do not have access to crond(8) on the list host. ezmlm-get(1) is in it's normal place in **DIR/manager** before ezmlm-manage(1), but a digest code is specified in the ezmlm-get(1) command line. To trigger digests requires a regular trigger messages generated from e.g. crond(8) (see below), but this can be done from _any_ host, not only the list host. ezmlm-make(1) sets up ezmlm-get(1) this way if a digest "code" is given as the 5th ezmlm-make(1) command line argument. However, you need to set up the trigger messages separately (see below):

```
% ezmlm-make DIR dot local host code
```

To also test for message volume with this setup, generate trigger messages with the granularity you'd like, and add a ezmlm-tstdig(1) line to **DIR/manager**. E.g., use a trigger message every 3 hours and the following ezmlm-tstdig(1) line before ezmlm-get(1):

```
|/usr/local/bin/ezmlm/ezmlm-tstdig -t24 -m30 -k64 DIR || exit 99
```

In general, a cron-triggered digest is preferred for very large lists and for lists with very low traffic. Again, the ezmlm-get(1) "-t" switch can be used for non-standard arrangements to redirect the digest. For most lists, the digesting from **DIR/editor** works very well, and does not require any extra setup work.

Combination setups

The default setup in the ezmlmrc(5) file included in the distribution is the **DIR/editor** triggered setup described above. If you in addition use ezmlm-cron(1) or crond(8) directly to generate trigger messages to `list-dig.code@host`, you can get regular digests (via the trigger messages and **DIR/manager**), with extra digest sent when traffic is unusually high (via the ezmlm-tstdig/ezmlm-get limits set in **DIR/editor**). This works best when the time argument on the ezmlm-tstdig(1) command line is the same as the trigger message interval, and the other ezmlm-tstdig(1) parameters are set so that they are only rarely exceeded within the normal digest interval.

4.23 How ezmlm-tstdig works.

ezmlm-tstdig(1) looks at **DIR/num** and **DIR/dignum** to determine how many messages and how much traffic (in terms of bytes of message body) has arrived to the list since the latest digest. It also determines how much time has passed since the last digest was generated. If any of the criteria specified by command line switches exists, ezmlm-tstdig(1) exits 0, causing the invocation of the next line in the .qmail file. If not, ezmlm-tstdig(1) exits 99 causing qmail to skip the rest of the .qmail file. ezmlm-tstdig(1) looks at LOCAL to determine if it is invoked in the command line, in **DIR/editor**, or in **DIR/manager**. In the latter two cases, ezmlm-tstdig(1) verifies that the list local address is correct. If invoked in **DIR/manager**, ezmlm-tstdig(1) exits 0 for all action requests except `list-dig`, so that is does not interfere with the normal functions of ezmlm-get(1) and ezmlm-manage(1). ezmlm-tstdig(1) uses **DIR/tstdig** as a flag to avoid problems caused by starting the program when another copy is already running.

ezmlm-make(1) automatically configures ezmlm-tstdig(1) with the parameters "-t48 -m30 -k64", which can be overridden with the "-3" switch.

4.24 How sublists work.

ezmlm uses the concept of sublists. Sublists are regular ezmlm lists, except that they only accept messages from their parent list, which is placed in the file **DIR/sublist**.

sublists are used to split the load of a large mailing list among several hosts. All you need to do to set up a local sublist of e.g. the `qmail@list.cr.yp.to` list is to create a ezmlm list, and put "qmail@list.cr.yp.to" into **DIR/sublist** of you list, and subscribe the sublist to the main qmail list. Now anyone can subscribe to your local list which handles its own bounces, subscribe requests, etc. The load on the main list is only the single message to your local list.

Sublists will not add their own mailing list header and they will not add a subject prefix. Normally, sublists will use their own message number, rather than that used by the main list. With ezmlm-idx>=0.23, sublists that are not archived and not indexed, will instead use the main list message number. This way, bounce messages from the sublist can refer the subscriber to the main list archive. This is not done for indexed/archived sublists for security reasons (an attacker could overwrite messages in the sublist archive).

With ezmlm-idx>=0.31, there is support for using ezmlm as a sublist of a mailing list run by another mailing list manager. To set this up, set up a normal ezmlm sublist, then edit **DIR/editor** so that the *ezmlm-send* line contains the command line option "**-h** *X-Listprocessor-Version:*" (before **DIR**). As the header text, you need to use a header that the main list manager adds to messages. Now your sublist will accept only messages from the main list requiring that they come from that list *and* contain the header specified.

ezmlm-idx>=0.313 also has added protection against the malicious subscription of the ezmlm list to mailing lists run by other list managers. If the ezmlm-reject(1) line in **DIR/editor** has "-h" and "**DIR**" on it, ezmlm-reject(1) will read **DIR/headerreject** and reject messages that have any header specified in that file. See the ezmlm-reject(1) man page for suitable headers.

4.25 How sublisting can be made transparent to the user.

Often you create a local sublist of a list that you do not control. Local users know to subscribe to your local list. However, occasionally, you want to run your own list as a main list and a series of sublists per geographic site, or split onto several hosts if the list is too large to be handled by a single computer. You may also want to split the load of a "well known" list host that is getting overwhelmed with traffic. ezmlm supports sublists, but here the fact that the user has to interact with the correct sublist is a problem. What if the user doesn't remember which sublist s/he is subscribed to? What if you change the name of a sublist host or move a sublist to a different host?

ezmlm-idx&-0.32 adds ezmlm-split(1), which allows sublisting transparent to the user. This program is invoked before ezmlm-manage(1) in **DIR/manager**. If it detects a `subscribe` or `unsubscribe` command, it will forward the command to the appropriate sublist based on a "split file" **DIR/split**. This file contains entries, one per line, of the format:

```
domain:lo:hi:sublistname@sublisthost
edu:::othersub@otherhost
:1:26:third@thirdhost
```

For each address, a hash in the range 0-52 is calculated. The "domain" is the last two parts of the host name, reversed. Thus, for `id.wustl.edu` it would be "edu.wustl". The domain is considered to match if the characters in the split file match. It is advisable to use only the last part of the domain for compatibility with the SQL version version (see section 5 (ezmlm support for SQL datbases)).

Thus, any address `*@*.domain` with a hash between "lo" and "hi" inclusive would match the first line and be forwarded to `sublistname@sublisthost`. `*@*.edu` (independent of hash) would match the second line and be forwarded to `othersub@otherhost`. Of remaining requests, a request for any target address with a hash between 1 and 26 would be forwarded to the sublist `third@thirdhost`. Remaining requests would be passed on to the local list.

The domain is useful for "geographic" splitting, and the hash for load splitting (within a domain). The user interacts only with the main list, and does not need to know from which sublist s/he is serviced.

ezmlm-idx sublists use the message number of the main list message if they are not indexed. This allows sublists to in bounce messages refer the subscriber to the main list archive. Use ezmlm-make(1) in conjunction with ezmlmsubrc(5) to set up the sublists. See man pages for further details.

Since the addresses are stored locally, the system is very fast and robust, but it is difficult to add new sublists. ezmlm-split(1) -D supports parsing addresses on stdin and splitting them to stdout (see man page). Thus, if you divide the domain of some sublist(s) onto a net set of sublists, you can use ezmlm-list(1) to collect the addresses, ezmlm-split -D with the new split file to split them, then after clearing the local subscriber databases use ezmlm-sub(1) to add the correct addresses to each new sublist. The section on SQL support describes an alternative way of managing sublists (see section 5 (ezmlm support for SQL datbases)).

4.26 How to service commands in the subject line.

Rfc2142 (standards track) says that for each mailing list `list@host`, there MUST be an administrative address `list-request@host`. This is not the default for ezmlm, but can be added with ezmlm-make(1) "-q", which adds a ezmlm-request(1) line before the ezmlm-manage(1) line in **DIR/manager**. This address is used to manage commands in the "Subject:" line, by translating them into appropriate ezmlm command messages.

When migrating from other mailing list managers which use this method to issue list commands, configuring ezmlm to respond to such commands may be useful. In addition, some software manufacturers sell MUAs and mail gateways that are unable to correctly transport rfc822-compliant Internet mail with certain characters in the local part of the address.

ezmlm-request(1) services the `list-request@host` address per rfc2142 (standards track). It is usually invoked in **DIR/manager** before ezmlm-get(1) and ezmlm-manage(1). It ignores all requests that are not for the list-request address. For requests to the `list-request@host` address, ezmlm-request(1) parses the "Subject:" line. If a ezmlm command address starting with the contents of **DIR/outlocal** (e.g. `list-get45`) is on the command line, ezmlm-request(1) generates the corresponding full ezmlm request message. If the subject does not start with the contents of **DIR/outlocal**, ezmlm-request(1) prefixes the line with the contents of **DIR/outlocal**, thereby building a complete ezmlm command. If a host name is specified, it must match the contents of **DIR/outhost**, i.e. ezmlm-request(1) in this function will only generate command messages for the local list.

Thus, a subject of "subscribe" to `list-request@host` will be auto-magically rewritten as a message to `list-subscribe-userlocal=userhost@host`. Similarly, any ezmlm command or "Reply-To:" address can be pasted into the subject field and sent to `list-request@host`. ezmlm-request(1) does not validate the command name, but invalid commands result in a "help" message in reply via ezmlm-manage(1). This allows ezmlm-request(1) to also service custom commands, like `list-faq@host` that you may have created for your list.

If the "Subject:" is empty or does not start with a letter, ezmlm-request(1) will attempt to interpret the first message body line that starts with a letter in the first position.

When ezmlm-request(1) has successfully processed a "request" command, it exits 99 to skip the rest of **DIR/manager**.

To set up a list to include ezmlm-request processing, use the ezmlm-make(1) "-q" switch. The default is to not do this.

4.27 How to support alternative command names.

ezmlm-idx>=0.23 allows alternate names for all user commands. This can be used to e.g. make a message to `list-remove@host` to result in an "unsubscribe" action. This may help migration from other mailing list managers and in non-English environments. The use of aliases allows ezmlm to respond to new command names, while always responding correctly to the standard commands. If ezmlm-request(1) is used it will automatically be able to deal with any commands you set up for the list, within ezmlm or as separate programs. See 19 (Multiple language support) on how to set up command aliases.

4.28 How to add your own commands.

The qmail/ezmlm mechanism makes it very easy to add your own commands. You can add them to **DIR/manager**, but this requires great care in terms of ordering and exit codes. Easier is to set them up separately with a **.qmail-list-command** file.

Let's assume you want to allow anyone to determine how many subscribers are subscribed to your list with the command `list-count@host`. Just create a program to do the work:

```
#!/bin/sh
DTLINE='Delivered-To: list-count@host processor'
grep "$DTLINE" > /dev/null &&
        { echo "This message is looping"; exit 100; }
{
  echo "$DTLINE"
  cat <<EOF
  From: list-help@host
  To: $SENDER
  Subject: list@host subscriber count

  Current number of subscribers:
  EOF
  ezmlm-list /DIR | wc -l
} | /var/qmail/qmail-inject -f list-return- "$SENDER"
exit 0
```

Then, create **DIR/count** containing "|/path/program" and then do "ln -sf DIR/count ~/.qmail-list-count". Now, the command will pass the message to "program". The first thing "program" looks for is its delivered-to line to detect looping. If not found, it goes on to print this header, followed by some minimal text and the subscriber number. This can of course be made prettier with ezmlm-list error checking, and maybe in perl, but shows how easy it is to extend ezmlm. All thanks to the DJB/qmail delivery mechanism.

4.29 How remote administrators can retrieve a subscriber list

A user with shell access can always manipulate subscriber lists with ezmlm-sub(1), ezmlm-unsub(1), and ezmlm-list(1) for the lists s/he owns.

Sometimes a remote administrator requires a list of subscriber E-mail addresses. At the same time, the list should be kept out of the hands of spammers and all unauthorized entities. By default, ezmlm does not allow remote subscriber list retrieval. You can enable the "-list" command for remote retrieval of a subscriber list by using the ezmlm-make(1) "-l" switch or by adding the "-l" switch to the ezmlm-manage(1) line in DIR/manager. With this switch, ezmlm will permit retrieval of a subscriber list, but only to remote administrators. Subscribers cannot get the list membership, and any outsider would have to be able to read a remote administrator's mail to get the list. *Note: This option is not functional unless the list is configured for remote administration, i.e. the ezmlm-make(1) "-rl" switches need to both be used.*

The list returned is unsorted for efficiency reasons. You can easily sort it or use your mail reader to find a specific entry. The number of subscribers is shown at the bottom of the list. To get the number of subscribers from the command line, use:

```
% ezmlm-list DIR | wc -l
```

4.30 How remote administrators can search the subscription log

The same conditions that enable remote administrators to retrieve a subscriber list (see 4.29 ()) also enable the remote admin to retrieve the subscription log, i.e. the log of changes made to the subscriber list. The command is `list-log@host`. The entries are of the form "date timestamp dir event address comment". "dir" is "+" for addition of an address, "-" for removal, "event" is empty for normal (un)subscribe "manual" for changes made with ezmlm-(un)sub, and "probe" for removals via bounce handling. "address" is the subscription address, and "comment" is empty or the subscribers "From:" line. The log can be used to look at recent additions/removals and to try to track down a subscriber address from e.g. the name on the "From:" line. The log is written on a best-effort basis. In contrast to the subscriber database, entries in the log may be lost at a system crash.

The remote administrator can do a case-insensitive search through the log with the command `list-log.xxx@host`, where "xxx" is any sequence of letters/numbers that must occur on a line in order for that line to be included in the reply. A "_" is a wild card and should be used for special characters as well. Thus, to search for any entry with a host name of `host*` mail `list-log._host` and to find entries for "Keith John..." etc, use `list-log.keith_john`.

For SQL-enabled lists, this command searches the "list_slog" table.

4.31 How text file editing works.

If a list is set up with the ezmlm-make(1) "-n" switch, or if the "-e" switch is added to the ezmlm-manage(1) line in **DIR/manager**, ezmlm allows remote administrators to edit the text files that make up most of the ezmlm responses. Of course, this will work only if remote administration is enabled for the list. Replies are sent only if the target address is a remote administrator. Thus, ezmlm does not rely on SENDER (easily forged) but on the notion that only the recipient receives the message. This is a reasonable assumption for remote administrators that receive mail on the local system.

With this switch, ezmlm replies to the `-edit` command with a list of the files in **DIR/text/**. Only files where editing seems reasonable are included in the list. The remote administrator can edit any file in **DIR/text/** by sending e-mail containing the new text to `-edit.file` where "`file`" is the name of the file replaced (edited). The file must exist and the name consist of only lower case letters and '-'. Any '-' (hyphen) must be substituted by a '_' (underscore). For remote administrator convenience, the substitution has been made in the list of files sent in reply to the `-edit` command.

In reply to this command, ezmlm sends a message with the file and editing instructions. A "cookie" based on the date, file name, and contents of the file is added to the "Reply-To:" address. The cookie becomes invalid as soon as the file has been changed, or after 27 hours, whichever is shorter. Also, the cookie cannot be used to edit any other file, even if the other file has exactly the same contents. If you sent an edit request, and decide not to edit the file, you can simply delete the message.

To apply standard changes to all your text files it is easier to edit **~/.ezmlmrc**. To reset the list's text files back to their default contents (as specified by **ezmlmrc(5)**), use the ezmlm-make(1) "-e" switch together with any other switches used to set up the list.

4.32 How subject line prefixes work.

First of all, it is against a number of RFCs to modify the "Subject:" header of messages. However, it is frequently requested by users who have seen it on other list managers. Second, it is many times worse to have a prefix that changes from message to message, such as a prefix with the message number. However, a number of lists, especially in Japan, use this feature and in its absence these lists might be unable to take advantage of ezmlm. Thus, while we recommend against using a prefix, ezmlm-idx supports it.

To add a subject prefix, just put the text into **DIR/prefix**. The only format that makes any sense is "list:" or "(list)" or such.

The message number prefix is activated by putting e.g. "(list-#)" into **DIR/prefix**. "#" is replaced by the message number. ezmlm refuses to make more drastic changes in the subject of a message. As a consequence, the message number prefix is added only when the subject does not already contain a prefix. Thus, replies will have the message number of the original message. Doing anything else and still supporting rfc2047-encoded subjects in the archive threading (much more important) would require decoding the subject, removing/editing the prefix, and re-encoding the subject. This is far too invasive.

The entire thread can always be retrieved by sending a message to `list-thread-x` where "x" is the message number in the prefix of any message in the thread.

4.33 How bounces are handled.

Ezmlm messages are sent with an envelope sender ("Return-Path") that directs bounces to **DIR/bouncer** and also via "VERP" contain information about the intended recipient. Thus, programs run from **DIR/bouncer** know the subscriber for whom the message bounced. ezmlm-weed(1) is used to weed out delivery delay notification and other junk. For others ezmlm-return(1) decides if the address is a subscriber. If so, it saves the first bounce message and a list of bounced-message numbers. ezmlm-warn(1) executed from e.g. **DIR/editor** goes through these bounce files. If it finds any that are older than 1,000,000 seconds (about 11.6 days) it sends a warning message to the subscriber. If this warning message bounces, ezmlm-return(1) sets up a "warning flag" for the subscriber. If ezmlm-warn(1) finds a warning flag older than 11.6 days, it sends a "probe" to the subscriber. If ezmlm-return(1) receives a bounced probe, the subscriber is automatically unsubscribed.

The ezmlm-warn(1) "-t" switch can be used to change the time-out (in days). The ezmlm-warn(1) "-d" switch causes processing of "list-digest" bounces rather than "list" bounces. ezmlm-weed(1) and ezmlm-return(1) can handle bounces for either list.

ezmlm-warn(1) also removes any files in the bounce directory that are older than 3 times the bounce time-out.

ezmlm-warn(1) is normally run from **DIR/editor**. This can take quite a lot of resources, if there are a large number of bouncing addresses (>>1000) on a busy list, since by default all bounces are stored in a single directory and ezmlm-warn(1) examines all of them with each invocation. ezmlm-idx->=0.32

changes bounce handling to improve performance for large lists. Bounces are stored in subdirectories of **DIR/bounce/d/**, one per 10,000 seconds. The corresponding address hashes are stored in 16 subdirectories of **DIR/bounce/h/**. Instead of looking at all bounces, ezmlm-warn(1) processes only the bounces in **DIR/bounce/d/** subdirectories that are "due". In addition, ezmlm-warn(1) uses **DIR/bounce/lastd** as a simple lockout, to assure that it will do work only at most once every 5.5 hours. (Times are scaled to the ezmlm-warn(1) "-t" argument if used.) Together, these changes assure that bounce handling will scale well in the default configuration, even for very large lists.

4.34 How the info and faq commands work.

The *-info* and *-faq* commands simply reply with the contents of the **DIR/text/info** and **DIR/text/faq** files. Edit these files directly or remotely (see 21.8 (How to remotely edit dir/text files)). The **DIR/text/info** file should start with a single line that is meaningful as is and describes the list. This will be used in later versions to allow automatic assembly of the global "list-of-lists" (see 15.2 (How to set up a global list address like majordomo@host or listserv@host)).

4.35 How the global ezmlm list address works.

Sometimes, it is desirable to have a host- or user-wide address that can list available mailing lists.

ezmlm-request(1) can be used to set up a global address, such as `ezmlm@host` which allows the user to see and interact with a number of different mailing lists. This is especially useful when your users are used to other mailing list managers, such as "majordomo" or "listproc". ezmlm-request(1) is set up to answer requests to the address (see 15.2 (How to set up a global list address like majordomo@host or listserv@host)). There, it interprets the first line of the message body as a command. It will reply directly to "lists" and "which" commands. All other commands will be used to construct messages to the respective lists. Where other mailing list managers use synonyms of ezmlm commands, ezmlm-request(1) recognizes these and translates them to the corresponding ezmlm commands. ezmlm-request(1) will build commands also of unrecognized commands. Thus, if you create new commands for a list, ezmlm-request(1) will automatically support them.

If the user does not specify the complete list address, ezmlm-request(1) will attempt to complete the name. See the ezmlm-reject(1) man page for more info.

4.36 How ezmlm-cron works.

If you are a user and have crond(8) access, if you do not need to get digests at specific times, or if you are a system administrator setting up lists, there is no reason for you to use ezmlm-cron(1). If you are a system administrator not allowing users crond(8) access or a user that needs digests at specific times, but without crond(8) access, read on.

ezmlm-cron(1) is a very restrictive interface to crond(8). ezmlm-cron(1) can be used to create digest trigger messages. If a list is set up with a digest code (see ezmlm-make(1) and ezmlm-get(1)) ezmlm will generate a digest from the list `joe-sos@host` sent to to subscribers of `joe-sos-digest@dighost` when receiving a message to `joe-sos-dig-code@host` where "code" is the digest code. ezmlm-cron(1) can be used to generate such messages at regular intervals. The file **ezcronrc** is set up by the sysadmin and controls what trigger messages specific users may set up via ezmlm-cron(1).

Usually, the ezcronrc of that use will have an entry like "user:user-:host:10" allowing "user" to create trigger messages for up to 10 lists with names starting with "user-" and on the host "host".

To list the ezcronrc line controlling your use of ezmlm-cron(1):

```
% ezmlm-cron -c
```

To list all entries that you've created:

```
% ezmlm-cron -l
```

To add an entry to trigger digests from `list@host` every morning at 0230:

```
% ezmlm-cron -t 02:30 -i24 list@host code
```

A new entry for the same list overwrites an old entry.

To delete the entry above:

```
% ezmlm-cron -d list@host
```

or use ezmlm-cron to trigger messages at a different time:

```
% ezmlm-cron -t 16:16 -i24 list@host code
```

4.37 How ezmlm-make works.

ezmlm lists allow almost infinite customization. The component build, together with the qmail delivery mechanism makes it possible to create any variant of list function imaginable. However, this complexity makes it somewhat daunting to the average user wanting to set up a mailing list. ezmlm-make(1) allows automated list setup, while permitting a large amount of configurability.

At first glance, ezmlm-make(1) has many complicated options. However, these can be applied iteratively through the ezmlm-make(1) edit mechanism. Also, they are intended to be relatively complete so that execution of ezmlm-make(1) by e.g. a GUI can be used to safely set up and edit any list.

ezmlm-make(1) reads its command line arguments and switches, then creates the list directory. If the "-e" edit or "-+" sticky edit switches are not specified, ezmlm-make(1) will fail if the directory already exists. The directory argument must be an absolute path starting with a slash. The dot-qmail file argument, if specified, must also be absolute.

ezmlm-make(1) next reads ezmlmrc(5) located in the /**etc**/ directory with a default install. If not found, the file in the ezmlm binary directory will be used. The second ezmlm-make command line argument specify the root name of the .qmail files. If the ezmlm-make(1) "-c" switch is used, ezmlm-make(1) will look in that directory for a **.ezmlmrc** file and use it instead. If this file does not exist, ezmlm-make(1) will print a warning and use the previously discussed ezmlmrc(5) files in the same order. You can also use "-C *ezmlmrc.alt*" to use *ezmlmrc.alt* as the ezmlmrc(5) file. Again, ezmlm-make(1) will fall back to the others with a warning, if the specified ezmlmrc(5) file is not found.

When not run in "-e edit" or "-+" sticky edit modes, ezmlm-make(1) first creates the list directory. It also as the last step of its action creates **DIR/key** containing the key used for cookie generation.

The ezmlmrc(5) file consists of a number of file names relative to the list directory, followed by conditional flags (see ezmlm-make(1) and ezmlmrc(5) for details). If all the conditional flags (controlled by the corresponding command line switches) are true, the lines that follow are entered into the named file. There are also tags to erase files. Tags in the format <#X#> (where "X" is any number, except "1" and "2") are replaced by the corresponding ezmlm-make(1) switch argument. The ezmlm-make(1) command line arguments and the ezmlm binary path can be similarly substituted into the text. Thus, ezmlmrc(5) controls (within reason) the entire operation of ezmlm-make(1). ezmlmrc(5) is also set up so that no messages or file containing list state information are lost. Therefore, ezmlm-make(1) can be used to safely edit existing lists. The only caveat is that the list state is undefined while editing is in progress. Thus, it is advisable to prevent mail delivery by setting the "sticky" bit on the user's home directory while editing lists.

ezmlm-make(1) will create the file **DIR/config**. This files saves all the flags that were set at the last execution of ezmlm-make, as well as all the switch and command line arguments. When editing a list, only "DIR" and the non-default letter switches need to be specified. Other command line arguments and the "digit switch" arguments are read from **DIR/config**. To remove a digit switch, simply use it with two single quotes as the argument.

You can also easily determine how a list was set up by looking at **DIR/config**.

4.38 What names can I use for my lists?

Rather than restrict you to a single E-mail address (`user@host`), qmail in the default setup gives you control over an infinite number of addresses `user-*@host`. Of course, you (normally) have no way of controlling `elsewhere@host` since that could lead to overlap between users' "e-mail address space". As a consequence, all you mailing lists have to be named `user-xx@host` where "user" is your user name and "xx" is anything. You cannot create e.g. `mylist@host`, only `user-mylist@host`. To create the list `user-list@host` do:

```
% ezmlm-make ~/list ~/.qmail-list user-list host
```

Notice that "user" is **not** part of the **.qmail** file name.

There are two way to create lists with names not starting with your user name: First, qmail can be set up so that you control a virtual domain (see below). Second, the system administrator can set up lists with arbitrary names within the **~alias**/ directory.

4.39 Lists in virtual domains

If you use qmail>=1.02 and ezmlm-idx>=0.32, lists under virtual domains work just like other lists and require no adjustments. You can choose any local name for the list and the ezmlm-make(1) argument "local" is that name; "host" is the name of the virtual domain.

4.40 How do I make customization simple for me/my users?

All non-default switches, ezmlm-issubn(1) setups, etc, can be made standard for new lists by customizing the ezmlm-make(1) configuration file named "**ezmlmrc**". A default **ezmlmrc(5)** is installed in the ezmlm binary directory. If installed, a system-wide customized ezmlmrc file in **/etc/ezmlmrc** (or symlinked from there) overrides this. Installing a **~/.ezmlmrc** file in a user **dotdir** and using the ezmlm-make(1) "-c" switch allows further per user customization (see 7 (Customizing ezmlm-make operation)).

5 ezmlm support for SQL databases.

5.1 Why use an SQL database with ezmlm?

MySQL support is experimental. The main advantages are that you are using an address database system that can easily be accessed from any number of other programs via ODBC, perl, java, PHP, ... You can easily hook up ezmlm with your customer database, etc. An even bigger advantage is for use with distributed lists, which can be easily managed, changed, and monitored via a central database. ezmlm programs compiled with MySQL support (and when available also those compiled with support for other SQL servers) are entirely backwards compatible. You can mix SQL dbs with normal ezmlm dbs, and convert lists between them.

5.2 Why not to use an SQL database with ezmlm.

The main disadvantages of the MySQL version are that you need to be familiar with MySQL, the binaries are quite a bit larger, and you are trusting your addresses to a large database program, rather than a small and easily audited set of ezmlm programs. Also, the SQL server becomes a single point of failure.

MySQL is *very* robust and is used in many mission-critical situations. Also, ezmlm with MySQL support continues to rely on qmail stability. If connection fails, ezmlm aborts with a temporary error causing redelivery at a later time point.

If you create list clusters with main list and sublists, security for normal ezmlm sublists scales well: compromise of one sublist may lead to loss of that sublist. Security for the SQL sublists does not scale well: compromise of one sublist may lead to the loss of the entire list. This is of minimal importance if you want to split the load of a list onto a few local hosts (such as when your main list host becomes overloaded due to the popularity of your lists). Here, all host are under your control and equally secure.

There are other ways to do this: First, we could send the addresses with the message. This would use a single communication channel. I would prefer this model with a compressed and PGP-encrypted address block. However, it is hard to do within the framework of ezmlm (backwards compatibility) and encryption support is made complicated by ITAR. The dual channel system has advantages: addresses are harder to capture and logging is more reliable. Also, you can easy to tunnel MySQL and SMTP through ssh, if you desire encrypted communication.

In future, QMTP might become more available and able to transfer messages with multiple recipient addresses. This would allow outgoing load splitting from an ezmlm list host. Our implementation with ezmlm has more overhead, with the advantage that it is available now and that it also distributes the load of bounce handling.

Most parts of ezmlm-idx have been thoroughly tested through several versions. MySQL support is new for this version of ezmlm-idx. It is not as well tested as other parts of ezmlm/idx, it is not as well documented, and it may change in the future. This means that you can help to improve it. If this "newness" bothers you, compile your binaries without MySQL support.

5.3 Tables used for (My)SQL support.

The basic philosophy is that the database can be on any host (if you use SENDER restrictions, connectivity to the main host is more important than to the sublists), and you choose the database and "table root" names. The default database is “ezmlm” and the default table root is “list”. Each list has a separate table root. Any number of lists can share a database.

The main list address table is named with the table root only, others have that name with various suffixes. In the following “list” is used as the table root.

5.3.1 Address tables.

list

List subscriber addresses.

list_digest

Digest list subscriber addresses.

list_allow

List subscriber alias addresses. Used only if SENDER restrictions are used for the list. This is configured in the default SQL list setup, but a local (ezmlm-style non-SQL) database could also be used.

list_deny

List deny addresses. This table is created, but the default configuration, if it uses the “deny” addresses at all, will do so with a local database.

list_mod

Moderator addresses. Created for completeness, but not used in the default configuration. If moderators are used, the addresses are stored in a local database.

5.3.2 Subscriber log tables.

For each of the above tables, there is a “*_slog” table that contains one row per transaction against the corresponding address table. The entries contain a time stamp, the subscription address; a direction indicator (“-” for removals, “+” for additions); a type indicator (blank for ezmlm-manage, “m” for “manual”, “p” for “probe, i.e. bounce handling; and the subscriber “From:” line contents (only additions and only when made by ezmlm-manage or by “ezmlm-sub(1) -n”).

5.3.3 Message logging tables.

For both the list and the digest list, there are a pair of tables that log messages:

list_cookie

The main list stores the message number and a pseudo-random cookie in this table when it processes the message. The cookie is derived from the secret **DIR/key**, the message sender and the message number. Thus, it is non-repeating and virtually impossible to guess beforehand. Sublists will check that the cookie sent with the message is the same as the one received with the message.

The digest list is created similarly, except that it is ezmlm-get(1) that originates the message and creates the cookie. This is done in “list_digest_cookie”.

list_mlog

Both the main list and the sublists make entries in this table. Each entry consists of a time stamp, a message number, a list number, and a code. The code is 0 for message arrival, 1 for "finished processing", 2 for "receipt received" and -1 for bounce. The lists will refuse to process messages that do not have the correct cookie, or if the message already has an entry with a code of greater than 0. To inject a message at the sublist, an attacker would have to inject a message with the correct code before the list has processed the "real" message, or subvert the SQL server. In practice, this is very hard to do, unless the attacker has broken security at the database server or a sublist. This authentication mechanism is intended to make it safe to sublist moderated lists. It also blocks any message duplication between main list and sublist from being propagated to the subscribers.

The codes 2 for "receipt received" and -1 for bounce are entered by ezmlm-receipt(1) at the main list. This program is configured instead of ezmlm-return(1) if the main list was set up with "ezmlm-make -w6". ezmlm-receipt(1) checks the cookie of messages addresses to `mainlocal-return-receipt@mainhost` and if correct enters the "receipt received" code. This address is normally in the subscriber database with a hash of 98, so that each list sends a message to the address *after* all subscriber addresses.

Bounces of sublist messages should not lead to removal of the sublist from the database. ezmlm-receipt(1) will instead log the bounce to the "list_mlog" table. It will also store up to 50 bounces in the bounce directory. This helps error detection and diagnosis. After the first 50 bounces, no more bounces are stored, until you manually remove the old ones. This is to prevent filling up your hard disk in case a configuration error causes a deluge of bounces.

The digest list is treated in the same manner. Here, the tables is "list_digest_mlog" and the feedback address is `mainlocal-digest-return-receipt@mainhost`.

5.3.4 Sublist split tables.

Two tables are responsible for defining which sublist is responsible for which addresses. These tables are used only for distributed lists.

list_name

This is the sublist split table for the main list. It contains the list number (listno) as an auto_increment column, the list address (name), a flag to make the entry not active (notuse), and a range of message numbers for which the entry is active (msgnum_lo, msgnum_hi). The latter are useful when updating running lists, but can be ignored for now.

There is a "domain" column which contains the first 3 letters of the last part of the address domain that this list services. Thus, it would be "jp" for `*@*.jp`, "edu" for `*@*.edu`>, and "hom" for the hypothetical domain "home". The default is that domain is empty.

In addition, there is a hash range (hash_lo, hash_hi). The entry will apply only to addresses with a hash in that range. Since the hash of addresses is approximately evenly distributed in the available range of 0-52, the range corresponds to the fraction of addresses covered.

list_digest_name

This works as the main list split table, but for the digest list.

5.4 How addresses are distributed among sublists.

To determine which addresses are handled by which sublists, the inactive sublist split table entries (notuse != 0 or current message number not in the msgnum_lo to msgnum_hi range) are first eliminated. Next, each sublist with a defined domain gets in that domain the addresses that fall within the hash range. Entries with a blank domain handle addresses with a blank domain as well as those not handled by other entries. The addresses handled by a list are the union of addresses handled by all entries for the particular list.

This means that the administrator needs to assure that there are lists with a blank domain covering the entire hash range 0-52. For each defined domain, the administrator must likewise assure that the entire hash range is covered. However, there is no corresponding requirement to cover the "domain range" since domains not specifically covered are handled by the entries with blank domains. This arrangement makes the database queries a little more complex, but makes administration considerable easier than other arrangements, while still allowing on-the-fly reconfiguration. See 14.4.3 (setting up a distributed ezmlm list with SQL support) for a detailed example.

For logging, the listno is determined as the lowest listno of a "list_name" entry that is active for the current message.

5.5 How to set up a simple list with SQL support.

To use SQL database support, you have to compile the programs with SQL support. Currently, only MySQL support is available. See **INSTALL.idx** in the package on how to do this.

The programs with SQL support will work exactly like the normal programs for standard lists. However, if the file **sql** exists in the basedir, it turns on the SQL mode and it is expected to contain SQL server connect info in the format

```
"host:port:user:password:database:table"
```

Here, "Host" is the SQL database server host, "port" can be left blank to use the default port, "user" and "password" are connection credentials for a user you need to define and grant access to the database. "Table" is the name of the address table ("list" in the examples above and "list_digest" for the corresponding digest list). For list clusters, ":sublist" is suffixed to this info and it is the name/address of the sublist.

For each address database, you also need to create the address table as well as the "*_slog" subscription log table. In addition, you should create a "*_cookie" and "*_mlog" table for message logging. This is all it takes to start using an SQL database.

5.5.1 Helper programs for SQL-enabled lists.

Two programs are supplied in the distribution to make it easier to create the database user and tables. Also, ezmlm-make(1) has support for setting up SQL-enabled lists.

Creating the tables

ezmlm-mktab(1) will create the necessary tables:

```
% ezmlm-mktab -d table
```

Pipe this into the SQL client with the appropriate administrator credentials needed to create tables (see MySQL documentation, e.g. `<http://www.tcx.se/>`).

For most lists, the only addresses that are stored in the SQL database are the subscribers of list and digest, and the "allow" aliases. It is NOT normally advisable to store moderator addresses there, since they are needed only at the main list and secrecy is more important. "Deny" addresses are few and again only needed at the main list. "Allow" are put in the SQL database when using the default ezmlmrc file only to make all relevant addresses manipulatable via the SQL server. The other tables are created, in case they are wanted (the cost for having them as empty table is zero). The basedir/sql file is the decision point. If it exists, an SQL table is used; if not a local ezmlm db is used.

Creating a user entry

ezmlm-grant(1) helps you to create a list user:

```
% ezmlm-grant -d database -t table host user password
```

This will grant minimal access to "user" with "password" to "database" on "host" with table "table, "table_digest", etc, conceivably needed. Just pipe this into the SQL client, e.g. mysql with the appropriate administrator credentials needed to modify grant tables (see MySQL documentation, e.g. `<http://www.tcx.se/>`).

Creating the list

ezmlm-make(1) supports SQL-enabled lists with the "-6" switch:

```
% ezmlm-make other_switches -6 'host:port:user:pw:db:table' \
        dir dot local host
```

Will create an SQL-enabled list that uses the SQL server for the main list subscribers, digest list subscribers (if configured) and "allow" poster alias addresses (if configured).

5.6 Manually manipulating the subscribers of a SQL-enabled list.

ezmlm-sub(1), ezmlm-unsub(1), and ezmlm-list(1) work as you would expect also with a SQL-enabled list. ezmlm-list(1) may be minimally slower (depending on network speed) if the SQL server is not local. ezmlm-sub(1) and ezmlm-unsub(1) will be faster, but this is noticeable only with very large subscriber lists and addition/removal of large numbers of addresses (more than several thousands).

5.7 Converting to and from and SQL database.

Just like other programs, ezmlm-list(1), ezmlm-sub(1), and ezmlm-unsub(1) will work with normal address databases in the absence of **DIR/sql**. However, they also have a "-M" switch to force this behavior even in the presence of **DIR/sql**. This is used to convert an address database from the standard type to the SQL type:

```
% ezmlm-list -M dir | xargs ezmlm-sub dir
```

or from the SQL version to the standard type:

```
% ezmlm-list dir | xargs ezmlm-sub -M dir
```

To synchronize the two, remove one and then update it with ezmlm-sub(1) from the other. Alternatively, sort the ezmlm-list(1) output for both, use diff and sed/awk to get separate files of the differences, and use ezmlm-sub(1) and ezmlm-unsub(1) to apply the differences to the appropriate database.

This type of conversion can serve as a convenient means to convert a list from one type to another, to back up databases, and to move subscriber addresses from a standard list to a SQL table for other purposes, or from a SQL database to a standard mailing list (you may need to use addresses from a SQL table, without wanting your lists to be dependent on an SQL server for day to day operation).

Note: This inter-conversion requires the **DIR/sql** file. If you do not run the list against an SQL server, you need to disable deliveries before you temporarily create this file. Otherwise, the list will run against the SQL database during the time **DIR/sql** exists.

5.8 Implementation of sublisting with SQL-enabled lists.

For more info on the SQL databases, see 5 (). In essence, we use one address table for the list and one for the digest list. For each address, a hash and domain is also stored. A "*_name" table has entries for sublists with sublist name, domain and hash range they serve. Multiple entries per list are allowed. In addition, there is a message number range and a "notuse" flag. These are provided to make it easier to make a new configuration take effect at exactly the same time as the old one ceases to be active. A sublist with an empty domain handles all domains not handled by another list. At least one such sublist is required. If you create an entry for a specific domain, it is your responsibility to make sure that the entire hash range is covered for that domain.

There are similar address databases for "allow", "deny", and "mod". Normally, none of these or only "allow" is used. ezmlm programs fall back on normal ezmlm databases, so it is quite possible (and advisable) to use a local "mod" database for a moderated list with an SQL table for subscriber addresses.

For each address database there is a "*_slog" table that contains one row per transaction, just like **DIR/Log**.

SQL sublists also use a "*_cookie" database. For each message processed by the main list, a pseudo-random "cookie" is generated and stored in this table. A header "X-ezauth:" with this cookie is added to the message. Sublists will explode a message only if the cookie matches the message number in the table, and if that sublist has not already processed that message. This makes it very hard (not impossible) for an attacker to inject messages at the sublist level.

To distinguish sublist addresses from normal addresses, sublists are added with a hash of 99. Thus, the main list only sends messages to addresses with this hash. The only way to make or remove such an entry is from the command line with:

```
% ezmlm-sub -s dir sublist@subhost
```

This makes it impossible for users to remotely subscribe sublists to each other, or to unsubscribe sublists from the main list.

A hash of 98 also receives special treatment. Both the main list and all sublists send a message to such addresses, *after* all other addresses. Thus, receipt of such a message (at a remote address) means that all remote subscriber addresses have been tried at least once for this message. Normally, you subscribe a feedback address "`mainlocal-return-receipt@mainhost`" with this hash:

```
% ezmlm-sub -r dir mainlocal-return-receipt@mainhost
```

This address is manned by ezmlm-receipt(1) which logs to the SQL database. A monitoring tool (e.g. **status.pl** supplied in the package) can be used to keep track of sublist function and efficiency.

Other hashed not in the normal (0-52) range can also be used. Messages are never sent to them. Since they cannot be remotely manipulated, their presence prevents normal subscription of the address. Here, you should add addresses that you wish to prevent from subscribing, such as possible aliases of sublist and mainlist. Even if done, the "X-ezauth" mechanism will thwart most attempts that we can think of.

5.9 Interpreting the message log.

This is easy. For each combination (listno,msgnum,done) there can be only one entry. Each entry has a time stamp in addition to this information. The first entry will remain and subsequent entries will fail. For any list, "listno" is the lowest list number for an entry in the "*_name" table which is active for the respective message. When the message arrives at the main list, a cookie is calculated for it, entered into the "*_cookie" table, and added to the message. An entry is made for message arrival (done=0) in the message log. When the list has finished processing the message and qmail has returned success, and entry for processed (done=1) is made. If the message processing fails, the message is redelivered, but the "arrival" entry first made stands. Thus, this is the time the message first arrived at the list/sublist.

At the end of all subscriber addresses, the return-receipt address is sent to qmail. If this is remote (usually) it will be tried after all (remote) subscriber addresses. Thus, sending it means that all subscriber addresses have been tried at least once. When the return-receipt is received at the main list by ezmlm-receipt(1), this program makes an entry for "receceived" (done=2) if the cookie is present and correct. If the message to the sublist bounces, ezmlm-receipt(1) makes an entry for "bounce" (done=-1) and saves the bounce in **DIR/bounce**.

Sublists will bounce messages for which there is already an entry with done of "processed" or "receipt received" (done > 0). If there is duplication of the message, or on rare occasions when the sublist operating systems misbehaves. Thus, you may have an entry both for "receipt" and "bounce". In this case, check your setup, but don't worry. The message was delivered ok.

If there is a long delay between "processed" on the main list and "arrived" on the sublist(s), there is an outgoing qmail problem at the main list, a network problem between the host, an incoming qmail problem at the sublist or possibly a list configuration problem. If there is a long delay between "arrived" and "processed" the sublist ezmlm-send(1) may be failing due to misconfiguration, memory shortage, etc. Alternatively, the sublist has trouble connecting to the SQL server. A long delay between "processed" and "receipt" suggests an outgoing qmail failure at the sublist, a network problem between sublist and main list, or an incoming failure at the main list. As ezmlm-receipt(1) at the main list needs to access the SQL server to log receipt, it is also possible that there is a problem in that communication. All this said, problems other than those caused by network outages and list server load are all that we've seen in testing, despite several network hardware failures, memory shortage, etc.

Since bouces can be ignored in the light of other entries, the best way to find out about the status of a message is to look for the entry on the particular sublist and message number with the highest "done" code. **utils/status.pl** in the distribution interprets status codes and presents an overview over the last 10 messages (default). Most table entries are links to message and list statistics and explanatory texts. Depending on your particular lists, you may want to adapt the specific breakpoints in time chosen in the program.

5.10 Optimizing MySQL for ezmlm.

5.10.1 Address SELECTs.

If you run a simple list without sublists, address selection from the SQL table is (almost) a simple dump of all records. This is very fast and does not require any optimization of MySQL irrespective of subscriber number.

For list clusters with sublists, each sublist runs a more complicated SELECT against the database. This uses keys, but does not require sorting. A table JOIN is used, but the second table ("list_name") is very small, so tuning of join and sort buffers should never be required.

MySQL is extremely efficient as a database for large ezmlm lists. The address table structure is optimized to make it even faster. On a 586-100 Mhz/32 Mb RAM IDE machine (Linux 2.0.34) with a key_buffer of 4M, 12 sublists, 19 "list_name" rows, and 200,000 subscribers it takes 100 s for all 12 address queries to complete. This can be improved by a factor of 10 by use of a faster processor, e.g. a Pentium II-266 MHz. More memory will also improve performance as will (if you have it) multiple processors. Assuming that each sublist host can deliver at a sustained rate on 1.5 M deliveries per day, the query time should be compared to the 16 min delivery would take. If you have lists with 100,000-1,000,000 subscribers, you might be able to afford a faster SQL server than the one used by us. Query time seems to scale about linearly with number of subscribers between 100,000 and 400,000 subscribers (not tested beyond, YMMV).

The format chosen where a (set of) list(s) handles all domains not handled by any other list is a trade-off. It leads to more complicated address queries, but without it administration becomes too difficult.

5.10.2 Address addition/removal.

INSERTS/subscribes run at about 400/s when the table is empty, down to 200/s above 150,000 subscribers staying at that speed at least up to 400,000 addresses (PII-266 MHz, 112 Mb RAM/IDE, key_buffer=4M). On the same machine adding addresses to a standard ezmlm database took 42 min for the first 18,000 scaling poorly above 3000 or so. Of course, with normal list use this is not a concern since subscriber address transactions are rare compared with message processing.

The speed can be increased by removing indices, then recreating them after the addition/removal of addresses. However, the address index itself ("a") is required in the process. Removing the "d" and "h" indices on our test system increased the speed to 900/s. Indices can then be added in a minute or two (the domain and hash indices are essential for fast SELECTS). See the MySQL documentation (`<http://www.rcx.se>`) for more info.

Even if you do not use sublists, a hash and domain are calculated for each address. This overhead is minimal. Addition/removal of addresses also requires updating of 3 indices (address, domain, hash) in the address table which takes some time. The latter 2 are used only for sublisting. Removal of these indices will speed up the process. If you frequently bulk add/remove large numbers of addresses (more than 10,000 or so) it might be worth to delete those indices. If the list does not use sublists, there is no need to recreate these indices.

5.10.3 Tunable MySQL parameters.

Key_buffer. No change necessary unless you use the sublist features in combination with very large lists. Lists under 100,000 subscribers should do fine with 4MB, and then increasing by 4MB/100,000 subscribers should yield near maximal performance.

Others. With up to 400,000 subscribers, we have failed to see any impact of increases in join_buffer, record_buffer, etc. However, this may be due to the excellent caching of linux. If you have information on how to improve performance, please E-mail details to `lindberg@id.wustl.edu`.

5.11 Maintenance of the MySQL database.

Weekly to monthly error checks on MySQL tables is recommended. Best is to use:

```
# isamchk -s -O readbuffer=2M */*.ISM
```

Other options allow automatic correction of errors, but are dangerous if tables are accessed while isamchk is running.

Other isamchk options allow recovery of space after frequent insert/delete of addresses (can also be done with "OPTIMIZE TABLE"), key optimization, etc. See the MySQL documentation (`<http://www.rcx.se>`) for more info.

6 Possible error conditions in ezmlm lists.

6.1 What do I do if ezmlm doesn't work?

Try to determine where the problem occurs and how to reproduce it:

- Do messages to ezmlm return an error message to the sender or not?
- What is/are the error message(s)?
- What does ezmlm log into the mail log?
- Are you using a setup with virtual domains, and qmail<1.02 or ezmlm-idx<0.31? If so, have you adjusted **DIR/inlocal** (see 7.5 (Adapting ezmlm-make for virtual domains))?
- Are posts sent out to the subscribers?
- Are there subscribers?

  ```
  %  ezmlm-list DIR
  ```

- Are there moderators?

  ```
  % ezmlm-list moddir
  ```

  where "moddir" is the contents of **DIR/remote** (for remote admin lists), of **DIR/modsub** (for subscription moderated lists) or **DIR/modpost** (for message moderation), if and only if the contents start with a forward slash. The default in all cases is **DIR/mod/**. If both **DIR/modsub** and **DIR/remote** contain directory names, the one in **DIR/modsub** is used for both subscription moderation and remote admin.

- Are the ownerships of all files correct, i.e. read/writable for the owner?

```
% chown -R user DIR
```

For lists under alias:

```
% chown -R alias DIR
```

If you use custom moderator databases, those directories and all their contents must also be readable for the user under which the list operates (i.e. the user qmail changes to during the delivery).

- Read the qmail log and capture relevant parts.
- Did you customize the package at all? If so, try the default settings which are known to work.
- Did you customize **ezmlmrc(5)**? Try to use the default copy (skip the -c switch).
- Did your customization of **.ezmlmrc** fail to have an effect? Remember to use the -c switch. The **.ezmlmrc** file used is the one in "dotdir", i.e. the directory where the **.qmail** files go, usually, but NOT necessarily, the one in your home directory.
- Make sure you followed the instructions in man pages and other documentation. Most of the problems are due to not closely following the instructions. Try again with a new test list.
- Make sure to take notes of how the list was created (which flags you used, etc.).
- use ezmlm-check(1) (see 6.5 (Using ezmlm-check to find setup errors)). and compare the variables identified by ezmlm-check to **DIR/inlocal**, etc. If you don't get a reply from ezmlm-check, then message was not delivered properly. Check your qmail setup.
- Try to find your problem or a question/item close to it in the FAQ.
- If this didn't resolve the problem, post to the ezmlm mailing list, describing how you set up the list, your general setup (especially the relevant control files for a virtual domain), what works and what doesn't and what results from different actions (log entries, error messages).

If you have solved a problem that you believe might be more general, please send a description of the problem and its solution to the authors, ideally as a FAQ item.

6.2 How do I report ezmlm bugs?

If you have found a bug in the ezmlm-idx additions, please send a bug report by E-mail to `lindberg@id.wustl.edu`. Describe the error, your setup, and your system in sufficient detail so that it can be reproduced by third parties. Include relevant sections of mail log, and information about any error messages returned. If you ran into a problem and resolved it on your own, include a fix as a context diff against the distribution.

If you have found a bug in ezmlm proper (unlikely), please send a similar bug report to `djb@cr.yp.to` or `djb-ezmlm@cr.yp.to`. If you're unsure where the bug is, you can start with `lindberg@id.wustl.edu`. If you have problems and questions, please refer to the documentation, then to mailing list archives, then E-mail the ezmlm mailing list or the authors.

6.3 Where do I send suggestions for ezmlm-idx improvements?

E-mail to `lindberg@id.wustl.edu`, ideally with a context diff. For ezmlm proper, `ezmlm@list.cr.yp.to` may be better.

6.4 Using ezmlm-test to check the ezmlm(-idx) programs.

ezmlm-test(1) tests the different ezmlm(-idx) programs. It is useful to test your installation. If this program succeeds, it is not likely that you have problems due to platform-specific ezmlm(-idx) bugs. If ezmlm-test(1) fails, this is the place to start. The program is good at finding problems but not that easy to use to determine the cause. Start by finding the place where it fails, recreate the conditions (add "exit 0" just before the point of failure and set the environment variables as set by the script), then try to run the command manually. **~/__TSTDIR__err** may contain a relevant error message. For further help, E-mail `lindberg@id.wustl.edu`.

6.5 Using ezmlm-check to find setup errors.

ezmlm-check(1) is included in the ezmlm-idx distribution. ezmlm-check(1) is an evolving shell script which when put into a **.qmail** file of a mailing list will return information about the environment variables passed by qmail to ezmlm as well as the list setup. It also attempts to check for common error conditions, such as HOST and **DIR/inhost** mismatch, missing files, etc. To use ezmlm-check(1), place a line:

```
|/usr/local/bin/ezmlm/ezmlm-check 'DIR'
```

where "DIR" is the list directory, as the first line in **DIR/editor** (for mail to list), **DIR/manager** (for mail to `list-subscribe`, list-help, etc), **DIR/moderator** (for mail to `list-accept`, `list-reject`). ezmlm-check(1) will send its output to SENDER. The rest of the **.qmail** file will be ignored. If you use a non-standard ezmlm binary directory, change the ezmlm-check(1) path accordingly.

ezmlm-check(1) in combination with mail logs and ezmlm error messages should make it easy to diagnose setup problems. When done, don't forget to remove the ezmlm-check(1) line. It is not security-proofed against SENDER manipulation and with it in place, the list won't work.

ezmlm-check(1) does not check all aspects of list generation, but catches all common errors when lists are created with ezmlm-make(1), an many other errors as well. The ezmlm-check(1) reply is also very valuable for support via E-mail.

6.6 Posts are rejected: Sorry, no mailbox here by that name (#5.1.1).

qmail tried to deliver the mail, but there is no mailbox with that name. ezmlm-make(1) was used with incorrect arguments, often in conjunction with a virtual domain setup. If the list is in a virtual domain, the "host" argument for ezmlm-make(1) should be the virtual domain, not the real host name. See 4.38 (What names can I use for my mailing lists?) and 4.39 (Lists in virtual domains) for more info.

Other possibilities are that your qmail setup is incorrect. For a virtual domain controlled by user "virt", create **~virt/.qmail-test** containing "|/bin/echo "It worked"; exit 100". Now send mail to `test@virtual.dom`. If delivery works, you should get an error message "It worked" back. If you get anything else, you need to adjust your qmail setup. Similarly, for a normal user, create **~user/.qmail-test** and mail `user-test@host` to test that you control extension addresses. If this fails, contact your system administrator or adjust your qmail setup.

If these tests worked, but your list still does not, you most likely supplied an incorrect "dot" argument for ezmlm-manage(1). It should be **~virt/.qmail-test** for the list `test@virtual.dom` and **~user/.qmail-test** for the list `user-test@host`.

6.7 Post are not sent to subscribers.

Non-moderated lists

1. Read the qmail log. Is your message delivered to the list? You can also:

```
% cat DIR/num
```

2. Send a message to the list.
3. See if it was received/processed:

```
% cat DIR/num
```

If the number was incremented, the message went to the list, and was successfully sent out in the opinion of ezmlm-send(1) (ezmlm-send(1) doesn't mind if there are no subscribers, so check that there really are both moderators and subscribers. These are added with ezmlm-sub(1). You can not just put addresses into a text file!).

Message moderated lists

1. Check number of queued messages awaiting moderation:

```
% ls -l DIR/mod/pending
```

2. Send a message to the list.
3. Check if another message was added to the queue:

```
% ls -l DIR/mod/pending
```

A new file should have appeared. If this file has the owner execute bit set, it was successfully processed by ezmlm-store(1). If this is true, but no moderation request was sent, then continue with 6.16 (Messages posted to the list do not result in moderation requests). If there is no new file, the message did not reach ezmlm-store(1), or ezmlm-store(1) failed early. In both cases, the mail log should tell you more.

If the message is there, but the owner execute bit is not set, ezmlm-store(1) failed. Check the mail log. Possible reasons include a failure to find the ezmlm-send(1) binary or **DIR/msgsize** is specified and the message body size is outside of the allowed range (again, this is accompanied by an error message and mail log entry).

General

1. If the message was not received/processed, there should be an error message in the mail log.
2. Fix temporary and permanent errors with the help of qmail and ezmlm documentation.

3. If there is no log entry at all, then the mail went to another host. Check your qmail setup.
4. If mail was delivered to the list, but not forwarded to the subscribers (check the qmail log - there should be an entry for a new delivery to the list), **the most common error is that there are no subscribers.** In this case, ezmlm-send(1) sends a message from `list-help@host`, and logs success, but no recipients are logged. To qmail, it is perfectly acceptable to send a message without recipients, so no error message is logged.
5. Check subscribers:

   ```
   % ezmlm-list DIR
   ```

6. Assure that ownerships are correct on the list directories:

   ```
   % chown -R user DIR
   ```

   For lists owned by the "alias" user (in ~alias):

   ```
   % chown -R alias DIR
   ```

7. Most other problems should be easily corrected with the help of the qmail log.

6.8 ezmlm-make fails: usage: ezmlm-make ...

The command line you specified is incomplete. Usually, a command line argument has been omitted or a switch was placed after the other arguments rather than before.

The same error is issued when you attempt to invoke ezmlm-make(1) with only the "DIR" argument without using the "-e" or "-+" switch. Other command line arguments can be omitted only when editing lists created or previously edited with ezmlm-make from ezmlm-idx>=0.23.

Some special situations use ezmlm-make(1) as a general script processor, e.g. the setting up of sublists with ezmlmsubrc(5) and of a global interface with ezmlmglrc(5). Here, there is no "memory" so all arguments have to be specified, even when using the "-e" or "-+" switches.

6.9 ezmlm-make fails: Unable to create ...

This error occurs when ezmlm-make is used to set up a list, and it tries to create a directory or a **.qmail-list** link that already exists. Usually, this occurs because the list already exists. If you are creating a new list, first erase remnants of any old test lists by deleting the list directory and the link files: *NOTE: DO NOT USE THESE COMMANDS WITHOUT UNDERSTANDING THEM.* You may erase more than you intended!

```
% rm -rf DIR
% rm -rf ~/.qmail-list ~/.qmail-list-*
```

If you want to save some files (such as in **DIR/text/**), make backup copies first, run ezmlm-make, then copy the backups to **DIR/text/**. Of course, it is usually easier to create a custom **.ezmlmrc**, and than use that for all your lists.

To use ezmlm-make(1) to modify an existing list, without changing the subscriber or moderator lists or the message archive, use the ezmlm-make "-e" switch. NOTE: any customization that you've made that is not specified by the **ezmlmrc(5)** file used will be overwritten. For instance, if you manually added checks to **DIR/editor**, a pointer to a custom moderator database in e.g. **DIR/modsub**, or made a change to a

DIR/text/ file (such as adding a "welcome" message to **DIR/text/sub-ok**) these changes will be lost. To retain such changes (especially ones that are common for several of your lists), place them in a local **~/.ezmlmrc** file instead. You can either make such changes the default for your lists, or you can configure **~/.ezmlmrc** so that they are added only if a specific ezmlm-make switch is used. (see 7 (Customizing ezmlm-make operation)).

6.10 ezmlm-make fails: ... ezmlmrc does not exist

There is no readable ezmlmrc(5) file in **/etc/** nor in the ezmlm binary directory. If you have **.ezmlmrc** in "dotdir" (see 1.3 (Terminology: dotdir)) use the ezmlm-make(1) "-c" switch (see 7 (Customizing ezmlm-make operation)).

6.11 Index/get/thread requests fail quietly or with errors from ezmlm-manage.

Make sure this is an indexed list and has an "ezmlm-get" line first in **DIR/manager**. If not, your commands are fed directly to ezmlm-manage(1). If they contain "-", ezmlm-manage interprets the rest as an address to which it sends the error message. Usually, this results in a "trash address" mail log entry and a bounce, which is why you don't see any error message. The same happens if you send non-existing commands followed by "-" and arguments. Thus, `list-gugu-54@host` results in an ezmlm-manage error, resulting in help text being sent to `54@localhost` ... When testing, try using syntax with a ".", not a "-", after the action command, e.g. `list-get.54_60@host`. This will assure that error messages get back to you.

6.12 Digest triggering requests fail.

(Digest triggering by mail is a relic from older versions. Use the standard setup with ezmlm-tstdig(1) as by ezmlm-make(1) "-d", or run ezmlm-get(1) directly from the command line via crond(8).)

If you get an error message, it tells you why the request failed. If you do not, see the previous item. Try using syntax without "-" after the "dig" command. Also, requests that would result in an empty digest are silently ignored, but the reason why no digest was created is logged to the mail log. This is done so that cron scripts generating daily digest will just fail silently, rather than generating an error, for what isn't really one.

6.13 Remote administration (un)subscribe confirm requests go to the user, not the moderator.

Either the list is not set up for remote administration (i.e. **DIR/remote** does not exist), or the moderator is sending the request from an address that is not in the moderator database (e.g. from `Fred@host.dom`, when `fred@host.dom` is in the moderator db, but `Fred@host.dom` is not). ezmlm-manage(1) has no way of knowing that the SENDER is a moderator and treats the request as coming from a regular user, i.e. it sends a confirmation request to the target address. Correct the SENDER address, the address in the moderator db, or create **DIR/remote**. If you are using a non-default moderator db location, make sure that the moddir name is in **DIR/remote** (for remote admin only) or **DIR/modsub** (if there is subscription moderation as well). In both cases, the contents will be ignored unless they start with a "/".

6.14 (Un)subscribers does not receive a (un)subscribe acknowledgement

With normal ezmlm lists, a subscriber confirming a subscription or a non-subscriber confirming a unsubscribe request results in a message to the target address. This message is suppressed when the list is set up for subscription and/or remote administration, so that confirmations from multiple moderators do not result in multiple messages to the target address. The target address is always notified if the subscriber status of the address changes (from non-subscriber to subscriber or vice versa).

6.15 Messages posted to a moderated list are sent out without moderation.

The list is not set up as a moderated list. Check **DIR/editor**. If should contain a ezmlm-store(1) line after the ezmlm-reject line if it is a moderated list. No ezmlm-send(1) line should be in **DIR/editor**. If there is, the list is not moderated. Also, **DIR/modpost** must exist. If it does not, ezmlm-store(1) will post the messages directly (via ezmlm-send(1)) without sending them out for moderation first. This makes it easy to temporarily remove message moderation by simply removing **DIR/modpost**, but may be confusing if the user is unaware of this ezmlm-store(1) feature.

6.16 Messages posted to a moderated list do not result in moderation requests.

- Check that **~/.qmail-list** is a link to **DIR/editor**.
- Check that **DIR/editor** contains ezmlm-store(1) and not ezmlm-send(1). If this is not the case, the list is not message moderated.
- Check for the presence of **DIR/modpost**. If this file is missing, the list is not moderated, even if **DIR/editor** is set up with ezmlm-store(1).
- Check qmail logs for error conditions during post delivery and correct these. If the messages are delivered correctly, verify that ezmlm-store(1) generated the moderation requests to the moderators.
- Check to see that there are indeed moderators:

  ```
  % ezmlm-list moddir
  ```

  where "moddir" is the contents of **DIR/modpost** if they start with a "/", otherwise those of **DIR/remote** (same "/" requirement), and **DIR/mod/** by default.
- Check file ownerships.

  Another common problem is directory ownerships, especially for lists under ~alias. To correct this error, issue the following command while in the ~alias directory (User the user/group of the list owner; for ~alias lists user=alias, group=qmail):

  ```
  % chown -R user DIR
  ```

6.17 Moderation request replies do not result in the appropriate action.

- Check that the address in the moderation request is correct.

- Check that the **~/.qmail-list-accept-default** and **~./qmail-list-reject-default** links exists and point to **DIR/moderator**.
- Check that **DIR/moderator** invokes ezmlm-moderate(1), and that there is a copy of ezmlm-send(1) in the ezmlm binary directory.
- Check the qmail log to see that the replies were delivered to this address.
- Check directory ownerships. For lists under alias:

  ```
  % chown -R alias DIR
  ```

  NOTE: This needs to be done every time you add/remove moderators as "root". For user-controlled lists (i.e. you are "user" when running e.g. ezmlm-sub(1)) this is not a problem.

  If setting up lists for *alias*, you can avoid many problems by setting them up as "alias", i.e. use "su alias" not "su".

  If setting up lists for a user controlling a virtual domain, you can avoid many problems by assuming that uid ("su user") before making any changes.
- Check the qmail logs: After the delivery of the moderation request, ezmlm-send(1) should run to send messages to all the list subscribers.
- Make sure there are list subscribers:

  ```
  % ezmlm-list DIR
  ```

Most error conditions, incorrect request cookies, etc, should result in informative error messages in the mail log.

6.18 Moderator comments with moderation request replies are not added to the post/sent to the poster.

Moderator comments are where the moderator chooses to "reject" the message and inform the person posting which his/her message was inappropriate. However, if a moderator wants to comment on **accepted** posts, the moderator may only do so via a follow-up post to the list. This is to avoid anonymously tagged-on text to posts. If a moderator has something to say to the list, they should (and can only) do so in regular posts. If you want to edit posts before sending them to the list, set up a moderated list with you as the only moderator. Into **DIR/editor** before the ezmlm-store(1) line, put a condredirect(1) line that redirects all messages with a SENDER other than you to your address. You can edit the contents ands repost, the message will pass condredirect(1), and hit ezmlm-store(1). You will be asked to confirm (needed to assure that nobody else can post directly) and when you do, the messages is posted.

Moderator comments for "reject(ed)" posts need to be enclosed between two lines (yes, the end marker is required), having "%%%" starting on one of the first 5 positions of the line. If there are characters before the marker, these will be removed from any comment line that starts with the same characters (e.g. the characters before "comment2" in the example below will be removed):

```
%%%
comment
%%%
```

or:

```
> %%%
comment
> comment2
> %%%
```

but not:

```
%%
COMMENT
%%
```

and not:

```
%%% this is my comment %%%
```

or

```
ezmlm said>%%%
comment
ezmlm said>%%%
```

6.19 Some headers are missing from messages in the digest.

By default, only a subset of message headers are sent out in any digest and archive retrieval requests. First, headers in **DIR/headerremove** are stripped. Most non-essential headers are excluded when the default archive retrieval format ("m") is used. Use the "v" or "n" format (see ezmlm-get(1)) to get all message headers that are in the archive.

6.20 Some Received: headers are missing from messages.

ezmlm-idx>=0.313 removes all but the latest "Received:" header from messages sent to the list. This is done since messages, especially sent via sublists, may have so many "Received:" headers that MTAs with primitive "loop detection" erroneously reject them. The subscriber can subscribe, since those messages have fewer such headers, and will receive warning and probe messages, but never see any posts.

To see all headers of a message for diagnostic purposes, mail `mainlist-getv.num@mainhost`, where "num" is the message number. All "Received:" headers are stored in the archive copy of the message.

To disable "Received:" header pruning, use the ezmlm-send(1) "-r" switch.

6.21 My Mutt users cannot thread their digest messages.

The digest by default removed non-essential headers like "In-Reply-To:" from messages. Modern MUAs, like *Mutt* can split out messages from a digest and then thread them based on such headers. To include these and all other headers in the digest messages, use the "v" or "n" format as described on the ezmlm-get(1) man page. Normally, the threading done by ezmlm is sufficient and the default format preferred to reduce message and digest size, often by 25% or more.

6.22 Posts fail: Message already has Mailing-List (#5.7.2).

The list you are trying to post to is used as a sublist (a list fed with messages from another (ezmlm) list), but not properly set up as a sublist. Put the name of the parent list ("origlist@orighost") which exactly matches the SENDER of the original (or parent) list into **DIR/sublist**. Check the ownership of **DIR/sublist**, to make sure that the user controlling the list can read it.

Alternatively, use the ezmlm-make(1) "-0 `origlist@orighost`" switch (see 7 (Customizing ezmlm-make operation)).

6.23 The last line of a DIR/text/ file is ignored.

Only complete lines ending with "newline" are copied. The last line in the **DIR/text/** file most likely lacks a terminal "newline".

6.24 No CONFIRM requests are sent to moderators.

Assuming that the user initiated the subscribe request, got a "confirm" request, and replied correctly, there are two possible causes for the problem: Either the list is not subscription moderated (in this case the user is subscribed and received a note saying so) or the list is subscription moderated but no moderators have been added (ezmlm-manage(1) sends out the request and doesn't mind that there are no recipients).

Check that the list is subscription moderated:

```
% cat DIR/modsub
```

If this fails the list is not subscription moderated. If it succeeds with a directory name with a leading "/", this is your "moddir". If not:

```
% cat DIR/remote
```

If this succeeds with a directory name with a leading "/", this is your moddir, otherwise the moddir is "**DIR/mod/**".

Check for moderators:

```
% ezmlm-list moddir
```

If there are none, this is your problem. If there are some, check the mail log to see what happened when the CONFIRM requests was supposed to have gone out. Assure correct ownerships for the moderator db:

```
% chown -R user moddir
```

For ~alias:

```
# chown -R alias moddir
```

Another possible problem is that you are trying to use the remote admin feature to subscribe a user, but you get no CONFIRM request. Usually, this is due to your SENDER address not being in the moderator database. The CONFIRM request went to the target address instead, since as far as ezmlm is concerned, you are a regular user.

6.25 Deliveries fail "temporary qmail-queue error"

Usually, this is due to a corrupted qmail queue (should affect all mail) or a corrupted ezmlm subscriber database (See 6.26 (How to deal with corrupted subscriber lists)).

6.26 How to deal with corrupted subscriber lists

Dan has made ezmlm very robust, but a subscriber list can still become corrupted due to e.g. disk errors. Usually, this will lead to a "temporary qmail-queue error" because an address does not conform to the standard format. Occasionally, two E-mail addresses are fused, e.g. "`addr1@hostTaddr2@host`". To diagnose and fix this type of error, disable deliveries (easiest is to "chmod 0 DIR/lock"), back up the contents of **DIR/subscribers/**, then:

```
% ezmlm-list DIR > tmp.tmp

        ( edit tmp.tmp to fix any problems )

% rm -rf DIR/subscribers/*
% xargs ezmlm-sub DIR < tmp.tmp
```

This will list all E-mail addresses, allow you to edit them, then re-subscribe them. Don't forget to re-enable deliveries.

6.27 Vacation program replies are treated as bounces by ezmlm.

Standard vacation programs do not reply to messages that contain a "Precedence: bulk" header. ezmlm-idx>=0.23 sets up lists with this header in **DIR/headeradd**. For older lists, use "ezmlm-make -+" or "ezmlm-make -e" to update them, or just add a "Precedence: bulk" line to **DIR/headeradd**.

6.28 Digests do not come at regular hours.

In the default setup, ezmlm-tstdig(1) determines if a new digest is due every time a message arrives to the list. Thus, even though ezmlm-tstdig is set to produce digests 48 hours after the previous digest, the digest will not be generated until a message arrives. If you'd like digests at a specific time each day, use crond(8) and crontab(1) to daily run:

```
% ezmlm-get DIR
```

6.29 Preventing loops from misconfigured subscriber addresses.

Occasionally, a subscriber address is misconfigured and automatically sends a message back to the list. Sometimes, the subscriber's setup has removed headers that ezmlm uses for loop detection or the generated messages has nothing in common with the send-out. To block such mail at the list, include the ezmlm-make(1) "-k" (kill) switch and add the offending address to **DIR/deny/** with

```
% ezmlm-sub DIR/deny badadr@badhost
```

ezmlm-unsub(1) and ezmlm-list(1) can be used similarly to remove or list the addresses. If your list is configured for remote administration (see 4.17 (How remote administration works)), and you are a remote administrator, you can add the address by sending mail to `list-deny-badadr=badhost@listhost`. Other subscriber database commands work as well for `list-deny`.

In other instances, a configuration error somewhere close to the subscriber creates a local mail loop throwing off messages to you. They are often bounces that are sent to the list address or to "list-help" due to configuration errors. Rather than accepting these, or the often resulting double bounces to "postmaster", just add a "|/path/ezmlm-weed" line first to **DIR/editor** or **DIR/manager**. This discards the bounce messages generated by the looping systems. ezmlm-weed(1) is also useful in other settings where excessive numbers of error messages are sent to the wrong address.

6.30 A user can subscribe and receives warning and probe messages, but no messages from the list.

ezmlm lists (ezmlm-idx>=0.31) remove "Received:" headers from incoming messages by default. This can be prevented with the ezmlm-send(1) "-r" switch. When the headers are propagated, especially sublist message may have many (15-20 or more), "Received:" headers. If there is a poorly configured sendmail host with a "hopcount" set too low, it will bounce these messages, incorrectly believing that the many "Received:" headers are due to a mail loop. The reason that administrative from the list do not bounce is that they have fewer "Received:" headers, since they originate from the sublist.

The message with all headers including the removed "Received:" headers can be retrieved from the list archive with the *-getv* command. The top incoming "Received:" header is added by qmail at the receipt to the list (or last sublist) host. This header is not removed, to allow the recipient to determine when the message reached the list.

7 Customizing ezmlm-make operation via ezmlmrc

7.1 Using ezmlm-make to edit existing lists.

With ezmlm-make(1) (from ezmlm-idx >=0.21) you can use the "-e" switch to edit existing lists. Invoke the ezmlm-make(1) command just as you would to create the list anew, but change the switches to reflect the desired change, and add the "-e" switch. ezmlm-make will accept preexisting directories and overwrite or remove files to change the setup. The message counter (**DIR/num**), digest counters (**DIR/dignum** and **DIR/digissue**), the key (**DIR/key**) and the message archive will not be affected.

If the list has been created or previously edited with ezmlm-make(1) from ezmlm-idx>=0.23, the list remembers (via **DIR/config**) the arguments and the switches. All you have to do is to use the ezmlm-make(1) "-+" switch and specify options you wish to change, or use the "-e" switch and specify all non-default options you'd like to use.

NOTE: ezmlm-make(1) "-e" will OVERWRITE any manual customizations you have made to text files other than **DIR/headeradd**, **DIR/mod-sub**, **DIR/sub-ok**, **DIR/info**, and **DIR/faq**. To make general customizations, please change **ezmlmrc(5)** (see 7.2 (What is ezmlmrc?) or read on) instead and use the "-c" switch as well. DO NOT use this option to change production lists without testing it on other lists first. Also, for some changes, removing or adding a flag is sufficient (see 17.1 (How do I quickly change properties of my list)).

7.2 What is ezmlmrc?

ezmlm-make(1) has a number of default switches that through **ezmlmrc(5)** have defined functions. These allow creation of many standard lists.

In addition, ezmlm-make(1) operation is fully customizable via modification of the template file, ezmlmrc(5) or .ezmlmrc. A default ezmlmrc(5) is installed in the ezmlm binary directory. The system administrator can install a system-wide default **ezmlmrc(5)** file in **/etc/ezmlmrc** (or symlinked from there) which overrides the file in the ezmlm binary directory. If the ezmlm-make(1) "-c" (custom) switch is used, ezmlm-make(1) will look for **.ezmlmrc** in the "dotdir", i.e. the directory in which the **.qmail-list** links are placed. This is usually a set directory for a given user/virtual domain (usually, the home directory for the user controlling the lists).

ezmlmrc(5) controls everything except creation of the list directory itself and the key used for cookie generation. The syntax of **ezmlmrc(5)** is documented in ezmlm-make(1), the ezmlmrc(5) man page, and in the ezmlmrc(5) file installed in the ezmlm binary directory. ezmlm-make limits its effects to within the list "dot" and "DIR" directories. In the "dotdir", only links to within "DIR" can be created.

7.3 Changing defaults for DIR/text/ files.

Copy the ezmlmrc(5) file from the ezmlm bin directory to **.ezmlmrc** in your **.qmail** file base directory (usually your home directory):

```
% cp /usr/local/bin/ezmlm/ezmlmrc ~/.ezmlmrc
```

The base **ezmlmrc(5)** file lives in the ezmlm binary directory, which may differ from "**/usr/local/bin/ezmlm/ezmlmrc**" if you do not have a default setup. If your system administrator has placed a ezmlmrc(5) file into the **/etc** directory, start with that one instead, as it is likely to already contain some useful local customization and comments.

Now edit **~/.ezmlmrc**. Find the tag corresponding to the text file you want to change, e.g. "</text/mod-request/>", and modify it appropriately. Some tags have conditional flags, so that succeeding text is copied only if specific switches are on/off. Thus, text succeeding "</text/file#rms/>" is copied into **DIR/text/file** if and only if the ezmlm-make(1) "-rms" switches are all used. For more info, see documentation in **ezmlmrc(5)** and the ezmlm-make(1) man page. To invoke a custom **.ezmlmrc** file, use the ezmlm-make(1) "-c" (custom) switch.

7.4 Changing default moderator directories.

See above. Edit the **.ezmlmrc** file to add a directory name to e.g. "</modsub/#s>". Also, you need to create that directory, and the subscribers subdirectory under it. NOTE: **DIR/mod/** is still required as the base directory for the message moderation queue.

7.5 Adapting ezmlm-make for virtual domains.

This is not necessary if you use qmail>=1.02 and ezmlm-idx>=0.32.

The problem with virtual domains is that ezmlm-make(1) by default puts the list name in **DIR/inlocal**. However, if the domain `host1.dom.com` is controlled by the user "virt", then the local part of the address for the list `list@host.dom.com` will be "virt-list", not "list". This is easily accommodated by putting a **.ezmlmrc** file in **~virt**/. In the "</inlocal/>" section of this file, enter "virt-<#L#>" instead of "<#L#>". Now, all lists created under **~virt** will be automatically set up correctly.

Similarly, if `host1.dom.com` is controlled by virt-dom1 and `host2.dom.com` by "virt-dom2", inlocal for list `list@host1.dom.com` should be "virt-dom1-list" and for `list@host2.dom.com` should be "virt-dom2-list". To accommodate this, put "virt-<#1#>-<#L#>" in "</inlocal/>".

Running:

```
% ezmlm-make -c ~virt/LIST ~virt/.qmail-dom1-list \
          list host1.dom.com
```

will produce a **LIST/inlocal** of `virt-dom1-list` by substituting the first part between two "-" (dom1) for "<#1#>". Two levels of dashes are accommodated, i.e. "<#2#>" will be replaced by the second part between two "-" (in this case empty (*Sic!*)). For more info, see ezmlm-make(1) and comments in **ezmlmrc**.

7.6 Setting up ezmlm-make for special situations.

Ezmlm-make is very flexible. There are only three sets of special command line switches: "-vV" for version info, "-cC" controlling the use of a custom file **.ezmlmrc** in the "dot" directory, and "-eE" for edit mode (i.e. reconfiguration of existing list setups). All other switches are soft, i.e. controlled through **ezmlmrc(5)**. Many switches, have special meanings via **ezmlmrc(5)** and are documented in the man page. Any other switches can be used for customization (*NOTE: we may use switches other than "-xyz" for specific purposes in future versions.*) The "-xyz" switches will always be available for your use, with the "-x" switch being configured for some demo/special features in the distributed **ezmlmrc(5)**. You can use them for anything you like. They are by default off=false. The complement of these switches is "-XYZ" (by default on=true). You can use these to cause specific changes in the list setup if a given switch is used. For an example, see the "-x" switch as used and documented in the default **ezmlmrc(5)** file. The switches "-aip" are set by default to be backwards compatible with ezmlm-0.53. Other switches are "off" by default.

Switches "-a-z" and "-A-Z" take no arguments. Switches "-0" and and "-3-9" take arguments. When the ezmlm-make(1) "-+" switch is used, the current settings for all these switches are read from the list's **DIR/config** (if available).

8 Restricting message posting to the list.

8.1 Requiring the list address in To:/Cc: headers.

SPAM or junk mail is usually sent by mailing a single message to a large number of (unwilling) recipients. As such, it usually does not contain the E-mail address of all recipients (remember, junk mailers pay for these address lists). By rejecting messages that do not have the list address in the To: or Cc: header(s) a large fraction of spam to the list can be filtered out.

This filter function is activated by default, but will work only if you specify the list directory on the ezmlm-reject(1) command line. To disable this restriction, remove the "DIR" argument from the ezmlm-reject(1) command line, or add the "-T" switch.

By default, this error is logged, and an error message is sent to the sender. Since virtually all the failures will be SPAM and virtually all spam has a faked SENDER, most of these error messages will go to the postmaster. Thus, you may want to use the ezmlm-reject "-q" switch (quiet) to suppress the sender notification.

8.2 Rejecting messages sent from other mailing lists.

ezmlm automatically detects are rejects messages that are sent from other ezmlm mailing lists. Some other mailing list managers do not use a rigorous mechanisms to verify subscribers. Thus, it is possible to subscribe an ezmlm list address to such a mailing list. You can easily block such a list by adding the address to the "deny" if you use the ezmlm-make(1) "-k" option. However, you can also configure ezmlm-reject(1) to reject messages based on specific headers placed into **DIR/headerreject**. A set of headers which will catch mailing list managers known to us are listed in the ezmlm-reject(1) man page. To activate this option, you must specify the "-h" switch and **DIR** on the ezmlm-reject(1) line in **DIR/editor**. Naturally, you can make this the default by editing ezmlmrc(5) (See 7 (Customizing ezmlm-make operation)).

8.3 Restricting posts based on the Subject line.

ezmlm-reject(1) is by default configured to reject posts with empty subject ("-s" switch) or with a subject that consists of only an administrative command word ("-c" switch), such as "subscribe". To remove these restrictions, use the ezmlm-reject(1) "-S" and "-C" switch, respectively. You can also into **DIR/editor** before the ezmlm-send(1) line add:

```
| grep -i 'subject:' | grep -if DIR/bad_words >/dev/null && \
        {echo "bad words found"; exit 100; }
```

to reject messages that have a line matching "Subject:" followed by any bad word listed in **DIR/bad_words**.

8.4 Restricting the size of posts.

If the "DIR" argument is specified on the ezmlm-reject(1) line in **DIR/editor** and **DIR/msgsize** exists and contains a number (in bytes) greater than "0", then any posts with a body larger than the number specified is rejected. The maximum message size can optionally be followed by ":" and a minimum message body size in bytes. For moderated lists, messages that are too large are rejected and not sent to the moderators. This feature can be used to prevent the posting an entire digest to the list by setting **DIR/msgsize** slightly below the message size set in your ezmlm-tstdig(1) innovation (if any). A minimum size can catch a few administrative request sent to the main list, but is otherwise not that useful. To always configure your lists with a message size restriction, add to **ezmlmrc(5)**:

```
</msgsize/>
max:min
```

The ezmlm-make(1) "-x" switch adds this with 40000:2.

8.5 Restricting posts based on MIME content-type.

ezmlm-reject(1) will look for **DIR/msgsize**, **DIR/mimereject**, and **DIR/mimeremove** if the "DIR" argument is specified ("DIR" can be left out to conserve resources on lists that do not use these features). *Note: The "DIR" argument is also required for the the To:/Cc: list address restriction (see 8.1 (Requiring the list address in To:/Cc: headers)).* If the message contains MIME parts that are of a content-type listed in **DIR/mimereject** they are rejected. If the message is a simple MIME message of a content-type listed in either **DIR/mimereject** or **DIR/mimeremove** it is also rejected.

There is currently no ezmlm-make(1) switch for **DIR/mimereject**, but it can easily be configured by editing **ezmlmrc(5)**. The ezmlm-make "-x" switch configures **DIR/mimeremove** (see 9.6 (mimeremove)) for a list of content-types). Messages consisting solely of these content-types (rare) will be rejected, and the corresponding MIME parts of composite messages will be removed.

8.6 Restricting posts to list subscribers.

Use message moderation. As an alternative, implement a check against SENDER by using ezmlm-issubn(1). The latter is easily defeated by faking SENDER. Also, it prevents posts from legitimate subscribers that are subscribed under a different address than the one they send from. Nevertheless, it may be useful in some situations. Add:

```
|/usr/local/bin/ezmlm/ezmlm-issubn 'DIR' 'DIR/digest' 'DIR/allow' ||
   { echo "Sorry, you are not allowed to post to this list.";
     exit 100; }
```

ALL ON ONE LINE to **DIR/editor** before the ezmlm-send(1) line. "DIR" is the main list directory. If your ezmlm binaries live in a different directory, change the ezmlm-issubn(1) path accordingly. If you would like denied posts to be dropped silently rather than bounced, change the exit code to 99.

See 7 (Customizing ezmlm-make operation) if you want your lists to have some of these features by default or set by specific ezmlm-make(1) switches. The ezmlm-make(1) "-u" switch by default sets up restrictions this way.

If you do not want to allow digest subscribers to post, remove **DIR/digest/** from the ezmlm-issubn command line. To allow posts from an address that is not a subscriber, simply add it to the addresses in **DIR/allow/**:

```
% ezmlm-sub DIR/allow address@host
```

The "allow" database can be manipulated remotely by sending mail to `list-allow-subscribe@listhost`, `list-allow-unsubscribe@listhost`, etc. If configured for the list, the "-list" command for remote administrators will work for the "allow" database as well.

Please note that this setup is not secure, as it is easy to modify the envelope SENDER. For more secure options, see 8.7 (Restricting posts to an arbitrary set of E-mail addresses (higher security option)).

8.7 Restricting posts to an arbitrary set of E-mail addresses (higher security option).

The easiest way to achieve this is to simply set up a message moderated list, and add all the e-mail addresses to the moderator db. Use a custom location, if you want a different set of moderators for subscription moderation/remote admin. If a "moderator" posts, only s/he will get a confirmation request. If anybody else posts, the post will be sent to all moderators.

To directly bounce posts from SENDERs not in the database, use the ezmlm-store "-P" (not public) switch. This is more secure than a simple ezmlm-issubn(1) construct, since faking SENDER to a moderator address will result in a confirmation request to that moderator (which s/he will reject/ignore), rather than a direct post. The draw-back is that each post has to be confirmed, but with the speed of ezmlm the request will arrive immediately after the post is made, so the overhead should is The best choice depends on your particular needs in the trade-off between security and convenience.

Setting a list up in this way with only the owner's address gives you a pretty safe owner-only list.

8.8 Completely restricting posts.

To completely prevent posting (for instance a message-of-the-day list), set up a normal list, and just remove **~/.qmail-list** and **DIR/editor** altogether. Make posts from the shell, or from shell scripts or crond, by simply piping a (complete) message to ezmlm-send(1):

```
% /usr/local/bin/ezmlm/ezmlm-send DIR < message
```

NOTE: This can be done by any user with write access to files within the list directory, so make sure your file modes are set correctly. The ezmlm-send(1) path may need to be changed to match your ezmlm binary directory. It's also a good idea to not allow others to read your list directory and **DIR/subscribers/** and other address lists.

8.9 A general solution to restricting posts based on SENDER.

As discussed above, the security afforded by SENDER checks is minimal, but nevertheless sufficient to keep out most spam and garbage. However, some subscribers post from e-mail addresses other than their subscription address, and users tend to become unfriendly when their posts are denied even though they are subscribers. This is a general solution to this problem which has minimal overhead for the list owner and is essentially completely transparent to the subscriber.

Set up the list with ezmlm-gate(1) in **DIR/editor** in place of the ezmlm-send(1) line. To the ezmlm-gate(1) command line add the list directory twice, then a digest directory **DIR/digest/** (if it exists), then **DIR/allow/**. Create **DIR/modpost**. Add the list owner as a message moderator.

With this setup, any message from a SENDER that is a subscriber of the main list, the digest list or added to **DIR/allow/**, will be posted directly, others will be sent to the list owner for approval. If the list wants to automatically approve posts from that address in future (e.g. it is an alias for a subscriber) s/he just adds it to the database in **DIR/allow/**. If the owner wants to approve this post, but not necessarily future posts from that address, s/he just accepts the message. To reject the message with a comment is equally easy. If the owner wished to have the option to silently ignore posts (and not have the SENDER notified that the post timed out), just add the ezmlm-clean(1) "-R" switch in **DIR/editor** and **DIR/moderator**.

In this way, the normal subscriber is always happy and the "behind the scenes" work of the owner is minimalized.

ezmlm-make creates lists with this setup if you specify the "-u" switch in addition to the "-m" switch:

```
% ezmlm-make -mu ~/list ~/.qmail-list joe-list host
```

If you omit the "-m" switch, the setup will reject posts from non-subscribers that are not in the "allow" database. ezmlm-both(1) uses a set of similar ezmlm-make(1) invocations to create a list with digest, optionally making you a remote admin, list owner, and subscriber to both lists.

9 Customizing outgoing messages.

9.1 Adding a trailer to outgoing messages.

Put the text in **DIR/text/trailer**. The text is NOT copied to the archived version of the message. This works also for sublists. Tags "<#h#>", "<#l#>", and "<#n#>" are replaced by the list host, local name, and current message number, respectively.

9.2 Adding a subject prefix to outgoing messages.

Put the exact text in **DIR/prefix**. You can include the message number assigned to the post in the list archive by adding the "#" character in the text in **DIR/prefix** (example: put "lsqb;listname-#rsqb;" in **DIR/prefix**). ezmlm does not modify the subject other than by prefixing it with the prefix. ezmlm knows about rfc2047 encoded subject and can detect a prefix within an encoded word. However, ezmlm will not modify the subject itself. It will add a prefix only of none has been added before. A consequence of this is that a message will have the message number prefix of the first message in the thread rather than a prefix with the number of the message itself. The entire thread can always be retrieved with a message to `list-thread-x@host`, where "x" is the number in the prefix.

We recommend against using the prefix feature and strongly against the message number prefix. If you use it, make sure you understand the drawbacks, of message modification and subjects that change between message and reply. ezmlm can deal with this, but other programs may not be able to.

Sublists ignore **DIR/prefix**.

If you add a prefix, especially if you previously added it by other means (procmail, etc.), use ezmlm-idx to re-index the archive. Due to the way ezmlm-get(1) does threading from the subject, it works best if you use exactly the same prefix as you did before.

9.3 Adding a header to outgoing messages.

Put the exact header text as a line in **DIR/headeradd**. Thus, if you'd like a "Precedence: bulk" header added to outgoing messages, put a line "Precedence: bulk" into **DIR/headeradd**. This particular header is already added via the default ezmlmrc(5). Any modifications you wish to be active for all future lists should be made via modification of ezmlmrc(5) (see 7 (Customizing ezmlm-make operation)). As of ezmlm-idx-0.32, the following tags can be used in **DIR/headeradd**, and will be substituted: <#n#> for the current message number, <#l#> for the local part of the list (this will be the digest list for digests), <#h#> for the host part of the list name. These substitutions are done at the time of message delivery, in contrast to the "capital letter" tags substituted by ezmlm-make(1) when the list is set up.

rfc2369 suggests putting unsubscribe and other list access information into special headers. This should be in the form of a URL, mainly because many mailing list mangers require commands in subject or body. ezmlm already provides the "Mailing-List:" header with the list -help address. A message to this address gives all other information on list commands. ezmlm-idx>=0.32 adds the "List-Help" and "List-Unsubscribe" headers by default to stimulate MUA support for this feature.

To add your own rfc2369 headers, simply put them into **DIR/headeradd**. For instance:

```
List-Post: <mailto: listlocal@listhost>
```

9.4 Adding a message number header.

A sequence header may be useful for users whose systems don't pass on the "Return-to:" header to the MUA.

Use **DIR/headeradd** with a header of the type "X-Sequence: <#n#>".

Bounced messages are identified by their local message numbers, i.e. when ezmlm sends you a message about which messages bounced, it refers to the message number of the sublist. To be consistent with these numbers, and a local sublist archive, use **DIR/sequence** on the sublist, not the main list. To get consistent message numbering in digests, digest have the message number of the first message in the digest.

ezmlm-idx tries to make message numbering problems with sublists a little easier: sublists use the incoming message number, but only when the sublist is not archived and not indexed. This restriction is necessary for security reasons. Otherwise, an attacker could wreak havoc in the local message archive by sending messages with faked message numbers in the SENDER.

9.5 Removing headers from outgoing messages.

Put the header up to, but excluding the ":" in **DIR/headerremove**.

9.6 Removing MIME parts from messages.

ezmlm-idx>=0.30 can strip parts from composite mime messages based on content type. Just put the appropriate content-types such as "text/ms-word" or "text/html" into **DIR/mimeremove**. This is automatically configured when using the ezmlm-make(1) "-x" switch.

9.7 Limiting "Received:" headers in outgoing messages.

Sendmail still is being used on the majority of mail hubs. Sendmail has very primitive loop detection, bouncing messages based on excessive "hopcount". The "hopcount" is determined by counting "Received:" headers. ezmlm by default propagates "Received:" headers to facilitate message tracking. Thus, messages, especially from a sublist, can have a number of "Received:" headers that exceeds the "hopcount" set on poorly configured sendmail hosts. Subscription confirmation requests, warning, and probe messages have fewer "Received:" headers. Thus, a user may be able to receive these, but not (some of the) list messages. Of course, the best is to correct the configuration on the bouncing host, but this is often under the control of neither list owner nor user.

To compensate for this problem, ezmlm-send(1) of ezmlm-idx->=0.313 by default removes all "Received:" headers except the top one. They are still written to the archive, an can be retrieved from there using the "-getv" command. To cause ezmlm-send(1) to pass on all the "Received:" headers, use the ezmlm-send(1) "-r" switch.

9.8 Setting "Reply-To: list@host".

This is not recommended, since it leads to dissemination via the list of messages returned from bad autoresponders and MTAs. Also, it may lead to public replies to the list where personal replies were intended. In addition, the original "Reply-To:" header is lost. If you do want to add a reply-to list header, put "reply-to" into **DIR/headerremove**, and "Reply-To: `list@host.dom`" into **DIR/headeradd**.

9.9 Configuring the list so posts are not copied to the original sender.

For most mailing lists, you want all subscribers, including the sender of a particular message, to get all messages. This way, the sender sees that the message reached the list. For small lists, such as a project group, it may be annoying for the members to receive their own posts.

ezmlm-send(1) can be configured to exclude the sender from the recipient E-mail addresses if configured with the "-C" switch. To add this switch, edit the ezmlm-send(1) line of **DIR/editor**.

9.10 Customizing ezmlm notification messages.

Most of ezmlm's more commonly used messages are stored in **DIR/text/**. These messages can be edited manually for a list once it is set up, or on a global basis via modification of **ezmlmrc(5)**. The messages may also be edited via E-mail by remote administrators (remote admin must also be enabled - ezmlm-make switch "-r") after the list is established by creating the list using the ezmlm-make(1) "-n" (new text files) (see 4.31 (How text file editing works) and see 7 (Customizing ezmlm-make operation)).

The most useful messages are **DIR/text/sub-ok** (and for subscription moderated lists **DIR/text/mod-sub**) for new subscriber information (such as the traditional “welcome” message, or a list charter or list posting rules/guidelines); **DIR/text/unsub-nop** is useful for messages to frustrated users unsuccessful in their unsubscribe attempts; **DIR/text/help** for general help information in reply to `list-help@host` or unrecognized commands, **DIR/text/bottom** for inclusion at the bottom of virtually all ezmlm messages; **DIR/text/mod-help** for moderator information; **DIR/text/trailer** for a (few) line(s) at the bottom of each post; **DIR/text/digest** for information in the “Administrivia” section of digests.

9.11 Specifying character set and content-transfer-encoding for outgoing ezmlm messages.

All ezmlm replies, except errors handled directly by qmail, can be sent in any character set and optionally with quoted-printable or base64 content-transfer-encoding. **DIR/text/** files are always 8-bit files, but even though qmail has no problems with 8-bit mail, other MTAs and MUAs do. Problems due to this can be avoided by assuring that outgoing ezmlm messages are 7bit by using the appropriate content-transfer-encoding.

To specify a character set, put the name in **DIR/charset** (default: us-ascii). To specify quoted-printable or base64 content-transfer-encoding, add “:Q” or “:B” after the character set name in **DIR/charset**.

10 Customizing archive retrieval.

10.1 Specifying the format for retrieved messages.

Add a format (f) specifier after the archive retrieval command:

```
list-getf@host
```

where “f” is “r” for rfc1153 format, “m” (mime; default) for MIME multipart/digest with subset of ordered headers, and “v” (virgin) MIME multipart/digest, i.e. with all headers retained from the archive, and “n” (native) the same as “v” except that no threading is performed and messages are returned in numerical order. Under some circumstances, it may be preferable to have a digest in “multipart/mixed”. The “x” (mixed) format is identical to “m” except for this header.

For ezmlm-cron(1), just suffix the format code to the digest code.

10.2 Specifying the default format for digests and archive retrieval.

The ezmlm-get(1) “-f” switch can be used to change the default format (MIME with removal of less relevant headers) to other formats. The format specifiers are the same as for individual archive retrievals (see 10.1 (Specifying the format for retrieved messages)).

10.3 Limiting the number of messages per -get/-index request.

By default, a single -get request returns a maximum of 100 messages, and a single -index request 2000 subjects entries (20 files of 100 subjects entries each). This can be changed by editing MAXGET, and MAXINDEX in **idx.h** and recompiling. Remember to edit **text/bottom**, **text/bounce**, and **ezmlmrc(5)** to reflect these changes so that your users won't get confused.

11 Restricting archive retrieval.

11.1 Restricting archive access to subscribers.

If you use ezmlm-get(1), archive retrieval can be restricted by using the ezmlm-make(1) "-g" (guard archive) switch. This in turn sets ezmlm-get(1) up with its "-s" switch, allowing access only to addresses that are subscribers of the list, or of the digest list, or that are present in an extra address database stored in **DIR/allow/**. Addresses can be added remotely by mailing `list-allow-useralias=userhost@listhost`. Other commands, such as "subscribe" work as expected. As you can see, the different programs have many options and ezmlm-make(1) organizes most of them into the most useful sets to make it easier. Don't hesitate to look at the ezmlmrc(5) man page and man pages for individual commands. There are many useful options to more finely tune your lists to your taste. Via modification of ezmlmrc(5) you can make your favorite options the default!

Since ezmlm-get always sends the reply to SENDER, this assures that only subscribers can get archive excerpts. Since SENDER is easily faked, anyone can still request archive info (and drain system resources), but replies go only to subscriber E-mail addresses. The **DIR/allow/** database can be used to manually add addresses that should be given archive access, but are not subscribers. This may be an address of a subscriber who posts from an address other than his or her subscription address.

11.2 Restricting available archive retrieval commands.

If you want to disable all archive retrieval except digest creation, simply add the "-C" command line switch to the ezmlm-get(1) line in **DIR/manager**. If you don't want digest creation via trigger messages and **DIR/manager**, but use other means to created digests, you can remove the ezmlm-get(1) line from **DIR/manager**.

11.3 Restricting archive retrieval to moderators.

If **DIR/public** does not exist, ezmlm-manage(1) and ezmlm-get(1) modify their behavior. They disallow user requests, but for remote administration lists, honor moderator requests. Thus, for a remote admin list without **DIR/public**, only subscription moderators or remote administrators can receive archive retrievals and only remote administrators can subscribe and unsubscribe user addresses.

If you'd like this restriction of archive retrieval with maintained user-initiated ezmlm-manage(1) subscription functions, use the ezmlm-get(1) "-P" (not public) switch, and retain **DIR/public**.

11.4 Allowing archive retrieval from a non-public list.

A non-public list lacks **DIR/public**. ezmlm-manage(1) will reject user requests for (un) subscription and for archive retrieval. The restriction on archive retrieval can be removed with the ezmlm-get(1) "-p" (public) switch.

12 Customizing digests.

12.1 Setting up a digest list.

Digests are integrated with normal ezmlm lists if you use ezmlm-idx>=0.30. Just add the ezmlm-make(1) "-d" switch to your list setup. To add digests to an existing list created with ezmlm-idx>=0.23 use:

```
% ezmlm-make -+d DIR
```

For ezmlm-0.53 or older lists, you just need to re-specify also other switches and the other ezmlm-make(1) arguments.

12.2 Generating daily digests.

The easiest way to generate trigger messages is to use crond(8) and execute ezmlm-get(1) daily. To do this, create the list with:

```
ezmlm-make -d dir dot local host
```

and add a line to your crontab file:

```
30 04 * * * ezmlm-get dir
```

and execute crontab(1). This will generate a digest each day at 04:30 am. In addition, a digest will be generated at any time when the latest post makes it more than 30 messages or more than 64 kbytes of message body since the latest digest. If you do not want these extra digests, edit **DIR/editor** and remove the ezmlm-tstdig(1) and ezmlm-get(1) lines.

If you do not need the digests to go out at a particular time, use the standard setup, but edit **DIR/editor** to put "-t 24" on the ezmlm-tstdig(1) line instead of the default "-t 48" for 48 hours. This is even easier. You can modify all parameters by editing **ezmlmrc** or by using the ezmlm-make(1) "-4" argument when creating/editing the list. This is described in the ezmlm-make(1) man page, and the options etc, are described in the ezmlm-tstdig(1) man page.

12.3 Generating the first digest.

If you want the first digest to start with issue 1 and the first message in your archive, no special action is required.

If you want the first digest to start at message 123 and you have shell access, put '122' into **DIR/dignum**.

If you want the next digest to start at message 456, you can always edit **DIR/dignum** to contain '455'. If you want the next digest to be named issue 678, put '677' into **DIR/digissue**.

12.4 Adding standard administrative information to digests.

The text in **DIR/text/digest** is copied into the "Administrivia" section of the digest. This information can be customized on a system-wide basis by editing **/etc/ezmlmrc**, on a user-wide basis by editing **~/.ezmlmrc**, or for the list by directly editing the **DIR/text/digest** file, or by a remote administrator by editing the file via e-mail, if the list has been set up using the ezmlm-make(1) "-nr" switches (see 4.31 (How text file editing works)).

12.5 Controlling the digest format.

You can control the default format that ezmlm-get(1) uses for its output by using the "-f x" switch. For individual digests triggered by mail or other archive access, add a format specifier after the digestcode:

```
list-dig.codef@host
```

For example:

```
joe-sos-dig.gagax@id.com
```

where "x" is "r" for rfc1153 format, "m" (default) for MIME multipart/digest with a subset of headers, "v" for virgin MIME multipart/digest, i.e. with all headers retained from the archive, "n" produces format similar to "v", without threading and with messages in numerical order. The "x" format is identical to the default "m" format, but the digest content-type is "multipart/alternative" rather than "multipart/digest". This helps with a pine bug if you are using quoted-printable/base64 encoding of ezmlm messages.

With digests triggered directly from crond(8), just use the "-f" format specifier:

```
ezmlm-get -fx DIR
```

The same switch can also be used for standard digest triggering from **DIR/editor**. Just add the "-fx" switch to the ezmlm-get(1) command line there. Edit **~/ezmlmrc** to assure that such customizations will be used for future list creations/edits.

12.6 Customizing bounce handling.

The time out for bounce messages is normally 11.6 days. This means that a bad address will take longer that 3 weeks to be removed. Usually, this delay is desirable. After all, it is much worse to remove a subscriber just because the address had temporary problems that to send a few extra messages and receive a few extra bounces.

However, for large lists, bounce handling can consume a considerable amount of resources. To decrease the load, remove all ezmlm-warn(1) lines from the **DIR/editor**, and **DIR/manager** files. Instead, execute:

```
/path/ezmlm-warn DIR
/path/ezmlm-warn -d DIR
```

daily during off-peak hours via a cron script. The second line can be omitted if you are not using the digest capability of the list.

This should not be necessary for ezmlm-idx>=0.32. That version adds much more efficient bounce handling, making this type of modification usable only for extremely large lists with many bad addresses (unusual for ezmlm lists) and for hosts that are working near the limit of their capacity (where shifting some qmail load to off-peak hours is worth the effort).

In addition, you may want to reduce the time out for bounces from 11.6 to a lower number of days, e.g. 5. To do so, add "-t 5" to the ezmlm-warn(1) command line.

If you start with a list from a list manager that does not have bounce handling, chances are that you have many bad addresses in your list. You can always execute:

```
/path/ezmlm-warn -t0 DIR
/path/ezmlm-warn -d -t0 DIR
```

to move bounce handling one step forward per execution. Users whose mail has bounced will be sent a warning. Users for whom the warning message has bounced will be sent a probe.

13 Remote administration.

13.1 How can I remotely add moderators, subscriber aliases, etc?

On any list, the **DIR/allow/** database can be manipulated remotely via mail to `list-allow-subscribe@listhost`, etc. The rules for adding/removing/listing addresses to this database are the same as for the main list. Thus, if a user on an open list wants to be able to post from `alias@al.host.com` s/he can send a message to `list-allow-subscribe-alias=al.host.com@listhost` and reply to the confirmation request. Now, s/he can post from this address even on a subscriber-only list and even though the address is not a real subscriber.

It can be confusing to some users that you use "subscribe" here, but you don't get any messages. If you explain to them that this is just another collection of addresses they will understand. You can also send the initial message on their behalf. If you are a remote admin, you can even complete the transaction adding the alias without subscriber participation.

Addresses can also be unsubscribed from the "allow" database. However, there is usually no good reason to do so.

If configured, the **DIR/deny/** database can be manipulated, but only by remote administrators, by mail to e.g. `list-deny-baduser=badhost@listhost`. Normal users cannot access this database.

To remotely administrate the **DIR/mod/** databases (i.e., without shell access), you need to set up a non-public, remotely administered list which "resides" within the **DIR/mod**. *Please carefully consider the implications of making it possible to remotely add, remove, and list moderators. In many circumstances, this is dangerous.*

After setting up your list with the specific functionality you need, use the following command for **DIR/mod/**:

```
% ezmlm-make -ePrIAl ~/list/mod ~/.qmail-list-mod joe-list-mod host
```

The '-l' flag is not necessary, but makes it easier to administrate your moderator database by permitting the "supermoderator" to see who is on the list.

The new list does not have a key. Using the key from the main list is inadvisable. Instead, create a dummy list, copy the key from this list to your "moderator" list:

```
% cp ~/DUMMY/key ~/DIR/mod/key
```

Erase the dummy list. Also, posts to this list should not be allowed. Erase the **~/.qmail-list-mod** and **~/DIR/mod/editor**. Then add the remote administrator of the "moderator" list:

```
% ezmlm-sub ~/list/mod/mod supermod@superhost
```

The "supermoderator" can now remotely administrate the moderators of the main list.

13.2 Moderating posts from a secondary account.

Request for moderation of posts can be forwarded to any address and acted on from that address. By default, all post moderation requests have subjects starting with "MODERATE for" followed by the list name.

13.3 Moderating subscription from a secondary account.

Requests for moderator approval of user subscribe requests can be forwarded to any address and acted on from that address. All subscription moderation requests have subjects starting with "CONFIRM" (or "CONFIRM subscribe to listname", since "CONFIRM unsubscribe from listname" is sent to the moderator only in reply to a moderator-initiated request on a list with remote admin).

Remote administration (initiation by the moderator of (un)subscribe requests on behalf of a user) CANNOT be initiated from an account that is not listed in the moderator database. If such attempts are made, these will be treated as regular requests, resulting in a confirm request to the user (which includes a copy of the initial request, revealing the moderator's address to the user). The user reply to a confirm request will on a non-moderated list result in the addition of the user address to the subscriber list, and in a moderated list a CONFIRM request to all the moderators. Replies to unsubscribe confirm requests always result in the removal of the address, without moderator intervention (except in some cases when the ezmlm-manage -U switch is used (see below)). With this caveat, moderation and remote administration can be done from a secondary address.

For the subscription moderator to temporarily use a different address, s/he needs to forward all "CONFIRM" messages to the new address. For a permanent move, it is better to remove the old moderator address and add the new SENDER address to allow moderator-initiated (un)subscribes without user intervention from the new address (of course, the list has to be configured for remote administration with **DIR/remote**).

13.4 Automatically approving posts or subscriptions.

Sometimes, it may be desirable for the moderator to automatically approve all moderation requests. This may be appropriate for a single moderator of a "civilized" list when away for the week.

Set up your client to auto-reply to the "Reply-To:" address for all messages with subjects "CONFIRM subscribe to listname" or "MODERATE for listname". Beware that this can be used by malicious people to trick your account to send mail anywhere. In practice, this should not be a problem. If you are worried, forward the messages to a (trusted) friend and ask him/her to appropriately reply to the requests.

13.5 Allowing remote administrators to get a subscriber list.

Access to the subscriber list is sensitive. Thus, this option is disabled by default. The ezmlm-manage(1) "-l" command line switch enables this option, but will send a subscriber list only to a moderator's address. This allows a moderator to also initiate a subscriber list retrieval from a secondary account (i.e. one to which the moderator's mail is delivered, but for which SENDER is not a moderator). The latter option does not decrease security, as it is trivial to fake SENDER (see 21 (Ezmlm-idx security) for a discussion of ezmlm-idx security aspects).

For maximum subscriber list security, do not enable this feature. To enable this feature by default, just modify **ezmlmrc(5)** (see 7 (Customizing ezmlm-make operation)).

13.6 Allowing remote administrators to retrieve or search a subscription log.

This is restricted and works as the subscriber list, since it contains information of equal sensitivity. To receive the entire log, mail `list-log@listhost`. See 4.30 (Howto get a subscription log) for more details on the reply format. As of ezmlm-idx-0.32, the subscription log also contains the From: line contents from the user's subscribe confirmation. This usually contains the user's name and can be helpful if the user cannot recall or determine the subscription address. To make life easier for the remote admin, ezmlm-idx-0.32 also supports searching the log, using exact matches for alphanumerics and "_" as a wild card character. Thus, to find records matching "Keith John*", the remote admin can mail `list-log.Keith_John`. See 4.30 (Howto get a subscription log) for more information.

13.7 Allowing users to get a subscriber list.

If you want any user to be able to get a subscriber list, you can set up a separate link to **DIR/list** and then put in a script using ezmlm-list (See 4.28 (adding your own commands) for more info.) . The authors strongly urge against this, since a common method for spammers to get valid E-mail addresses from mailing lists is to exploit unrestricted -list commands. A subscriber with questions about who is on the list should contact the `list-owner@host`. A subscriber wishing to confirm that they are still on the list can just send a message to `list-subscribe@listhost`, and reply to the confirm request. The following message will be a "ezmlm response" if the user was already a subscriber, and a "WELCOME to listname" if s/he was not.

13.8 Changing the timeout for messages in the moderation queue.

Put the time, in hours, into **DIR/modtime**. This value may not exceed the range of 24-120 h set at compile time by the defines in **idx.h**.

13.9 Finding out how many messages are waiting for moderation.

```
% ls -l DIR/mod/pending
```

and count lines with the owner execute bit set (rwx——). Others are remnants from failed ezmlm-store runs (ignore - ezmlm-clean(1) will remove them).

There is currently no way to see this remotely, although you could easily install a script mailing the 'ls' output in response to a message to e.g. **list-chkqueue@host**. (See ezmlm-check(1) and 4.28 (adding your own commands) for examples.)

13.10 Using the same moderators for multiple lists.

Set up a moderator dir:

```
% mkdir /path/moddir /path/moddir/subscribers
% touch /path/moddir/lock
% chown -R user /path/moddir
```

For alias:

```
# chown -R alias /path/moddir
```

For example:

```
% mkdir ~joe/mods ~joe/mods/subscribers
% touch ~joe/mods/lock
```

Then for the lists, put **/path/moddir** into **DIR/modsub** (for moderation of subscribes), **DIR/remote** (for remote admin if **DIR/modsub** does not exist), and **DIR/modpost** (for moderation of messages).

For example:

```
% echo "/home/joe/mods" > ~joe/DIR/modsub
```

NOTE: The path must start with a '/'.

13.11 Using different moderators for message and subscription moderation.

Proceed as in the previous point, but set up two different moddirs. Naturally, one of these can be **DIR/mod/** (preferably the one for posts, to keep it cleaner). Then modify the appropriate files (**DIR/modpost** and **DIR/modsub**) to contain absolute paths to the correct moddir.

13.12 Setting up moderated lists with the list owner as the "super moderator" able to add/remove moderators remotely.

This is done by crating a list that has **DIR/mod/** as it's main list directory, then adding the "super moderator" to **DIR/mod/mod/** (see 13.1 (remotely adding moderators)).

If this is a common setup for you, you can write a simple script creating both lists (plus a digest list, if desired) with one simple action (see ezmlm-both(1) for an example).

13.13 Customizing ezmlm administrative messages.

Subject lines, and other ezmlm output for moderation are controlled by defines in **idx.h** and by files in **DIR/text**. To customize these, change **idx.h** and recompile or for **DIR/text** files, edit **ezmlmrc(5)** (see 7 (Customizing ezmlm-make operation)).

You can also configure the list to allow remote administrators to edit files in **DIR/text/** via E-mail (see 4.31 (How text file editing works)).

13.14 Manually approving a message awaiting moderation.

All you have to do is to pipe the corresponding message to "ezmlm-send DIR". Messages awaiting moderation are kept in **DIR/mod/pending/**. To find a particular file, grep the contents. Thus, to find a file from `user@host.dom`, try:

```
% grep 'user@host\.dom' DIR/mod/pending/*
```

(Depending on your setup, you may not have to escape the period.) Check the files for the owner execute ("x") bit. It is set on all messages queued successfully. Ignore other files!

To then accept the message (change the ezmlm-send(1) path if you've installed in a non-default directory):

```
% cat DIR/mod/pending/filename \
% /usr/local/bin/ezmlm/ezmlm-send DIR
```

Alternatively, use ezmlm-accept(1). It checks the 'x' bit, ezmlm-send(1) return codes, removes the file, etc.

For example:

```
% ezmlm-accept ~joe/SOS ~joe/SOS/pending/*
```

will accept all messages in the queue of the list in **~joe/SOS/**.

13.15 Manually rejecting a message awaiting moderation.

Simply deleting the file from **DIR/mod/pending/** will do it. If you want to notify the sender, just send him/her an E-mail. There is an easy way to get ezmlm-idx programs to do it for you: just wait and let ezmlm-clean(1) take care of it for you, once the message has timed out (number of hours settable within 24-240 in **DIR/modtime**; default 120).

14 Sublists.

A sublist is a list that receives its input from another mailing list, rather than from users directly. The sublist is just a regular subscriber of the main list. A sublist in e.g. Tasmania is very useful since only one message is sent from the main list and then the sublists servers all subscribers in Tasmania. Bounces and all administration is handled locally. The local sublist can have a digest, even though the main list may not. (See 4.24 (How sublists work) for more info on how sublists work).

14.1 Sublists of ezmlm lists.

To set up a sublist to an ezmlm list, just use the ezmlm-make "-5 mainlist@mainhost" switch. This will configure your list as a sublist to the `mainlist@mainhost` mailing list.

14.2 Sublists of non-ezmlm lists.

To set up a sublist to an ezmlm list, just use the ezmlm-make "-5 mainlist@mainhost" switch. This will configure your list as a sublist to the `mainlist@mainhost` mailing list. Since the main list may not use the "Mailing-List" header, you must identify another header that the main list adds to all messages. See the ezmlm-reject(1) man page for examples. Next, edit **DIR/editor** of your sublist and add a "-h *Listprocessor-Version:*" option to the ezmlm-send(1) line, but replacing "*Listprocessor-Version:*" with your mainlist header.

Now your list will accept only messages from `mainlist@mainhost` and with the header specified.

14.3 How to set up a cluster of list and sublists with standard databases.

ezmlm-0.53 allows sublists. The difference between a sublist and a main list is that the sublist requires that the SENDER of the message is the main list and that the message has a "Mailing-List:" header. Sublist messages have their own subscriber database and subscription mechanism, and use their own message number. This is very convenient if you want to create a private sublist. Since the subscribers have to interact with the appropriate sublist, it is difficult to administrate if you want to use it to distribute the load of a very large list, since users will have to address administrative requests such as unsubscribe to the correct sublist. Also, bounce messages refer to the sublist archive with sublist message numbers.

ezmlm-idx modifies this in several ways: First, the message number of the incoming message is used also for the outgoing message so that subscribers see the same message number no matter which sublist they get it from. For security reasons, this is enabled only if the sublist is NOT ARCHIVED. With this feature, bounce messages can refer the user to the main list archive instead, obviating multiple archives.

Second, ezmlm-split(1) can be used to forward administrative requests sent to the main list, to the appropriate sublist. Thus, subscribers interact only with the main list, and do not need to know which sublist that servers them. With bounce and administrative messages referring them to the main list, subscribers will usually be unaware of the sublisting.

To set this up:

- **create the main list**

  ```
  ezmlm-make dir dot local host
  ```

- **add an ezmlm-split(1) invocation**

  Before the ezmlm-manage(1) line in **DIR/manager** add:

  ```
  |/path/ezmlm-split dir
  ```

- **decide how to split the load**

  The main list sends to sublists and to any addresses not covered by the split table. You can split the load by domain ("geographically"), and any domain (including ") can be subdivided by "hash" by using different parts of the 0-52 range. Of course, you can also use hash alone. The request will go to the first row that matches, so although overlaps are not advisable (in case you later want to add sublists of switch to an SQL server-based system (see 5 ())), they have no negative effects. The domain for ezmlm-split can be the last TWO parts, i.e. "edu.wustl" to handle all *.wustl.edu subscribers. This is useful, but remember that the SQL version supports only one level.

An example:

```
domain:hash_lo:hash_hi:sublistname
edu:0:52:sub1@here.edu
com:0:26:sub2@there.net
com:27:52:sub3@some.com
:0:13:sub4@what.org
:14:39:sub5@what.org
```

As you can see, the entire "edu" domain is handled by `sub1@here.edu`. The "com" domain is about evenly split between `sub2@there.net` and `sub3@some.com`. Everything else is split so that approximately 1/4 goes to `sub4@what.org`, 1/2 to `sub5@what.org` and the rest falls through, i.e. is handled by the main list.

Why are there 2 sublists on the same host? This is in preparation of adding a host. It easy to just move the entire `sub5@what.org` list to a new host. All we have to do it to set up the new list, copy over the subscribers, and change the name in the split table entry.

To split the split the `sub5@what.org` load onto 2 lists requires a little more work. First, create a dummy split table in a directory "temp":

```
:14:26:new1@new.net
:27:39:new1@other.net
```

Next, split the subscribers of sub5@what.org into these 2 groups, as detailed in the ezmlm-split(1) man page. Create the two new lists, add the respective subscribers, and replace the `sub5@what.org` line with the two lines above.

To add a totally new domain, e.g. `jp:0:52:sub6@niko.jp` requires collection or subscribers from all lists that currently handle these subscribers, (the ones with blank domain in the example), re-splitting them, and adjusting the subscribers. Easiest here is to just unsubscribe the `sub6@niko.jp` subscribers to be from the other list with ezmlm-sub(1). Since that program will silently ignore any addresses that are not on the respective list, it will work fine.

- **Create the sublists**

  Use ezmlmsubrc which sets up a minimal non-archived sublist with bounce texts pointing to the main list:

  ```
  % ezmlm-make -Cezmlmsubrc -3mainlocal -4mainhost \
          DIR dot sub1local sub1host
  ```

- **subscribe the respective sublists to the main list**

  If you forget, the sublist will not get any messages to distribute. Add these addresses with ezmlm-sub(1) as subscribers to the main list.

A strong point of this system is that it is relatively simple and that only a fraction of the addresses are available to any given sublist. Thus, compromised security at a sublist threatens only the addresses and functions handled by that sublist.

As you can see, this works quite well, but it's not trivial to change the setup. If you modify it while the list is running, some subscribers may get duplicate messages or miss messages. Therefore, you should disable deliveries to the main list before the final step of the changes (removal of subscribers from old lists and adding new lists as subscribers to the main list). For most lists, this should work flawlessly, and some minimal planning and extra lines in "split" can markedly facilitate future expansion.

Another weak point is the authentication of messages between list and sublist. The requirements the sublist places on the message can be easily faked. This allows injection of messages at the sublist level as a way to circumvent moderation or other access control.

An associated disadvantage is that not even the main list has access to all the addresses. Thus, SENDER checks for archive access (relatively secure) and posts (relatively insecure) cannot directly be used. Also, sublist cooperation is required to determine the number of subscribers, or to access subscriber addresses for a purpose other than distribution of list messages.

14.4 Setting up an ezmlm list "cluster" of main list and sublists using a central SQL server.

This is a little more complicated. Instead of forwarding (un)subscribe requests to a sublist, all administrative requests are handled locally working against a central SQL database. All addresses are stored in the same table. Which addresses are served by a particular sublist is decided at the time of processing. In order to be compatible with how ezmlm works with normal databases, SQL-based ezmlm list clusters use 2 communication channels. One is the message itself, the other is the communication with the SQL server.

14.4.1 Advantages.

The main advantage is ease of administration. It is easy to add new sublists, temporarily work around a defective sublist, etc. Backups of addresses can be centralized. Subscriber-only restrictions are more easily enforced since the main list has access to all addresses. Also, the 2 band communication allows better monitoring of sublist function and better authentication between list and sublist.

14.4.2 Disadvantages.

Clearly, this is more complicated to set up. First, you need to run a SQL server. Second, you need to be able to reconfigure the list to take advantage of the ability to work around broken sublists and add new ones on-the-fly. Another disadvantage comes from centralization. Each sublist needs to have access to the address table. Thus, compromise of any sublist access credentials reveals all subscriber addresses. Since sublists handle their own bounces, sublists must also have DELETE access to the addresses. Thus, a sublist compromise allows the attacker to remove all addresses, not just those handled by the sublist. There are various ways around this. Some make administration much harder, others require special programming. For now, you have to trust your sublists. This is no concern if what you have is 5 local hosts that you'd like to share the burden of a few lists.

14.4.3 Setting up a list cluster using a SQL database.

Create the database

Use commands appropriate for the SQL server, and set up a user-id with appropriate access restrictions for the administrator of the list cluster. Minimally, this includes SELECT privileges to all tables, INSERT and DELETE privileges to address tables, and "*_name" tables, as well as INSERT privileges to all "*_slog" tables. We usually grant this user full access to the specific tables and use the same uid for the main list.

Create the tables for the list cluster

You need to specify the "table root", i.e. the name of the main list address table, which will be used as the name root for all other tables. Thus, with a table root of "list" the digest list subscriber log would be "list_digest_slog". This creates tables with the root "list" in the preexisting database "ezmlm". The "-d" switch will cause removal of any preexisting tables with the same name.

```
% ezmlm-mktab -d table | mysql -uuser -ppw -hhost -f ezmlm
```

The database server is on host "host" and "user" with password "pw" is assumed to have sufficient privileges to create the table. For other SQL interfaces than MySQL, the exact means of creating the tables from the table definition printed by ezmlm-mktab(1) will differ.

Create one sublist user per list

Minimal access needed is SELECT on all tables except "*_slog", INSERT on "*_slog", "*_cookie", and "*_mlog", and DELETE on the main address table and digest subscriber table "*_digest". Easiest is to use the list local name as the userid (YMMV). For MySQL, restrict to both user and host and use different passwords for different sublists.

ezmlm-grant(1) outputs statements to create the userid:

```
% ezmlm-grant -d db -t troot s_host s_uid s_pw | \
        mysql -uuser -ppw -hhost
```

Here, sublist user "s_uid@s_host" with password "s_pw" is granted minimal access privileges to the database "db" with table root "troot". The MySQL access info for this action is specified as before. Again, with an alternative SQL server, the manner in which the ezmlm-grant(1) output is used will differ.

Insert rows into troot_name, one per list

You need to decide how to split the load. The main list only sends to the sublists. If you split by domain ("geographically") you need one list with domain=". It will handle domains not handled by other list. Any domain (including ") can be subdivided by "hash" by using different parts of the 0-52 range. Do not overlap hash ranges, or some subscribers will get multiple messages. If you leave some out (or don't have a list with domain=") some subscribers will not receive posts. The main list should have hash=99. Sublists are entered with this hash.

An example:

```
   name         domain  hash_lo hash_hi Comment
main@host.com   ''      99      99      sends to sublists
sub1@here.edu   'edu'   0       52      all @..edu subscribers
sub2@there.net  'com'   0       26      about half of the @..com subscribers
sub3@some.com   'com'   27      52      the rest of the @..com subscribers
sub4@what.org   ''      0       13      About 1/4 of all other subscribers
sub5@what.org   ''      14      26      Another 1/4. Same list as above
sub6@host.us    ''      27      52      remainder.
```

As you can see, the " part is split in about 2 x 1/4 + 1/2. A list can occur several times, and the addresses served are the union of the entries.

Create the main list

We create this one with ezmlm-receipt(1) to take advantage of feedback logging ("-w") and for "-6" using the access credentials of our main list user:

```
% ezmlm-make -6 host::uid:pw:db:troot -w dir dot mainlocal mainhost
```

Create the sublists on the respective hosts

Here, we use a special "ezmlmrc" file to get all the bounce messages, etc, to point correctly to the main list:

```
% ezmlm-make -C/usr/local/bin/ezmlmsubrc \
        -6 host::s_uid:s_pw:db:troot -3mainlocal -4mainhost \
        subdir dot sublocal subhost
```

Repeat for each sublist.

Subscribe the sublists and feedback address to the main list

You need to do this at the main list, since the sublists are not allowed to insert into the address table (you can verify this!)

```
% ezmlm-sub -s subdir sub1@here.edu
% ezmlm-sub -s subdir sub2@here.net
% ezmlm-sub -s subdir sub3@some.com
% ezmlm-sub -s subdir sub4@here.edu
% ezmlm-sub -s subdir sub5@what.org
% ezmlm-sub -s subdir sub6@what.org
% ezmlm-sub -r subdir main-return-receipt@mainhost
```

Test access

From each sublist, try:

```
% ezmlm-list -a subdir
```

You should see all the above addresses. If something is wrong with the access info, you will be told so. Correct the setup. Subscribe a few addresses.

```
% ezmlm-list -a subdir
```

should again show you all addresses.

```
% ezmlm-list subdir
```

should show you only the addresses served by the particular sublist including the feedback address. *Note:* If you later modify the "*_name" table and use the message number interval, ezmlm-list(1) may not give you the correct answer, since it normally ignores the message number. To test for a specific message number, supply it with the ezmlm-list(1) "-n msgnum" switch.

Remove all test addresses at any list:

```
% ezmlm-list -a subdir | ezmlm-unsub subdir
```

This works since all sublists have DELETE access to the address tables. Don't worry! The sublists and feedback addresses won't disappear. Since they have a hash outside of the normal range, you need to use the ezmlm-unsub(1) "-s" and "-r" switches, respectively, to remove them.

Monitor list cluster status

That's it! To conveniently monitor the list use **status.pl**, a small perl script found in the **utils/** subdirectory of the ezmlm-idx distribution. Set up a MySQL uid with SELECT privileges to [assuming rootname=list] list_name, list_mlog, list_cookie (and the corresponding digest lists if used). Any sublist or main list uid would work, but remember that MySQL restricts access based not only on uid, but also on host. Thus, if the http server runs on a host that is not a list host you may have to set up a special user for **status.pl**.

Edit **status.pl** to reflect your installation and this uid and place the program in the cgi-bin directory of your http server. The program is [should be] self-explanatory. Don't forget to copy in the util/images/* files and set $IMAGES and $URL correctly. Also, it needs "x" and "r" bits set for the httpd user.

14.4.4 Adding a new sublist.

Let's add a new list. FIRST, subscribe `sub6@niko.jp` to the main list with hash 99 (using ezmlm-sub(1) "1-s"). Then add the following line to "*_name":

```
sub6@niko.jp    'jp'    0       52      Handles all @..jp addresses.
```

By just adding this, we took that set out of the addresses handled by sub4-6, without having to modify other entries. You can also see that we've prepared to split " further and can do this by just changing the list name of one of the `*@what.org` entries.

If we just add this while the list is running, these subscribers might get a message duplicated or miss a message. For this reason, the "*_name" table has a few more columns:

notuse

This can be set to non-zero to inactivate the entry. This is useful to temporarily remove list, or to add new ones while verifying, etc.

msgnum_lo, msgnum_hi

These default to the lowest and highest message number respectively. These columns can be used to change the split on running lists. Assume we want to add the `sub6@niko.jp` list. When we add it, some sublists may have already received the message, others not. If we remove it, the list that should have sent to those subscribers may already have processed the message, but this list has not yet. If instead we add `sub6@niko.jp` with msgnum_lo a few messages higher than the highest one sent, all lists will use the old split for lower messages (`sub6@niko.jp` will defer them leading to bounce). Then when the message number has reached the limit, all lists will use the new split.

Similarly, when inactivating `sub6@niko.jp` we would set the high end so that all lower messages (correctly) assume a working list and all higher ones determine their address space assuming that `sub6@niko.jp` is inactive.

To replace sub5 with would edit its msgnum_hi and add the replacement list with msgnum_lo set one higher.

To maximize efficiency, DELETE rows that are no longer active. Logging is done to the lowest listno entry for a given sublist name. If you need this information, collect and save it before removing any rows.

NOTE: You can have several entries for a list, e.g. to set it up to service both 'se', 'no', and 'fi'.

14.4.5 Restrictions on cluster addresses.

There are important restrictions to addresses and sublist entries:

The main list can handle ONLY sublists, NOT other subscribers.

Bounces from sublists are stored (max 50) and bouncing sublists are never automatically unsubscribed! To handle subscribers, just set up a separate sublist on the same host.

The main list domain entry MUST be empty ”.

This is the default. Likewise, sublist subscriber entries MUST have an empty domain entry. This is enforced by ezmlm-sub. This is a minor trade-off for easy sublist administration.

You SHOULD have at least one sublist with an empty domain.

This list will service all domains not explicitly serviced by another sublist.

You MUST make sure that for each domain (including the empty domain) the entire hash range 0-52 is cov

If you don't, some subscribers won't get mail! Thus, hash range cover is manual, whereas domain range cover is automatic (see c).

Overlap in hash range will result in some subscribers getting duplicate messages.

To e.g. split the 0-52 range from 2 to 3 lists, create 3 new entries in the name table, using msgnum_lo to make them take effect at the same time. Adjust msgnum_hi for the old entries so that they stop being active when the new ones start, i.e. msgnum_hi for the old lists should be 1 lower than msgnum_lo for the new ones, and all should be high enough that they will still be higher than the current message when you've finished making the changes.

15 Migration to Ezmlm from other Mailing List Managers.

This section describes differences and similarities between ezmlm and other mailing list managers. It also details functions of ezmlm-idx that allow you to configure ezmlm to respond to commands utilized by such other mailing list managers so the command syntax will be familiar to such users. Contributions to complete this sections are welcome.

15.1 Basic Concepts.

Ezmlm is different from other mailing list managers in that it is *list-centric* rather than *host-centric*. With a *list-centric* interface, you address the list directly with administrative commands. With ezmlm, the command is embedded in the list address thus becoming part of it (i.e., the "command address".) With smartlist, again you address the list, but send all administrative commands to the `list-request` address. Ezmlm lists can support this if you use the ezmlm-make(1) "-q" switch to configure ezmlm-request(1) in **DIR/manager**.

Other mailing list managers are *host-centric*, i.e. administrative commands for any list on that particular host are addressed to a central address such as `majordomo@host`, `listserv@host`, or `listproc@host`. Then the user is required to place the command in either the subject header or more commonly in the body text of the message. The listname has to be included with the command. [*Note:* The above concept is not universally applicable to all host-centric mailing lists. While intended to to used in a host-centric manner, many such mailing list managers also support `listname-request@host` addressing. See the applicable list manger documentation for details. Coverage of this aspect of other mailing list manager functionality is beyond the scope of this FAQ.] To make the migration to ezmlm easier, support for a *host-centric* style mailing list manger is available. This is based on the use of ezmlm-request(1) with the "-f **config_file**" switch.

15.2 Setting up ezmlm to respond to host-centric commands.

ezmlm-request(1) can be used a a "majordomo/listserv-emulator". You can create the necessary accessory files manually. However, ezmlm-idx>=0.32 contains ezmlmglrc(5) which makes is very easy for you:

```
% su
# su alias
# ezmlm-make -C/usr/local/bin/ezmlmglrc dir dot local host
```

where "local" may be e.g. "majordomo". Even easier is to set it up under a virtual domain "host" controlled by a user "user". Just put "user" in place of "alias" in the example.

If you use a character set other than US-ASCII, put it's name, optionally followed by ":" and the desired content-transfer-encoding character ("Q" for quoted-printable and "B" for base64) into **ezdomo/charset**.

All that remains is to set up **DIR/ezdomo.cf** with information on the lists (local and/or remote) that you want to make accessible via this interface. Another script, ezmlm-glconf(1) can help you with this for your local lists. To configure for all your lists:

```
ezmlm-glmake ~/ > ~/dir/ezdomo.cf
```

See man page for details. Alternatively, do it manually:

The **DIR/ezdomo.cf** contains a list of mailing lists which the "majordomo" (in this case) can provide information about in the following syntax:

```
list@host:listdir:description
```

To show a list in "lists", but not include it in a "which" search, simply omit the "listdir" for that line:

```
list@host::description
```

For the "which" command to work, the **DIR/**, which contains the subscriber database, must be readable by the user under which mail is delivered. This means that "which" is usually limited to lists owned by the user or virtual domain under which the "ezdomo" interface is set up.

15.3 Commands of other mailinglist managers recognized by ezmlm.

15.3.1 Listproc/Listserv.

When set up as above, substituting "listproc" or "listserv" for "majordomo" as appropriate, ezmlm will recognize and respond to the following commands placed in the body of the e-mail with the syntax below. **Note: ezmlm will only respond to one command per message.**

syntax: command listname [subscriber@host]

Supported commands

subscribe, sub, unsubscribe, unsub, list, help, review.

Additional supported commands

All ezmlm commands, such as "thread", "index" and "get" as well as the list owner's commands.

This interfaced makes information available via command messages to the appropriate mailing list. Thus, "list" and "review" will send a subscriber list only to remote administrators and only if specifically allowed by the list owner.

15.3.2 Majordomo.

syntax: command listname [subscriber@host]

Supported commands

lists, subscribe, unsubscribe, help, which, who.

Additional supported commands

All ezmlm user and ezmlm owner commands.

This interfaced makes information available via command messages to the appropriate mailing list. Thus, "who" will send a subscriber list only to remote administrators and only if specifically allowed by the list owner.

15.3.3 Smartlist.

Unlike "listproc/listserv" or "majordomo", "smart-list" does not provide "host-centric" services. Rather, commands are addressed to `listname-request@host` and the command placed on the "Subject:" line:

```
To: listname-request@host
Subject: command [subscriber@host]
```

The body of the message is normally ignored. If the subject is empty, the first body line that starts with a letter is interpreted.

Supported commands

subscribe, unsubscribe.

Additional Supported Commands

All ezmlm user and ezmlm owner commands.

16 Optimizing list performance.

Ezmlm-idx is designed to make it as easy as possible to set up mailing lists. The default setup works well for small and medium-sized lists. For large lists, the lists can be made more efficient with a few simple changes.

16.1 Crond-generated digests for better performance.

With the default setup, ezmlm-tstdig(1) in **DIR/editor** tests if a digest should be sent out. On lists with a lot of traffic this is inefficient. Also, you may want digests to be delivered as a specific time. To do this, use crond(8) to execute ezmlm-get(1) directly, as described elsewhere.

16.2 Optimizing execution of ezmlm-warn(1).

ezmlm-idx>=0.32 comes with much improved bounce handling. Modification as described below should be considered only when you expect thousands of bouncing addresses (virtually never). The description remains, for users of ezmlm-0.53 or earlier versions of ezmlm-idx. For users of ezmlm-0.53 alone, we recommend a patch (`<ftp://ftp.id.wustl.edu/pub/patches/ezmlm-return.diff>` which fixes a bug in ezmlm-0.53 bounce handling. The patch is superseded by ezmlm-idx.

To redistribute the load of bounce warning and probe addresses to off-peak hours, you may want to set up the list without ezmlm-warn(1) by using the ezmlm-make "-w" switch, and instead execute "ezmlm-warn DIR" via crond(8). You also need to run "ezmlm-warn -d DIR" for digest bounces if your list is configured with digests. Normal ezmlm list with ezmlm-idx>=0.32 will have an insignificant bounce load, except if you bulk add addresses, e.g. from a MLM without bounce handling. In the latter case, the load will be higher for the first 2-4 weeks, then decrease drastically. If you feel you need to run ezmlm-warn(1) from crond(8), you should seriously consider sublisting your lists.

Note: the ezmlm-make(1) "-w" switch has a special meaning if used at the same time as enabling SQL-support ("-6"; see man pages).

16.3 Decreasing ezmlm-warn time out to increase performance.

With ezmlm-idx, you may alter the ezmlm-warn(1) timeout to a number of seconds with the "-t seconds" switch. The default is 1,000,000 seconds or about 11.6 days. This is the time from the first bounce until ezmlm-warn(1) sends a warning message and the time from the warning message bounce until ezmlm-warn(1) sends a probe (which if bounced leads to removal of the address from the subscriber list). If you have a digest list, remember to execute ezmlm-warn(1) with the "-d" switch as well.

Decreasing the default to e.g. 5 days will cut in half the average number of files in the bounce directory and the number of messages sent at each crond(8)-directed invocation of ezmlm-warn(1). The trade-off is that worst case, a subscriber may be unsubscribed if his/her mail path is defective for more than twice the timeout. Removing a subscriber after 10 days seems reasonable on a busy list. Do this by adding the "-t" switch to all the ezmlm-warn(1) invocations. This timeout should be larger than the interval between ezmlm-warn(1) invocation.

To be aggressive, use "ezmlm-warn -t0". This will minimize the time your lists spends servicing bounces, but will for some errors lead to subscribers to be also lead to subscribers being removed if messages to them bounce for two consecutive ezmlm-warn(1) runs. This is useful to rapidly clean up a low quality address collection.

16.4 Use ezmlm without ezmlm-idx for maximum performance.

ezmlm-idx adds a number of functions to ezmlm. It indexes the archive, and adds an index entry for each message, it can remove MIME parts, it can add a subject prefix and message trailer, decode rfc2047-encoded subjects, etc. Although designed to impact minimally on performance, these options when used take time. Even when they are not used, time is spent looking for e.g. the prefix. However, the performance penalty is small, as the absolutely dominating cost of a mailing list is the work qmail does to deliver the messages to subscribers.

In bench marking, we have not found a significant difference in performance between ezmlm-0.53 and ezmlm-0.53+ezmlm-idx-0.32 when ezmlm-idx features are not used. Thus, a non-indexed list with ezmlm-idx-0.32 performs the same as the corresponding ezmlm-0.53 list. Adding an index adds the overhead of another safe write (the index file). Use of other features adds very marginally to execution time. For virtually all lists, the ezmlm execution time is negligible compared to the resources needed by qmail to disseminate the message to the subscribers.

16.5 Not archiving to maximize performance.

An archived list needs to write the message to the archive. If you don't need an archive, don't archive. However, the archive is very useful to allow users to catch up on messages that they didn't receive due to delivery problems.

16.6 Sublists to maximize performance.

Consider splitting your list into sublists, ideally geographically. The main list deals only with a subset of subscribers (or only the sublists), and each sublist deals with a subset of subscribers, bounces, etc. This is the most rational way to scale ezmlm to large lists (see 4.24 (How sublists work) for more info on how sublists work and 14 (Sublists) on how to set up sublists).

17 Miscellaneous.

17.1 How do I quickly change the properties of my list?

```
ezmlm-make -+ [changed_switches] dir
```

ezmlm-make(1) stores configuration info in **DIR/config** and uses that info as the default when you use the "-+" switch. If the list was created with a very old version or ezmlm-0.53 ezmlm-make(1) you have to restate all arguments the first time you edit the list.

The "-e" switch works the same, without stickiness for switches.

A message arriving during reconfiguration may be handled incorrectly. The prudent user will set the sticky bit on the home directory to prevent delivery, then clear it after the list has been changed.

17.2 Open archived list with daily digests.

This is the default setup. The main list generates digests in response to a mailed request or when a message arrives and the amount of messages since the last digest exceeds set limits (see ezmlm-tstdig(1)). Alternatively, ezmlm-get(1) can be invoked from the command line. In both cases, the generated digest message is disseminated to the subscribers stored in **DIR/digest/subscribers/**, i.e. the subscriber database with the base directory **DIR/digest/**.

- See 12.1 (setting up a digest list) on how to set up the lists.

17.3 Variations in moderation

You can set up lists with combinations of message moderation, subscription moderation, and remote administration, easiest by combining ezmlm-make(1) "-m" ,"-s", and "-r" switches. You can use a non-default moderator db, by specifying a directory starting with a slash in **DIR/modsub** or **DIR/remote** (for remote admin and subscription moderation - always the same db for both functions) or in **DIR/modpost** for message moderation. You can point several lists to the same moderator db, thus using the same moderators for several lists. *NOTE:* The user controlling the list must have read/write access to the files (specifically, must be able to write the lock file).

Some of these setups are not trivial. However, you can make them trivial by modifying ezmlmrc(5) so that ezmlm-make(1) can set up the desired lists by default or when the user uses e.g. the "-y" or "-z" switches (see 7 (Customizing ezmlm-make operation)).

17.4 Lists that allow remote admin, but not user initiated subscription or archive retrieval.

Create a regular remote admin list, but remove **DIR/public**. This allows moderators to (un)subscribe users and have archive access, but rejects all user requests. Posts work as usual. Naturally, this can be combined with message moderation or ezmlm-issub SENDER checks (see 8 (Restricting message posting to the list)).

17.5 Lists that allow remote admin, user archive retrieval, but not user-initiated subscription.

Create a regular remote admin list, remove **DIR/public**, and add the "-p" [public] switch to the ezmlm-get(1) command line in **DIR/manager**. This overrides the normal **DIR/public** effect on ezmlm-get(1) and archive retrieval, allowing full archive access to anyone, but rejecting user -help and subscription commands. It is assumed that the users know archive retrieval commands without help. If you want to provide specific help, just link **~/.qmail-listname-help** to **DIR/help**, and invoke a script that copies help info from there. See ezmlm-check(1) for an example.

17.6 Lists that restrict archive retrieval to subscribers.

Use a standard list, but add the ezmlm-get(1) "-s" command line switch in **DIR/manager**. Only subscribers can receive archive excerpts. Digests work as usual. This can be set up using the ezmlm-make(1) "-g" switch.

17.7 Lists that do not allow archive retrieval at all.

Use a standard list, but add the "-C" switch to both the ezmlm-get(1) and ezmlm-manage(1) command lines in **DIR/manager**. No archive retrieval commands will be honored. Digest can be created as usual (See 11 (Restricting archive retrieval)).

17.8 Lists that do not allow archive retrieval and do not allow digest triggering per mail.

For maximal archive security, set up a normal indexed and archived list, then remove the ezmlm-get(1) line from **DIR/manager** and add the "-C" switch to the ezmlm-manage(1) command line. You can still create digests by direct invocation of ezmlm-get(1) from a script or crontab entry.

17.9 Lists that allow archive retrieval only to moderators, but allow user-initiated subscription.

Create a normal remote admin (+ subscription moderated) list, and add the "-P" (not public) switch to the ezmlm-get(1) command line in **DIR/manager**. Subscription will not be affected, but ezmlm-get(1) will send archive excerpts only to moderators. Digests are unaffected.

17.10 Lists that do not require user confirmation for (un)subscription.

The need for a user handshake can be eliminated by the ezmlm-manage(1) "-S" (subscribe) and/or "-U" (unsubscribe) switches. Alone, this is very insecure. However, there may be some use for it in local lists with subscription moderation, or alone for notifications where ease of use is more important than preventing users from (un)subscribing others. If the list has subscription moderation or remote administration, any user subscribe or unsubscribe request is forwarded to the moderators if the SENDER and target address do not match, even if the "-U/-S" switches are specified. This is put in place to make a "-U/-S" list similar to other list managers, not for security (it's not secure, since a malicious outsider can easily fake the SENDER address). Unsubscribe confirmations are sent also to the target in this case, to avoid situations where the user needs moderator "permission" to get off the list.

17.11 Announcement lists for a small set of trusted posters

Set up the list with ezmlm-store(1) "-P" and add the "trusted E-mail addresses" to **DIR/mod/** with

```
% ezmlm-sub DIR/mod address@host
```

A post from a "trusted address" is sent back to that address for approval, assuring that the user at that address really sent the post. Posts from other e-mail addresses are rejected.

17.12 Announcement lists allowing moderated posts from anyone.

This is useful in many circumstances. A list announcing new programs for a system, where both the main developers and other users may have contributed programs.

Set up the list with ezmlm-store(1) and the main developers as moderators. When any of these posts, that user alone is asked to confirm. Posts from other E-mail addresses are sent to all moderators/developers. To use a different set of E-mail addresses as "trusted e-mail addresses" and moderators for other posts, use the ezmlm-store(1) "-S" switch and make a separate address database for the "trusted E-mail addresses". Put the name of the basedir for the "trusted e-mail addresses" database in **DIR/modpost** (needs leading "/"), and add the post moderator(s) to **DIR/mod/** using ezmlm-sub(1) as shown above.

17.13 Announcement lists with less security and more convenience.

A general solution for SENDER checking is to configure list with ezmlm-gate(1). ezmlm-gate(1) takes as arguments any number of basedirs for subscriber lists. Posts from SENDERs that are found are posted. For others ezmlm-store(1) is invoked. If **DIR/modpost** exists, ezmlm-store(1) will send out other messages for moderation. To bounce such messages, create **DIR/modpost**, and use the ezmlm-gate(1) "-P" switch (will be passed on to ezmlm-store(1) to bounce any posts not from a moderator).

By default, ezmlm-gate(1) accepts messages from subscribers. However, this is overridden if any "basedirs" are put on the ezmlm-gate(1) command line. Common would be to create a address list and put its "basedir" on the ezmlm-gate(1) command line. Trusted E-mail addresses can then be added with:

```
% ezmlm-sub basedir trusted@host
```

As this relies on SENDER checks it is less secure than the ezmlm-store based confirmation-requiring setup.

18 Ezmlm-idx compile time options.

18.1 Location of binaries.

This is configured via **conf-bin** as for other ezmlm programs. The default is **/usr/local/bin/ezmlm**.

18.2 Location of man pages.

This is configured via **conf-man** as for other ezmlm programs. The default is **/usr/local/man**.

18.3 Base directory of qmail-installation.

This is configured via **conf-qmail** as for other ezmlm programs. The default is **/var/qmail**.

18.4 Short header texts, etc.

Ezmlm-idx text (short lines, such as "Administrivia" for digests), command names, etc, are defined in **idx.h**, used at compile time. You can change them by changing the defines in this file.

18.5 Arbitrary limits.

idx.h contains defines for some ezmlm-idx arbitrary limits, such as the maximum number of messages per "-get" request. They can be changed here.

18.6 Command names.

There is support for one alias per user command for internationalization. (See 19 (Multiple language support).)

18.7 Error messages.

All ezmlm-idx error messages are defines in **errtxt.h**, used at compile time. These can be changed for special situations, but we would advise against doing so. If you do for some reason produce such a translated file, we would appreciate if you sent a copy to the authors. NOTE: These do not affect error messages from programs that are not part of the ezmlm-idx package, nor of some subroutines used by ezmlm-idx programs (getconf_line.c comes to mind).

Hopefully, the error messages for all parts will be synchronized in later versions of ezmlm, and possibly handled from a run-time changeable separate file (maybe as a .cdb database).

18.8 Paths and other odd configuration items.

idx.h also has defines for **/etc/ezmlmrc**, default formats for moderation enclosures, default character set, default digest format, etc. Since most of these items are easily changed at run time, there is usually no need to change the compiled-in defaults. If you do need to, this is where they are.

19 Multiple language support.

19.1 Command names.

ezmlm commands can have aliases for use in translations for non-English use. Due to the use of commands in mail e-mail addresses, the character set is limited by rfc822 to us-ascii. To enable the command aliases, remove the comment marks around the INTL_CMDS define in idx.h. Also, remove the comments from the define corresponding to one language (currently, only LANG_FR - French) available.

The INTL_CMDS define results in the compilation of all ezmlm programs with support for alias commands for those commands listed in the INTL section (all that are used directly by users). All aliases MUST be defined, but should be the normal English commands. The language-specific sections un-define and redefine the commands for which alternative names should be used. This allows use of e.g. "inscription" as an alias in addition to the standard "subscribe".

19.2 Text files.

Most ezmlm responses are made from text files in **DIR/text/**. These are created from the template file "ezmlmrc". Thanks to Frank Denis, and Masashi Fujita, Wanderlei Antonio Cavassin, Sergiusz Pawlowicz, Frank Tegtmeyer, Torben Fjerdingstad, Jan Kasprzak, and Sebastian Andersson, French, Japanese, Portuguese (var. Brazil), Polish, German, Danish, Czech, and Swedish versions are available. Just:

```
% make jp
```

before

```
# make setup
```

or just copy **ezmlmrc.jp** to **/etc/ezmlmrc**, where it will override the copy installed in the ezmlm binary directory. For rpm packages, the en_US version is installed, but the other versions are available in the **/usr/doc/** hierarchy.

If you have made an **ezmlmrc(5)** version for another language, please make it public domain and E-mail it as an attachment to `lindberg@id.wustl.edu`. It will then be put into the **ezmlmrc** directory of the distribution site. Please take advantage of the "Content-transfer-encoding" capability of ezmlm-idx>=0.30, if needed, as this avoids problems when messages are sent via non-8-bit MUAs.

Other ezmlm responses, such as words in subject lines, are defines in **idx.h** and can be changed there. Error messages should ideally not be altered. However, it may make sense to change a few of them which are used as messages to e.g. remote administrators. The defines for all error messages are in **errtxt.h**.

19.3 Multi-byte character code support.

ezmlm, as far as we know, places no restrictions on character sets. The configurable default character set allows you to use other character sets for out going ezmlm messages. ezmlm-make does not *per se* support other character sets. However, any single-byte character set is supported, as long as the us-ascii character sequence "</" does not occur anywhere as the first characters of the line, and the character sequence "<#x#>" (where "x" is any number, or A, B, C, D, F, H, L, R, T) does not occur anywhere is text (if it does, it risks being substituted). Also, any occurrence or "<#A#>" and "<#R#>" that is the first on any text line will be substituted by ezmlm-manage and ezmlm-store. Any occurrence of "!A" and "!R" as the first characters on a line will be substituted by ezmlm-manage and ezmlm-store.

For multi-byte character codes, the same restrictions apply. Thus, "</" at the start of a line will confuse ezmlm-make, and any "<#x#>" sequence within the text risks substitution. In practice, both of these should be very rare and easily avoidable when setting up an ezmlmrc(5).

20 Subscriber notification of moderation events.

20.1 General opinions.

This is a collection of the authors opinions and an explanation of ezmlm-idx moderation design, which you may or may not agree with.

20.2 Users should know that the list is subscription moderated.

List subscribers should be informed that subscriptions to the list are controlled by a moderator. ezmlm-idx in its default setup handles this notification during and after the subscribe handshake. Most of this can be disabled by manipulation of the **DIR/text/** files.

20.3 Subscribers should know that posts are moderated.

List subscribers should be informed that posts to the list are moderated. ezmlm-idx does this by adding the "Delivered-To: moderator for ..." header, but IOHO, the list owner should make the fact of list moderation plain in introductory messages, or other means, to the list subscribers.

20.4 Senders of posts should be notified of rejections.

With normal use of ezmlm-idx, the sender of a rejected post is notified that the post has been rejected and if the moderators chooses to comment, the sender receives this comment, usually describing why the post was rejected. This ezmlm behavior cannot be disabled at run time.

If post are neither accepted or rejected, they time out. ezmlm-clean(1) notifies the sender when this happens. This behavior can be disabled with the ezmlm-clean(1) "-R" (not return) switch, which has to be placed on the command line of all invocations of ezmlm-clean(1) (normally in **DIR/editor** and **DIR/moderator**). If you for some reason do not wish to inform the sender of your editorial decision, you can use this switch and let undesirable posts time out, rather than actively rejecting them. IOHO, it is better to be "above board" and use the normal notification mechanisms, together with active rejection and informative rejection comments.

The ezmlm-make(1) "-u" switch uses moderation in a slightly different way. Here, posts are restricted to subscribers, but posts from non-subscribers are sent to the moderator(s) rather that being ignored. This to help the subscriber that posts from an alias of the subscribed address, or the occasional non-subscriber. In this case it is perfectly acceptable to just ignore non-accepted posts. Thus, using the ezmlm-make(1) "-u" switch configures the ezmlm-clean(1) invocations with the "-R" switch.

21 Ezmlm-idx security.

21.1 General assumptions.

This document discusses security aspects of ezmlm-idx addition to the ezmlm-0.53 mailing list manager. This is the authors' understanding of security aspects of ezmlm-idx functions and not to be taken as a warranty. If you find any errors in this document or the ezmlm-idx package in general, please inform the authors.

In general, ezmlm with or without the ezmlm-idx package is more secure and less resource hungry than most other mailing list managers. Better security than afforded by ezmlm +/- ezmlm-idx would require encryption or PGP/digital signatures. Such an addition would make it difficult, if not impossible, to run the mailing list from a standard MUA. The ezmlm-idx package adds a number of functions and options, which under some conditions may decrease security. The purpose of this document is to discuss security aspects of using/enabling these different functions.

21.2 SENDER manipulation.

We assume that the cost of manipulating/falsifying the SENDER address of a message is zero. Thus, any mechanism relying on SENDER alone is insecure. However, such a mechanism may help in case of simple mailer or user errors. We also assume that the "cookies" used by ezmlm are secure, i.e. that it is very hard for someone to generate a valid cookie for a given address. SENDER is used to identify a moderator for remote administration of subscriptions. The result of the action or the confirmation request are sent back to that moderator address. Thus, providing a false SENDER is useless, unless the attacker can also read that moderator's mail.

21.3 ezmlm cookies.

Since ezmlm doesn't rely on the SENDER, the security lies entirely within the action-time-cookie-address combination. Anyone obtaining a valid "combination" can do whatever the combination is meant to do, but nothing else. Also, the cookie times out 1000000 seconds (approximately 11.6 days) after it was issued. Since the "combinations" are specific for a particular action and address, they can only be reused for that particular purpose, and within 11.6 days. Ezmlm (un)subscriptions for a given address are usually pointless to repeat. Message moderation "combinations" are useless after they've been used, since the message is no longer in the moderation queue.

21.4 Lists without remote admin/subscription moderation.

Maliciously (un)subscribing an address with ezmlm-0.53 requires that the attacker is able to read mail sent to the subscription address.

With the ezmlm-idx add-on, a non-moderated list works exactly the same way. Ezmlm-idx introduces the moderator for moderated and remote admin lists. For any moderator functions, an attacker needs to be able to read mail sent to a moderator's address. If s/he can do this, the attacker can affect anything the moderator is allowed to do (since falsifying SENDER is trivial). To minimize risks, give moderators only the power they need, do not use more moderators than necessary, and use moderators whose mail is hard to intercept (on the same machine/same internal/secure network or by encryption via e.g. ssh).

21.5 Message moderation.

A basic message moderated list keeps ezmlm subscriber security, but interpolates the moderator(s) between the address of the list and the list itself. An attacker able to read moderator mail can accept/reject a post, if s/he can do it before a regular moderator has taken action. The potential for abuse can be minimized by using few and local moderators. Mail logs are needed to trace which moderator address was misused.

21.6 Subscription moderation.

A basic subscription moderated list retains ezmlm subscriber security, but adds a moderator handshake. An attacker would need to be able to both read mail to the subscriber address and to at least one moderator.

21.7 Remote administration.

A remote admin (-r) list adds the ability of the moderator to (un)subscribe any address. The price of this is that an attacker able to read moderator mail can (un)subscribe any address. The moderator handshake message will be delivered to the abused moderator address, which will alert that moderator and reveal the compromise. Another basic assumption is that action-date-cookie-address combinations are only sent to the target address or a moderator and that moderator action "combinations" are never sent to non-moderators.

21.8 Remote editing of ezmlm text files.

ezmlm-manage(1) can allow remote administrators to edit files in **DIR/text**. First, this option is disabled by default. Second, the "-edit" command is accepted only when the target (the recipient) is a remote administrator. Third, only existing files within **DIR/text** are editable. It is not possible to create files.

ezmlm replies to a valid request with an informative message and the contents of the file. In addition, the "Reply-To:" address contains a cookie based on the file name and contents, as well as the current time. Anyone possessing this cookie can save a new version of the text file. As with other ezmlm security, the security of this process depends on only the remote administrator receiving remote administrator mail. If this is not sufficiently secure for you, do not enable this option. As always, an increase in accessibility results results in a decrease in security.

Cookies for editing expire in approximately 27 hours. Also, as soon as a file is changed, the cookie is invalidated since the file contents change. This also means that an outstanding edit request cannot be completed if the files has been updated in the interim.

A potential attacker obtaining a valid cookie has a window of opportunity while you edit the file, or for at most 27 hours. S/he can overwrite and existing text file with potentially offensive material. Usually, this can be achieved more easily by posting to the list. S/he can also potentially fill your disk with a large amount of data (up to two times 10240 bytes (limited by MAXEDIT in **idx.h**)) and could put part of this data onto messages leaving the list. Again, this is much more easily achieved by e.g. sending the equivalently sized message to your list.

21.9 Digest generation and archive retrieval.

The archive retrieval functions added by ezmlm-idx are digests (protected by a "code") and other functions. Anyone who knows the digest code (through reading mail logs, reading **DIR/manager** of the list, or reading any scripts used to send digest triggering messages) can trigger a digest. Protect these locations accordingly! For default lists with digests triggered from **DIR/editor** via ezmlm-tstdig(1) and ezmlm-get(1), you do not need the digest code and can thus disable the possibility to trigger digest by mail. For other functions, the output is sent to SENDER and can be restricted to subscribers (the "-s" switch). ezmlm-get(1) functions (apart from digest) can be entirely disabled with the i"-C" switch, or restricted to moderators with the "-P" switch or by removing **DIR/public**. Other sections of this document discuss several other options. All switches are documented in the man pages.

The moderator support functions added by the ezmlm-idx package (extended help and subscriber list) are sent only to a moderator address, i.e. an attacker again needs to be able to read moderator mail to read the output. The help info (**DIR/text/mod-help**) should not contain secrets. The "-list" function is normally disabled, but can be enabled with the ezmlm-manage -l switch to aid the remote administrator(s).

21.10 Convenience for security: the ezmlm-manage "-S" and "-U" switches.

ezmlm-manage(1) functions can be made more convenient, at the expense of security. There have been many requests for these options, so they have been added, although we recommend against using them:

The ezmlm-manage(1) "-S" switch eliminates the subscriber handshake from subscribe requests. Thus, it is no longer necessary for the subscriber to confirm the subscription. This is not secure, but may be convenient for some moderated lists. Use only with extreme caution. The ezmlm-manage(1) "-U" switch similarly eliminates subscriber confirmation from unsubscribe requests. Again, this is insecure and useful only under special circumstances. If the list has any moderators (remote or modsub), requests to (un)subscribe an address other than sender are still routed to a moderator. This is similar to how some other lists work. Naturally, this is insecure because it relies on SENDER. Unsubscribe requests are always non-moderated, since, IOHO, it seems un-ethical to force a subscriber to remain on a list. Where an unsubscribe confirm request is sent out it is (also) sent to the target, except when the request was initiated by a moderator on a list with remote administration (**DIR/remote** exists).

The (un)subscription target is always informed about completed (un)subscribe request, whether initiated by that address, another address, or by a moderator. Thus, attempts of a user or moderator to subscribe an address will be brought to the attention of the user receiving mail at that address.

21.11 Denial of service.

ezmlm-get(1) archive retrieval functions can be used to deplete system resources. However, this can also be done by posting messages to lists, mail bombing, etc. If you are worried about this, you can use a combination of ezmlm-manage/ezmlm-get "-C", "-s", and "-P" switches, removal of **DIR/public**, and removal of the mail-triggered digest function (by removing the digest code from the ezmlm-get(1) command line) to decrease availability of these functions (see man pages). Digest can also be triggered by direct execution of ezmlm-get from within a script from **DIR/editor** as in the default setup with the ezmlm-make(1) "-d" switch.

21.12 Moderator anonymity.

Anyone getting messages from the list can see the "Delivered-To: Moderator for ..." header and realize that the list is moderated. In the authors opinion, this is fair and appropriate. If this bothers you, edit the source of **ezmlm-store.c**.

While the fact that the list is moderated will be disclosed by the headers, the moderator(s)' identity will not be disclosed by the header. Moderators are anonymous to anyone who cannot directly read the mail log, the moderator list, or monitor your outgoing and incoming mail. Anyone intercepting the acting moderators' mail or able to read the mail log can determine who took a particular action.

Moderator E-mail addresses are not (to our knowledge) disclosed by any ezmlm mechanism. Thus, the poster does not know who rejected/accepted the message. Other moderators can find out that the message was accepted (by seeing it on the list or by themselves committing to a reject/accept reply) or rejected (by being informed by the poster or by themselves committing to a reject/accept reply). If no moderator takes any action for a given time (120 h - configurable to anything 24-240 h via **DIR/modtime** - and the parameters are likewise configurable at compile time via **idx.h**) the message times out, an act for which no particular moderator can be held accountable.

Subscription requests are acted upon only if a moderator completes the transaction by approving the requests.

Requests can not be directly disapproved, but the associated cookie becomes invalid after approximately 11.6 days. Neither the subscriber nor the other moderators know which moderator accepted the subscription request. Requests to unsubscribe from the list are never moderated or otherwise controlled, except by requiring confirmation from the subscriber (normal unsubscribe) or the moderator that initiated the request (remote administration). If several moderators approve the same subscribe request, the user gets multiple notifications.

The triggering message (the moderation approval or the moderator's completion of the subscription request) are not returned or logged. This protects moderator anonymity, but makes it harder to track down the offender in case of abuse. Only a good mail log will help. IOHO, abuse of these mechanisms requires considerably more effort that it is worth to (un)subscribe someone to a list. Also, IOHO, moderator anonymity is more important. If this increased difficulty in tracking down abusive behavior bothers you, don't use the remote administration and moderated subscription features.

21.13 Confidentiality of subscriber E-mail addresses.

The optional "-list" command enabled by the "-l" ezmlm-manage(1) command line switch returns a subscriber list to the moderator. Again, anyone who can intercept a moderators' mail can fake SENDER and use this command to obtain a subscriber list. The use of local moderators minimize the risk. If the risk of subscriber disclosure is not worth this convenience, do not enable this feature.

21.14 Help message for moderators.

ezmlm-manage sends **DIR/text/mod-help**, rather than **DIR/text/help** in reply to messages to `list-help@host` if the target address is a moderator. **DIR/text/mod-help** should not contain secrets or other confidential information.

21.15 Sublists.

ezmlm sublists require that the message envelope sender is the main list, and that the message has a "Mailing-List:" header. Both are easy to fake, allowing an attacker to inject messages at the sublist level. Other than the possible ramifications of only a subset of subscribers seeing the message, this is of no concern for open lists. For a "subscriber-only" list based on SENDER checks, it is no harder to set SENDER to the address of a subscriber than to fake the headers required by the sublist. However, for a moderated list the mainlist to sublist communication becomes the weakest link. Sublists using a SQL database also use better authentication in this step (see 5 (SQL-enabled ezmlm lists)).

A sublist user can unsubscribe a normal ezmlm sublist from the main list. To guard against this, you need to prevent propagation of unsubscribe confirm requests by the sublist. Easiest is to add a line to **DIR/editor** before the ezmlm-send(1) line:

```
|grep -i '^Subject: CONFIRM' >/dev/null 2>&1 && exit 99; exit 0
```

Another option would be to take advantage of the fact that **DIR/headeradd** headers at the main list are added to normal messages, but not to administrative messages. Thus, one could discard messages that lack the default "Precedence: bulk" header:

```
|grep -i '^Precedence: bulk' >/dev/null 2>&1 || exit 99; exit 0
```

For lists with SQL-support, users cannot unsubscribe sublists (see 5 (SQL-enabled ezmlm lists)).

Break-in at a sublist host for normal ezmlm lists leads to loss/compromise of the addresses handled by the sublist. For MySQL-enabled lists, the sublist access credentials give DELETE and SELECT access to all addresses serviced by the list. Thus, a successful sublist attacker can completely disable the list. The MySQL log (if used) will reveal from which host the attack was done. Although the potential damage to a SQL-enabled list is greater, the results are of the same order of magnitude. The risk in minimized by keeping control over all sublist hosts. A successful sublist attacker cannot normally add addresses, since the sublist users by default are set up without INSERT privileges to the address database.

21.16 SQL databases.

For SQL-enabled lists, the database contains all list information. Subversion of your database server allows an attacker to add/remove addresses at will. This is also true for normal ezmlm lists. In addition, modification of the "*_name", "*_cookie", and "*_mlog" tables can cause the list to misbehave in a manner that doesn't immediately suggest a security breach. Keep your ezmlm list and database servers secure.

21.17 Reporting security problems.

Please send private E-mail about any security problems with the ezmlm-idx additions to Fred Lindberg, *lindberg@id.wustl.edu*. For ezmlm, please send them via private E-mail to Dan J. Bernstein, the author of ezmlm proper.