ListProcessor®

ver 7.1

List-Owner Manual©

The Corporation for Research and Educational Networking

ListProcessor
Owner Manual
Table of Contents

1. Introduction

This List-Owner Reference Manual provides an overview to the list-owner commands of the CREN® ListProcessor® list- and file-management software, also known as ListProc®. The commands documented in this reference are not generally available to subscribers who do not own the list affected by the command being used.

This reference assumes familiarity with the ListProcessor User Manual and does not replicate that document's explanation of ListProc's subscriber commands.

This Owner's manual is one of five documents list owners should be aware of. (1) The List Owner's Manual explains in detail how to manage a list; (2) the List Owner's Reference Card is a short reminder of the material covered in the Manual; (3) the Listprocessor User's Manual is a detailed explaination for users of how to interact with the list processor; (4) the User's Reference Card is a short reminder of material covered in the Manual; and (5) the Moderator's Manual is a short discussion of the basics of how to moderate a discussion list. It is strongly recommended that you, as list owner, obtain and read all five documents. You should be familiar with end-user commands, which are not covered in this document. It is recommended that you either print this Owner's Manual out or use a multi-window program to read it so that you can refer to different sections of this manual as you read it.

Each of the five ListProc documents is available in plain text format by sending the command **`get doc`** ***`refdoc`***, where refdoc is replaced by one of **`userman`**, **`ownercard`**, **`usercard`**, or **`moderatorman`**, as the body of a mail message to *listproc@listproc.net* or any other CREN ListProcessor. These documents are available from *info.cren.net* via anonymous ftp, as the PostScript file */listproc/ps/doc-ownercard.ps* and as the RTF file */listproc/rtf/doc-ownercard.rtf* (open the latter with Microsoft Word or WordPerfect®, and set the orientation to landscape before printing).

2. What is ListProcessor?

The Internet provides thousands of discussion groups via e-mail. Participants in these discussion groups place themselves on mailing lists, called "lists" for short, in order to be able to send messages to and receive messages from the various lists. ListProcessor, ListProc for short, is a powerful mailing list agent that keeps track of thousands of people subscribed to any number of mailing lists. When a user subscribes to a list, that person's name and e-mail address are automatically added to the list of subscribers. The new subscriber will receive a form letter of welcome telling about the list. Then all mail sent to the list by other subscribers will be sent to the new subscriber also. If the new subscriber wants to reply to these messages and sends in a response to a message, the message will be sent out to everyone who is subscribed to the list. Users may post messages, review members of lists, review the configuration set up of lists, etc.

ListProc allows full control over the list and its configuration to the list owner, freeing up the system manager from having to make major modifications for the list owners when needed. ListProc is also a powerful file archiver, allowing search and retrieval of text files based on regular expressions and e-mail retrieval of binary files automatically uuencoded and divided into managable sized portions. It is automated, and eliminates the need for user intervention and maintenance of multiple aliases of the form "list, list-owner, list-request", etc. There is support provided for public and private hierarchical archives, moderated and non-moderated lists, peer lists, peer servers, private lists, address aliasing, news connections and gateways, mail queueing, digests, list ownership, owner preferences, crash recovery, batch processing, configurable headers, regular expressions, archive searching, and live user connections via TCP/IP.

3. What Are the Responsibilities of a List Owner?

List owners are individuals responsible for list administration via mail commands. Thus, list owners may be remotely located. Each list has to have at least one list owner. These owners may be different than the system's manager, and have special privileges: they may issue commands on users' behalf (add a user, remove a user, etc.) overriding system restrictions set on regular users (including overriding disabled commands), obtain reports about the lists they administer, append to the ".aliases" and ".ignored" files, change the welcoming (".welcome") and informative (".info") messages, as well as other system files such as the aliases file (".aliases"), the ".ignored" file, the subscribers file (".subscribers"), the news file (".news") and the peers file (".peers"). In addition, they may moderate their lists and they receive various error messages pertaining to their lists. All administrative commands are author authenticated and password protected. Whenever a message cannot be author authenticated, the list's owner and manager are notified. On the other hand, list owners may not add restricted users; this service can be provided by contacting the system's manager. List owners may also receive copies of user commands and/or error messages such as invalid postings, syntax errors on commands, etc.

List owners may assign various parts of these responsibilites to other people. List owners may assign moderators, whose function is to approve messages and guide the online discussion, subscription managers, whose function is to add or remove individuals to the list, and recipients of error messages, whose function is to contact users who generate error messages by submitting invalid commands and educate them as to the error, also to remove users whose addresses are no longer valid.

4. Special Files that ListProc Uses

ListProc makes use of a number of text files during the course of its operation. Some files, such as the list’s config file, are changable via commands to ListProc and are not editable, but others can be requested from ListProc using the **`EDIT`** command, edited, and replaced using the **`PUT`** command (see **`EDIT`** and **`PUT`** in the command section below). These seven files all have filenames starting with a dot “.” and are .aliases, .ignored, .info, .news, .peers, .subscribers, and .welcome. When requesting any of these files for editing from ListProc you do not include the dot in the filename in the command line, so the discussion below will not include the dot in the filename in order to avoid confusion.

The alias file allows the list owner to rewrite incoming e-mail addresses. Due to the wide variation in the manner in which different hosts are set up, it is possible that e-mail can come in to listproc with a different return address than the sender’s real address. For example, if mail comes in from BITNET there is no way of knowing which gateway will be used for the mail. The mail may come in as user%node.bitnet@interbit.cren.net on one occasion and from user%node.bitnet@pucc.princeton.edu on another occasion. A user can likely get back a error message saying that he/she is not subscribed to a particular list. In order to get the ListProc to accept the address user@node.bitnet no matter what getway it comes through an alias can be set up in the alias file.

The ignored file filters out messages sent by certain users. You may, for example, want to prevent error messages coming from postmaster@anywhere.org from being posted to your list. Or perhaps you have a particularly annoying individual whose messages you want filtered out. By putting these addresses in the ignored file you prevent anything coming from these addresses from going out to your list.

Two very important files are the welcome and info files. The welcome file is a message automatically sent out to all new subscribers to a list. The info file is sent out to anyone requesting information from ListProc about your list. Both files should give the list name, names and e-mail addresses of list owners or moderators, purpose of the list, and any special rules and regulations pertaining to the list. The welcome file should additionally contain some welcome message.

The subscribers file contains a list of all subscribers to a list, their e-mail addreses, names, and switches for their subscriptions such as concealed, ack, noack, digest, etc. The subscribers file is one file that can be EDITed or can be modified by using ListProc's internal commands such as **`ADD, DELETE, SET,`** etc. The subscribers file *must* be in a specific format. If you chose to edit that file and then replace it with an edited version and if your text editor makes a minor reformatting such as wordwrap or if you make a small error in one character in your editing, then the newly placed subscribers file will not only not work, it will cause ListProc to crash. For this reason it is important that you not try to edit the subscribers file, but instead use ListProc's commands to modify the subscribers file.

Lists can be gatewayed to newsgroups; that is, postings to a particular newsgroup will be forwarded to a mailing list and postings to the mailing list will be placed in the newsgroup. One way gateways are also possible in which postings to the list go to the newsgroup but no postings to the newsgroup will go to the list. Or the opposite one way gateway can be set up allowing only postings from newsgroup to list. The news file contains the information necessary for establishing the gateway between newsgroup and list. Peer lists are mailing lists that are subscribed to one or more mailing lists at local or remote sites. They handle local distribution of messages just like any other list and also distribute messages that originate in the peer lists. The peers file is a special file which coordinates the peering of lists. The syntax and use of peer files and news files is beyond the scope of this owner manual and can be found in the ListProcessor site manager manual. It is suggested that the peers and news files should be edited only by the ListProc system manager, not by list owners.

5. Configuring Your List From the Beginning.

The first thing you need to do when your list is set up by the system administrator is configure it and create WELCOME and INFO files. This section will discuss some of the more important issues to consider when configuring your list and refer you to sections of the Command Reference section for commands related to each item discussed.

a) It is important to have WELCOME and INFO files in most cases. The INFO file serves the purpose of informing potential users what your list is all about. The INFO file should have a full description and any restrictions imposed on the list membership. Potential subscribers should be able to get the INFO file and decide upon reading it whether they want to subscribe to your list. The WELCOME file should also tell what the list is about but additionally should give instructions for how to get in touch with list owners and/or moderators and give any rules and regulations of membership in the list. For creating INFO and WELCOME files see section 6.A.1 **EDIT** and section 6.A.2 **PUT**.

b) Do you want your subscriptions to your list to be open, closed, or owner-controlled? Open means that anyone can subscribe to your list without needing your prior approval. Subscription is by simply sending a message to the ListProcessor asking to be added to the list. Owner-controlled means that all subscription commands from users get forwarded to someone who you designate as your subscription manager. If you, the owner, do not designate a subscription manager, then you, the owner, become the subscription manager by default. Closed means that no one can submit a subscribe command to your list. You, the owner, may add people to your list but if anyone tries to send a subscribe command they will get back a message saying that the list is closed to subscriptions. For configuring your list as OPEN, CLOSED, or OWNER-CONTROLLED see section 6.x.x **CONFIGURATION *listname password* OPEN-SUBSCRIPTIONS**, section 6.x.x **CONFIGURATION *listname password* CLOSED-SUBSCRIPTIONS**, and section 6.x.x.x **CONFIGURATION *listname password* OWNER-SUBSCRIPTIONS**.

c) Do you want your list to be moderated or unmoderated? Moderated means that all messsages sent to the list by subscribers must be approved by the list moderator before anyone else can see them. There are two types of moderation. In moderated-no-edit the moderator is presented with the message and a tag number. The moderator may then approve or discard the message as is. In moderated-edit the moderator may not only approve the message but also may edit it before sending it back to be posted to the list. Unmoderated means that all messages sent to the list automatically get forwarded to everyone subscribed to the list without the moderator's approval. If you, the owner, do not designate a list moderator, then you, the owner, become the subscription manager by default. For configuring your list as moderated or unmoderated see section 6.B.x.x **CONFIGURATION *listname password* MODERATED-EDIT**, section 6.B.x.x **CONFIGURATION *listname password* MODERATED-NO-EDIT**, and section 6.B.x.x **CONFIGURATION *listname password* UNMODERATED**.

d) Do you want your list to be hidden or visible? Hidden means that anyone requesting a list of all lists known to the server will not see your list. Visible means that people requesting information about lists supported by your server will see your list. You can have a visible list and still restrict the amount of information people can obtain about your list. For configuring your list as hidden or visible see section 6.B.x.x **CONFIGURATION *listname password* HIDDEN LIST** and section 6.B.x.x **CONFIGURATION *listname password* VISIBLE-LIST**.

e) How much information can people get about your list? Every list has an info file, a one line description, a series of attributes or settings, and a list of its subscribers. You can restrict access to each individual piece of this information, either only to list subscribers, to the general public, or to no one. To control who can revew your list see section 6.B.x.x **CONFIGURATION *listname password* REVIEW-TO-** To control who can obtain obtain your list statistics see section 6.B.x.x **CONFIGURATION *listname password* STATISTICS-TO-** To determine whether anyone can see a list of your subscribers see section 6.B.x.x **CONFIGURATION *listname password* DEFAULT CONCEAL YES|NO**

f) Will there be archives of your list, and if so, who will have access to list archives? In addition to archives of material sent out by your list you can also have archives of supplementary material. You control access and determine whether a password is needed to access these materials. For information on configuring archives of the list messages see section 6.B.x.x **CONFIGURATION *listname password* ARCHIVE** To determine who can get access to the archives see section 6.B.x.x **CONFIGURATION *listname password* ARCHIVES-TO-**

g) How much information do you want to get about your list's operation? Getting list information and error messages is discussed in section 6.B.x.x **CONFIGURATION *listname password* DEFAULT PREFERENCES** and in section 6.F.x **SET PREFERENCES** with the following options for the preferences:
1. If your list is open, do you still want to know about all new subscribers? **CCSUBSCRIBE CCUNSUBSCRIBE**
2. Every time someone sends a request for a list of recipients, for information, for list statistics for a review of the list or a run command, do you want to receive a copy of the listproc's response? **CCREVIEW CCSTATISTICS CCINFORMATION CCRECIPIENTS**
3. Everytime a message comes in from someone in your 'ignore' file or someone not subscribed to your list if private, do you want to know? **CCIGNORE**
4. Do you want copies of all error messages? **CCERRORS CCALL**
These questions must be considered seriously because the more information you request, the more mail you will be getting. If you want all of these informational messages and you have an active list, you could get hundreds of informational messages filling up your mailer each day. You can designate someone else as recipient of all error messages if you want error messages monitored but cannot take the time to do it yourself. To designate someone as recipient of error

messages see section 6.c.x.x **CONFIGURATION *listname password* DELIVERY-ERRORS-TO *address***

g) Is your list going to be mailed out as a digest or not? A digest is a group of many messages sent out together in a single 'digest' message and has the advantage of not filling up your users' mailboxes with large amounts of mail from an active list. On the other hand, it delays the sending of individual messages until enough messages have been received to make up a digest. In a less active list this may be undesirable. To determine whether your list is sent out as a digest or not, see section 6.B.x **CONFIGURATION *listname password* DIGEST *frequency when***

Keep in mind when a new list is set up that listproc has some built in defaults. These defaults may be changed by the listprocessor manager.

ListProc defaults are listed below with short explainations. Longer explainations can be found in the various command sections below dealing with each of these options. Listproc distribution defaults are:

stats-to-all	The general public can request list statistics
review-to-all	The general public can review a list’s settings
auto-delete	Automatically delete users whose mail bounces
digest-daily	Digests are sent out daily if mailmode digest is chosen
send-by-all	Anyone can send mail to the list whether subscribed or not
no-archives	The list is not archived
open-subscription	Anyone requesting a subscription to the list is added

Everything is open, that is the default list allows maximum access to all users. If you want a restricted list you must change those defaults.

5. Command Reference.

Here is a complete description of the set of list management-related commands recognized by ListProcessor and examples of how to submit commands to the listprocessor. List managers should also obtain the User Reference for a brief description of user commands and the User Manual for more detailed explanation of user commands.

Everything appearing in [] below is optional; everything appearing in <> is mandatory; all arguments are case insensitive. The vertical bar ("|") is used as a logical OR operator between the arguments. Please note that the brackets or braces or parentheses themselves are NEVER a part of the command. In the syntax examples below, the word "list" must be replaced by your list name, the word "password" must be replaced by your list password, the word “options” must be replaced by any options for the command, and the word “args”, short for arguements, must be replaced by any additional arguements that must be added after the options. Commands may be abbreviated, but you must give at least as many characters as needed to distinguish the command from other commands or at least enter the first three characters.

All lists have four levels of management commands: (1) the listprocessor administrator, (2) list owners, who have ultimate control over the list,(3) list moderators, who do the day-to-day list management, i.e, receivemessages to REVIEW, APPROVE, DISCARD, or FORWARD reviewed messages, and(4) subscription managers and error message recipients, both of whom have access only to commands relating to subscriptions to the list; specifically,they can ADD, DELETE, SET ...FOR, and REVIEW; however, error message recipients do not receive commands to subscribe to closed lists which subscription managers receive.

Any command which spans more than one line must have an ampersand (&) at the end of each line to indicate that the command is continued on the next line.

Recognized commands are:

A. List configuration commands
   EDIT
   PUT
   REPORTS
   CONFIGURATION

B. Commands affecting list subscriptions
   ADD
   ALIAS
   DELETE
   IGNORE
   LOCK
   SET
   SYSTEM
   UNLOCK

C. Commands affecting posting of messages
   APPROVE
   DISCARD
   HOLD
   FREE

D. File archives as lists
   AFD
   FUI

A. List Setup Commands:

EDIt *list password filename* [-nolock]

Obtain specified file for editing; filename can be one of:

subscribers, aliases, news, peers, ignored, info, or welcome. (For further explaination of these files, see section on Special Files that ListProc Uses above.) List will be automatically locked to list-specific commands until the list is UNLOCKed, a new file has been PUT, or **`-nolock`** is specified in the **`EDIT`** command. It is strongly recommended that list owners wanting to use the **`EDIt`** command on the subscribers file do so only for the purpose of examining the file and not for the purpose of modifying it. Small errors in modifying the subscribers file will cause ListProc to crash, bringing down *ALL* local lists. When examining a file, the -nolock option should be used.

Examples:

Let's assume there is a list called "mylist" and the owner's list management password is "beqrt". The owner of the list wants to edit the info file. The owner of "mylist" sends the following commands:

ed mylist beqrt info	The info file from "mylist" will be sent to the owner; the list is locked until the info file is replaced or unlocked with an unlock command.

The owner of mylist now wants to modify the welcome file. The owner of "mylist" sends the following commands:

ED mylist beqrt welcome -nolock	The welcome file from "mylist" will be sent to the owner; the list is not locked.

`PUT` ***`list password filename`*** **`[`*`args`*`]`**

Replace system files. Filename is as in EDIT command. These files are obtained with the EDIT command, can then be edited and replaced with the PUT command. If the PUT command is for an alias or ignore file the [args] consists of the address(es). When PUTting a file, its contents start immediately after the command and span the entire message. No other commands can follow a PUT and no signature should be in a PUT command. Neither an alias nor an ignore file can have comments in the file. It is strongly recommended that list owner *NOT* use the PUT command on a subscribers file because of damage that may be done to ListProc from an improperly edited subscribers file. All modifications desired to the subscribers file can be easily and safely done using ListProc's internal commands such as ADD, DELETE, SET, etc.

Examples:

Let's use the example from the EDIT command above of a list called "mylist" and the owner's list management password is "beqrt". The owner of mylist now wants to replace the info and welcome files with new ones. The owner of "mylist" sends the following commands separately:

`put mylist beqrt info`
Mylist is a discussion of ideas related to what I think about life.

`PUT mylist beqrt welcome`
Welcome to Mylist. We will be discussing life in general and in particular.

In the first example above the list's info file will be replaced with a file saying that "Mylist is a discussion of ideas related to what I think about life." In the second example the list's welcome file will be replaced with a file saying "Welcome to Mylist. We will be discussing life in general and in particular." Notice that these are sent as two separate commands in two separate e-mail messages, and not in a single e-mail message. This is because all information put into the message after the command line is incorporated into the file. If the owner's mailer automatically appended the owner's signature to the message then that signature will be included in the file being placed.

REPorts *list password*

Obtain all reports about the specified local list.

CONfiguration *listname password* [*option* [*args*]] &
[,*option* [*args*] ...]] [,*option* [*args*] ...]]

Set a list's configuration options. The command, **CONFIGURATION**, without options, returns the list's current settings. If the configuration command spans more than one line then each line must end in an ampersand (&) in order to indicate that the command is continued on the next line. Options are a comma-separated list of one or more of the following keywords:

ARChive [*password*] [messages|digests]
NO-ARChive

Turn on/off archiving of lists. Specify a password for subscribers' access to the archives and whether single messages are to be archived or digests. The command ARCHIVE with no options results in non-password protected archive with default archiving of individual messages. If the owner wants to specify digests or messages without a password, a dash (-) is used in place of the password. If the owner wants to remove a password and not replace it, an empty string delimited by single or double quotes ("" or '') will remove it. The directory, archive name, file name of the archived material, and whether archives are stored compressed or not are set by the list manager by request from the list owner. By default every list's messages are stored in the default archive directory with the archive named after the list and no password. File names can be any letters of the alphabet, numbers, the percent character, and the underscore character. Naming may be by any one of a number of character sequences:

- -The contents of a ARCHIVE-NAME header line. This method cannot be used if archiving digests.
- -The current message count.
- -Any combination of day of month, week of month, month name, and year.
- -Any combination of issue number and volume number, which is extracted from a line saying Volume # Number #.
- -The first word of the first non-blank line of the file.
- -Digest number.

Example:

For a list named "mylist" whose owner password is "foo"

```
configuration mylist foo archive erty messages
```

will cause the list "mylist" to be archived as individual messages which can be obtained from ListProc with a **GET** command (See **GET** below) using the password erty.

```
configuration mylist foo archive - digest
```

will cause the list "mylist" to be archived as digests according to the default digesting (See **DEFAULT** below) with no password required to obtain copies of the archived material.

ARCHIVES–TO–ALL

ARCHIVES–TO–OWNERS

ARCHIVES–TO–SUBSCRIBERS

Specify who can request material from the archives. ALL: anyone can request index and archive material. OWNERS: only owners may request archive material and index. SUBSCRIBERS: only owners and subscribers may request index and archive material.
Example:
For a list named "mylist" whose owner password is "foo"

configuration mylist foo archives–to–subscribers

will allow only subscribers to the list "mylist" to obtain archives of the list.

AUTO–DELETE–SUBSCRIBERS

NO–AUTO–DELETE–SUBSCRIBERS

Listprocessor has the capability of automatically deleting from a list any users whose mail bounces. This allows for automatic removal of subscribers who have lost or turned off their e-mail accounts, but did not unsubscribe from your lists, or people who for some reason their e-mail addresses become unreachable. These commands turn on and off automatic deletion of subscribers whose mail bounces. When a user is automatically deleted the list owner is notified. If someone other than the list owner is defined as a recipient of system error messages, then that person is notified of the automatic removal of a user.
Example:
For a list named "mylist" whose owner password is "foo"

configuration mylist foo auto–delete–subscribers

will allow ListProc to automatically delete any subscriber whose mail bounces. If a list is set to **auto–delete–subscribers** then the list owner should be aware of the fact that some list subscribers will occasionally complain of not having received any mail from the list recently. Users frequently are not aware when their home system is rejecting mail. Deleting subscribers whose mail bounces is a useful function because bounce mail takes up bandwidth and wastes cpu time. On a small machine with a slow connection to the Internet, large amounts of bounce mail can significantly slow down service on all the machine's local lists.

CLOSED–SUBSCRIPTIONS

OPEN–SUBSCRIPTIONS

OWNER–SUBSCRIPTIONS

When closed, the list will not accept any new subscribers at all. However, the owners may still submit an **ADD** command. When open, the list will automatically accept all people who send in subscribe requests. Although the owner does not have to approve new subscribers in an open list, the owner can receive notification of new subscribers added to a list; see **SET ccpreferences**. **OWNER–SUBSCRIPTIONS** makes a list private. When a user sends a subscribe command for the list to ListProc a message is sent to the subscription manager notifying of the request for subscription. Subscriptions are then approved by the designated people, either subscription managers or owners. **OWNER-SUBSCRIPTIONS** turns on ARCHIVES-TO-SUBSCRIBERS, REVIEW-TO-SUBSCRIBERS, STATISTICS-TO-SUBSCRIBERS, and SEND-TO-SUBSCRIBERS, all limiting access to information from the list to subscribers.

Example:
For a list named “mylist” whose owner password is “foo”

configuration mylist foo open-subscriptions

will allow ListProc to automatically add anyone sending a subscribe request without obtaining approval from the list subscription manager or owner.

configuration mylist foo closed-subscriptions

will cause ListProc to send a message to the person requesting a subscription to the list saying that no new subscriptions are being accepted at this time.

configuration mylist foo owner-subscriptions

will make a list private to subscriptions. The owner or designated subscription manager will have to manually **ADD** new subscribers.

WIDE-OPEN-LIST

Make a list visible, allows subscribers to add themselves to a list, allows anyone to post messages to the list, and allows anyone to request a review, statistics, or archives from a list.
Example:
For a list named “mylist” whose owner password is “foo”

configuration mylist foo wide-open-list

has the same effect as sending a command to ListProc saying
configuration mylist foo open-subscriptions, unmoderated, & review-to-all, statistics-to-all, archives-to-all.

COMMENT string

NO-COMMENT

Sets the one line list comment (description) string. To remove the comment send an empty string delimited by single or double quotes ("" or ''). Note that only the following characters are permitted in the comment string: All the alphabetic letters from a - Z including both upper and lower case, all the numbers from 0 - 9, and the following characters **[\ t + = : ; ' . , @ # % ! _ -]** while comment strings may ***NOT*** contain the following characters: **< > ` * ? , \ n**
(Please note that in this discussion the symbol **\t** is a combined symbol which stands for the tab character and not the separate characters slash-t. Likewise **\n** is a combined symbol which stands for the newline character.) If a comment string is to have a colon “:” then the colon must have an escape character “\” before it “\:”
Example:
For a list named “mylist” whose owner password is “foo”

configuration mylist foo comment MyList\: This is my list

The listing of all lists will say “mylist@machine.address MyList: This is my list”. Additionally all mail coming from “mylist” will have the comment appended to the header, “MyList: This is my list”.

DEFAULT *mailmode option* [*mailmode option*]

Set the default subscription options. Mailmode choices and options are:

`DEFAULT` *`address`* `variable|fixed`

- **`variable`** allows a subscriber to change addresses with a command sent from a different address than the one the user is subscribed from.
- **`fixed`** requires a subscriber to unsubscribe from the subscribed address and resubscribe from the new address.

`DEFAULT mail ack|noack|postpone|digest`

- ack means a subscriber will receive a copy of his/her posting.
- noack means a subscriber will not receive a copy of his/her posting.
- postpone means all mail to a subscriber will be held until the subscriber sends a command changing mailmode to something else.
- digest causes all messages to be sent out concatenated together in groups. The user can change this by sending a command requesting a different mailmode. For example, changing mailmode to ack will turn off digests for the individual user. In contrast, if the list is set to sending out all mail as digests because it was set that way using the **DIGEST** ***frequency*** command, then the individual user will be unable to change that designation.

`DEFAULT` ***`password string`***

- sets a default user password. If not set a random password is assigned to new users.

`DEFAULT conceal yes|no`

- **`yes`** allows other people to see the user's name and e-mail address in a list of subscribers to a list.
- **`no`** prevents others from seeing user's s name and e-mail address.

`DEFAULT preference` ***`CCoption`***

- determines which commands sent by users are copied to owners. ***`CCoption`*** can be one of: **`CCUNSUBSCRIBE, CCRECIPIENTS, CCINFORMATION, CCSTATISTICS, CCPRIVATE, CCRUN, CCIGNORE, CCERRORS, CCREVIEW, CCALL`**

When any user sends a command to the listprocessor concerning a list the owner can be set up to receive a copy of the response. This is useful if owners want to keep track of who subscribes, unsubscribes, requests information, etc. or error messages sent out to users. The more ***ccoptions*** set, the more mail a list owner or designated recipient of ccoptions will receive. It may not be advisable to set **ccall** on a very active list because of the very large volume of mail that may ensue. The 'default preference' only effects owners that are added after the change is made, current owners must change their options using the **SET preference** command (see below).

A mailmode with an empty string as an option, designated by either two double quotes("") or two single quotes(''), causes the mailmode to revert back to the system default. The user can also revert to default with a **`SET`** ***`mailmode`*** command and no arguements.

Examples:
For a list named “mylist” whose owner password is “foo”

configuration mylist foo default address fixed

This will prevent any user from changing a subscribed address from another machine; it forces users to unsubscribe from their subscribed machine and resubscribe from the new address.

configuration mylist foo default mail ack

This will cause ListProc to return a copy of all messages posted to a list by any user to that subscriber as a way of acknowleging to the user that the message was sent out to the list.

configuration mylist foo default mail digest

This setting will cause the ListProcessor to mail out all messages as digests in the manner specified by the **DIGEST** command (See **DIGEST** below). Users can override this by sending in a **SET *listname* MAIL ACK command** or any other **SET *listname* mail** command.

configuration mylist foo default mail noack

With this option set, ListProc will not return a copy of all messages posted to a list by any user to that subscriber unless the subscriber uses the **SET** command to set his/her mailmode to **ack**. The subscriber’s messages will, however, go out to everyone else on the list.

configuration mylist foo default conceal yes

With this option set, ListProc will conceal the names and addresses of all subscribers to a list. In the event someone sends a **REVIEW** command to the ListProc, they will receive back the number of subscribers only.

config mylist foo default preference ccsubscribe, ccunsubscribe

With this option set, ListProc will forward all subscribe and signoff (unsubscribe) requests from users to the list owner or person designated to receive error messages (See **DELIVERY-ERRORS-TO** below). When this is not set by the list owner or by the ListProc site manager, the system default is for **ccignore** only.

DELIVERY-ERRORS-TO *address* [*address*]

REMOVE-ERRORS-TO *address* [*address*]

Specifies to which addresses copies of error messages are to be sent, and removes recipients of error messages. If no address is designated then all error messages wil go to the list owner by default. "ERRORS-TO" recipients may send commands to ADD, DELETE, SET ...FOR, and REVIEW. This allows the recipients of error messages to take corrective action without having to bother the list owner.

Examples:
For a list named “mylist” whose owner password is “foo”

configuration mylist foo DELIVERY-ERRORS-TO george@somewhere

This sets the recipient of all error messages set in the **DEFAULT preference *CCoption*** command (See **DEFAULT preference *CCoption*** above) to george@somewhere. The list owner will continue to receive error messages unless the list owner removes him/herself from receiving such messages with the **REMOVE-ERRORS-TO** command.

configuration mylist foo REMOVE-ERRORS-TO harriet@somewhere

This will remove the person harriet@somewhere from receiving error messages.

DIGEST *frequency* [*when*] [*lines bytes*]

DIGEST daily *hh:mm* [*lines bytes*]

DIGEST weekly [*day-of-the-week*] [*lines bytes*]

DIGEST monthly [*lines bytes*]

NO-DIGESTS

Turn on/off collection of digests; define when they will be distributed. Requires specification of how frequently the digest should be sent out. Digests can be sent out daily, weekly, or monthly. If weekly, the day of the week can be specified. Digests are always sent out at midnight if the time is not indicated. In contrast, if the default for each subscriber is set to sending out all mail as digests because it was set that way using the **DEFAULT** ***mail digest*** command, then the individual user will be able to change that designation using a **SET** ***mailmode*** command. If lines and/or bytes are specified, digests will then be distributed at the specified time or whenever the number of lines exceeds the specified limit, or whenever the size of the digest exceeds the specified limit. If an owner wants to change the lines or bytes spec only, the owner may use the dash (-) as a place holder (one dash is enough for all cases) for frequency.

<u>Example</u>:

For a list named "mylist" whose owner password is "foo"

configuration mylist foo digest daily 00:01

will mail digests daily at one minute after midnight

configuration mylist foo digest weekly wednesday 06:00

will mail digests once a week on Wednesdays at 6 am

configuration mylist foo digest monthly 250 2000

will mail digests once a month but if the digest reaches 250 lines or 2000 bytes first it will be sent out. Additional material will be sent out when the month turns over as usual. Since a time is not specified the digest will go out at midnight.

configuration mylist foo digest - 100 1024

will not change the method of sending out digests or time but in the event the digest reaches 100 lines or 1024 bytes in size the digest will be mailed out immediately, not waiting for the designated time. The "-" is used as a place holder for frequency of mailing. To turn off the specs, owners may specify zero:

configuration mylist foo digest - 0 1024

This will send a digest out whenever it is supposed to go out, or every 1024 bytes (or more); no line-count restriction is set.

DISABLE *command* [*command*]

ENABLE *command* [*command*]

Enable or disable specific user commands. This applies to specific user commands such as `REVIEW` or `STATISTICS` but does not apply to user `SET` commands. Disable prevents users, but not owners, from issuing these commands.

<u>Example</u>:

For a list named "mylist" whose owner password is "foo"

configuration mylist foo disable review

will prevent all users, whether subscribed or not, from getting any output from a REVIEW command to ListProc on the "MyList" list.

configuration mylist foo disable subscribe

will prevent anyone from subscribing to "MyList". Anyone wanting to subscribe will have to send a message to the owner asking to be put on and the owner may add the person. However, this should not be routinely used as a means of keeping a list closed. The **closed-subscriptions** option is better used for this purpose.

SET-DISABLE *mode* [*value*] [*mode* [*value*]]

SET-ENABLE *mode* [*value*] [*mode* [*value*]]

mail ack|noack|postpone|digest
conceal yes|no
password

Enable or disable specific user SET commands. This applies to commands in which users set their mail mode such as SET MAIL ACK, conceal, or change their password. Owners can still SET ...FOR any commands which have been disabled.

Example:

For a list named "mylist" whose owner password is "foo"

configuration mylist foo set-disable conceal

will prevent all users from changing their visibility on a **REVIEW** command. If all users are concealed, then they will remain so. If all users are visible, then they will remain visible.

configuration mylist foo set-disable password

will prevent all users from changing their list passwords.

configuration mylist foo set-enable password

will re-enable changing of passwords on a list in which changing passwords was previously disabled.

FORWARD-REJECTS

DONT-FORWARD-REJECTS

Enable or disable forwarding of reject/error messages to the list owners instead of to the original senders. If **FORWARD-REJECTS** is on, then listprocessor forwards listproc-generated error messages to the list owners, not the message sender.

Example:

For a list named "mylist" whose owner password is "foo"

configuration mylist foo forward-rejects

will cause all user-generated error messages to go to the list owner or person designated as recipient of error message.

configuration mylist foo dont-forward-rejects

will allow all user generated error messages to go to the user whose command or message caused the error. The owner or designated recipient of error messages will also receive a copy of the error message.

KEEP-RESENT-LINES

DONT-KEEP-RESENT-LINES

When forwarded mail is sent to a list, enables or disables a header line indicating that the message is forwarded mail.

Example:

For a list named "mylist" whose owner password is "foo"

configuration mylist foo keep-resent-lines

When a user forwards mail to a list this will cause ListProc to add a line to the message header saying that the message was forwarded.

HIDDEN-LIST

VISIBLE-LIST

Determines whether a list is visible or hidden when listprocessor receives a command for a list of lists. **HIDDEN-LIST** turns on ARCHIVES-TO-SUBSCRIBERS, REVIEW-TO-SUBSCRIBERS, STATISTICS-TO-SUBSCRIBERS, SEND-TO-SUBSCRIBERS, all limiting access to information from the list to subscribers.

Example:

For a list named "mylist" whose owner password is "foo"

configuration mylist foo hidden-list

will prevent prevent anyone from seeing the list named "MyList" in a LISTS command. Also prevents anyone from requesting archives, a review of the list, statistics, or from posting to the list if they are not subscribed.

MAX-MESSAGES-PER-DAY *number*

or

MESSAGE-LIMIT *number*

NO-MESSAGE-LIMIT

Determines the maximum number of messages a list will process per day. Messages above the maximum will be held until the next day and processed, or will be sent if the list is FREEed. It is a good idea to set a message limit for all lists. If the owner does not want a message limit, the limit can be set to a high number. This way, if a message loop occurs there will be a limit to how many messages will get sent out in a single day before the owner discovers and corrects the loop.

Example:

For a list named "mylist" whose owner password is "foo"

configuration mylist foo message-limit 300

configuration mylist foo max-messages-per-day 300

will not allow more than 300 messages to go out from MyList in a single day. If more than 300 messages are posted to the list, all messages over 300 are stored until the following day before posting. The two commands, **message-limit** and **max-messages-per-day** are synonyms for each other.

MODERATED-EDIT *address* [*address*]

MODERATED-NO-EDIT *address* [*address*]

UNMODERATED

REMOVE-MODERATORS *address* [*address*]

Determine moderation of list. MODERATED-EDIT sends all messages to the moderator(s) for editing and approval. MODERATED-NO-EDIT sends all messages to the moderator(s) for approval and includes a tag identifier in the first line. Moderator sends back an approval command giving the tag identifier only. See APPROVE command. Moderators may be removed with the REMOVE-MODERATORS command. If a list is set up as MODERATED with a specified address as moderator and an additional MODERATED command is given, either MODERATED-EDIT, or MODERATED-NO-EDIT, with a new moderator address, the new moderator address is added to the pre-existing moderator address such that both addresses become moderators. In other words, the old moderator is not replaced. The only way to replace a moderator is to use the REMOVE-MODERATORS command to remove the moderator to be replaced. However, if a list is MODERATED-NO-EDIT and a MODERATED-EDIT command is given, then the list is changed, and vice versa. If a list is MODERATED but no address is specified, messages are sent to owners for approval. See APPROVE command.

Example:

For a list named “mylist” whose owner password is “foo”

configuration mylist foo moderated-no-edit george@somewhere

will set the list so that all messages posted to the list by any subscribers will go to george@somewhere for approval. A paragraph is appended to each message telling the moderator to send an approve command if the moderator wants to allow the message to be posted.

configuration mylist foo moderated-edit george@somewhere

will set the list so that all messages posted to the list by any subscribers will go to george@somewhere but then that moderator will be able to edit the messages and send them back to ListProc for posting to the list.

configuration mylist foo moderated-no-edit

will set the list to moderated without editing of messages but since no moderator was specified, the list owner becomes the default moderator of the list.

OWNERS *address* [*address*]

New owners may be added with the OWNERS command. This command does not replace owners, it only adds new owners. To replace owners, it is necessary to use both the REMOVE-OWNERS command and the OWNERS command.

Example:

For a list named “mylist” whose owner password is “foo”

configuration mylist foo owners george@somewhere

will add george@somewhere as an owner in addition to the current owner(s). It will *not* replace current owners; it only *adds* new owners.

REMOVE-OWNERS *address* [*address*] ...

List manager, or list owners, if list is OWNER-CONTROLLED, may remove owners.

Example:

For a list named “mylist” whose owner password is “foo”

configuration mylist foo remove-owners george@somewhere

will remove george@somewhere from being an owner of the list “MyList”.

PASSWORD *string*

Set or change the list management password.

Example:

For a list named "mylist" whose owner password is "foo"

`configuration mylist foo password gobbledygook`

will change the list management password from "foo" to "gobbledygook".

`SUBSCRIPTION-MANAGERS` ***`address`*** **`[`*`address`*`] ...`**

`REMOVE-SUBSCRIPTION-MANAGERS` ***`address`*** **`[`*`address`*`] ...`**

`REMOVE-ALL-SUBSCRIPTION-MANAGERS`

Add or remove subscription managers. The SUBSCRIPTION-MANAGERS command does not remove or replace the existing subscription managers, it adds new managers. To remove managers the REMOVE-SUBSCRIPTION-MANAGERS command must be used.

Example:

For a list named "mylist" whose owner password is "foo"

`configuration mylist foo subscription-managers george@somewhere`

will set george@somewhere as the subsciption manager for the MyList list. All requests for subscriptions will go to that address for approval. Then george@somewhere will have to use the **`ADD`** command to subscribe the individuals requesting subscription. If the list is open to subscriptions, however, the subsciption manager does not get requests because new subscribers may add themselves.

`configuration mylist foo subscription-managers george@somewhere`

will remove george@somewhere from being a subscription manager leaving the other subscription managers in place. If there are no other subscription managers then the list owner becomes subscirption manager by default.

`REPLY-TO-LIST`

`REPLY-TO-LIST-ALWAYS`

`REPLY-TO-SENDER`

`REPLY-TO-SENDER-ALWAYS`

`REPLY-TO-OMITTED`

Determines whether replies automatically go to the list or to the sender of the individual message. If list is a digest replies always go to the list. When a list is set as **`REPLY-TO-LIST`**, **`REPLY-TO-SENDER`**, or **`REPLY-TO-OMITTED`**, if the sender includes a Reply-To: in the message header it takes precedence. However, if the list is set as **`REPLY-TO-[LIST|SENDER]-ALWAYS`** then the user's inclusion of a Reply-To: in the message header will be ignored.

Example:

For a list named "mylist" whose owner password is "foo"

`configuration mylist foo reply-to-list`

will cause all replies to all messages to go to the list except if the sender of the reply adds a REPLY-TO: line to the message header.

`configuration mylist foo reply-to-sender-always`

will cause all replies to all messages to go to the sender of the message being replied to even if the person replying tries to put a REPLY-TO: line in the message header.

`configuration mylist foo reply-to-omitted`

will cause all replies to go neither to the list or the person replying. Usually replies go to the list owner.

REVIEW-BY-ALL

REVIEW-TO-ALL

REVIEW-BY-OWNERS

REVIEW-TO-OWNERS

REVIEW-BY-SUBSCRIBERS

REVIEW-TO-SUBSCRIBERS

Determines who can submit a REVIEW command for a list. See below for additional details.

STATISTICS-BY-ALL

STATISTICS-TO-ALL

STATISTICS-BY-OWNERS

STATISTICS-TO-OWNERS

STATISTICS-BY-SUBSCRIBERS

STATISTICS-TO-SUBSCRIBERS

Determines who can issue a command for list statistics. See below for additional details.

SEND-BY-ALL

SEND-BY-OWNERS

SEND-BY-SUBSCRIBERS

Determines who can post to a list.
ALL: Anyone, whether subscribed to a list or not. OWNERS: Limited to list owners. SUBSCRIBERS: Both subscribers and owners. Commands using the word TO or BY are equivalent.
<u>Examples</u>:
For a list named "mylist" whose owner password is "foo"

`configuration mylist foo review-by-all`

will allow anyone, whether subscribed to MyList or not, to send a **REVIEW** command to ListProc and receive the output.

`configuration mylist foo review-to-all`

will allow anyone, whether subscribed to MyList or not, to send a **REVIEW** command to ListProc to REVIEW MyList and receive the output. In all three commands allowing or disallowing a **REVIEW** command, **STATISTICS** command, or ability to **SEND** messages to the list, the **TO** and **BY** are synomyms and can be substituted for each other.

`configuration mylist foo review-to-owners`

will allow only the owners of MyList to send a **REVIEW** command to ListProc to REVIEW MyList. Subscribers and non-subscribers alike will get back an error message saying this command is not available.

`configuration mylist foo review-to-subscribers`

will allow only subscribers to MyList to send a **REVIEW** command to ListProc to REVIEW MyList. In all cases owners are always allowed to send a REVIEW command.

`configuration mylist foo statistics-to-all`

will allow anyone, whether subscribed to MyList or not, to send a **STATISTICS** command to ListProc to get use statistics for MyList.

`configuration mylist foo send-by-all`

will allow anyone, whether subscribed to MyList or not, to send or post messages to the list. However, non-subscribers will not receive any messages from the list.

```
configuration mylist foo send-by-owners
```

will allow only the owners of MyList to send or post messages to the list. This creates a one-way read-only list.

Review Examples for the **CONFIGURATION** Command:

All of the following examples assume a list called bajor-l located at somewhere.net. The owner's password is r45678.

```
CON bajor-l r45678 owners jim@machine.org sally@machine.org
```

For the list, bajor-l, adds two people as co-owners, jim@machine.org and sally@machine.org, giving their e-mail addresses. Notice that the configuration command is abbreviated as CON.

```
con bajor-l r45678 send-by-sub reply-to list review-by-subscr &
stat-by-owners arc shazam digests arc-to-all
```

In this example, for the list bajor-l, **send-by-sub** allows only subscribers to post to the list, **review-by-subscr** restricts issuing a review command concerning the list to subscribers. **reply-to list** directs all replies to messages sent from the list back to the list. **stat-by-owners** restricts sending a request for list statistics to owners only. **arc shazam digests** causes all messages sent out from the list to be archived as digests which can be requested by anyone (**arc-to-all**), whether subscribed to the list or not by giving the archive password **shazam**. Notice that while the previous example used CON in upper case, this example uses con in lower case. The commands are generally case insensitive. Notice also that because the command spans more than one line it has an ampersand (&) at the end of the first line to signify to the listprocessor that the command is continued on the next line.

```
Configuration bajor-l r45678 message-limit 1205 visible-list &
no-auto-delete-subscribers keep-resent-lines dont-forward-
rejects
```

In this example, for the list bajor-l, **message-limit 1205** sets a limit on the number of messages the listprocessor sends out per day to 1205 messages; once 1205 messages have been sent out, any additional messages will be held until the next day. The owner here has set an extremely high limit which is unlikely to be met unless the list is echoing bounces back to the list. In this case the echo of bounces will continue until the 1205 message limit is reached or the owner or list manager stops the resending of these messages. On the other hand subscribers whose mail bounces will not be deleted from the list automatically (**no-auto-delete-subscribers**). **keep-resent-lines** tells the listprocessor to keep the lines saying "Forwarded " in all messages which are forwarded by subscribers to the list. **dont-forward-rejects** causes the listprocessor to send rejected messages back to the sender of the messages instead of forwarding them to list owners or moderators.

```
configuration bajor-l  r45678
```

As mentioned previously, the configuration command without any options results in a list of the configuration options as they currently are set for the list. Below is a sample response:

```
Configuration of ListProcessor list bajor-l@somewhere.net

VISIBLE-LIST
OWNER-SUBSCRIPTIONS
SUBSCRIPTION-MANAGERS [owners]
SEND-BY-SUBSCRIBERS
STATISTICS-TO-SUBSCRIBERS
REVIEW-BY-SUBSCRIBERS
ARCHIVES-TO-SUBSCRIBERS
NO-ARCHIVE
UNMODERATED
DELIVERY-ERRORS-TO [kira@machine.org]
DIGEST daily 00:01 0 0
NO-MESSAGE-LIMIT
COMMENT Ongoing discussion of politics on planet Bajor
AUTO-DELETE-SUBSCRIBERS
DONT-FORWARD-REJECTS
REPLY-TO-LIST
KEEP-RESENT-LINES
OWNERS kira@machine.org jim@machine.org sally@machine.org
PASSWORD r45678
OWNER-CONTROLLED
```

Compare the response above with the response a user gets when sending a review command to the listprocessor. The review command returns almost the same information but in a format that is simple to understand for the user. The configuration command with no options returns the information in a format that makes it easier for the list owner to decide on options to change.

`review bajor-l short` returns the following response:

```
***
*** list bajor-l@somewhere.net: Ongoing discussion of politics on planet Bajor
***
***  Date created: Sun Sep 11 03:39:41 1994

--- The current list settings are as follows:

PRIVATE: subscriptions controlled by jim@machine.org sally@machine.org .
SEND: open to subscribers and owners only.
VISIBLE: the list shows up in listings.
NO-ARCHIVE: no logs are kept.
STATS: open to subscribers and owners only.
REVIEW: open to subscribers and owners only.
ARCHIVES: available to subscribers and owners only.
UNMODERATED: postings not controlled.
DIGEST: digests distributed daily at 00:01
MESSAGE-LIMIT: unlimited daily postings.
FORWARD-REJECTS: no; all listproc-generated errors sent to sender.
REPLY-TO-LIST
AUTO-DELETE-SUBSCRIBERS: yes.
KEEP-RESENT-LINES: yes; Resent- header lines preserved.
DELIVERY-ERRORS: non-delivery reports are sent to the owners.
OWNERS: kira@machine.org jim@machine.org sally@machine.org


bajor-l is a private, closed list used by the Bajoran council to discuss local politics.
Minutes of the meetings of the council are posted to the archives.
```

In addition to returning the same information as the configuration command, the review command returns the info file and, if the review command is not a "**`review listname short`**" command, then it also returns the list of non-concealed subscribers.

End of options for **`CONFIGURATION`** command.

II. Commands Affecting List Subscriptions

`[quiet] ADD` ***`list password address user-name`***

or, for adding multiple user-names:

`[quiet] ADD` ***`list password {address user-name} &`***

`{ address user-name }{ address user-name }`

Add specified user(s) to the list. If more than one user is added, enclose each one in brackets {}. Multiple names may be placed on one line or place one name per line in brackets. Commands spanning more than one line require an ampersand (&) at the end of each line to indicate that the command is continued on the next line. If the optional "quiet" switch is added before the command line, the user(s) will not be notified. When adding a user to a list only the following characters are permitted in a user's address or user-name:

All the alphabetic letters from a - Z including both upper and lower case, all the numbers from 0 - 9, and the following characters **[\ t + = : ; ' . , @ # % ! _ -]** while user names may ***NOT*** contain the following characters: **` " < > [ ] { } | $ ~ * ? ! **
(Please note that in this discussion the symbol **\t** is a combined symbol which stands for the tab character and not the separate characters slash-t.)

Although a user-name can be as many words as the user wants, the portion that shows up in a review command is limited to five words including the e-mail address.
Examples:
For a list named "mylist" whose owner password is "foo"

`add mylist foo ht56@springfield.org Albert Parker`

will subscribe Albert Parker to MyList with the address ht56@springfield.org and Albert will receive a welcome message from ListProc.

`quiet add mylist foo trans@springfield.org MaryLou Parker`

will subscribe MaryLou Parker to MyList with the address trans@springfield.org but MaryLou will *NOT* receive a welcome message or any indication of having been added to the list from ListProc because in this case the **`quiet`** option was used.

```
add mylist foo {ht56@springfield.org Albert Parker} &
{trans@springfield.org MaryLou Parker} &
{fort@machine.org The National Janitorial Crew}
```

will subscribe Albert Parker, MaryLou Parker, and The National Janitorial Crew to MyList with their respective addresses. All will receive a welcome message from ListProc. Notice that the subscriber from fort@machine.org used a four-word name for the subscription. ListProc will accept up to five separate words in a subsciption, one word is used for the address of the subscriber; the other four words can be used for the subscriber's name. Note that the ampersand (**&**) is used here to indicate that the **`ADD`** command is continued on the next line.

`ALIas` ***`list password new-address address-as-subscribed`***

Alias an existing subscriber to a new address. Used in cases where a subscriber needs to be able to send mail from another machine or the subscriber's email address can appear in multiple formats. Mail will be accepted from both the address-as-subscribed and from the new-address. The new-address pattern may be an extended standard

UNIX regular expression. For more information on regular expressions, see the section on that subject at the end of this file.

Examples:

For a list named “mylist” whose owner password is “foo” the user ht56@nebbuch.org has informed the list owner that occasionally she would like to be able to send commands to ListProc from another machine but doesn’t want to have to subscribe from that machine also. The list owner complied by setting up an alias for the second account.

alias mylist foo ret@kabb.net ht56@nebbuch.org

so that all mail from the second address, ret@kabb.net, is seen by ListProc as being from ht56@nebbuch.org. Using regular expressions, the list owner can map a whole range of e-mail addresses to aliases. For example, a second user has complained that she has an account on a system called whatsoever.net which has 3 machines networked together. Mail can be addressed to her at susie@whatsoever.net. However, when she logs into her account on whatsoever.net she can randomly log into any one of the three machines and her return address can be any one of the 3 machine addresses. The 3 machines are named red, blue, and green. So when she sends mail her message header can say that the mail is from susie@red.whatsoever.net or susie@blue.whatsoever.net or susie@green.whatsoever.net. This results in ListProc having her subscribed to MyList from only one of the three machines and when she is logged into either of the other two machines her commands to ListProc are rejected. This situation is very bothersome for her and she asks the list owner if he can do anything to help her. The list owner responds by sending to ListProc the command

alias mylist foo (.+)@.*\.whatsoever.net \1@whatsoever.net

The addresses being given to the **ALIAS** command are regular expressions. The subject of regular expressions is covered in another section later in this manual. The end result of the above command is that whenever any mail comes in from any user whose address ends in “whatsoever.net” the person’s username is attached to the string “@whatsoever.net” and the first portion, either red, blue, or green, is stripped off.

[quiet] DELETE *list password address* [*address*]

Delete the specified user(s) from the list. Multiple names may be placed on one line. If the command spans more than one line an ampersand (&) must be placed at the end of each line to indicate that the command is continued on the next line. If the optional "quiet" switch is added before the command line, the user(s) will not be notified.

Examples:

For a list named “mylist” whose owner password is “foo”

delete mylist foo ht56@springfield.org

will remove ht56@springfield.org from MyList and this user will receive a notification that he has been dropped from the list.

quiet delete mylist foo trans@springfield.org

will delete trans@springfield from MyList, but this user will not receive any notification of being dropped because the **quiet** option was used on the command.

delete mylist foo ht56@springfield.org trans@springfield.org &

fort@machine.org

will remove, or unsubscribe, all three users from the list. All will receive a notification message from ListProc that they have been removed from the list. Note that the ampersand (&) is used here to indicate that the **DELETE** command is continued on the next line.

IGNore *list password address*

Add a user to a file of troublesome users whose mail to a list should be discarded. The address pattern may be an extended standard UNIX regular expression. For more information on regular expressions, see the section on this topic at the end of this file.

Examples:

For a list named "mylist" whose owner password is "foo" a user has been sending abusive messages to the list and the owner wants ListProc to ignore anything coming from this user. The owner sends ListProc a command saying:

```
ignore mylist foo ht56@springfield.org
```

which directs ListProc to totally ignore anything sent to it from ht56@springfield.org. The list owner is notified that this person is still sending messgaes to the list but none of the messages sent gets posted to the list. Using regular expressions, the list owner can have ListProc ignore a whole range of e-mail addresses. For example, suppose the list MyList is being run for the benefit of only people from the University of Southern Madagasgar. The domain for that organization is usmaf.edu. By sending the command:

```
ignore  mylist foo ~<.*\.usmaf.edu>
```

the list owner instructs ListProc to ignore any address that doesn't end in "usmaf.edu". The subject of regular expressions is covered in a section later in this manual.

LOCk *list password*

Suspend execution of list-specific commands and queue them up for later processing. Owners may still issue such commands, unless the list is locked by the listprocessor manager. The list will still process messages.

UNLock *list password*

Resume execution of list-specific commands, including those queued up while the list was locked. All owners may unlock a list, unless it's locked by the listprocessor manager.

Examples:

For a list named "mylist" whose owner password is "foo"

```
lock mylist foo
```

will prevent all user commands from being sent concerning the list called MyList. Messages posted to the list will still go out. Commands sent to ListProc concerning MyList will be held until the list is unlocked.

```
unlock mylist foo
```

will unlock the list and cause any commands being held from users to be executed. Anytime a list owner sends an **EDIT** command without using the **–nolock** option, the list will be locked until the list owner sends an **UNLOCK** command or **PUT**s the file that was **EDIT**ed.

[quiet] SET *list* [*option arg*[*s*]] for *address* [*address*]

Allows list owner to **SET** subscriber mail options for the subscriber(s). If quiet, the user(s) will not be notified. The **SET** command also allows the list owner to SET the amount of list control messages that are sent to the person who is designated to receive error messages.
Valid options and args are:

[quiet] SET *list* mail ack|noack|postpone|digest

- **ack | noack** determines whether sender of message gets a copy of his/her posted messages back from the list. When ack is set, the user gets a copy of all mail sent to the list. When noack is set the sender does not get a copy of his/her own mail sent to the list.
- **postpone** causes user's mail to be held until the user releases it. This is useful if a user goes on vacation for a period of time. When the user releases the mail by sending another SET command all held mail is sent.
- **digest** causes a user's mail to be sent collected into a digest. Certain places like CompuServe charge for mail on a per-message basis. In this case a user may want to have all mail from a list sent in a digest instead of separate messages.

[quiet] SET *list password current-password new-password*
change list password for user.

[quiet] SET *list address password new-address*
change the address by which the list knows a user. Using the SET address command you can change the address to which your user receives mail from the list.

[quiet] SET *list* conceal yes|no
hide user name in commands for lists of subscribers.

[quiet] SET *list* preference *CCoption*

- determines which commands sent by users are copied to owners. When a user sends any command to ListProc concerning your list a copy of the output from ListProc will come to either the list owner or the designated recipient of error messages or both. ***CCoption*** can be one of:

CCSUBSCRIBE - The list owner gets a copy of all subscription requests.
CCUNSUBSCRIBE - The list owner gets a copy of all unsubscribe requests.
CCRECIPIENTS - The list owner gets a copy of all requests for list recipients.
CCINFORMATION - The list owner gets a copy of all requests for information.
CCSTATISTICS - The list owner gets copied on all statistics requests.
CCPRIVATE -
CCRUN - The ListProc manager gets copied on all run requests.
CCIGNORE - The list owner gets all messages sent by people on the ignore file.
CCERRORS - The list owner will receive a copy of all error messages.
CCREVIEW - The list owner will be copied on all review requests.
CCALL -The list owner and/or ListProc manager gets copied on all of the above. Be aware that this option will generate *a lot* of mail.

SET *list* preference *CCoption* is used to make changes from the defaults which are set by the owner using the '**configuration *listname password* default preference**' command as detailed above.

<u>Examples</u>:
For a list named "mylist" whose owner password is "foo"

SET mylist foo mail digest for mark@where.ever.com

will tell ListProc to send all mail from the list called MyList to user mark@where.ever.com as a digest instead of single messages. The user can, of course, send a **SET mail digest** command himself but if either the set command is disabled by the owner or the user is so unsure of himself that he asks the list owner to do it for him this allows the list owner to do it.

SET mylist mark@last.address foo mark@where.ever.com

will tell ListProc to change the address for mark@last.address in the list called MyList to mark@where.ever.com. This would be in the event that user mark@last.address suddenly lost his account and was unable to unsubscribe from the list. If mark@last.address remembered his list subscriber password, he might have been able to change his address on the list himself using the **SET *list-name password old-address new-address*** command which is detailed in the ListProc User Manual.

SET mylist preference ccsubscribe ccunsubscribe ccerrors

will tell ListProc to send all requests to subscribe or unsubscribe and all error messages to the list owner or other designated recipient of error messages. (See the section describing the **Configuration delivery-errors-to** command above.)

SYStem *list password user-address #user-command*

Allows the list owner to issue any command on a user's behalf. This command is a vestigal command from previous versions of ListProc and it has been replaced by other commands. Its presence in this version of ListProc is only for the convinence of list owners who are accustomed to previous versions of ListProc.

III. Commands Affecting Posting of Messages to a List

APProve *list password tag* [*tag*][*tag*][*tag*].... [*tag*]

DIScard *list password tag* [*tag*][*tag*][*tag*].... [*tag*]

If the list is set up as MODERATED-NO-EDIT, discards or approves the message identified by the tag number for posting to the specified moderated list. Both APPROVE and DISCARD will accept multiple tags on one line or on several lines, each ending in an ampersand (&) except for the last line.

Examples:

For a list named “mylist” whose owner password is “foo”

APPROVE mylist foo 124 749 325
DISCARD mylist foo 097

on a list which is moderated-no-edit will tell ListProc to send messages with tag numbers 124, 749, and 325 out to the list while the second line tells ListProc to discard message number 097.

HOLd *list password*

Suspend distribution of messages but allow users and owners to send commands to the list. This command is complementary to the **LOCK** command which stops all commands to the list but allows messages to continue to be delivered.

FREe *list password*

Resume delivery of a held list, or reset the message-limit to zero. If the listprocessor manager held the list, only he/she can free it.

Examples:

For a list named “mylist” whose owner password is “foo”

HOLD mylist foo
will tell ListProc to suspend distribution of mail to MyList. However, commands concerning the list will still be accepted.

IV. File Archives as Lists

AFD, Automatic File Distribution, is a mechanism for the automatic distribution of file archives or individual files via listproc. Users subscribe to an archive just as they would a message-based list. When the archive manager updates the files in the archive, the files are automatically sent out to the archive subscribers, or, if subscribers prefer, they may simply receive a notification of a file update, leaving it up to the subscriber to **GET** the file. The same AFD and FUI commands can be sent by users and by archive managers but the commands have different meanings depending on who gives the command.

User Commands:

afd <*action*> {<*archive*> [/*password*] [*files*]} &
[{<*archive*> [/*password*] [*files*]}]

fui <*action*> {<*archive*> [/*password*] [*files*]} &
[{<*archive*> [/*password*] [*files*]}]

Where action can be one of **add**, **delete**, or **query**. This allows a user to subscribe to, drop from, or query an archive . Please see the *CREN ListProc User Manual* for more information on the user commands.

Owner Commands:

[quiet] afd <*action*> {<*archive*> [/*password*] [*files*]} &
user-address user-address user-address user-address

This will allow you to subscribe a user to an archive such that whenever the archive is updated the user will receive the new files automatically. You may optionally specify filenames in the archive so that the user will receive only those files when they are updated. The name of the archive and optional file names must be in brackets {}. If the command is preceeded by the word “quiet” then the user will not be notified of being added to or subtracted from the archive list.

[quiet] fui <*action*> {<*archive*> [/*password*] [*files*]} &
user-address user-address user-address user-address

This will allow you to subscribe a user to an archive such that whenever the archive is updated the user will automatically receive notification of the new files but will not receive the new files themselves as in the **afd** command. You may optionally specify filenames in the archive so that the user will receive notification of only those files when they are updated. The name of the archive and optional file names must be in brackets {}. If the command is preceeded by the word “quiet” then the user will not be notified of being added to or subtracted from the archive list.

Options for ***action*** are: **add, delete, query, deliver, deliver_files, or deliver-debug.** Both commands take identical arguements and the examples below can be applied to both **AFD** and **FUI**.

ADD will add a user to an archive.

DELETE will delete a user from an archive.

QUERY will tell a user to which archives he/she is subscribed.

REVIEW will send a list of subscribers to an archive.

DELIVER will force delivery of a file whether it has been updated or not. If the **deliver** command is given and "archive" is missing, it will scan all archives for **afd/fui** files. Syntax of the **DELIVER** command is:

afd deliver <*password*> [*archive*]
fui deliver <*password*> [*archive*]

DELIVER-DEBUG will produce a report of activity without touching system files or sending mail. It will list all newly updated files and who will be notified. This command does not list old files; it only reports on what will happen the next time a delivery is schedualed. Syntax of the **DELIVER_DEBUG** command is:

afd deliver-debug <*password*> [*archive*]
fui deliver-debug <*password*> [*archive*]

DELIVER_FILES *frequency* [*when*] allows an archive owner to determine when files will be delivered after new updates are posted. Valid options are:

deliver_files hourly <*min*> <*password*> [*archive*]
deliver_files daily <*hh:mm*> <*password*> [*archive*]
deliver_files weekly <*day*> <*password*> [*archive*]
deliver_files monthly <*password*> [*archive*]

<u>Examples</u>:

For file archives called “master” having a password of “shazam” and an archive called “doc” having no password, both being owned by the same person:

quiet fui add {master /shazam } peter@system.com & mary@machine.org

This will add these two users to the “master” archive. Since this is an **fui** command it means the two users will be notified when the Master archive’s or any subarchive's files are updated (if, of course, that subarchive if marked for afd/fui functions).

If the owner sends the command:

afd add { master /shazam file1 file2 } { doc } peter@system.com

This will add the user to to an archive named “master”, but not to the entire archive; the user is only added to files “file1” and “file2” notify list and will receive these files when either of these two files are updated, and be added to archive “doc” list (anything updated in “doc” will be sent to sender also).

afd deliver shazam master
fui deliver shazam master

This will force delivery of all files in the “master” archive and notifications of the files whether they are new or not.

afd deliver-debug shazam master
fui deliver-debug shazam master

This will cause a report to be delivered to the archive owner in the following form:

FST: File Subscribers Table: list of files and subscribers.
SFT: Subscribers Files Table: list of subscribers and files subscribed to.
FMT: File Modification times Table: list of files and modification times.

```
afd deliver_files hourly 15 shazam master
fui deliver_files hourly 15 shazam master
```

This will distribute new files every hour on the 15 minute mark after they have been updated.

```
afd deliver_files daily 12:00 shazam master
fui deliver_files daily 12:00 shazam master
```

This will distribute new files once a day at 12:00 pm (ListProc uses a 24 hour clock) after they have been updated.

```
afd deliver_files weekly Monday shazam master
fui deliver_files weekly Monday shazam master
```

This will distribute new files once a week on the first Monday after the new files have been updated. Default ListProc delivery time is midnight if no time is specified.

VII. Interactive ListProcessor (ilp)

Commands to ListProcessor may be issued interactively, i.e., directly to ListProc without using e-mail, if both the host running ListProc and the system you use are networked directly on the Internet. ILP clients are available to run under UNIX, X-windows, Macintosh, and MS-Windows. An ilp session offers a command line interface in UNIX and graphical interface in the other clients. Copies of ilp should be available on the server hosting the list that you interact with. If not you may contact CREN for the latest version.

If you are using UNIX you will need to compile the ilp client before you use it. Then you connect to your desired server with the command "ilp host@domain" where "host@ domain" is replaced by the Internet domain name for the server to which you are connecting. It may be necessary to give an optional port address; check with your ListProc system manager if this is necessary. Once you connect you will be prompted for a username and password. Log in with the e-mail address that ListProc knows for you in your list as a login name and use your list password as a password. If you are not subscribed to any lists on the system you may login as "test" giving a password of "new-user". Once a connection has been established, you will get the ilp command prompt "REQUEST>" to which all commands are entered. In the examples below the command prompt is in regular type as above and the commands that you type are in **bold** type. If ListProc cannot match your e-mail address and password with one in a known list then ListProc will give you "casual user" privleges. Casual users may only issue commands for **help**, **information**, **recipients**, and **statistics** for public lists and may issue **index**, **get**, **view**, and **search** requests in the archives. If you are a subscriber to a list then you may issue all these plus the **set**, **run**, **subscribe**, **unsubscribe**, and **which** commands. List owners may additionally issue all the owner commands for their lists. To review your command privleges type a question mark "**?**" or "**privleges**" at the prompt. ListProc accepts all ilp commands exactly as it does via e-mail. Long requests may be continued on multiple lines with an ampersand "&" at the end of each line. With the UNIX version of ilp the output of every request may be redirected to a file by using a standard UNIX redirection ">" or ">>" followed by a file name. For example, to get an index of an archive typing "REQUEST> **index > listproc.index**" will obtain an index of the archives and save it in a file called listproc.index. Like ftp, ilp allows you to specify a filetype for all files transfered using the keywords "binary" and

“ascii”. Input may also be redirected from a file using the UNIX redirection “<“ followed by a filename. Therefore you can create a file called BATCH.REQUESTS containing the lines
index>index.text
get ListProc info >>index.text
quit
and run it with the command:
“REQUEST> **<BATCH>REQUESTS**”.
You can also use UNIX porting of output with the “ | ” such that the command:
“REQUESTS> **review listname | more**”
will pipe the output of the “review “ command to the UNIX “more” command. As a further example, you can look for a specific subscriber in a list by issuing the command: “REQUESTS> **review listname | grep -i name**” where “name” is replaced by the person’s name.

To end an ilp session just enter quit or exit at the prompt.

VIII. Regular Expressions

Many ListProc command lines take regular expressions as arguments. A regular expression is a group of symbols which describe a unique string of characters. An example of a simple regular expression is the word “donkey”. In a group of words, the regular expression “donkey” matches only other instances of the word “donkey” and nothing else. So if we had a text file and did a search for “donkey” every time that word appeared in the text it would show up in our search. This definition can be expanded if we define the period “.” as a wild card replacement for a single character. Then the regular expression “.onkey” would include “honkey” or “tonkey” or “bonkey” as well as “donkey”. We can continue to expand on the definition by adding the asterisk to the period “ .* ” as a substitute for any number of characters. Then the regular expression “don.* ” will not only include “donkey” but will include all strings of characters beginning with “don” including “don” itself. A search of a text file for “don.* ” will turn up dongle, donkey, donner, dondoodit, dondiddle, etc. Regular expression matching, therefore, puts together a whole set of rules that allow you to test whether a string fits into a specific syntactic shape. You can also search a string for a substring that fits a pattern, and just as importantly, you can replace one string with another. The command line of a ListProc command is more like a regular expression as used in a database search.

Regular expressions have a syntax in which a few characters are special constructs and the rest are "ordinary". An ordinary character is a simple regular expression which matches that character and nothing else. The special characters are `$', `^', `.', `*', `+', `?', `[', `]' and `\'. Any other character appearing in a regular expression is ordinary, unless a `\' precedes it.

For example, `f' is not a special character, so it is ordinary, and therefore `f' is a regular expression that matches the string `f' and no other string. (It does *not* match the string `ff'.) Likewise, `o' is a regular expression that matches only `o'.

Any two regular expressions A and B can be concatenated. The result is a regular expression which matches a string if A matches some amount of the beginning of that string and B matches the rest of the string.

As a simple example, we can concatenate the regular expressions `f' and `o' to get the regular expression `fo', which matches only the string `fo'. Still trivial.

The following are the characters and character sequences which have special meaning within regular expressions. Any character not mentioned here is not special; it stands for exactly itself for the purposes of searching and matching.

`.'

is a special character that matches anything except a newline. Using concatenation, we can make regular expressions like `a.b' which matches any three-character string which begins with `a' and ends with `b'.

`*'

is not a construct by itself; it is a suffix, which means the preceding regular expression is to be repeated as many times as possible. In `fo*', the `*' applies to the `o', so `fo*' matches `f' followed by any number of `o''s. The case of zero `o''s is allowed: `fo*' does match `f'.

`*' always applies to the *smallest* possible preceding expression. Thus, `fo*' has a repeating `o', not a repeating `fo'. The matcher processes a `*' construct by matching, immediately, as many repetitions as can be found. Then it continues with the rest of the pattern. If that fails, backtracking occurs, discarding some of the matches of the `*''d construct in case that makes it possible to match the rest of the pattern. For example, matching `c[ad]*ar' against the string `caddaar', the `[ad]*' first matches `addaa', but this does not allow the next `a' in the pattern to match. So the last of the matches of `[ad]' is undone and the following `a' is tried again. Now it succeeds.

`+'

`+' is like `*' except that at least one match for the preceding pattern is required for `+'. Thus, `c[ad]+r' does not match `cr' but does match anything else that `c[ad]*r' would match.

`?'

`?' is like `*' except that it allows either zero or one match for the preceding pattern. Thus, `c[ad]?r' matches `cr' or `car' or `cdr', and nothing else.

`[...]'

`[' begins a "character set", which is terminated by a `]'. In the simplest case, the characters between the two form the set. Thus, `[ad]' matches either `a' or `d', and `[ad]*' matches any string of `a''s and `d''s (including the empty string), from which it follows that `c[ad]*r' matches `car', etc.

Character ranges can also be included in a character set, by writing two characters with a `-' between them. Thus, `[a-z]' matches any lower-case letter. Ranges may be intermixed freely with individual characters, as in `[a-z$%.]', which matches any lower case letter or `$', `%' or period.

Note that the usual special characters are not special any more inside a character set. A completely different set of special characters exists inside character sets: `]', `-' and `^'. To include a `]' in a character set, you must make it the first character. For example, `[]a]' matches `]' or `a'. To include a `-', you must use it in a context where it cannot possibly indicate a range: that is, as the first character, or immediately after a range.

`[^ ...]'

`[^' begins a "complement character set", which matches any character except the ones specified. Thus, `[^a-z0-9A-Z]' matches all characters *except* letters and digits. Note that the ^ has to be within brackets. Outside of brackets it has a different meaning as mentioned below. `^' is not special in a character set unless it is the first character. The character following the `^' is treated as if it were first (it may be a `-' or a `]').

`^'

is a special character that matches the empty string -- but only if at the beginning of a line in the text being matched. Otherwise it fails to match anything. Thus, `^foo' matches a `foo' which occurs at the beginning of a line.

`$'

is similar to `^' but matches only at the end of a line. Thus, `xx*$' matches a string of one or more `x''s at the end of a line.

`\'

has two functions: it quotes the above special characters (including `\'), and it introduces additional special constructs. If you wanted to search for the dollar sign within a text you would have to use \$ in place of just $ since the dollar sign has a special meaning. Because `\' quotes special characters, `\$' is a regular expression which matches only `$', and `\[' is a regular expression which matches only `[', and so on.

For the most part, `\' followed by any character matches only that character. However, there are several exceptions: characters which, when preceded by `\', are special constructs. Such characters are always ordinary when encountered on their own.

No new special characters will ever be defined. All extensions to the regular expression syntax are made by defining new two-character constructs that begin with `\'.

`\|'

specifies an alternative. Two regular expressions A and B with `\|' in between form an expression that matches anything that either A or B will match. Thus, `foo\|bar' matches either `foo' or `bar' but no other string. `\|' applies to the largest possible surrounding expressions. Only a surrounding `\(... \)' grouping can limit the grouping power of `\|'. Full backtracking capability exists when multiple `\|''s are used.

`\(... \)'

is a grouping construct that serves three purposes:

1. To enclose a set of `\|' alternatives for other operations. Thus, `\(foo\|bar\)x' matches either `foox' or `barx'.

2. To enclose a complicated expression for the postfix `*' to operate on. Thus, `ba\(na\)*' matches `bananana', etc., with any (zero or more) number of `na''s.

3. To mark a matched substring for future reference. This last application is not a consequence of the idea of a parenthetical grouping; it is a separate feature which happens to be assigned as a second meaning to the same `\(... \)'

construct because there is no conflict in practice between the two meanings. Here is an explanation of this feature:

`\DIGIT'

After the end of a `\( ... \)' construct, the matcher remembers the beginning and end of the text matched by that construct. Then, later on in the regular expression, you can use `\' followed by DIGIT to mean "match the same text matched the DIGIT'th time by the `\( ... \)' construct." The `\(... \)' constructs are numbered in order of commencement in the regexp.

The strings matching the first nine `\( ... \)' constructs appearing in a regular expression are assigned numbers 1 through 9 in order of their beginnings. `\1' through `\9' may be used to refer to the text matched by the corresponding `\(... \)' construct.

For example, `\(.*\)\1' matches any string that is composed of two identical halves. The `\(.*\)' matches the first half, which may be anything, but the `\1' that follows must match the same exact text.

`\b'

matches the empty string, but only if it is at the beginning or end of a word. Thus, `\bfoo\b' matches any occurrence of `foo' as a separate word. `\bball\(s\|\)\b' matches `ball' or `balls' as a separate word.

`\B'

matches the empty string, provided it is *not* at the beginning or end of a word.

`\<'

matches the empty string, but only if it is at the beginning of a word.

`\>'

matches the empty string, but only if it is at the end of a word.

`\w'

matches any word-constituent character.

`\W'

matches any character that is not a word-constituent. What the `\(... \)' groupings matched.

Here are examples of commands that use regular expressions:

lists global 'health|mental'~death

The above will compile a list of lists that contain either the word 'health' or 'mental' in either their list name or description comment but will exclude lists with the word 'death'. The way you should read 'health|mental'~death out loud is; "health or mental but not death".

lists global move$&dan$

will search for all lists containing BOTH the characters 'move' AND 'dan' so that move$ will return both movement and movies and dan$ will return both dancing and danger. But in order for you to receive a reply, the list will have to contain BOTH words. So a list about Dangerous Movies will show up in your search as well as a list about Movement and Dancing.

ignore <listname> <password> bart@^ar..+beta.org

This example will ignore a user named bart whether he posts from ar1.beta.org or from ar2.beta.org or art.beta.org.

alias <listname> <password> (.+)@host.domain \1@domain

will take anything inside the () and store it in \1 which is then accessed. For example, if homer@cs.domain sends a message, it will be translated to homer@domain.

Your Feedback Desired

Your suggestions for improving this document are eagerly solicited by CREN and should be sent to *suggestions@listproc.net*.

® CREN is a registered Service Mark of the Corporation for Research and Educational Networking (CREN). CREN has applied for registration of ListProcessor and ListProc.