L-Soft international, Inc.

List Owner's Manual
for
LISTSERV®, version 1.8e

23 May 2002
Initial Release

The reference number of this document is 0205-UD-02.

All of L-Soft's manuals for LISTSERV are available in ascii-text format via LISTSERV and in popular word-processing formats via `ftp.lsoft.com`. They are also available on the World Wide Web at the following URL:

http://www.lsoft.com/manuals/index.html

L-Soft invites comment on its manuals. Please feel free to send your comments via e-mail to **MANUALS@LSOFT.COM**, and mention which manual you are commenting on. (However, please do not send support questions to this address.)

"Hot fix" revisions to this and other L-Soft manuals are posted as they are made to the master document, on the announcement-only mailing list:

LSOFT-DOC-UPDATES@PEACH.EASE.LSOFT.COM

A word about formatting: This manual was written in Microsoft Word 2000, and originally formatted to be printed on 8-1/2"x11" paper on a Canon BJC-2100 inkjet printer. When printing the manual on a different type of printer, or converting to a different word-processing program, it is highly likely that the formatting and pagination will change and it will be necessary to update the tables of contents and figures as well as the index prior to printing. The author has taken great pains to ensure that the pagination and formatting works properly with the particular printer mentioned above, and cannot be held responsible for what is, in the end, a limitation of the software used to produce the manual.

Reference Number 0205-UD-02

Table of Contents

Table of Figures

List Owner's Manual
for
LISTSERV®, version 1.8e

23 May 2002
Initial Release

The reference number of this document is 0205-UD-02.

Preface: LISTSERV Command Syntax Conventions

LISTSERV Command Syntax Conventions

Generally, parameters used in this document can consist of 1 to 8 characters from the following set:

```
A-Z 0-9 $#@+-_:
```

Deviations from this include:

fformat	Netdata, Card, Disk, Punch, LPunch, UUencode, XXencode, VMSdump, MIME/text, MIME/Appl, Mail
full_name	first_name [middle_initial] surname (*not* your e-mail address). Must consist of at least two space-separated words, eg, "John Doe".
listname	name of an existing list
node	Either: the fully-qualified domain name (FQDN) of an Internet host; or the BITNET nodeid or Internet hostname of a BITNET machine which has taken care of supplying an ':internet' tag in its BITEARN NODES entry;
host	Generally the same as *node*, but normally refers specificllly to the fully-qualified domain name (FQDN) of an Internet host rather than to a BITNET nodeid.
pw	a password containing characters from the set: `A-Z 0-9 $#@_-?!|%`
userid	Any valid RFC822 network address not longer than 80 characters; if omitted, the 'hostname' part defaults to that of the command originator
internet_address	Similar to `userid`, but specifically refers to a complete RFC822 network address in `userid@fqdn` format. When we use this nomenclature a fully-qualified hostname is required.

Other deviations from the standard set will be noted along with the affected commands.

Also please note the following conventions for representing variable or optional parameters:

italic type	always indicates required parameter names that must be replaced by appropriate data when sending commands to LISTSERV

< >   Angle brackets may sometimes enclose required parameter names
             that must be replaced by appropriate data when sending commands
             to LISTSERV. Sometimes used for clarity when italic type is
             inappropriate

[ ]   Square brackets enclose optional parameters which, if used, must be
             replaced by appropriate data when sending commands to LISTSERV

1. About Mailing Lists and LISTSERV

LISTSERV® is a system that allows you to create, manage and control electronic "mailing lists" on a corporate network or on the Internet. Since its inception in 1986 for IBM mainframes on the BITNET academic network, LISTSERV has been continually improved and expanded to become the predominant system in use today. LISTSERV is now available for VM, OpenVMS^{TM}, unix®, the Windows NT "family" (including Windows 2000), and Windows 95/98/ME.

Consider for a moment what the users of your electronic mail system actually use electronic mail for. Do they discuss problems and issues that face your organization, down to the departmental level? In an academic setting, do your faculty and students communicate via electronic mail? As with "real world" distribution lists, electronic mailing lists can make it possible for people to confer in a painless manner via the written word. The electronic mail software simply replaces the copying machine, with its associated costs, delays and frustrations. In fact, electronic mail lists are easier to use than most modern copiers, and a lot less likely to jam at just the worst possible moment.

Because electronic mail is delivered in a matter of seconds, or occasionally minutes, electronic mailing lists can do a lot more than supplement the traditional paper distribution lists. In some cases, an electronic mailing list can replace a conference call. Even when a conference call is more suitable, the electronic mailing list can prove a powerful tool for the distribution of papers, figures and other material needed in preparation for the conference call. And, when the call is over, it can be used to distribute a summary of the discussion and the decisions that were made. What before might have been an exchange of views between two or three people can now become an ongoing conference on the issue or problem at hand. Announcement lists and even refereed electronic journals can be made available to your audience, which can be as small as a few people or as large as the entire Internet community.

If you need a further overview, please see Appendix D, *Related Documentation and Support*, for information on how to get one.

2. Starting a Mailing List – The Basics

Note: This chapter (and much of the balance of this manual) assumes that you are administering your list by mail. LISTSERV 1.8d and following includes a web-based administration interface for lists which is described in Chapter 11 and which can handle most of the operations described in this chapter.

Lists that are coded "**Validate= Yes,Confirm,NoPW**" or "**Validate= All,Confirm,NoPW**" must imperatively be managed by mail, since the web administration interface is secured by passwords and these settings reject password validation, instead requiring validation by the "OK" method.

2.1. Avoid duplication of effort[1]

Before you start your list, it pays to do a careful search in several places to find out if you are duplicating an already-existing list, or if the name you are considering is already in use for a list on a differing subject.

The first place to check is the "CataList" service maintained by LISTSERV itself. This service lists all public lists running on LISTSERV servers worldwide. Point your Web browser of choice at the URL **http://www.lsoft.com/CataList.html** to access CataList.

If you don't have a web browser, you can alternately send the command

    LISTS GLOBAL *search_string*

in the body of mail to LISTSERV@LISTSERV.NET (or to LISTSERV at any host site). You will receive a mail message in return containing a list of all lists known to LISTSERV where either the name of the list or the short list description contains your search string. For instance, **LISTS GLOBAL IBM** would result in the following being returned to you:

```
 Excerpt from the LISTSERV lists known to LISTSERV@HOME.EASE.LSOFT.COM
 --------------------------------------------------------------------
                         12 Dec 2001 13:35

                        (search string: IBM)

            Copyright 2001 L-Soft international, Inc.

L-Soft international, Inc. owns the copyright to this compilation of
Internet mailing lists (the "Compilation") and hereby grants you the
right to copy the enclosed information for the sole purpose of
identifying, locating and subscribing to mailing lists of interest. Any
other usages of the Compilation, including, without limitation,
solicitation, tele-marketing, "spamming", "mail-bombing" and "spoofing"
are strictly prohibited.

***********************************************************************
* To subscribe, send mail to LISTSERV@LISTSERV.NET with the following *
* command in the text (not the subject) of your message:             *
*                                                                     *
*                       SUBSCRIBE listname                            *
*                                                                     *
* Replace 'listname' with the name in the first column of the table.  *
***********************************************************************

Network-wide ID  Full address and list description
```

---

[1] Parts of this section were adapted from "Some Lists of Lists", compiled by Marty Hoag.

```
---------------   --------------------------------
AIX-L             AIX-L@NEW-LISTS.PRINCETON.EDU
                  IBM AIX Discussion List
```
Figure 2.1. Sample output of `LISTS GLOBAL IBM`

(Quite a few more lists were deleted for brevity)

You might want to make your search more specific, as this particular search locates every list that has IBM somewhere in its title. For instance, if you wanted to start a list on some aspect of the IBM 370, you might do better to search for `IBM 370`.

Alternative searches you can do include:

- Check the NEW-LIST archive at

  http://listserv.classroom.com/archives/new-list.html

  where the NEW-LIST project (originally at North Dakota State University, later hosted by the Internet Scout Project, and now hosted by Classroom Connect) stores list announcements starting with June 2000[2]. The NEW-LIST archive contains information about LISTSERV lists as well as about lists running on other types of servers.

- Check the Usenet newsgroups `news.announce.newusers` and `news.lists` , if they are available to you via your local news feed.

- Use one of the World Wide Web search engines such as Alta Vista or Google to search for matches to the name you want to use.

2.2. What skills do I need to start and maintain a LISTSERV mailing list?

You should already be familiar with your mailing system and text editor. Otherwise, there are no special skills required. It is the goal of this manual to give you what you need to know about LISTSERV user commands, privileged LISTSERV owner commands, and how to read and interpret RFC822 Internet-style mail headers. LISTSERV itself is designed to operate in an identical manner no matter which operating system it is running under. Thus the fact that LISTSERV is running under VM, VMS, some flavor of Unix, or Windows NT should not be a concern to the list owner, who may not even know which version of LISTSERV his lists are running on.

Additionally, we have made an attempt to give you a basic "list owner's course" in anticipation of some of the issues you may encounter in the course of moderating a list.

2.3. Creating a mailing list – Where can it be done, and Who can do it?

If you are looking for a site to host a list, consider the following:

- First, find out if your computing center maintains a LISTSERV host.
- If not, you might consider a commercial LISTSERV site. There are a number of such sites, including L-Soft's own EASE[SM] service. You can get more information on

---

[2] Older archives for NEW-LIST (January 1989 through June 1998) are available at **http://LISTSERV.NODAK.EDU/archives/old-new-list.html** but there are no guarantees as to how long the old archives will remain available. Similarly, the archives of the list for June 1998 through June 2000 are available at **http://scout.cs.wisc.edu/addserv/new-list-archive.html** with the same lack of guarantee as to how long they will remain available.

EASE^{SM} by pointing a WWW browser at

http://www.lsoft.com/ease-head.html

Please note also that many sites (predominantly, but not necessarily limited to, those in .EDU domains) will not host commercial or potentially-controversial lists because of internal policies regarding appropriate use of their computing facilities. In such a case, your only option may be to seek a commercial LISTSERV site.

Physically creating the list is the task of the LISTSERV maintainer (sometimes referred to as the "LISTSERV postmaster") at a given LISTSERV host site.[3] Specific procedures for requesting a list startup vary from institution to institution. It is usually best to contact the computing center at the site for more information.

Because most list owners do not have the appropriate permissions to create lists, instructions on how to physically create lists are not included in this manual. If you are a LISTSERV maintainer, you can find these instructions in the *Installation Guide* that came with the software, or in the *Site Manager's Operations Manual for LISTSERV.*

2.3.1. Naming Conventions

When choosing a name for a list, there are a few conventions and restrictions that you should keep in mind.

The "-L" convention

The "-L" convention isn't required, but it can help people to realize that the mail is coming from a mailing list rather than from a real person. The people we are referring to here are people who run Internet mail systems, who may see a great deal of mail coming from a single host and begin to wonder why. If it comes from a userid that ends in a "-L", they will be more likely to recognize it as list mail.

Reserved names

You may not create lists whose names match the following wildcards:

owner-*
*-request
*-search-request
*-server
*-signoff-request
*-subscribe-request
*-unsubscribe-request

For instance, lists cannot be made with names like "owner-loyalty", "linux-server", and "donation-request".

These "pseudo-mailboxes" have a special meaning to LISTSERV, which has internal rules that govern how mail sent to these addresses is handled. See chapter 17.3 of the *Site Manager's Operations Manual for LISTSERV* for more information on what happens to mail sent to these special addresses.

---

[3] Note that the "LISTSERV postmaster" is not identical to the regular POSTMASTER address at a host site. The term "LISTSERV postmaster" is a canonical term from early in LISTSERV's history, and refers only to the person who is the actual LISTSERV maintainer at the host site. The term has fallen into disuse and its use is discouraged because of the potential confusion it may cause.

Reserved characters

Generally you want to avoid "special" characters such as the ones above the number keys on your keyboard. For example, don't use:

!   which can be confused for "bang-path" addressing, e.g., UUCP
@   which is a reserved character
#   which can cause problems with some mail software which uses it for addressing
$ which may have a special meaning to the unix shell
%   another addressing character that could cause problems
&   is sometimes reserved by non-unix systems (specifically on NT it has a special meaning to the shell). However, please note that *use of this character in the name of a list or in a sendmail alias for a list will cause LISTSERV on unix to choke.* Note that it *is* possible under unix to create a list with a "&" character in the name quite easily, and it is also possible to create a sendmail alias with a "&" character in the alias. That does not mean it will work.
*   is, of course, the wildcard character.
()   Parenthesis are generally reserved and can't be used in file names.
+   The plus character should be avoided because recent versions of sendmail deliver mail addressed to "user+whatever@somedomain" to "user@somedomain." Whether or not this is an intelligent thing to do on sendmail's part is left as an exercise for the user, but it can affect mail being sent to a list with a "**+**" character in the listname.
/   The slash character is reserved and can't be used in file names.
.   Although on some systems it is physically possible to create lists with a dot character in the name, LISTSERV will not accept this nomenclature. The only place a dot can or should be used is before the word "LIST" in the **PUT** command; e.g., **PUT MYLIST-L.LIST** is equivalent to **PUT MYLIST-L LIST**.
"   Double-quote characters are not allowed.

It is best if you avoid the use of special characters altogether and stick exclusively to the letters A-Z, numbers 0-9, and the underscore and hyphen characters when naming lists. Note that the "_" (underscore) character may cause problems with some non-compliant receiving systems. Also note that the space character (ASCII 0x20) is illegal in a list name, and L-Soft recommends that, although apostrophes (aka "single-quotes", ASCII 0x27) are valid in an RFC822 username, they not be used in list names since some mail programs may not accept them. (Prior to 1.8d, not all LISTSERV commands will work for lists whose names contain an apostrophe.)

If you have any question about the validity of a particular name, you can of course refer to RFC822 (**http://nis.nsf.net/internet/documents/rfc/rfc822.txt**) for the Internet standards for e-mail addressing.

Maximum length of the list name

The length of the list name (that is, the name of the list file and thus the "official" name of the list) is restricted as follows:

VM: 8 characters
Non-VM: unlimited (starting with 1.8c; but see below)

If you need a longer list name for a list running on a VM server, you should use the **List-ID=** keyword (see Appendix B).

PLEASE NOTE CAREFULLY that L-Soft recommends using names of 32 characters or less whenever possible as they provide for correct alignment of the results returned by certain commands. Very long (program generated) list names are likely to conflict with mail system limits and L-Soft recommends other solutions to the problem of dynamically generated lists. As a rule, list names in excess of 70 characters are likely to result in mail delivery problems.

Make it easy on your users

While you can (within limits) name a LISTSERV mailing list just about anything you want, you will probably want to follow a couple of simple guidelines:

1. Keep the name simple.
2. Keep the name as short as possible without causing confusion.

No doubt you could name a list MY-LIST-FOR-MATH-STUDIES, but who wants to type that? Conversely, MLFMS-L wouldn't mean much to Joe Random User. Somewhere in the middle is a reasonable compromise, e.g., MATH-STUDIES (or even just MATH-S).

2.4. List Header Keywords and what they do

How a LISTSERV mailing list performs its tasks is defined by its header keywords. There are several different categories of keywords, each of which is discussed below in general terms. A complete alphabetical listing of list header keywords, including default settings and all options available, is provided in Appendix B.

Access Control Keywords. These keywords designate the level of "openness" for a list. They determine who can post to the list, who can review the list of subscribers, and whether or not the list is open to general subscription.

Distribution Keywords. This group has to do with how LISTSERV distributes postings to subscribers, including whether or not acknowledgments are sent back to posters, how many postings may go through the list daily, whether or not the list is available in digest form and whether it is available to USENET through a gateway. These keywords also determine whether or not list topics are enabled, and how LISTSERV will configure outgoing postings for replies.

Error Handling Keywords. Included under this group are the keywords controlling automatic deletion, loop-checking, and to whom error messages are sent for disposition when received by LISTSERV.

List Maintenance and Moderation Keywords. A fairly large group of keywords having to do with how the list is operated, including definitions for the list owner, list editor, and the list archive notebook; whether or not (and whom) to notify when users subscribe and sign off; how often subscriptions must be renewed, and so forth. These are perhaps the most basic keywords that can be set for a given list, and one of them ("Owner=") *must* be set for a list to operate.

Security Keywords. These keywords control who can "see" the list (that is, whether or not the list appears in the List of Lists for a given user, based on the user's host site), and the level of security necessary for changes to the list itself. The "Exit=" keyword is also contained in this group.

Subscription Keywords. These control whether or not the list is open to general subscriptions, whether or not a mailing path confirmation is required, and what user

options are set by default upon subscription.

Other Keywords. These control other aspects of list management that are not generally changed from their defaults, and which do not fit readily into the categories listed above.

2.5. Sending commands to LISTSERV

In the following sections, you will see numerous references to "sending commands to LISTSERV". All LISTSERV commands are sent to the server either by email or (in LISTSERV 1.8d and following) via the web administration interface described in Chapter 11. For mailed commands, this means that you must create a new mail message using whatever command this requires for your mail client (click on "New message" or its equivalent for most mail clients) addressed to the LISTSERV address. Let's say for the sake of argument that the list you want to subscribe to (or are currently subscribed to) is running on a server called `LISTSERV.MYCORP.COM`. In order to send a command to that server, you would create a new message and address it to `LISTSERV@LISTSERV.MYCORP.COM` , and place the command(s) in the body (not the subject) of the message.

Depending on how you have security set up for your lists, some or all commands may require that you validate them with a personal LISTSERV password.

2.6. Defining Personal Passwords

The passwords recognized by LISTSERV for various operations (assuming that the `NOPW` parameter is not used with the "`Validate=`" keyword) are of two distinct types:

- Personal Passwords. LISTSERV can store a personal password in its signup files corresponding to your userid. This password not only can be used for list maintenance operations, but also protects your FUI (file update information) and AFD (automatic file distribution) subscriptions (if available on your server) and must be used to store your archive files, if any, on the server.

- List Passwords. Beginning with 1.8c, list passwords are obsolete (we are mentioning them here only because users upgrading from earlier versions will be aware of their existence). You should define and use a personal password for all protected operations.

To add a personal password, send mail to LISTSERV with the command

<p align="center">PW ADD <i>newpassword</i></p>

in the body of the message. LISTSERV will request a confirmation via the "OK" mechanism (see above) before it adds the password.

If you want to remove your password altogether, send the command

<p align="center">PW RESET</p>

This command will also require confirmation.

And finally, if you simply want to change your personal password, send the command

<p align="center">PW CHANGE <i>newpassword</i> [PW=<i>oldpassword</i>]</p>

If you do not include the old password in the command (e.g., you've forgotten it), LISTSERV will request an "OK" confirmation. Otherwise, it will act on the command without need for further confirmation (unless, of course, the *oldpassword* provided is incorrect).

Personal passwords may also be defined via the web administration interface at login time.

2.7. Retrieving the list – some considerations

If you are a LISTSERV maintainer, never attempt to hand-edit a production list file in place and restart the server. The GET and PUT operations (or the web administration interface/TCPGUI functions) are the only supported methods. Particularly under unix and Windows, LISTSERV will not always accept the hand-edited list file because some editors will insert control characters or CR-LF combinations that LISTSERV cannot parse. Under VM or VMS, it is always possible that hand-editing the list will introduce some sequence that will cause an operational error. L-Soft suggests that this method be used sparingly, if at all, and does not support it.

Once your list has been created by the LISTSERV maintainer, you can have a copy of the list sent to you for editing purposes. Simply issue the command

GET *listname* (HEADER

to LISTSERV. This will cause the server to mail you a copy of the list header only (without the subscriber list).

Note that you can retrieve the entire list, subscribers and all, by omitting the **(HEADER** switch. However, L-Soft strongly discourages getting the entire list at any time. This is because you do not need the entire list file if all you want to do is to change list header keyword settings. Also, since LISTSERV has well-documented commands available to manage user subscriptions, you should never attempt to hand-edit a list file in order to add or delete subscribers. Therefore there should normally be no reason to issue the **GET *listname*** command without the **(HEADER** switch.

The **GET** command automatically locks the list so that no changes can be made to the operating copy on the server until you do one of two things:

- Issue the **UNLOCK *listname*** command (if you decide no changes are needed)
- Send the list back to the server with the **PUT** command.

Leaving the list locked also prevents new subscribers from signing up. It is therefore not advisable to leave the list locked for long periods of time. This necessitates remembering to issue the **UNLOCK** command if you decide not to make any changes.

It is possible to request that LISTSERV not lock the list when it is sent to you. This is accomplished by adding the **(NOLOCK** switch to the **GET** command. You can use **(NOLOCK** and **(HEADER** together as in the following example:

GET *listname* (HEADER NOLOCK

(Note that the "**(**" switch character is used only once.)

CAUTION: It is not advisable to use the `(NOLOCK` switch in at least two cases:

- Don't use the `(NOLOCK` switch if you are not the sole owner of the list. This prevents conflicting GETs and PUTs by different list owners. For instance, Owner(A) GETs the list without locking it. Owner(B) then also GETs the list. The owners make differing changes to the list header. Owner(B) PUTs his changes back first. Owner(A) then PUTs his changes back, erasing every change Owner(B) made. If Owner(A) had not used the `(NOLOCK` switch, Owner(B) would not have been able to GET a copy of the list until Owner(A) either unlocked the list or PUT his copy back. (Owner(B) could also unlock the list himself, but it would be advisable to ask Owner(A) if he was finished editing the list header before doing so.)

- Don't use the `(NOLOCK` switch if you get the entire list rather than just the header. You will erase all subscriptions for users who subscribed between the time you GET the list and PUT the list back. It is easier to deal with questions as to why they got the "listname has been locked since time by list-owner" message than to explain why they got a subscription confirmation and now aren't getting list mail.

Another caution (1.8c and earlier): If you `GET` the header with the `(HEADER` switch, do not add new subscribers "on the fly" to the bottom of the header. If you do, your subsequent `PUT` will replace the entire list online with what you have sent, canceling the subscriptions of every user on the list (except for the ones you added to the header). Note carefully that LISTSERV will parse a signature file as if it were new subscribers; you should therefore turn off your signature file whenever you `PUT` your list header.

Under 1.8d and following the above problem has been alleviated by the new `PUTALL` command and a modification to `PUT`. A `PUT` command containing new subscribers added "on the fly" will result in only the header of the list being updated and a warning being generated that says if you really wanted to `PUT` the entire list, subscribers and all, that you should use the `PUTALL` command.

LISTSERV maintainers should note one further caution: It is considered extremely inadvisable to "hand-edit" subscriber lists, as columns at the far right of each subscriber's entry contain list control codes corresponding to the subscriber's personal option settings. The only case in which it might be appropriate to "hand-edit" would be to delete a user entirely, and then only if all attempts to delete the user via the DELETE command fail. For instance, X.400 or X.500 addresses can cause DELETE to fail because of their use of the "/" character. You can use wildcards to delete these subscriptions. You can also enclose the address in double quotes:

    DELETE XYZ-L "/ADMD=ABC/PRMD=DEF/...../@X400.SOMEHOST.COM"

Finally, note that depending on your list configuration, you may have to use a password or respond to a confirmation request in order to GET your list header. The syntax for using a password with the GET command is

    GET *listname* (*options* PW=*password*

For instance,

    GET MYLIST-L (HEADER NOLOCK PW=MYPASSWORD

See the sections below regarding list passwords, personal passwords, and the "OK" command confirmation feature.

2.8. Editing the list header

Once the LISTSERV maintainer has notified you that the basic list has been created, you can send a `GET` command to the server to make any modifications necessary, as explained above. For instance,

    GET MYLIST (HEADER PW=MYPASSWD

might cause LISTSERV to send you the following list header file:

```
PUT MYLIST.LIST PW=XXXXXXXX
* The Descriptive Title of My List
*
* Owner= NATHAN@LSOFT.COM (Nathan Brindle)
* Notebook= Yes,E:\LISTS\MYLIST-L,Monthly,Public
* Errors-To= Owner                 Send= Public
* Subscription= Open,Confirm       Ack= Yes              Confidential= No
* Validate= No                      Notify= No            Reply-to= List,Respect
* Review= Public                    Default-Options= NoFiles,NoRepro
*
* This list installed on 96/11/02, running under L-Soft's LISTSERV
* for Windows NT.
*
* Comment lines...
*
```
Figure 2.2. A sample list header file for a list called MYLIST.

You can now physically edit this file in a couple of different ways, depending on what tools you have on your workstation:

- Cut and paste the header from your mail program into an ASCII text editor such as vi, emacs, or Notepad. Edit the various keyword values and set the password in the PUT command appropriately. Finally, cut and paste from your editor into a new mail message addressed to LISTSERV and send the message.

- Cut and paste from your mail program into the body of a new mail message addressed to LISTSERV and do your editing in the mail program.

- If you use the '`mail`' or '`mailx`' programs under unix, you can save the header into a text file (for instance, named `myheader.txt`) and mail it back to LISTSERV with the command line syntax

    mail listserv@myhost.com < myheader.txt

In Figure 2.3, we've made some changes to the list header and it is ready to be included in a mail message and sent back to LISTSERV. Note that the `PUT` command has been modified to include your personal password (see 2.6 for instructions on how to obtain a personal password).

```
PUT MYLIST.LIST PW=MYPASSWD
* The Descriptive Title of My List
*
* Owner= NATHAN@LSOFT.COM (Nathan Brindle)
* Owner= Quiet:,nathan@linus.dc.lsoft.com,ncbnet@linus.dc.lsoft.com
* Notebook= Yes,A,Monthly,Public         Auto-Delete= Yes,Full-Auto
* Errors-To= ncbnet@linus.dc.lsoft.com    Subscription= Open,Confirm
* Ack= Yes                Confidential= No               Notify= No
* Validate= Yes,Confirm
* Reply-to= List,Respect    Review= Public               Send= Public
* Default-Options= NoFiles,NoRepro
```

```
*
* This list installed on 96/11/02, running under L-Soft's LISTSERV
* for Windows NT.
*
* Comment lines...
*
```
Figure 2.3. The edited list header file ready to be sent back to the server.

If LISTSERV responds to your **PUT** operation with error messages, bear in mind that the most common problems are:

1. You sent the header back from an account which is not defined as a list owner
2. You used the wrong password or didn't specify a password in the **PUT** command
3. Your mail program indents paragraphs by default, such that each header line does not start in column 1 and LISTSERV does not recognize your message as a list header
4. You sent the header file back as an attachment rather than as plain text in the body of the message.
5. Your mail client wrapped the lines of the header so that LISTSERV received header lines that did not begin with "*" and attempted to treat them as subscriber addresses. In this case you can probably solve the problem by increasing the right margin setting in your mail client so that messages at least 80 columns wide do not get wrapped.

2.9. Defining list owners

List owners should be persons who will undertake the responsibility of managing the list in all of its aspects. A list owner may be a moderator; a list owner may be called upon to determine why a user can't unsubscribe from the list, or to handle delivery errors, or to fix other problems that may arise.

The primary list owner (the first owner defined) has special responsibilities as well. This owner is considered the Editor and the primary Moderator for lists that have **Send= Editor** but do not have **Editor=** or **Moderator=** defined. This owner receives all error messages when **Errors-To=** is set to "Owner". In short, the primary list owner is generally the person who is ultimately responsible for the workings of the list.

Secondary list owners fall into two categories: Quiet and non-Quiet.

- Non-Quiet list owners receive mail sent to the *listname-request* address, and will receive error messages if **Errors-To= Owners**.

- Quiet list owners will never receive delivery errors or other administrative mail from LISTSERV.

Here is a sample list header excerpt for a list with all three types of list owners defined:

```
* Owner= NATHAN@EXAMPLE.COM (Nathan Brindle)
* Owner= nathan@linus.example.com
* Owner= Quiet:
* Owner= ncbnet@linus.example.com,cheng@linus.example.com
```
Figure 2.4. Example: How to define list owners in the list header file.

Note that all list owners defined after the *** Owner= Quiet:** line will be quiet list owners.

You can define multiple owners on a single line by separating them with a comma. Note

that if you put "Quiet:" on a line with list owner userids, you must place a comma after "Quiet:", e.g.

```
* Owner= Quiet:,ncbnet@linus.example.com,cheng@linus.example.com
```

There must **always** be at least one non-quiet list owner. Otherwise LISTSERV sends all error messages and other administrative mail to the LISTSERV maintainer by default.

2.10. Storing the list on the host machine

When you are ready to store your list back on the host, include the list file in a mail message to LISTSERV. Ensure that the `PW=XXXXXXXX` command is in the first line of the mail body. Change `XXXXXXXX` to the personal password you have previously defined with the `PW ADD` command (see section 2.6). Then send the message.

If LISTSERV has trouble processing the edited list file, it will return a discrepancy report to you with each error noted. If the errors are categorized as "warnings only," LISTSERV will go ahead and store the list. However, if any one error is categorized as a serious error, the list will not be stored and the old version will be retained.

Caution: If you are using a mailer such as Pine or Microsoft Mail that allows "attachments" to mail, do not "attach" the list file to your mail message. It must be in plain text with the `PUT` line at the top. LISTSERV will not translate encoded attachments.

2.11. Fixing mistakes

LISTSERV always backs up the current list file before it stores a new copy. Should you discover that you have made a mistake (for instance, you have deleted all users by storing a header and adding users "on the fly"), it is possible to retrieve the previous copy of the list by issuing a `GET listname (OLD` command to the host server. You must then add the `PUT listname LIST PW=XXXXXXXX` command to the top of the file and store it. (In LISTSERV 1.8d and later you should use the `PUTALL` command for this purpose since you will be storing the entire list, not just the header.)

2.12. Security Options

LISTSERV's security options are wide ranging, from almost no protection (easiest to administer your list, but also most open to hacker attacks) to total protection requiring validation of each and every command sent to LISTSERV for your list. It is also possible to limit access to various aspects of your list, such as who can subscribe, who can review the list of subscribers, and who can access the list archives. You can hide your list from the LIST command, either at the global level or from all requests, including those from users on LISTSERV's local machine, or from a definable range in between.

2.12.1. First line of defense: The VALIDATE= keyword

The `Validate=` keyword controls the level of command validation desired for your list. The default, `Validate= No`, requires password validation only for storing the list on the server. This is often sufficient for general needs. However, when a list is set this way, LISTSERV only compares the RFC822 "Sender:"/"From:" headers against the `Owner=` keyword(s) in the list header to determine whether or not the person ostensibly sending the commands has authority to do so. Otherwise at this level LISTSERV does not validate commands it receives for the list, under the assumption that the mail it receives is genuinely coming from a list owner. This level of validation does not protect the list from commands issued by hackers who have forged mail in the name of the list owner. If

you run a list on a controversial topic or just don't feel comfortable without at least some security, `Validate= No` is probably not for you.

The next level is `Validate= Yes`. At this level, LISTSERV requires a password for all of its "protected" commands. This password is the sender's personal LISTSERV password as defined by the `PW ADD` command. The commands protected by this level are those that affect subscriptions or the operation of the list, e.g., `DELETE` or `ADD`. Users will also have to validate most commands that affect their subscriptions, but generally can do so using the "OK" mechanism rather than defining a personal password. Note that some user commands will be forwarded to the list owner for validation rather than accepting password validation from the user.

The next level is `Validate= Yes,Confirm`. At this level, LISTSERV will require validation with the "OK" mechanism (see below) by default, but will still accept passwords where appropriate. While the less-secure passwords are still accepted, this is considered a good compromise between list security and list owner and user convenience.

The next level is `Validate= YES,Confirm,NoPW`. At this level, LISTSERV will no longer accept passwords as validation for protected commands. The logic is that because of the way the "OK" mechanism is implemented, passwords are not as safe as "magic cookies". This is the recommended setting for lists that must be kept secure.

Two other levels are `Validate= All,Confirm` and `Validate= All,Confirm,NoPW`. These levels require "OK" validation for all commands that cause a change in state except for the `PUT` command. If `NoPW` is not specified, passwords are accepted where appropriate. With these levels, commands that do not cause a change in state (e.g., `QUERY`) do not require validation.

Note that LISTSERV requests coming from the local system via CP MSG or CP SMSG on VM systems or via LCMD on NT, VMS or Unix systems never require validation, as they cannot be forged.

Lists which are set to either `Validate= Yes,Confirm,NoPW` or `Validate= All,Confirm,NoPW` may not be managed via the web administration interface, which is password-driven.

See Appendix B for more information on the `Validate=` keyword.

2.12.2. Controlling subscription requests

You can control subscription requests by use of the `Subscription=` keyword. By default, this keyword is set to `Subscription= By_Owner`, meaning that all subscription requests will be forwarded to the list owner for disposition. You can also refuse all subscription requests by setting `Subscription= Closed`.

To code a list for open subscriptions without list owner intervention, you set `Subscription= Open`. If you would like to add protection against forged subscription requests or bad return mailing paths, code `Subscription= Open,Confirm`. The latter will cause a subscription confirmation request to be sent to the prospective subscriber, which he or she must respond to using the "OK" confirmation mechanism.

In order to restrict subscriptions to persons in a specific service area, see the next section.

2.12.3. Controlling the service area of your list

The Service= keyword is not available in LISTSERV Lite.

It may be desirable to restrict access to your list to people in a small area. For instance, you probably would not want a list for students in a class section at a university to be advertised or accessible by people all over the world. However, without setting certain keywords appropriately, such a list will be visible to a **LIST GLOBAL** command.

If you wish to simply hide your list from a **LIST** command, but still allow people to subscribe to it if they know it is there, use the keyword **Confidential= Yes**. Note that users subscribed to the list as well as the list owner(s) *will* be able to see the list if they issue a **LIST** command.

If you wish to hide your list from and refuse subscription requests from users outside the local area, you define two keywords:

```
* Service= bitnode1,bitnode2,some.host.edu
* Confidential= SERVICE
```

Service= can also be set to **Service= Local**, meaning it will use either LISTSERV's global definition of which machines are **Local**, or the machines defined by the list keyword **Local=**. If you wish to set **Service** to **Local**, you should check with your LISTSERV maintainer to find out which nodes are considered local. If the global definition is not suitable, you can override it by defining the **Local=** keyword:

```
* Local= bitnode1,bitnode2,some.host.edu,another.host.com
* Service= Local
* Confidential= Service
```

If there are many subdomains within your primary domain, you may wish to use the wildcard when defining the **Local=** or **Service=** keywords. For instance:

```
* Service= HOST.COM,*.HOST.COM
```

defines the service area as "**HOST.COM** and all subdomains ending in **.HOST.COM**".

2.12.4. Controlling who may review the list of subscribers

For whatever reason, you may wish to restrict the ability to review the subscriber list either to subscribers or to list owners. This is done by setting the **Review=** keyword appropriately.

To restrict reviews of the list to subscribers only, set **Review= Private**. This is the default starting with LISTSERV 1.8c.

To restrict reviews of the list to list owners only, set **Review= Owners**.

To allow anyone, including non-subscribers, to review the list, set **Review= Public**. Prior to LISTSERV 1.8c this was the default; it is no longer recommended to use this value unless your LISTSERV server is operating on an intranet.

You can also restrict reviews to users within the list's service area by setting **Review= Service** , and defining the **Service=** keyword appropriately (see the preceding section).

2.12.5. Controlling who may access the notebook files

Restricting access to the list's notebook archive files is similar to controlling who may review the list. It is accomplished by setting the fourth parameter of the **Notebook=** keyword to an appropriate value. For instance,

`* Notebook= Yes,A,Monthly,Public`

defines a monthly notebook on LISTSERV's A disk that is accessible by anyone. Change **`Public`** to **`Private`** if you wish only subscribers to be able to access the notebooks. The same access-levels are available for this keyword as for **`Review=`**. (See Appendix B for a discussion of access-levels.)

Note: The location (second) parameter of the **Notebook=** keyword may be changed only by the LISTSERV maintainer.

If enabled, notebook archives are private by default.

2.12.6. Controlling who may post mail to the list

The **`send=`** list header keyword is the basic control for who may post mail to the list. If the list allows non-subscribers to post, set **`Send= Public`**.

For a list that does not allow non-subscribers to post, set **`Send= Private`**. (This is the default.)

If you want a further level of security for **`Send= Private`**, you may set **`Send= Private,Confirm`** , which requires each poster to confirm (via the "OK" mechanism) that the posting actually came from them. This can help in cases where a hacker might be trying to "spoof" mail from an otherwise legitimate subscriber. It is not recommended to set this in normal circumstances.

For a list where all posts should be forwarded to a moderator/editor, there are three settings:

- **`Send= Editor`** forwards all postings to the list editor (see the **`Editor=`** and **`Moderator=`** keywords). This setting allows the editor to make changes before forwarding the message back to the list. Note that your mail program must be capable of inserting "Resent-" header lines in your forwarded mail—if it is not capable of this, all such posts forwarded to the list will appear to be coming from the editor. Check with your system administrator if you are not sure whether or not your mail program inserts the "Resent-" headers.

- **`Send= Editor,Hold`** forwards a copy of the posting to the editor but differs from **`Send= Editor`** in that LISTSERV holds the posting for a period of time (usually 7 days) until the editor confirms the message with the "OK" mechanism (see below). Unconfirmed messages simply expire and are flushed by LISTSERV, so there is no need to formally disapprove a posting. This method of message confirmation is well-suited to lists where it is not often necessary to modify the text of a posting, and also is an excellent workaround if the editor's mail program does not generate "Resent-" headers in forwarded mail.

Below is a sample of the editor-header for a list set to **`Send= Editor,Hold`**:

```
Date:           Tue, 4 Aug 1998 10:47:21 -0500
From: "L-Soft list server at PEACH.EASE.LSOFT.COM (1.8d)"
                <LISTSERV@PEACH.EASE.LSOFT.COM>
Subject:        B5-L: approval required (9723A0DD)
To: Joe ListOwner <joe@PRUNE.EXAMPLE.COM>

This message was originally submitted by  jack@UNIX.FOO.COM to the B5-L list at
PEACH.EASE.LSOFT.COM. You can  approve it using the "OK"  mechanism, ignore it,
or repost an edited copy. The message  will expire automatically and you do not
need to do  anything if you just want  to discard it. Please refer  to the list
owner's  guide  if  you  are  not  familiar  with  the  "OK"  mechanism;  these
instructions  are  being  kept  purposefully  short  for  your  convenience  in
processing large numbers of messages.

----------------- Original message (ID=9723A0DD) (13 lines) -------------------
```

Figure 2.5. The editor-header for a list set to `Send= Editor,Hold`

- `Send= Editor,Hold,Confirm` is identical to `Send= Editor,Hold` except that postings coming directly from an editor must be confirmed (with the "OK" mechanism) by the editor who sent the message. *This is the recommended setting for any moderated list or announce-only list as it protects the list from hackers who might try to forge mail from a legitimate editor address.*

A fourth method (called "self-moderation") exists for lists where subscribers should be allowed to post freely, but non-subscriber posts should always be sent to an editor for approval. To enable self-moderation, set

```
* Send= Editor
* Editor= userid@host,(listname)
```

Ensure that "listname" is in parenthesis. Note that self-moderation will catch all posts from non-subscribers—including posts from subscribers who are posting from a different address. For instance, if the subscriber originally signed up as `joe@foo.com` but is posting from `joe@unix1.foo.com`, LISTSERV will treat his mail as non-subscriber mail. Self-moderation may require some slight changes in individual user subscriptions in order for it to work seamlessly.

2.12.7. The "OK" confirmation mechanism

Depending on the setting of the `Validate=` list header keyword, certain LISTSERV commands have always required a password for execution. However, with a recognition that mail can be forged ("spoofed") by just about anyone on the Internet today, L-Soft introduced a "magic cookie" method of command validation that is considered much more secure than passwords.

In essence, the "magic cookie" method requires that the sender of the command must confirm his command via a reply containing only the text "OK". (This is actually simplistic; see below.) If mail is spoofed from the list owner's user id, the command confirmation request will always be sent to the list owner's user id, thus preventing the spoofer from confirming the command. Moreover, the "cookie" itself (a six-digit hexidecimal number) is registered to the "From:" user id of the original command.

A typical command confirmation request looks like this:

```
Date:           Wed, 5 Aug 1998 09:50:06 -0400
From:           "L-Soft list server at LISTSERV.EXAMPLE.COM (1.8d)"
                <LISTSERV@LISTSERV.EXAMPLE.COM>
Subject:        Command confirmation request (5C019D91)
To:             joe_user@EXAMPLE.COM
```

```
Your command:

                        PW REP XXXXXXXX

requires confirmation. To  confirm the execution of  your command, simply
point your browser to the following URL:

        http://listserv.example.com/scripts/wa.exe?OK=5C019D91

Alternatively, if  you have no WWW  access, you can reply  to the present
message and type  "ok" (without the quotes) as the  text of your message.
Just the word "ok" - do not  retype the command. This procedure will work
with any mail  program that fully conforms to the  Internet standards for
electronic  mail. If  you receive  an error  message, try  sending a  new
message  to  LISTSERV@LISTSERV.EXAMPLE.COM  (without  using  the  "reply"
function - this is very important) and  type "ok 5C019D91" as the text of
your message.

Finally, your  command will be  cancelled automatically if  LISTSERV does
not receive your confirmation within 48h. After that time, you must start
over and resend the command to get a new confirmation code. If you change
your mind and decide that you do  NOT want to confirm the command, simply
discard the present message and let the request expire on its own.
```
Figure 2.6. A typical command confirmation request.

The general method of replying to a command confirmation request is as follows:

- Under 1.8d and following, the suggested method is to use the web browser confirmation method outlined in the confirmation request.

If you prefer, or if you are using 1.8c or 1.8b, you can use the old method of responding by mail:

- **REPLY** to the command confirmation request with the text "ok" in the body of the reply. (Non-case-sensitive) LISTSERV reads the "cookie" from the subject line and if it corresponds to a held job, the job is released and processed.

If this does not work, it is possible that the Subject: line was corrupted in transit and you may ne ed to try the following:

- **SEND** a new message to LISTSERV with the text "ok xxxxxx" (where xxxxxx is the command confirmation number from the original confirmation request) in the body of the reply.

It is also possible to confirm multiple command confirmation requests with a single message (for instance, if you have **Send= Editor,Hold** and have a number of requests to be responded to). This eliminates multiple "Message approved" mails from LISTSERV. However, make sure that you send the confirmations in a new mail message rather than replying to one of them.

Prior to LISTSERV 1.8e, when using "OK" cookies for moderation, note that confirmation requests for messages containing MIME attachments will show the "raw" attachment. This is because LISTSERV does not generate MIME headers for confirmation request messages. When the "OK" is sent, MIME attachments will be processed correctly.

Prior to LISTSERV 1.8d, the "OK" confirmations must come from the userid that originated the command, i.e., you cannot send a command from one account and then approve it from another.

From LISTSERV 1.8d the "OK" can be sent from any address, which helps when the

address field of your mail gets changed somewhere along the line. For instance if you are logged into the web administration interface as joe@example.com and issue a command that requires mail confirmation, LISTSERV will send the request to joe@example.com (as expected). If your mail system expands joe@example.com to Joe_Doakes@mail.example.com, responding to the request under 1.8c would result in a failure because the cookie and the address in your From: line wouldn't correspond to what LISTSERV has on file. Under 1.8d and later the "OK" will succeed and Joe_Doakes@mail.example.com will get a message that says

```
> ok
Confirming:
> QUIET DELETE * jane@example.com
[reply sent to joe@EXAMPLE.COM]
```

while as a protection against "spoofed" commands the actual command response will be sent to joe@example.com like this:

```
jane@EXAMPLE.COM has been removed from the TEST list. No notification has
been sent.

Global deletion process complete, one entry removed.
```

Three further enhancements were added to the "OK" confirmation mechanism in 1.8d:

- An "OK" without an argument (confirmation number) flushes the job stream, so any text following an "OK" on a line by itself will not be seen by the LISTSERV command processor.
- Bracketed "OK" functionality. This feature allows you to send multiple commands for which LISTSERV will request only a single "OK" (where normally you would expect to have to "OK" each individual command). The syntax is as follows:

```
    OK BEGIN
    command1
    command2
    [more commands, one per line]
    commandn
    OK END
```

- A command confirmation ("OK") may now be sent by clicking on a web URL provided in the command confirmation request (mailed "OK"s are still perfectly acceptable, of course).

> **Documented restriction**: In a "bracketed OK" the aggregate length of the data stream (ie, the total number of characters in the command lines falling between **OK BEGIN** and **OK END**) MUST be less than 32K characters. In practice you should use bracketed OKs for limited numbers of commands only, say no more than 10-12 at a time. In particular, if you have many ADD or DELETE commands to send, it is far more efficient (and strongly preferred) to use the bulk ADD and bulk DELETE syntaxes described in chapters 4.4 and 4.5, above.

2.12.8 Explicitly cancelling "OK" cookies (1.8e)

In LISTSERV 1.8e and following, it is possible to explicitly cancel an OK confirmation cookie if so desired. The command is simply

```
OK CANCEL xxxxxxxx
```

(for instance, "OK CANCEL 8F2E8F4B"), and if the cookie is valid, LISTSERV will respond "Confirmation code 8F2E8F4B cancelled." If the cookie is not valid (eg has expired, has already been cancelled, or is simply incorrect), LISTSERV will send its standard message telling you in part that "The confirmation code 8F2E8F4B does not correspond to any pending command."

2.12.9. Restricting subscriber privileges

Another security issue involves protecting the list from people who refuse to play by the rules. LISTSERV includes several different levels of privilege restriction for these users, some of which are available for use by list owners without the intervention of the LISTSERV maintainer.

1. *The `REVIEW` personal option setting.* By issuing a `SET listname REVIEW FOR userid@host` command to LISTSERV, you can moderate postings at the individual subscriber level. Postings from subscribers set to REVIEW are passed on to the Editor(s) or Moderator(s) of the list, or, if neither of these keywords are defined for your list, the postings are passed on to the primary list owner. At this point, the person who receives the postings can determine whether or not to approve them. Note that the subscriber always receives notification that his or her posting has been forwarded to a moderator for approval. This is to avoid the impression that the subscriber's posting has been lost before reaching LISTSERV.

2. *The `NOPOST` personal option setting.* By issuing a `SET listname NOPOST FOR userid@host` command to LISTSERV, you can prevent a subscriber from posting to the list entirely. LISTSERV will reject postings from these subscribers and will not pass them on to a moderator. As with the REVIEW setting, note that the subscriber always receives notification that his or her posting has been rejected.

3. *The `FILTER=` list header keyword.* You can filter individual users from subscribing and/or posting to your list by adding them to the `Filter=` list header keyword. For instance, if you have a list called MACTALK-L and you want to discourage redistribution lists from using the same name as your list, you can add

    ```
    * Filter= Also,MACTALK-L@*
    ```

See Appendix B for more information on the `Filter=` syntax.

2.12.10. Restricting the number of postings per user to the list per day

Beginning with 1.8c, you can control the maximum number of postings per day *per subscriber* on a list-by-list basis by setting the new (optional) second parameter of the "Daily-Threshold=" list header keyword. The default is to have no such daily limit per user.

If set, when the per-subscriber threshold is reached, the subscriber is told that his message cannot be processed because he has reached the limit for today, and that he should repost his message at a later time. The counter for this limit resets to zero at midnight for all lists.

This limit is waived for the list owner(s) and any list editors/moderators.

If you want to set this limit, note that an overall daily threshold must be set for the list in

the first parameter of the keyword. If no "Daily-Threshold=" keyword is already present in your list header, the default is "Daily-Threshold= 50". Thus, to leave the default value in force and to add a daily limit of 5 postings per day per user, you would code:

```
* Daily-Threshold= 50,5
```

For more information see Appendix B.

2.13. How to set up lists for specific purposes

2.13.1. Public discussion lists

Public discussion lists have always been the "classic" type of LISTSERV mailing list. Such lists are available to discuss just about everything imaginable. In the last few years it has become desirable to secure mailing lists against random spamming and mailbombing, but no discussion of different types of lists would really be complete without talking about this kind of list.

Typically, a public discussion list is wide-open (although some things, like the ability to review the subscribership, may be restricted). Anyone can subscribe (with a confirmation to verify the mailing path), anyone can post, anyone can read the messages in the archives, and security is set fairly low. Very large lists (hundreds or even thousands of users with hundreds of postings every week) may likely be set up this way as it is a "low-maintenance" way to run a list (and most spams tend to be caught by LISTSERV's anti-spamming filters anyway). For instance you might have

```
* My public discussion list (MYLIST-L)
* Subscription= Open,Confirm
* Ack= Yes
* Confidential= No
* Validate= No
* Reply-to= List,Respect
* Review= Owners        Send= Public      Errors-To= Owner
* Owner= joe@example.com
* Notebook= Yes,E:\LISTS\MYLIST-L,Weekly,Public
```

For more security, you might want to code

```
* Validate= Yes,Confirm
```

and if you want to cut down on the amount of "me-too"ism on the list, you could set

```
* Reply-to= Sender,Respect
```

to force the default Reply-To: header to point back to the original poster instead of to the list. Note that the ",Respect" option means that if a user sends mail to the list that contains a "Reply-To:" header pointing back to the list (unlikely that this may be), LISTSERV will "respect" that header and use it. If you absolutely do not want this to be possible, you should code

```
* Reply-to= Sender,Ignore
```

instead.

There is one major caveat with regard to the use of the `Reply-To=` list header keyword. You should note carefully that not all subscriber-side mail clients either recognize or

properly handle an RFC822 "Reply-To:" header. This may result in users posting replies to your list even though LISTSERV put the correct Reply-To: header on the mail. There is absolutely nothing that L-Soft can do to correct this problem since it exists on the subscriber's end in non-compliant mail software that L-Soft does not and cannot support.

2.13.2. Private discussion lists

Private discussion lists are similar to public discussion lists, but with varying restrictions on who may subscribe, who may post and who may view the archives. Such lists are relatively safe from random spamming since typically only a subscriber can post (but note that a spammer spoofing mail from a subscriber's address will probably be successful unless first caught by the anti-spamming filters). For instance:

```
* My private discussion list (PRIVATE-L)
* Subscription= By_Owner
* Ack= Yes
* Confidential= Service
* Validate= No
* Reply-to= List,Respect
* Review= Owners
* Send= Private
* Errors-To= Owner
* Owner= joe@example.com
* Notebook= Yes,E:\LISTS\PRIVATE-L,Weekly,Public
```

is a low-security private discussion list where subscriptions requests are passed on to the list owner(s) for review, only subscribers may post, and only subscribers may view the list archives. Here again, for more security you might want to set "**Validate= Yes,Confirm**", and of course you can have replies go to the original poster rather than to the list with "**Reply-To= Sender,Respect**" (with the same caveats as noted above in 2.13.1).

2.13.3. Edited lists

An edited list is one which requires a human editor to approve messages sent to the list. Some list software and most USENET newsgroups refer to this as "moderation", but to avoid confusion between two types of moderated LISTSERV lists, the present example will be referred to as an "edited" list.

Examples of edited lists range from refereed electronic journals to lists where the list owner simply wishes to exercise control over which postings are allowed to go to the list.

To set up a basic edited list, simply add

```
* Send= Editor
* Editor= someuser@somehost.com
```

to the basic list header. Note that the primary Editor= specification (that is, the first editor defined by an Editor= keyword for the list) must be a human person who will be able to act on postings sent to him or her for approval. You may not use an *access-level* specification (such as "Owner") when defining the primary editor for a list.

Please note that L-Soft recommends setting "Send= Editor,Confirm" so as to add a level of security against malicious users forging mail from an "Editor=" address to get around your moderation settings, or against badly-configured "vacation" programs that simply reflect the message back to the list in a manner that makes it appear that the mail is

coming from the editor's address. The "Confirm" option causes LISTSERV to request an "OK" confirmation from an editor when it receives mail claiming to be from that editor.

You can define multiple editors, but only the first editor will receive postings for approval. Anyone defined as an editor may post directly to the list without further intervention. Multiple editors can be defined on separate Editor= lines or can be grouped several on a line, e.g.,

```
* Editor= someuser@somehost.com,anotheruser@anotherhost.com
* Editor= yetanotheruser@his.host.com
```

To approve postings with the above configuration, the editor simply forwards (or "resends", or "bounces"--the terminology is unclear between various mail programs) the posting back to the list address after making any desired changes to the content. *This should be done with a mail program that supports "Resent-" fields; if "Resent-" fields are not found by LISTSERV in the headers of the approved posting, the posting will appear as coming from the editor's address rather than from the original poster. If your mail program does not support "Resent-" fields, you should use the "Send= Editor,Hold" option and approve messages with the "OK" mechanism described below.*

If you do not need to physically edit the content of your users' posts (for instance, to remove anything considered "off-topic" or to remove included mail headers and so forth), you can code

```
* Send= Editor,Hold
```

The "Hold" parameter causes LISTSERV to send you a copy of the posting along with a "command confirmation request". To approve the posting, you simply reply to the confirmation request with "ok".

For security purposes, you can code

```
* Send= Editor,Confirm
```

which will cause LISTSERV to request a command confirmation ("ok") from the editor sending the approved posting back to the list. This makes it impossible for an outside user to "spoof" mail from an Editor address.

Naturally, you can also code

```
* Send= Editor,Hold,Confirm
```

Finally, please note that the **NOPOST** subscriber option will take precedence over **Editor=**, if set for someone defined as an editor. This means that if you have "**Default-Options= NOPOST**" for your list and you add an editor as a subscriber, you will have to manually reset the editor to **POST** (with "**SET** *listname* **POST FOR** *userid@host*") before things will work properly. You will know that this is necessary if your editor can successfully approve postings but is then told that he or she cannot post to the list.

2.13.4. Moderated lists

Note: The Moderator= keyword is disabled in LISTSERV Lite.

A moderated list is similar to an edited list, but for LISTSERV's purposes it refers to a list

that uses the Moderator= list header keyword to "load-share" posting approvals among several editors. It is set up similarly to an edited list, as follows:

```
* Send= Editor,Confirm
* Editor= someuser@somehost.com
* Moderator= someuser@somehost.com,anotheruser@anotherhost.com
* Moderator= yetanotheruser@his.host.com
```

This list will "load-share" the approval process between the three moderators, who will each receive one-third of the postings for approval. Note that a primary editor should still be defined.

If it is desired to have one editor handle more than a single share of the approvals, you simply define the editor more than once in Moderator=. For instance,

```
* Send= Editor,Confirm
* Editor= someuser@somehost.com
* Moderator= someuser@somehost.com,anotheruser@anotherhost.com
* Moderator= someuser@somehost.com,yetanotheruser@his.host.com
```

would cause every other posting to be forwarded to someuser@somehost.com for approval.

Beginning with 1.8c, if the parameter "All" is coded at the beginning of the list of moderators, LISTSERV will send copies of all postings to all moderators, any of whom may approve the message. An example of this would be

```
* Moderator= All,kent@net.police.net,joe@bar.edu
```

Please note that something like

```
* Moderator= kent@net.police.net,All,joe@bar.edu,alex@reges.com
```

is not valid. "All" must appear at the beginning of the list of moderators.

Assuming "Send= Editor, Hold", once a message is approved by one of the moderators, any other moderator attempting to approve the same message will be told that the message cannot be found and has probably expired (since the cookie for that message will be gone).

If the message body is edited in any way before it is approved (i.e., by forwarding an edited copy back to the list), and more than one moderator is involved, duplicates are possible. Thus it is important that the moderators of any list set up this way pay close attention to whether or not the posting has already been approved by another moderator. Note carefully that this means if the "All" parameter is used in "Moderator=" with "Send= Editor" (that is, without the "Hold" parameter), again a separate synchronization method will have to be used to prevent duplicates, as two moderators are unlikely to make exactly the same edits to the message. Even if LISTSERV were able to identify the two submissions as being the same message, it would not know which to choose over the other.

2.13.5. Semi-moderated lists

"Semi-moderation" was developed some years ago after a great debate on whether or not an "urgent" message should be allowed to be posted to an edited list without having to go through the approval process. Although this option is still available, it can be

misused by anyone who knows about it, and is therefore not generally recommended for use. However, should this feature be deemed necessary, it is activated by setting

`* Send= Editor,Semi-Moderated`

Then anyone needing to send an "urgent" message to the list simply types "Urgent:" in the subject line of their mail, followed by the subject of the message. Messages that do not have the "Urgent:" subject are forwarded to the list editor for approval as usual.

2.13.6. Self-moderated lists

So-called "self-moderated" lists were invented in 1993 or 1994 when the current epidemic of spamming was beginning to get cranked up and before the "spam filter" was developed by L-Soft. With the spam filter in operation, self-moderation is not as much of an issue anymore, but some lists still run this way.

Self-moderation takes advantage of the ability to make an *access-level* a secondary list editor, and is implemented as follows:

`* Send= Editor,Confirm`
`* Editor= someone@someplace.com,(`*`listname`*`)`

(The "Hold" and "Confirm" parameters for "Send=" may naturally be used if required. L-Soft recommends that "Confirm" be used by default.)

Usually, one of the list owners is the primary editor (here "someone@someplace.com") and the specification of **`(`*`listname`*`)`** makes all of the subscribers of the **`listname`** list editors, and thus eligible to send messages directly to the list without editor intervention. Postings from non-subscribers (e.g., spammers) are deflected to the primary owner for his or her disposition.

There is one caveat to this kind of list. If a user subscribes to the list, and later his mail address changes (for instance, the hostname changes slightly but mail sent to the old address is automatically forwarded to the new address), any postings from him to the list from the new address will be forwarded to the editor because the new address is not subscribed to the list. Thus there is a certain amount of list-owner overhead on this kind of list in keeping track of users whose addresses have changed and modifying the subscriber list to reflect those changes.

2.13.7. Private edited/moderated lists

This type of edited or moderated list allows subscribed users to post with editor or moderator intervention, but rejects postings received from non-subscribers with a note to the poster stating that they are not allowed to post.

Using the same header you would create for an private discussion list (see 2.13.2, above), simply add the following line to the header:

`* Default-Options= REVIEW`

You should also add Editor= and (optionally) Moderator= keyword settings to the list. At least one editor must be defined to handle the message approval chores, otherwise the first listed list owner will receive the messages for approval.

Note the following carefully:

- For brand-new lists or existing lists which have no subscribers, all subscribers added to the list after this option is set will be set to REVIEW, and nothing further needs to be done.

- For existing lists with existing subscribers, you will need to set the existing subscribers to the REVIEW option manually, ie, with the command

QUIET SET *listname* **REVIEW FOR *@***

New subscribers who sign up or are added after you add the Default-0ptions= keyword setting will automatically be set to the REVIEW option.

Finally, note that the list editor will also be set to REVIEW if he is subscribed to the list under this scenario. This can be important if the list editor wants to approve even his own postings (for instance, to help avoid someone spoofing mail to the list from his address). If the list editor does not require this "suspenders and belt" level of security, he can simply set himself to NOREVIEW.

2.13.8. Auto-responders

Since LISTSERV Lite does not support list-level mail templates, this functionality is effectively not available in LISTSERV Lite.

An "auto-responder" is a type of list that simply responds with a set message whenever it receives mail from someone. This kind of list can be useful for things like service messages or upgrade availability, or even to simply send back a standardized message to a user who has sent mail to a "support" address.

A simple auto-responder header might look like this:

```
* Auto-responder for service messages
* Owner= someone@someplace.com
* Send= Public     Notebook= No     Subscription= Closed
```

In other words, it can be very simple, since you probably don't want notebook archives for this kind of auto-responder, you don't want people to subscribe to the list as it isn't really a mailing list, and so forth. To make the auto-response message for this list, you'd then create a `listname.MAILTPL` file (see chapter 10 for details) that includes a `POSTACK1` template, like the following:

```
>>> POSTACK1 Service Message for &MYNAMES
&MYNAMES will be down Sunday from 0200 EST until 0500 EST for
backups    and    upgrades.    For    more    information    contact
LSTMAINT@&MYHOST.
```

This particular template would inform the user that LISTSERV would be down (`&MYNAMES` translates to `LISTSERV@NODE` where `NODE` is the value of `NODE=` in the system configuration file) and to send questions to `LSTMAINT@` the local host. In order to change the service message, it would be necessary only to change the `POSTACK1` template.

2.13.9. Announce-only lists

An "announce-only" list would be used to distribute a newsletter or other timely information where responses to the list are neither expected nor desired. A typical

announce-only list header might look like this:

```
* The FOO Product Announcment List
*
* Owner= foo@myhost.com
* Owner= Quiet:
* Owner= anotheruser@myhost.com,yetanotheruser@myhost.com
* Editor= foo@myhost.com
* Editor= anotheruser@myhost.com,yetanotheruser@myhost.com
* Notebook= No
* Errors-To= Owner
* Subscription= Open,Confirm
* Validate= No
* Review= Owners
* Send= Editor,Confirm
* Reply-To= foo@myhost.com,Ignore
* Sender= None
```

This list is set up so that generally any response to postings will go back to foo@myhost.com, which might be a special account set up specifically to handle such things, or a mail alias pointing to another account. The newsletter can be posted by **foo**, or **anotheruser**, or **yetanotheruser**, all of whom are editors, but the likelihood is that it would be posted from the **foo** userid so that the From: line would read "From: foo@myhost.com".

2.13.10. Restricted subscription lists with automatically-generated questionnaire

Since LISTSERV Lite does not support list-level mail templates, this functionality is effectively not available in LISTSERV Lite.

Sometimes it is desired to send out a little questionnaire before approving a subscription to a list with a very narrowly-defined topic or to lists created for members of specific organizations. By setting "**Subscription= By_Owner**", you can of course force all potential subscriptions to require list owner approval. In the "old days", if you wanted more information before you approved the subscription request, you had to manually send a questionnaire out to the user and wait for him or her to return it to you.

By setting "**Subscription= By_Owner**" and adding two simple template forms to your *listname*.**MAILTPL** (as explained in chapter 9), you can now have LISTSERV send your questionnaire out automatically, as soon as the subscription request is received.

The first template form you need to add to *listname*.**MAILTPL** is called **SUB_OWNER**, and in this case it would typically look like this:

```
>>> SUB_OWNER &LISTNAME: &WHOM requested to join
.TO &WHOM
A copy of the &LISTNAME membership questionnaire has been sent
to you.  Please read it carefully and follow the instructions
to complete it and return it to the list owners.
```

The **.TO &WHOM** directive is required so that the message is sent to the subscriber rather than to the list owner. If you want the non-quiet list owners to receive a copy of this message (which is admittedly unlikely), you can simply add **CC: &OWNERS** to the end of the **.TO** line, e.g.,

```
.TO &WHOM CC: &OWNERS
```

Or, if you want to cc: a specific user such as joe@unix1.example.com, use

```
.TO &WHOM CC: joe@unix1.example.com
```

Note that you cannot format the `SUB_OWNER` template; it all comes out as one long paragraph without formatting no matter what you do, because it is a "linear" template. But you should modify it from the default to let people know that they will receive a questionnaire to be filled out and returned.

The second template form you need to add to *listname*`.MAILTPL` is called `ADDREQ1` and it can be as simple or as detailed as you want. All of the available template formatting commands can be used in `ADDREQ1`. For instance:

```
>>> ADDREQ1 &LISTNAME Membership Survey
.RE OWNERS
.TO &WHOM
.CE &LISTNAME Membership Survey
NOTE:  Please make sure when you send this back that it goes to
the address &LISTNAME-Request@&MYHOST.  Thanks.

This is a standard questionnaire required for all prospective
subscribers to &LISTNAME. Blah blah blah...
```

In this case you want the message to go to the subscriber, with a Reply-To: header pointing back to the (non-quiet) list owners. The first line indicating the return address is added for those users with mail clients that don't recognize Reply-To: headers.

You can also put a pre-formatted `ADD` job into the questionnaire to simplify your job when the questionnaire comes back. For instance,

```
.fo off
------------------------------------------------------------------
For List Owner's Use Only --  Be sure to include with your Reply
------------------------------------------------------------------
// JOB
  ADD &LISTNAME &WHOM &USERNAME
// EOJ
------------------------------------------------------------------
.fo on
```

For more detailed information on mail templates, see chapter 9.

2.13.11. Peered lists

This functionality is not available in LISTSERV Lite.

Please consult your LISTSERV maintainer before peering lists.

Occasionally the need to split a very large list may arise. This was more common when LISTSERV ran only on BITNET, whereas the TCP/IP version of LISTSERV is not limited by BITNET constraints. However, because of the fact that subscribers may be scattered all over the world, in rare cases it can make sense to split (or "peer") a list and share the mail load among two or more LISTSERV servers. Peering also makes it possible to have

list archives located in more than one place; for example, a list might be peered between a European host and a North American host, making it possible for subscribers on each continent to retrieve archives from the nearer host.

Although there is no problem about peering to another L-Soft LISTSERV list, linking to a non-L-Soft mailing list manager is *not* supported and can and will cause serious problems (including mailing loops) for which L-Soft international, Inc. could not be held responsible.

After the link operation has been completed, it is recommended that you define "Peers=" keywords on lists you just linked. For lists running on LISTSERV for VM, this makes it possible to **EXPLODE** them for better network efficiency. (Because peering is not widely used today, it is unlikely that the EXPLODE command will be ported to other platforms.)

Note also that the peer lists MUST have the same list password (**PW=** list header keyword setting) or messages approved on one peer will not be accepted by the other peer and an error message will be generated, i.e.,

```
The approval request code received together  with your posting for the MYLIST-L
list is  incorrect. For  a peered  list, this  may be  a normal  condition. The
approval protocol  is not  guaranteed to  work among  peer chains with pre-1.8b
servers, and  will also  fail if  the peers  have a  different password.  For a
non-peered list, the only likely explanation is a failure in the mail system or
a recent change in mail  system version or  configuration. At any  rate, please
resubmit your message and go through  the approval procedure a second time, and
contact the LISTSERV administrator if the problem persists.

------------------------ Rejected message (73 lines) -------------------------
```

This means that under LISTSERV 1.8c and later you must explicitly set the **PW=** list header keyword for each peer and not use the password LISTSERV generates automatically at list creation time. This is the only situation in which the **PW=** keyword must explicitly be set to a specific value.

Moving users from one (peer) server to another:

You should be aware of the fact that a **MOVE** operation is not just an **ADD** to the new server and a **DELete** to the current one. This would effectively transfer the person from the old server to the new one but his distribution options would be lost in the process. Besides, you should make sure that the user does not lose any mail in the process. The proper course of action to be taken when people are moved from one list to the other is the following:

1. Send mail to the list telling people that a new peer server is being linked to the list, and that some subscribers will be moved to it.

2a. If the prerequisites for using the **MOVE** command are met, you should use either individual **MOVE** commands (in the case that there are very few users to move) or a batch-**MOVE** command with associated **DDname** (see the LISTJOB MEMO guide for more information on commands-jobs) to move the users. You may want to use the **QUIET** option to suppress notification if there are a lot of users to move.

Warning: the **MOVE** command should not be used to move peer list servers. See the **MOVE** command description for more details.

If you cannot use the **MOVE** command, you should try one of the following two methods:

2b. For each user to be moved, issue the following commands in the following order:

- **Query** *listname* **FOR** *userid@host* (old server), write down the options.
- **QUIET ADD** *listname userid@host full_name*
- **QUIET SET** *listname options* **FOR** *userid@host*
- Wait until you get confirmation for the two previous commands
- **QUIET DELete** *listname userid@host* (old server)

2c. If there are a lot of users to move, the following method is preferred:

- **GET** *listname* (old server)
- **GET** *listname* (new server)
- ***If you are using VM XEDIT:*** Receive both files and use the XEDIT "PUT" and "GET" commands to move users from one list to the other. You ***must*** preserve the contents of columns 81-100 across the move.
- ***If you are using another text editor:*** Make sure that the editor you are using does not "imbed" control codes such as line breaks, tabs or word-wrapping characters into the text when you edit it. Use the cut and paste controls to copy lines in their entirety. You ***must*** preserve the contents of columns 81-100 across the move. Imbedded control codes and/or word wrap will generate errors when the list is stored back on the server.
- Store the two lists back on their respective servers.

Special commands for peered lists only

ADDHere *listname userid@host* **<**full_name**> <PW=***list_password***>**

The **ADDHERE** command is strictly identical to ADD, with the exception that the placement of the user is not checked against the list of peer servers, i.e. the specified user is added to the local list without any further verification. (By comparison, the ADD command causes LISTSERV to check automatically to see if there is no better-suited peer list for the specified user.)

EXPLODE *listname* **<F=***fformat***> [VM only]**

The **EXPLODE** command provides a means whereby a list can be automatically analyzed by LISTSERV to optimize the placement of its recipients over the various peer servers hosting the list. It requires a "Peers=" keyword to be defined in the list header (see Appendix B). Non-BITNET userids will be exploded according to the network address of the corresponding gateway (as per the SERVICE NAMES file), or ignored if the gateway could not be identified. LISTSERV will create a commands-job file containing the necessary **MOVE** command to transfer all the users which were found to be (possibly) mis-allocated to the peer server which is nearest to them. This file will then be sent to you so that you can review it before sending it back to the server for execution.

MOVE *listname userid@host* **<TO>** *newhost* **<PW=***list_password***>**
  DD=*ddname listid@newhost* **[VM only]**

The **MOVE** command allows list owners to easily move users from one peer server to another. It will move the complete user entry from the source server to the destination one, including full name as it appears in the specified list and all list distribution options. The **MOVE** operation will be done in such a way that no mail can possibly be lost by the target while the **MOVE** operation is in progress (duplicate mail might be received for a short duration, however). Notification will be sent to the target user unless the **QUIET** option was used.

If the source and destination list names are identical, only the destination node ('newhost') needs be specified. Otherwise, the full network address ('listid@newhost') must be specified.

The `MOVE` command requires both source and destination lists to have the same password. Since each server will have to send a password to the other to validate the (special) `ADD/DELETE` commands it is sending to the other, it has potentially a way to trap the password specified by the server, thus thwarting any attempt at inventing a protocol to allow use of this command on lists which have a different password. Besides, no `MOVE` operation will be accepted on lists which do not have a password at all, because for technical reasons it would allow unauthorized users to easily add someone to a list (since there would be no password validation).

The `MOVE` command is the proper way to effect a move operation. You should not use any other command/set of commands unless you cannot use `MOVE`. THE `MOVE` COMMAND SHOULD NOT BE USED TO MOVE DISTRIBUTION LISTS!!! Since a `MOVE` is basically an `ADD + DELETE`, with the latter being done only AFTER the `ADD` is completed, moving a distribution list address with the `MOVE` command can cause a duplicate link to be defined for a short period of time. This could result in a transient mailing loop, which could become permanent if the size of the looping mailfiles is less than the size of the inter-servers "DELETE" command jobfile, and the RSCS priority of the latter has been altered.

2.13.12. "Super-lists" and "sub-lists"

This functionality is not available in LISTSERV Lite.

Please note that the LISTSERV maintainer must create the super-list.

In LISTSERV Classic 1.8c and following it is possible to define a "super-list" (as in opposite of sub-list), that is, a "container" list that includes all the subscribers in a predefined set of sub-lists. This can be done recursively to any depth. Only the LISTSERV maintainer can create a super-list, for security reasons. Concretely, the "Sub-lists=" keyword is protected from owner tampering in the same fashion as "Notebook=". The value is a comma separated list of all the sub-lists, which must all be on the same (local) machine. For instance:

```
* Sub-lists= MYLIST-L,MYOTHERLIST-L
```

The default value for this keyword is null, e.g., to have no sublists.   Please note that the super-list and all of its sublists must reside on the same LISTSERV server.

The only difference between a normal list and a super-list is what happens when you post to it. With the super-list, the membership of all the sub-lists is added (recursively) and duplicates are suppressed. Other than that, the super-list is a normal list with its own archives, access control, etc. You can even subscribe to it, and this is actually an important aspect of the operation of super-lists. If you are subscribed to the super-list itself, the subscription options used to deliver super-messages to you are taken from your subscription to the super-list, just like with any other list. All combinations are allowed, and in particular NOMAIL is allowed, meaning you don't want to get messages posted to the super-list. When you are subscribed to multiple sub-lists, on the other hand, things work differently:

1.   NOMAIL subscriptions are ignored. You will get the super-message if you have an

active (not NOMAIL) subscription to at least one sub-list. The idea is that the super-message must be equivalent to posting to all the sub-lists, without the duplicates. Since all it takes to get a message posted to all the sub-lists is a single non-NOMAIL subscription, this is how the super-list works. The only way not to get the super-messages is to subscribe to the super-list directly and set yourself to NOMAIL.

2. The DIGEST and INDEX options are ignored and internally converted to MAIL. The first reason is that, since in most cases the user will be on multiple sub-lists (otherwise you don't need a super-list in the first place), the only safe method to set subscription options for super-messages is by subscribing to the super-list so that there is no ambiguity. The second reason is that, in most cases, super-lists will be used for out of band administrative messages rather than for large volume discussions, so it is actually preferable to have the message sent directly. The third reason is that the super-list and sub-lists may not necessarily offer the same options (DIGEST and INDEX). In particular it is expected that many super-lists will not have archives. If you want a DIGEST or INDEX for the super-messages, you must subscribe to the super-list directly.

3. In LISTSERV 1.8c and 1.8d, the REPRO option is NOT inherited by sub-lists. That is to say, even if the sub-list subscriber is set to REPRO on the sub-list AND the super-list is set up such that sub-list subscribers may post directly to it, he will NOT receive a copy of his own posting. REPRO is effective only for users who are directly subscribed to the super-list. This restriction has been removed in LISTSERV 1.8e.

Topics, if defined, are evaluated on a per-list basis. That is, for every sub-list (and for the super-list), LISTSERV determines whether the topic of the message is one that you want to see. If not, it acts as if you were not subscribed to this particular list. Roughly speaking, this works very well if all the sub-lists have the same set of topics (or a well-defined set of common topics), and doesn't work well at all if every list has its own set of topics.

Postings to a super-list are always archived in the super-list's notebooks (if enabled), and never in the notebooks of the sub-lists. This is because by its nature a posting to the super-list is not equivalent to cross-posting a message to all of the sub-lists. Rather, LISTSERV recurses into the sub-lists and generates an "on the fly" listing of all of the users on the super-list and the sub-lists (this is how it avoids duplicates, among other things) and then treats this "on the fly" listing as if it were the subscriber list of the super-list itself. You will note that a super-list posting is always identified as coming from the super-list, regardless of whether a given user is subscribed to the super-list or to one or more of the sub-lists.

Note carefully that a **REVIEW** command sent for the super-list *will not* recurse into the sub-lists pointed to by the super-list. If you have a super-list called SUPER and you send a **REVIEW SUPER** command, LISTSERV will respond with only the people who are subscribed directly to SUPER.

LISTSERV 1.8c and 1.8d: Also note that the REPRO option is honoured only when the user posting to the super-list is subscribed to the super-list with the REPRO option set. The REPRO option is not evaluated when LISTSERV recurses into the sub-lists. Thus if you have a super-list that is set up so that all of the subscribers of the sub-lists are able to post to it without actually being subscribed to the super-list, they will not receive copies of their own postings even if they are set to REPRO on the sub-lists.

LISTSERV 1.8e and following: The above restriction has been removed and REPRO works for users who are subscribed to the sub-lists.

Similarly, access to the super-list's notebook archives is not automatically recursive. If

you want sub-list subscribers to be able to access the archives of the super-list (but don't want the sub-list subscribers to have to subscribe to the super-list), then you must configure the Notebook= keyword for the super-list so that it contains references to each of the sublists. For example, say we have a super-list called SUPER and two sub-lists called SUB-A and SUB-B. We want the subscribers of both SUB-A and SUB-B to be able to read the archives of SUPER (since postings to SUPER won't be archived in SUB-A or SUB-B), but we *don't* want people who aren't susbcribed to any of the three lists to be able to access the archives. So we set

```
* Notebook= Yes,C:\LISTS\SUPER,Monthly,Private,(SUB-A),(SUB-B)
```

and anyone subscribed to the SUPER list or to the SUB-A or SUB-B lists can access the SUPER archives.

If you have many sub-lists, you can specify multiple Notebook= lines, e.g.,

```
* Notebook= Yes,C:\LISTS\SUPER,Monthly,Private,(SUB-A),(SUB-B)
* Notebook= (SUB-C),(SUB-D),(SUB-E),(SUB-F)
```

LISTSERV will read these two (or more) Notebook= lines and concatenate the values.

2.13.13. "Cloning" lists

Some sites may have a need for many lists that are essentially identical. For instance, a series of class section lists for a university department may have the same owner, allow the same class of users to subscribe, and so forth. LISTSERV makes it possible to maintain large collections of lists by "including" keywords from an external file.

For instance, consider a mathematics course with ten sections. Each section should have its own list (for instance, called **M101-001**, **M101-002**, and so forth), but the lists will otherwise be identical. The LISTSERV maintainer simply creates a text file (in this case called **M101 KEYWORDS**) containing the keyword definitions that will be shared by the lists, as follows:

```
PUT M101 KEYWORDS PW=createpw
* Owner= mathwhiz@someuni.edu (Professor J. Random User)
* Owner= Quiet:
* Owner= gradasst@someuni.edu (Joe Doakes, Graduate Assistant)
* Notebook= Yes,/home/listserv/archives/m101,Monthly,Private
* Auto-Delete= No
* Errors-To= gradasst@someuni.edu
* Subscription= Closed
* Notify= Yes              Confidential= Yes        Validate= Yes,Confirm,NoPW
* Reply-to= List,Ignore    Review= Owners           Send= Private
* Default-Options= Repro
```

Next, the LISTSERV maintainer stores this file in the usual way, by first making a filelist or catalog entry for it (as outlined in chapter 8) and then storing it with a **PUT** operation. Generally the **GET** and **PUT** FACs for this file should specify that the list owner(s) should be able to retrieve and store it. The file *must* be stored in LISTSERV's A directory (the same directory that contains the ***.LIST** files).

Note that it is also possible to create this file directly in LISTSERV's A directory with a text editor; if you do so, make sure that you do not include the PUT command shown above. You should still make the filelist or catalog entry for the file so that the list owners can retrieve and store it.

Next, the LISTSERV maintainer creates and stores a skeleton list header for each of the section lists. The first section list (`M101-001`) is illustrated below:

```
PUT M101-001 LIST PW=createpw
* Math 101 Section 001 Mailing List
* .IK M101
```

The `.IK` command tells LISTSERV that whenever it uses this list, it should read the keyword definitions from the file `M101 KEYWORDS` (note carefully that the syntax is "`.IK M101`", *not* "`.IK M101 KEYWORDS`"). Now, whenever the professor in charge of the class wants to make a change to all of the M101 lists (for instance, he has a new graduate assistant), he simply `GET`s the file `M101 KEYWORDS`, makes the changes, and `PUT`s the file back, instead of having to `GET` separate headers for each list and make the changes to all of them individually.

> Note: On some servers the LISTSERV maintainer may have to stop and restart LISTSERV (or do a `GET+PUT` of all of the list headers involved) to make changes to the `KEYWORDS` file appear. This is because LISTSERV may have the `KEYWORDS` file and/or the list headers that use it cached at the time you modify it.

In order to see the complete list header, send a `REVIEW listname` command. The response to a `GET` will be only the skeleton header with the `.IK` command.

Special note: The sample `KEYWORDS` file above includes a `Notebook=` keyword. This will cause the notelogs for all of the lists that use this `KEYWORDS` file to be written in the same directory, per the example, `/home/listserv/archives/m101` . This means that in that directory you would have notelogs for the `M101-001` list, the `M101-002` list, and so forth (depending of course on what lists use the example `M101 KEYWORDS` file). If this behavior is not desired, simply don't put a `Notebook=` keyword in the `KEYWORDS` file, and define it in the list header for the cloned list instead, either before or after the `.IK` directive.

For the web archive interface, note carefully that if you do use the same directory for all of the cloned lists' notelogs, the LISTSERV maintainer will still have to make separate web archive directories for each list under your `WWW_ARCHIVE_DIR` directory if you intend to serve the archives via the web interface. In other words, the web interface doesn't care where you keep a list's notelogs as long as it has a directory specified under `WWW_ARCHIVE_DIR` for it to write the list's web archive indexes into. So while all of your notelogs may go into `/home/listserv/archives/m101` , regardless of the name of the cloned list, the LISTSERV maintainer still needs to make (for example) `/usr/local/etc/httpd/htfiles/archives/m101-001` and so forth in order to serve the notelogs on the web.

2.14. List passwords are now obsolete

In LISTSERV 1.8c and later, when creating the list, a random password is assigned for security if the LISTSERV maintainer does not define one explicitly. In 1.8c and later it is no longer necessary to use the list password in all but one situation; it is simply another line of defense, and you can substitute a personal password in any command that formerly called for a list password. See section 2.9, above, to learn how to create a personal password.

The only situation in which a list password MUST be defined explicitly in a list header is

in the case of peered lists, where the `PW=` list header keyword must be set to the same value on all peers.

2.15. Allowing/Blocking MIME Attachments

LISTSERV 1.8d kits starting in May 2000 contain a new MIME attachment-filtering feature which is configured by setting the new `Attachments=` list header keyword. The new keyword allows three distinct modes:

- Allow all MIME attachments, no filtering or blocking

- Reject MIME attachments with notice to the poster

- Filter MIME attachments out of messages transparently

In addition, you can configure specific MIME types to reject or filter while allowing other types through (for instance, you can block executable files but allow images or word processing files based on their MIME type).

For information on the various settings, please see the section on the `Attachments=` keyword in Appendix B of this manual.

2.16. Content filtering

This feature requires LISTSERV 1.8e or later. It is not available in LISTSERV Lite.

This feature is intended primarily to filter out-of-office messages and the like. *It is not intended as a profanity filter.* Attempts to configure it to filter profanity will most likely prove to be futile in the long run and are not recommended by L-Soft.

The `CONTENT_FILTER` mail template form, if present, contains filtering rules, one rule per line, empty lines ignored. Each rule has the following format:

`[prefix:] pattern`

The prefix, if present, can be a mail header tag (eg "Subject:"); "Header:" to check the whole header; or "Text:" to search the message text. The latter is the default if no prefix is supplied, it is provided in case the pattern contains a colon in the first word. If there are multiple mail header tags with the specified name (eg "Received:"), each such tag is searched and it is enough for one of them to match the pattern. If the requested tag is not present in the header, there is (surprise!) no match. A text search will search every line of the first text/plain part in the message. If there is no text/plain part, there is no match. Again, this is designed to filter read receipts, loops, chain letters, spam, you name it. There was no attempt on the developers' part to make this a profanity filter, and future versions will not be "enhanced" to make futile attempts at (for instance) decoding Word documents to look for obscene words.

Regular comparisons such as those described above are not case sensitive. Patterns are standard LISTSERV patterns, that is, the asterisk is the wildcard character. If there is no asterisk in the pattern, it is replaced with "*pattern*" much like the SCAN command.

The content filter also supports "exact match" comparisons, which are triggered by a double colon. For instance:

`Subject::`

There are two significant differences between exact and regular match:

a. You must supply your own wildcard characters in an exact match (if you want to use wildcards, that is). A regular match will insert leading and trailing wildcards if none are found. Thus, an exact match is the only way to make a comparison without wildcards.

b. You can make an exact match for the empty string. Empty regular matches are ignored since they map to a wildcard comparison for **, which would be always true. This also makes it possible to apply an exact match to a message that does not contain a specified header. For instance, if you want all messages to contain a (mythical) KABOOM: RFC822 header, with an exact match you can tell LISTSERV to perform one of the content-filtering actions if the the header is not present. This is not possible with a regular match.

Note however that you cannot differentiate a header with an empty KABOOM field from a header with no KABOOM field.

One of the most handy uses for the exact match syntax is to be able to write a rule to reject messages with blank subject lines. For instance:

```
Subject::
Action: REJECT Please resubmit your message with a non-blank subject.
```

Every rule can, optionally, be followed by an action rule. This has the following format:

```
Action: ALLOW
Action: REJECT reason
Action: DISCARD comment
Action: MODERATE
```

(The available actions are the same for both regular and exact comparisons.) For instance,

```
>>> CONTENT_FILTER
Subject: Out of office
Action: REJECT OOO messages are not allowed on this list.
Subject: Auto-Generated:
Action: REJECT
Text: Click here to be removed
Action: REJECT Buzz off, spammer.
Subject::
Action: REJECT Please resubmit with a non-blank subject.
Subject: copyright
Action: MODERATE
To: friend@public.com
Action: DISCARD This guy is a spammer
```

The default is "Action: REJECT" with no specified reason. REJECT means that the message is rejected. MODERATE means that the message is to be forwarded to the list editor to be manually approved or rejected. DISCARD means that the message is to be dropped on the floor without further processing; any text following DISCARD is echoed to the LISTSERV console (and is thus logged).

ALLOW means that the message is allowed and all remaining rules are ignored. This could be used in moderated lists to allow the list moderator to bypass certain filters, for

instance:

```
>>> CONTENT_FILTER
Subject: Out of office
Action: REJECT OOO messages are not allowed on this list.
From: JOE@EXAMPLE.COM
Action: ALLOW
Text: Click here to be removed
Action: REJECT Buzz off, spammer.
```

In the example above, messages with Subject: lines containing "Out of office" are rejected. Messages containing the text "Click here to be removed" are also rejected UNLESS they come from joe@example.com .

The text of the rejection is fetched from the **BAD_CONTENT** mail template form, with the reason supplied as a variable called **&COMMENT**. The rejection message looks like this:

```
Date:        Tue, 4 Dec 2001 22:03:42 -0500
From:        "L-Soft list server at LISTSERV.EXAMPLE.COM (1.8e)"
             <LISTSERV@LISTSERV.EXAMPLE.COM>
Subject:     Rejected posting to TEST@LISTSERV.EXAMPLE.COM
To:          Joe User <joe@EXAMPLE.COM>

Your posting to the TEST list has been rejected by the content filter. OOO
messages are not allowed on this list.
```

followed by the text of the posting including all mail headers. (In this case the body of the message contained the text "out of office" and the rule above was applied.)

A default site-wide **CONTENT_FILTER** template form may be defined in $SITE$.MAILTPL for use by lists whose owners do not prefer to provide their own custom versions in their listname.MAILTPL files.

3. Advertising Your Public Mailing Lists

3.1. Lists of Lists maintained by LISTSERV

LISTSERV automatically produces a List of Lists that may be reviewed by users anywhere on the Internet in one of two ways:

- L-Soft's CataList service at **http://www.lsoft.com/CataList.html**

- The **LISTS GLOBAL** *searchtext* command. This list of lists is made up of one-line entries containing the short listname and the descriptive title of the list (up to about 60 characters in length). A sample of the List of Lists format was shown in Chapter 2.

Note that it is possible to code a descriptive title in your list header that is more than 40 columns long, but the List of Lists will include only the first 40 columns of that title. It is therefore important from this respect to be sure that the descriptive title of your list is succinct and to the point.

3.2. Adding HTML to a list header for the CataList

L-Soft's CataList service (**http://www.lsoft.com/lists/CataList.html**) allows users to search the global list of public LISTSERV lists via the World Wide Web. Adding an HTML description to a list is easy, and can do a lot to enhance the appearance of a list in the database. All you have to do is update your list header and add the text of your choice. Here is an example:

```
* The coffee lovers' list
*
* Review= Public     Subscription= Open          Send= Public
* Notify= Yes        Reply-to= List,Respect
* Notebook= Yes,L,Monthly,Public
*
* Owner=  claudia@espresso.xyz.it (Claudia Serafino)
*
* <HTML>
* COFFEE-LOVERS is an open list for, well, coffee lovers! Our
* motto is: <cite>"Instant – just say no!"</cite>
* That's pretty much our whole charter, although there are a
* few other <a href="http://www.coffee.org/charter.html">
* rules</a> that you may want to read before joining. For
* instance, we don't allow flame wars about decaf: if you like it,
* well, it's your body after all.
*
* <p>The list is maintained by
* <a href="http://www.coffee.org/claudia.html">Claudia
* Serafino</a> (that's me!) and you will find all sorts of
* useful info about coffee on my home page.
* </HTML>
*
```

In other words, you just insert your HTML text in the list header and bracket it with **<HTML>** and **</HTML>** tags (these tags tell the web interface where the HTML text begins and ends – they are not actually sent to the web browser). There are three simple rules that you must follow when inserting your HTML data:

1. The **<HTML>** and **</HTML>** tags must appear on a separate line, as shown in the

example above. You cannot have anything else on that line and, in particular, you cannot mix keyword definitions with HTML data.

2. The HTML data you are providing is embedded into the document shown by the web interface when users query your list. Because you are given some space between two horizontal rules on an existing page, rather than a whole new page. you should not include tags that affect the whole document, like for instance **`<TITLE>`**.

3. While this procedure is compatible with all versions of LISTSERV, there are a few restrictions on the placement of equal signs within your HTML text with versions that do not have any specific support for the **`<HTML>`** and **`</HTML>`** markers. In practice, you can ignore this rule unless you get an error message while storing your list.

When reformatting your list header description for HTML, bear in mind that the text will not always be viewed using a web browser. It is best to keep the formatting as clear as possible and minimize the usage of HTML tags, since there are still many people without WWW access. For instance, do not hesitate to use white space between paragraphs for clarity.

3.2.1. Update latency

Barring network outages, a list header update takes a maximum of 24h to be reflected in the distributed LISTS database. Database updates are usually scheduled to be broadcast at night, so the changes take place overnight. Once the LISTS database has been updated, it can take a maximum of 24h for the frozen copy of the database used by the web interface to be updated. In most cases, both the LISTS database and its frozen copy on the web server will be updated overnight. However, if the site hosting your lists is several time zones west of the site hosting the web server, and if that server only updates itself once a day, you may have to wait two days for your update to be reflected.

3.2.2. Inserting a pointer to another list

Sometimes it may be useful to link a number of related lists together so that the viewer can quickly examine all the lists without having to go back to the search screen and retyping the names you are providing. You can do this using the special HTML sequence:

`<!--#listref listname@hostname-->`

This sequence is internally translated to an **`<a>`** tag with a URL that will bring up information about the list you indicated. You must then provide a suitable caption and a closing **`</a>`** tag. Example:

```
Don't forget to take a look at
<!--#listref COFFEE-L@COFFEE.ORG-->
the coffee list!</a>
```

3.2.3. Restrictions on the placement of equal signs

While all versions of LISTSERV are supported, servers which have no specific support for the **`<HTML>`** and **`</HTML>`** tags will process your HTML data as an ordinary list header line and attempt to determine whether it contains a list header keyword or descriptive text. The exact algorithms vary from one version to another, but in general the parser looks for a single word followed by an equal sign. With HTML text, it is possible (if unlikely) to generate such patterns. Here is an example:

```
*
* Sample list with problem pattern
*
* <HTML>
* For more information on the list, just check <a
* href="http://www.xyz.edu/mypage.html">my home page.</a>
* </HTML>
*
```

In that case, you can just reorder the HTML data so that the equal sign does not appear in this position. Alternatively, if the equal sign was meant to be actually displayed as an equal sign (as opposed to being part of some HTML tag), you can use the HTML escape sequence `=` instead.

3.3. Defining search categories in a list header for the CataList

> Note: The complete list of search categories may not yet be available when LISTSERV 1.8e is released. Note also that during the "pilot" phase of categories implementation, all categories will be "open", and you can define search categories for your list as long as the categories you define are in compliance with the rules for defining categories. When the "production" phase begins, only categories defined below as "open" will be open, and if a list is created or modified without a "Categories=" keyword, LISTSERV will issue a warning (but will go ahead and store the list without it).

Another feature of the CataList service discussed in the preceding chapter is the ability to search for lists based on topic categories. For instance, a user might be looking for lists that discuss various aspects of opera. The same user might want to search not just for lists that discuss opera in general, but great operatic tenors in particular.

In order to implement search categories for your list, you use the new "Categories=" list header keyword, in conjunction with the list of categories that can be found at the CataList site. The URL for the category list is `http://www.lsoft.com/listcat.html`.

If you do not have a web browser, you can issue the command

GET LISTCAT FILE

to LISTSERV@LISTSERV.NET or any LISTSERV server running version 1.8c or higher to have a list of categories mailed to you.

A typical category listing is in two parts. The first part is the category title itself (this is what you code in the "Categories=" keyword). The second part is an optional description of what the category covers. For instance:

`Category:SubCategory:MinorCategory Description of this category`

There are two types of categories that you need to be aware of.

Open Categories: These categories have a description indicating that they are open and can be added to. Taking our example of great operatic tenors above, you might see the following category listed:

`Arts:Music:Opera:Singers Operatic Singers (Open)`

You notice that there are further subcategories like

```
Arts:Music:Opera:Singers:Te_Kanawa_Kiri
Arts:Music:Opera:Singers:Caruso_Enrico
```

and so forth, but (gasp!) no category for your favorite tenor, Luciano Pavarotti! And your list is PAVAROTTI-L. Not to worry, however. Because the category of "Singers" is open, you can simply code:

```
* Categories= Arts:Music:Opera:Singers:Pavarotti_Luciano
```

and LISTSERV will accept the new subcategory "Pavarotti_Luciano".

Note that when you create a new category, it will not show up until the central categories list has been updated.

Note also that there are two "root level" open categories, **Misc** and **Local**. The **Misc** category is world-searchable. If, however, you code a **Local** category, it will only be searchable from the search engine running on the server hosting your list.

<u>Closed Categories:</u> These are categories that cannot be added to. In other words, if you see a category like:

```
Computers:Internet:Mailing_List_Managers:LISTSERV:Manuals:List_Owners_Manual   List
Owner's Manual for LISTSERV
```

whose description does not indicate that it is open, then you cannot add new categories after the last term. If you try to create a new subcategory under a closed category, you will receive an error message when you PUT your list header, and your updated header will not be stored.

3.3.1. Examples of category settings

Categories are defined by the new "Categories=" list header keyword. Each category string's subcategories are internally delimited with colon ("**:**") characters. Each separate category string is separated from the others with commas. If your "Categories=" keyword setting gets too long to fit on one line, simply define multiple "Categories=" keywords. Note that spaces are not allowed in categories; therefore

```
* Categories= Arts:Music:Opera:Singers:Luciano Pavarotti
```

is not legal, but

```
* Categories= Arts:Music:Opera:Singers:Luciano_Pavarotti
```

is.

A simple category setting would be:

```
* Categories= Arts:Music:Opera:Singers
```

and if someone searched on that category, they would find our list. But we saw above that we can create a new category if we are running a list dedicated to Luciano Pavarotti. So instead, we might code

```
* Categories= Arts:Music:Opera:Singers:Pavarotti_Luciano
```

If, however, we're running a list for the Three Tenors, we might want to code:

```
* Categories= Arts:Music:Opera:Singers:Pavarotti_Luciano
* Categories= Arts:Music:Opera:Singers:Domingo_Placido
* Categories= Arts:Music:Opera:Singers:Carreras_Jose
```

Or even:

```
* Categories= Arts:Music:Opera:Singers:Three_Tenors
```

depending on our preference.

If you code a sub-category that does not exist in a "closed" upper-level category, LISTSERV will respond with an error message that will list the legal sub-categories that you can use.

3.4. The INFO <listname> command and how to implement it

This functionality is not available in LISTSERV Lite.

Chapter 9, *Customizing LISTSERV's Default Mail Templates*, includes details on how to include an informative paragraph in the information mail template file for your list. When a user sends the command **INFO** *listname* to your server, LISTSERV responds with either:

- The default response, which simply sends a copy of the list header to the user; or
- The customized paragraph included in the *listname*.MAILTPL file.

If *listname*.MAILTPL does not exist, the default response is sent. Also note that the user may send the **INFO** *listname* command to any L-Soft LISTSERV host (including the Global List Exchange discussed below), which will forward the request to the appropriate server.

3.5. The NEW-LIST project

The NEW-LIST project was started in 1989 to promote mailing lists via a mailing list. Originally hosted on NDSUVM and later on LISTSERV.NODAK.EDU at North Dakota State University, NEW-LIST was moved to the Internet Scout Project in July of 1998, and then in June, 2000 to Classroom Connect (www.classroom.com). The NEW-LIST administration asks only that your list be well-tested and ready for new subscriptions before you send your announcement to them. You also want to make sure that your announcement is as correct and comprehensive as possible, as news on the Internet spreads quickly and a mistake in a NEW-LIST announcement may cause problems for both you and other users months later.

For more information on the NEW-LIST project and what you need to use it, you should point a World Wide Web browser at the URL

http://listserv.classroom.com/archives/new-list.html

(As a matter of historical note, the NEW-LIST Project published a hard-copy version of their archive in 1992 with a newer edition in 1993 under the title *Internet: Mailing Lists* [ISBN 0-133-27941-3], edited by Edward T. L. Hardie and Vivian Neou.)

3.6. The Internet Network Information Center (INTERNIC)

Unlike many other lookup services on the Internet, the INTERNIC is not necessarily free. Its three distinct sections are run by General Atomics, Network Solutions, Inc. (NSI), and AT&T.

You can register your list with the INTERNIC, but be forewarned. A "basic" listing is free, while an "extended" listing is not. (On the other hand, anyone with net access can search the INTERNIC databases for free.)

For more information, point a WWW client at the INTERNIC web site at `http://www.internic.net` .

3.7. The Global List Exchange (GLX) and why you should mention it

The Global List Exchange, or GLX, is a central clearinghouse for LISTSERV subscriptions and List of List requests. For instance, If a user knows the name of a list but not the name of the host server, GLX simplifies the process by giving the user a single address where all subscription requests for lists running on L-Soft's LISTSERV can be sent.

By adding the GLX address in all advertisements for your list, you help other list owners as well as yourself by making it simple for users to subscribe to any list. Additionally, if for some reason a user is unable to contact your server directly, the GLX gives him an alternate subscription method.

The GLX address is LISTSERV@LISTSERV.NET.

3.8. How NOT to advertise a mailing list

It is generally considered a breach of netiquette to invade the privacy of other lists with a broadcast announcement that your list is up and running. The only time when this *might* be acceptable is when your list addresses a concern of people already subscribed to another list. If you feel it necessary to post an announcement on someone else's list, it is good manners to first send private mail to the owner of that list and ask his or her permission to do so. (The same policy applies to USENET newsgroups, though it may be more difficult to find out who the moderator is.)

It is certainly a breach of netiquette (and many networks' appropriate use policies) to blindly post multiple copies of your announcements to multiple lists. This kind of behavior is termed a "spam", something about which you may read more in Chapter 6, *Moderating and Editing Lists.* This kind of announcement is guaranteed to reap a good deal of bad will and may well result in the revocation of your network privileges.

4. Managing Subscriptions

4.1. How to add and delete subscribers to/from a list

A list owner may add and delete subscribers manually. The command syntax is:

```
ADD listname netaddress full_name
DELete listname netaddress
```

In a perfect world, subscribers would understand intuitively how to subscribe and unsubscribe from mailing lists. Unfortunately, this is not always the case. Depending on an individual's style of list management, a list owner may choose to add or delete subscribers to the list manually, or send the potential subscriber instructions on how it is done. (See Appendix C for sample "boilerplate" instruction files that can be modified to suit local purposes.) And for lists coded **Subscription= By_Owner** or **Subscription= Closed**, it is of course necessary to use the **ADD** command to subscribe a user.

If the list is set to confirm mailing paths for new subscriptions (**Subscription= Open,Confirm**), it is probably wisest to use the latter option, since if a subscriber is added manually to a list, the confirmation process is bypassed.

Note that *full_name* should contain at least two discrete words, but it is also possible to add users without knowing the value for *full_name.* Simply use an asterisk ("*") character. Note that if the user is already subscribed to another list on the same host, LISTSERV will pick up the value for *full_name* from its signup files. Examples are:

RIGHT:	**ADD GOV-L vice-president@whitehouse.gov Al Gore**
RIGHT:	**ADD GOV-L vice-president@whitehouse.gov ***
WRONG:	*ADD GOV-L vice-president@whitehouse.gov Al*
WRONG:	*ADD GOV-L vice-president@whitehouse.gov Al-Gore*

When adding users, **ADD** will also accept a full RFC822 address that you can cut and paste from the "From:" line of a message. Be sure that you remove the "From:" part of the line. For example, the "From:" line

```
From:  Joe User <JoeUser@example.com>
```

becomes an **ADD** command as follows:

```
ADD MYLIST-L Joe User <JoeUser@example.com>
```

4.1.1. Adding users whose address and real name exceed 80 characters

This problem happens particularly with the X.400 and X.500 addressing schemes, but can happen as well with any system which allows users to have a very long "local part" (i.e., the part to the left of the "@") in their userid, or with users on systems that just have very long names, such as some of the hosts in the **.US** domain generally have. For instance, you might try to send the following ADD to LISTSERV:

```
QUIET ADD MYLIST someone.with.a.real.long.userid.that.wraps@hishost.com
His Name
```

"His Name" wraps to the next line. If you send this to LISTSERV, LISTSERV treats the

two lines as separate commands even though you did not hit RETURN after the user's address, and it responds:

```
> QUIET ADD MYLIST someone.with.a.real.long.userid.that.wraps@hishost.com
someone.with.a.real.long.userid.that.wraps@HISHOST.COM  is not  yet in  the
signup file. Please specify the full name of that person, as in "ADD MYLIST
JOE@XYZ.EDU Joe H. Smith".

> His Name
Unknown command - "HIS". Try HELP.
```

To avoid this problem, set up your ADD command with a "continuation card" as follows:

```
// QUIET ADD MYLIST someone.with.a.real.long.userid.that.wraps@hishost.com ,
His Name
```

4.1.2. X.400 and X.500 addressing--Special Problems

X.400 and X.500 addressing schemes can cause problems for the list owner who is trying to add or delete one. These addressing schemes use the "/" character to separate address elements, but to LISTSERV, "/" is a special character and you would not be able to add or delete one of these addresses by simply cutting and pasting it into an **ADD** or **DELETE** command.

For instance, you might have an address like:

```
/G=Joe/S=Randomuser/OU=403402ABD/O=SOME.CORP/@LANGATE.SOME.HOST.COM
```

In order to either add or delete this address, there are two issues:

1. The address may wrap to the next line once you add the **DELETE listname** command, and LISTSERV will not accept it.
2. The address contains characters that LISTSERV will reject as illegal (the "/" character).

For adds, to get around both of these issues, you must use a LISTSERV JOB syntax as follows:

```
ADD MYLIST-L DD=X500 PW=MYPASSWORD
//X500 DD *
/G=Joe/S=Randomuser/OU=403402ABD/O=SOME.CORP/@LANGATE.SOME.HOST.COM
/*
```

Any other method will trigger an RFC822 parser error because of the "/" characters in the address. (Note that a user who is *subscribing* from an address like this will have no trouble; it is only when the list owner uses the **ADD** command that this difficulty surfaces.)

For deletions, to get around both of these issues, the wildcard character ("*") can be used. You may not need the entire address in order to delete it, so you might just use

```
        DELETE MYLIST *G=JOE*S=RANDOMUSER*@LANGATE.SOME.HOST.COM
```

which solves both the line wrap problem and the illegal character problem at the same time.

You can also use double-quotes around the address if it contains illegal characters, and a "continuation card" (see 4.1.3) if the address is too long to fit on one line:

```
// DELETE MYLIST ,
"/G=Joe/S=Randomuser/OU=403402ABD/O=SOME.CORP/@LANGATE.SOME.HOST.COM"
```

4.1.3. Continuation card syntax

The basic syntax of a continuation card is

```
//<space><beginning of command><space><comma>
<continuation of command><space><comma>
<continuation of command>....
```

for example,

```
// QUIET ADD MYLIST someone.with.a.real.long.userid.that.wraps@hishost.com ,
His Name PW=mypassword
```

or, for instance, for a large **GETPOST** job,

```
// GETPOST MYLIST 10769-10770 10772 11079 11086 11095 11099-11100 11104 ,
11111 11115 11118 11121 11124 11131 11144 11147 11153 11158 11166 11168
```

Without going into a lot of detail, the "**//<space>**" at the beginning of the command causes LISTSERV to look for a comma at the end of the first line and, if if finds the comma, to add anything following the comma on the second line to the end of the first line. Be sure to put a space before the comma at the end of the first line, as LISTSERV will not add the space for you.

For more information, see the chapter on LISTSERV's Command Jobs Language Interface (CJLI) in the *Developer's Guide to LISTSERV.*

4.2. Finding users who do not appear in the list

Sometimes the list owner will get a message from a subscriber who says, in essence, "I keep trying to (unsubscribe/change to digest/etc.) and LISTSERV says I'm not subscribed. Can you help?" This requires some detective work.

There are a couple of strategies for figuring out what is wrong. List owners should first use the powerful **SCAN** command to search for a pattern anywhere in the subscriber list. The syntax is:

> **SCAN** *listname search-text*

For instance, "**SCAN TEST-L Nathan**" might return:

```
> scan test-l Nathan
Nathan Brindle <nbrindle@INDYCMS.IUPUI.EDU>
Somebody Else <nathan@BAZ.NET>
Jonathan Smith <jsmith@FOO.BAR.COM>
SCAN: 3 matches.
```

Note that **SCAN** is not case-sensitive. "Nathan", "NATHAN", and "nathan" all return the same results.

Searches with **SCAN** should start out simple and become more complex as needed. For instance, if there are only three people in the list with the string "NATHAN" as part of their subscription record, it will be unlikely that you will need to make the search any more

complex. If you are looking for "SMITH", however, it may be necessary to further qualify your search string, say to look for "JOE SMITH". Another reason it is important to begin with a simple search string is that your user may not be subscribed under the exact address the error is returning to you. For instance, say you don't have the user's id, but you have a host name. You can search for all occurrences of the host name, but note that the search:

```
SCAN TEST-L MAIL.FOO.BAR.COM
```

will *not* find the user jsmith@foo.bar.com. If you run the following search:

```
SCAN TEST-L BAR.COM
```

however, you will find Mr. Smith's subscription.

Another possibility is that the subscriber may be using more than one address to work with his subscription. For instance, say the user's complaint to you came from JOE@SUN6.SOMEUNI.EDU. Looking at the list, you find a subscription for JOE@SUN8.SOMEUNI.EDU. LISTSERV has no way to know that JOE@SUN6 is the same person as JOE@SUN8, even though Joe and you know they are. The solution to Joe's problem above is for you to delete his SUN8 subscription and add his SUN6 address. Then Joe needs to be sure that he uses SUN6 in the future, if not for reading mail, then at least for managing his own subscription.

Another strategy would be to submit a wildcard `QUERY` to the list. The drawback to this method is that it might require multiple tries to find the subscription, depending on the complexity of the wildcard query.

Note also that not only can this sort of problem arise from a subscriber using more than one workstation to read mail, but it can also arise when a particular site changes its domain configuration, forwards mail from the old addressing scheme to the new addressing scheme, and doesn't inform its users of the change. In these cases, users often don't realize there is a problem until they try to unsubscribe or change personal options, because the change has been transparent to them.

4.3. Converting existing lists from other systems to LISTSERV

4.3.1. Converting mailing lists

Currently there are no supported conversion programs that will take (for instance) a Majordomo or ListProc mailing list and convert it to LISTSERV format. However it should be possible to extract the address list from the non-LISTSERV list and use a bulk add operation (see chapter 4.4) to populate your new LISTSERV list.

4.3.2. Converting message archives

Note also that existing list archive notebooks will probably not be in LISTSERV format (a modified VM MAILBOOK format), but rather, in the standard unix mailbox format. Again there are no supported programs to convert such archives to the LISTSERV format, but the basic format is as follows:

```
Message separator
Body of Message
Message separator
Body of Message
```

The MAILBOOK message separator is a line of 73 "=" characters (ASCII &H3D). Each archive notebook file must start with a message separator as the first line, e.g.:

```
=========================================================================
Date:          Tue, 3 Mar 1998 10:36:55 -0500
Sender:        Test list <TEST@LISTSERV.EXAMPLE.COM>
From:          Nathan Brindle <nathan@example.com>
Subject:       Test
Mime-Version: 1.0
Content-Type: text/plain; charset="us-ascii"

First test message
=========================================================================
Date:          Tue, 3 Mar 1998 10:39:11 -0500
Sender:        Test list <TEST@LISTSERV.EXAMPLE.COM>
From:          Nathan Brindle <nathan@example.com>
Subject:       Test2
Mime-Version: 1.0
Content-Type: text/plain; charset="us-ascii"

Second test message
```

and the notebooks must be named in a standard LISTSERV format, e.g., **TEST.LOG9803A**, so that LISTSERV will see them and include them in the output of the **INDEX** *listname* command.

Note that for unix mailbox-formatted archives you must remove the first line of each message, which begins with "From" followed by a space (this is the standard unix mailbox delimiter). Below is an excerpt from a unix mailbox for illustration purposes:

```
From owner-test@listserv.example.com  Mon Dec 29 15:17:20 1997
Return-Path: <root@listserv.example.com>
Received: from localhost (root@localhost) by listserv.example.com (8.8.3/8.8.3) 0
Date: Mon, 29 Dec 1997 15:17:20 -0500 (EST)
From: root <root@listserv.example.com>
To: test@listserv.example.com
Subject: Test
Message-ID: <Pine.LNX.3.95.971229151703.711A-100000@listserv.example.com>
MIME-Version: 1.0
Content-Type: TEXT/PLAIN; charset=US-ASCII
Status: RO
X-Status:

This is a sample message from Majordomo in unix mailbox format.

Root
From owner-test@listserv.example.com  Mon Dec 29 15:23:51 1997
Return-Path: <root@listserv.example.com>
Received: from localhost (root@localhost) by listserv.example.com (8.8.3/8.8.3) 0
Date: Mon, 29 Dec 1997 15:23:51 -0500 (EST)
....
```

Each of the lines beginning with **From owner-test@listserv.example.com** is the delimiter separating one message from the next.

4.4. Adding subscribers to lists in bulk

If you are moving a list from a non-LISTSERV site, you can quickly and easily convert the existing subscriber list to the LISTSERV format by following these instructions:

1. Have the LISTSERV maintainer at your new site create the new list header and install it on the machine.

2. Create an add job as follows. The `QUIET` and `IMPORT` command words are optional; omit the square brackets if you use them. The "full name" field is optional as long as you use the `IMPORT` option; otherwise you must either specify "`*`" (for an anonymous subscription) or a full name consisting of at least two separate words.

```
[QUIET] ADD listname DD=ddname [IMPORT] PW=yourpassword
//ddname DD *
userid1@host1.com [*|full name]
userid2@host2.com [*|full name]
...more users, one per line...
useridn@hostn.com [*|full name]
/*
```

For example (what fun:),

```
ADD B5-L DD=MYDD IMPORT PW=BLAHBLAH
//MYDD DD *
SHERIDAN@BABYLON5.MIL John Sheridan
IVANOVA@BABYLON5.MIL Susan Ivanova
LONDO@CENTAURI.PRIME.GOV Londo Mollari
GKAR@KAARI.NARN.GOV Citizen G'Kar
/*
```

If you are importing from an existing non-LISTSERV list, you should remove any lines from the original list that do not actually identify subscriber addresses. If you are converting to LISTSERV from ListProc, note that LISTSERV will not convert ListProc user options to their LISTSERV equivalents; you must take a line like

```
user1@somehost.com POSTPONE NEWLIST NO user's name
```

and reduce it at least to

```
user1@somehost.com user's name
```

Otherwise, the ListProc options will become part of the `full name` field.

3. Send the job to LISTSERV.

The `IMPORT` option implies a `QUIET ADD` (in other words you do not need to specify `QUIET` if you use `IMPORT`) and otherwise vastly speeds up the `ADD` process by loosening syntax checking and omitting success messages. If you do not use the `IMPORT` option and do not specify `QUIET`, the users you bulk add will receive the normal SIGNUP message and/or WELCOME file as usual.

4.5. Deleting subscribers from lists in bulk

If you have a large number of users to delete at one time, you can use a bulk delete syntax that is similar to the bulk `ADD` documented above. However please note that there is no "`IMPORT`"-type option for this feature, and as usual for the `DELETE` command you specify only the user's address in the data DD.

There is, however, a `BRIEF` option that can be specified which is good when you don't want a long list of "userid@host has been deleted from list xxxx" messages, one for each user deleted. Use of the `BRIEF` option tells LISTSERV to return only a count of the users that were deleted.

Once again you construct a LISTSERV JOB framework as follows and then send it to LISTSERV:

```
[QUIET] DELete listname DD=ddname [BRIEF] PW=yourpassword
//ddname DD *
userid1@host1.com
userid2@host2.com
...
useridn@hostn.com
/*
```

You will probably want to use the `QUIET` modifier when doing a bulk delete, in order to suppress the notification message to the users being deleted.

4.6. Using the QUIET option with commands

Prepending the command word "`QUIET`" before any LISTSERV command that you issue on behalf of a subscriber causes LISTSERV to suppress any notification to the subscriber of the changes you have made. This is particularly helpful when deleting subscribers whose accounts have expired and when setting subscribers with full mailboxes to NOMAIL, as it will help avoid another error message from the host when the notification message bounces. It is also helpful when adding subscriptions to the list that should not receive any welcome mail, such as redistribution lists and USENET newsgroups.

Examples of the usage of `QUIET` include:

```
QUIET ADD EXCEL-L comp.spreadsheets.excel@netnews.somenode.edu
QUIET DELETE EXCEL-L Bouncemeister@somenode.edu
```

4.7. Dealing with bounced mail

4.7.1. What is a bounce, and what can typically cause one?

A bounce is simply an undeliverable e-mail message. The term "bounce" is used to describe it because normally the system that discovers the delivery error "bounces" a copy of the message back to you with some sort of delivery error message. Sometimes these messages are easy to decipher – "No such user at foo.bar.com" – but uncomfortably often they are not that easy. Certain systems, as noted above, kindly format error notifications in a format that LISTSERV can understand, and if your list is configured for auto-deletion, these bounces will be the least of your worries – in fact, they will not be worrisome at all.

4.7.2. The owner-*listname* address

If you receive bounces processed through LISTSERV you will note that they normally say something like the following at the top:

```
The enclosed message has been identified as a delivery error for the MYLIST-L
list because it was sent to 'owner-mylist-l@LISTSERV.EXAMPLE.COM'.

---------------------------- Message in error -------------------------------
```

What this message means is simply that LISTSERV has received mail sent to the owner-*listname* mailbox for your list. Mail sent to this special address is automatically forwarded

by LISTSERV to the address(es) you have defined in the **Errors-To=** list header keyword. The little "error-header" shown above is prepended to the actual error message to let you know that this is an error for your list (rather than unceremoniously dumping it into your mailbox and making you wonder "why did I get this?", since some delivery errors aren't specific about what list or even what user they are for). So whenever you get mail saying it was found in the owner-*listname* mailbox, it means that it is an error that you need to deal with for the list referred to by *listname*.

If you find that you have users trying to contact you (as list owner) at the owner-*listname* address, you should tell them that the correct generic address for contacting the list owner(s) is *listname*-request, not owner-*listname*. Mail sent to the *listname*-request address will be sent to all non-quiet list owners and furthermore will be automatically responded to with the **REQACK1** mail template form from your **listname.MAILTPL** file (or its default from **DEFAULT.MAILTPL**; see chapter 9) while mail to owner-*listname* will not be responded to at all unless you do so explicitly. The nice thing about having people use the *listname*-request address is that you can store your list's FAQ (if you have one) in the **REQACK1** mail template form and probably not have to answer all of the questions you get as list owner--like "how do I subscribe?" and "how do I sign off?".

4.7.3. What to do about several types of bounces

Please note carefully that it is not the intent of (nor would it be reasonably possible for) this manual to document each and every kind of delivery error that you may ever see as a list owner. Unfortunately, and completely outside the control of anyone at L-Soft, new types of cryptic, difficult to understand, and totally misleading errors appear all the time. If you run across something not otherwise documented here, the best place to ask for help is the LSTOWN-L mailing list (see chapter 10.6).

That being said, here are a few of the typical mail errors you will have to deal with as a list owner. Newer, so-called "Notary" format error codes are documented in RFC1893, which can be found at the WWW URL

http://nis.nsf.net/internet/documents/rfc/rfc1893.txt

1. no such user at *host* , user unknown (or "notary" format error number 5.1.1)

   Most of the time, this is authoritative and indicates that the user's access has been curtailed for some reason (graduation, no longer employed, etc.). A **quiet delete** (syntax: "**QUIET DELETE listname userid@host**") is in order unless you have reason to believe that the message is *not* authoritative. Variations on this message include "Recipient unknown" and "Ambiguous address: *userid*". The latter doesn't really mean the user doesn't exist, but it's almost as bad, and many list owners choose to classify it as "no such user".

   Microsoft Exchange servers send back the following message for an unknown user:

   ```
   Joe@EXCHANGE.EXAMPLE.COM on Wed, 4 Mar 1998 13:31:50 -0600
       The recipient name is not recognized
       MSEXCH:IMS:Example Corp:EXAMPLE:EXCOMEXCH 0 (000C05A6)
   Unknown Recipient
   ```

2. no such host, host unknown (or "notary" format error number 5.1.2)

   This is sometimes authoritative and sometimes not. If a host goes down or a gateway fails, often this message is returned by an intermediate host or gateway. If the user is bouncing a great deal of mail from a high-volume list, it is probably best to

set the user to **NOMAIL** (syntax: "**SET *listname* NOMAIL FOR *userid@host*"**) rather than to summarily delete him. This way, the error messages stop, the user is sent an automatic message telling him his personal options have been changed by the list owner, and the user doesn't have to go through the subscription process again if the problem has been solved in the interim.

The problem is that some hosts go down on a regular basis and this error makes it impossible to tell if the host in question is gone forever or gone until the local sysadmin reboots his machine. After a while, you will begin to recognize the transient hosts and may elect to ignore them. If you choose to set the user to NOMAIL, you should send a message to the user just in case the system has come back up, and you should keep some sort of record of the users you've set this way so you can follow up later with another message.

3. no MX or A records for *host* (or "notary" format error 5.4.4)

Similar to "no such host". This means that the Domain Name Service (DNS) can't find any routing information for *host* but has found at least one reference to it. This generally indicates a DNS configuration error and may or may not be transient.

4. Transient failure: cannot deliver for *n* days

A host is experiencing periodic failures, and the gateway or intermediate host has not been able to deliver the message for *n* days. Usually the host will attempt redelivery. Usually there is nothing wrong with the user address, so it is a list owner decision as to whether it is worth waiting out the transient failure or going ahead and setting the user to **NOMAIL**. Unfortunately, by the time you get this message, the failure is *n* days in the past, the "transient failure" is very probably over, and you are likely to receive further error messages for *n* more days until the intermediate host's queue is exhausted.

5. mailbox full, quota exceeded (or "notary" format error number 4.2.2)

Self explanatory. This usually happens on systems with tiny user mailbox space, but it can happen on any system if a user subscribes to too many lists or goes on an extended vacation without setting lists to **NOMAIL**. The best solution is to set the user to **NOMAIL** yourself. Variations on this message include VMS's "file extend failed writing to [disk.user]MAIL.MAI".

6. unknown mailer error x

This is a favorite Unix sendmail configuration bounce. **NOMAIL** or **DELETE**, according to your preference. Since it is a configuration problem, it is *usually* transient. One system sent the following under an "unknown mailer error 1" heading:

```
binmail: /usr/spool/mail/userid: too big to accept new messages.
       It's size is 205735 bytes which is 935 bytes over quota.
mail: cannot open dead.letter
554 <userid@node>... unknown mailer error 1
```

This is apparently a "mailbox full" error, as "userid's" mail spool is "over quota". It is also possible that it means your message would put the user over quota by 935 bytes. Either way, there isn't enough space in the user's mailbox to store your message (in this case, it was a daily digest). Note that "unknown mailer error x" does not always mean the user's mailbox is full – what it always means is that sendmail

cannot identify the cause of the error.

7. Bounced, but sent successfully

This error comes from cc:Mail systems and is extremely misleading. It claims that the mail bounced to one address, but was sent successfully to another.

```
While talking to smtp.ccabc.com:
>> DATA
<< 554 I/O error to mailbox
554 MILLERT@smtp.ccabc.com... Service unavailable

----- Recipients of this delivery -----
Bounced, cannot deliver:
   MILLERT@smtp.ccabc.com
Sent successfully:
   <MILLERT@ABC.COM>
```

What this means (assuming that the mail hasn't gone through a redistribution list or a mirror site) is that you have a user MILLERT@ABC.COM on your list, and the server accepted the mail for that address successfully. However, that address actually maps to a different internal address (in this case MILLERT@smtp.ccabc.com) and for whatever reason, the server can't forward the mail on. This is the equivalent of a "user unknown" error for MILLERT@ABC.COM.

8. Too many hops (or "notary" format error number 5.4.6)

Means that the message has transited through too many intermediate mail systems (1 transit = 1 hop). Most of the time this will be due to a temporary looping condition on the user's end (despite the "permanent" 5.4.6 error). For instance, the following Internet routing headers indicate a loop between three different mail machines (starting from the bottom and working back to the top):

```
Received: from un1.sample.com (root@un1.sample.com [200.9.212.3])
        by un7.sample.com (8.8.7/8.8.7) with ESMTP id RAA22765
        for <user@example.com.ar>; Wed, 4 Mar 1998 17:17:10 -0300
Received: from ul1.sample.com (root@ul1.sample.com [200.0.224.2])
        by un1.sample.com (8.8.7/8.8.7) with ESMTP id RAA27352
        for <user@example.com.ar>; Wed, 4 Mar 1998 17:16:00 -0300
Received: from un7.sample.com (un7.sample.com [200.9.212.4])
        by ul1.sample.com (8.8.8/8.8.8) with ESMTP id RAA13496
        for <user@example.com.ar>; Wed, 4 Mar 1998 17:15:40 -0300 (GMT-3)
Received: from un1.sample.com (root@un1.sample.com [200.9.212.3])
        by un7.sample.com (8.8.7/8.8.7) with ESMTP id RAA22034
        for <user@example.com.ar>; Wed, 4 Mar 1998 17:15:27 -0300
Received: from grape.EASE.LSOFT.COM (grape.ease.lsoft.com [206.241.12.34])
        by un1.sample.com (8.8.7/8.8.7) with ESMTP id RAA25235
        for <user@example.com.ar>; Wed, 4 Mar 1998 17:08:43 -0300
```

The problem here appears to be that the mailers at `sample.com` are MX (mail exchanger) sites for `example.com.ar`, but that they can't decide which one of them should hold onto the mail until it can be delivered to `example.com.ar`. So it looped through 7 iterations until the `un7` machine finally decided that enough was enough (including the passage through LISTSERV it had taken 26 hops and `un7` was set to accept a maximum of 25 hops) and generated an error.

You may occasionally see a "too many hops" message that isn't a loop. Usually the non-looping variant is due to the recipient being many hops away from the mail originator and the maximum hop count being set too low on the recipient's machine. Many older sendmail installations, for instance, will accept only 10-15 hops before they reject the message. With today's Internet a setting of 30-40 is probably much

more reasonable.

A particularly annoying error you may have to deal with comes from Banyan networks and is of the form:

```
LLONG@StarShip@Dora:   Mailbox full
```

Obviously this is not a properly-configured address (at least, not as far as LISTSERV is concerned), and if you SCAN or QUERY the list for it, you will get a negative response. If, however, you SCAN the list for LLONG, you may find a user such as:

```
> scan test-l LLONG
Bill Smith <LLONG%StarShip%Dora@BOONDOCK.TERTIUS.COM>
SCAN: 1 match.
```

This user can now be set to **NOMAIL** and the errors will stop after the Banyan host has emptied its queue. If you do not find the user on the first **SCAN**, try using another part of the address as your search text. Note that a user may have his mail forwarded from the account that is actually subscribed to an account on another machine where he reads his mail. If the second machine is bouncing the mail, it may not be immediately apparent from the bounce messages that the mail is actually being forwarded. It is important to check for variants of the userid in the bounce message as it may be related to the userid that is actually subscribed to the list.

Note that there are many forms of error messages. Many mail systems do not conform to Internet "standards" (some of them even return non-English error messages!) and LISTSERV's auto-deletion feature will not always catch their bounces.

4.7.4. Redistribution and forwarding

Perhaps the worst type of bounce is one that comes from a user who is "hiding" behind an account that redistributes mail (a "redistribution list"), or a user whose Internet address has changed slightly but who is still subscribed to your list under his original address.

Redistribution lists typically (but not always) take some form of your list's name (such as "xxxxx-L-REDIST@foo.bar.com"), and thus their subscriptions tend to be easy to find. What is difficult is that you have no way of knowing which users (or how many users) are hidden behind this interface, nor any way of knowing what their userids are.

Forwarded accounts generally fall into one of two categories – those where the user has forwarded his own mail from one account to another rather than changing his subscription, and those where the user's system name has changed and the old address is still valid but is forwarding mail to the new address without the user being aware of it.

Let's say that suddenly you are bombarded with delivery errors for someuser@baz.net. Your immediate reaction is to set this person to **NOMAIL** or (in some cases) to delete him/her altogether.   You   therefore   send **set   xxxxx-L   nomail   for someuser@baz.net** to LISTSERV.   LISTSERV responds: "No subscription for someuser@baz.net in list XXXXX-L."

In a best-case scenario, you can query the list for *@*.baz.net and find either a user like someuser@glork.baz.net (the address has changed and the local sysadmins didn't inform the user) or a redistribution-list account like xxxxx-L@baz.net. These are easily-fixed redistribution bounces. In the first case, you delete the user and let him or her

resubscribe. In the second case, you can try sending a message to owner-xxxxx-l@baz.net with a cc: to postmaster@baz.net and inform them of the problem. If it persists, you could send a further message informing them that you are suspending the redistribution list's subscription until such time as they tell you the problem on their end is fixed, and simply set xxxxx-l@baz.net to NOMAIL.

The worst-case scenario is as follows: baz.net may be bouncing the mail to you, but there may not be a single subscription for baz.net in your list. Here's where you have to do some careful sleuthing. First, run a wildcard query such as `QUERY xxxxx-l FOR *@*baz*` or `QUERY xxxxx-l FOR *baz*@*`. The former will find users at baz.com, for instance, where baz.net is a synonym for baz.com. The latter query may seem somewhat strange, but it's possible that the mail is being routed through a gateway and the actual subscription is for xxxxx-l%baz.net@cunyvm.cuny.edu or something of that sort.

4.7.5. "Sender:", "From:" or "Reply-To:" field in body causes bounce

Sometimes you will receive bounces from LISTSERV with a error header like this:

```
The enclosed message, found in the VISBAS-L mailbox and shown under the spool
ID 19630445 in the system log, has been identified as a possible delivery error
notice for the following reason: "Sender:", "From:" or "Reply-To:" field
pointing to the list has been found in mail body.
```

Sometimes this is a legitmate bounce from a mail system that isn't compliant with Internet standards for mail, and the reason the "Sender:", "From:", and/or "Reply-To:" headers are significant is because if this mail were to be allowed through to the list it could very possibly start a loop with the non-compliant mail server. Normally this is a good thing; however, an unfortunate side-effect of the loop-checking code that catches this kind of bounce means that LISTSERV may treat replies to list mail from some mail clients as if they are delivery errors. LISTSERV has no way to know the difference between a bounce and a legitimate message that just happens to have unquoted included headers so it takes the conservative route and bounces it to the list owner as a "possible" delivery error. This way the list owner can (if he or she wants to) return the message to the user in question and ask them to either quote out or delete the headers from their replies.

In any case this is specifically known to be a problem with Pegasus Mail and some incarnations of the Microsoft Exchange Client, but there are probably other mail programs that do the same thing. The problem arises when the user's mail client includes the "Sender:", "From:", or "Reply-To:" fields that point back to the list itself (for instance, the above error was for VISBAS-L@PEACH.EASE.LSOFT.COM) in the quoted material and doesn't quote them correctly--that is to say, without a quoting character, or with a space between the quoting character and the included text. For instance, a reply from Pegasus with quoted material would include the following lines:

```
User's reply, blah blah blah

> Date:          Tue, 31 Dec 1996 17:00:00 -0700
> Reply-to:      Visual Basic List <VISBAS-L@PEACH.EASE.LSOFT.COM>
> From:          Joe User <JOE@UNIX.FOO.COM>
> Subject:       Re: 97 Style ToolBars
> To:            VISBAS-L@PEACH.EASE.LSOFT.COM
```

The quoted lines below the user's reply would trigger LISTSERV's loop detection functions because there is a space between the ">" character and the "Reply-To:" and the "From:" headers.

The correct, netiquette-approved method of quoting these headers is to delete them entirely from the body of your message. Quoting is generally done for reasons of context and message headers are not needed for context. (Pegasus actually lets you toggle this on and off via the "Advanced options for replies" dialog. Other clients don't seem to have this function.) Note that Eudora quotes messages with no space between the ">" character and the quoted text, so this is not an issue with Eudora.

If necessary, subscribers using Pegasus can change the quoting character (at least they can in the current version of Pegasus) by editing their copy of **PMAIL.INI** and changing the value in

[General]
...
Commenting string   = >

Normally this variable contains "> ", that is, ">" followed by a space character. If you remove the space, Pegasus quotes "properly" and this is no longer a problem. Other mail clients may or may not have similar configuration settings.

(See also 10.2, below).

4.7.6. LMail error codes

LMail is an L-Soft mailer product for VM mainframes. When it receives error mail from remote hosts it translates the error into a standard format recognized by LISTSERV so that LISTSERV can take action as necessary. From time to time you may see such errors in your mailbox; they look like this:

```
------------------------------------------------------------------------
Date:     Fri, 8 Jan 2000 20:04:50 +0100
Reply-To: Postmaster@SEARN.SUNET.SE
From:     RFC822 mailer (LMail release 1.2d/1.8d) <MAILER@SEARN.EXAMPLE.SE>
Subject:  Undelivered mail
To:       ERIC@SEARN.EXAMPLE.SE
cc:       Postmaster@SEARN.SUNET.SE
X-Report-Type: Nondelivery; boundary="> Error description:"

An error was detected while processing the enclosed message. A list of
the affected recipients follows. This list is in a special format that
allows software like LISTSERV to automatically take action on incorrect
addresses; you can safely ignore the numeric codes.

--> Error description:
Error-For:  JACK@SEARN.SUNET.SE
Alias:      JACK@SEARN.BITNET
Error-Code: 3
Error-Text: No such local user.

Error-For:  JOE@SEARN.SUNET.SE
Alias:      JOE@SEARN.BITNET
Error-Code: 3
Error-Text: No such local user.

Error-End:  Two errors reported.

------------------------ Rejected message (5 lines) --------------------------
Date:        Fri, 8 Jan 1993 20:04:47 +0100
From:        Eric Thomas <ERIC@SEARN.BITNET>
To:          JACK@SEARN.BITNET, JOE@SEARN.BITNET

Testing.
```

The LMail codes stand for the following:

0 is used for all the weird errors that can't easily be classified, and LISTSERV takes no action on these codes by definition. This includes errors such as "invalid device name" or "device full" which have meaning only for the postmaster of the host that bounced the message.

1 means "don't know that/don't know how to get there" (as opposed to "can't get there right now"), which is used when the host can't be found (eg, "host unknown").

2 means "configuration error". This means that LMail has detected an error in its routing tables which prevents it from delivering the mail. It does not necessarily mean that the address is bad.

3 is "no such local user".

4 is "not allowed to mail to this user".

The difference between 3 and 4 is that a 3 indicates there is no way to successfully send mail to the user, whereas 4 indicates the user cannot receive mail from the address your message came from. LMail uses code 4 when a local LMail user directs it to reject all mail coming from mailing lists, but to let private mail through. Typically you will not see very many 4 codes.

5 means "mailbox full", "quota exceeded", and so on.

LISTSERV itself takes action on error codes 1, 3 and 4, and forwards anything else to the list owner.

4.8. Delivery error handling features

LISTSERV supports several levels of automatic deletion based on error messages passed back to it in LMail format by certain remote systems. While auto-delete will not solve all of your bouncing mail problems, it has the potential to take care of most "permanent" errors (including "no such user" and "no such host"). However, note that auto-delete ignores "temporary" errors such as "host unreachable for 3 days", "system error", "disk quota exceeded", and so forth, such that users whose accounts generate "temporary" errors are not summarily deleted from the list.

By default, lists running under LISTSERV Classic 1.8b and higher generate a report which lets the list owner know what userids are causing problems, rather than deleting users at the first error LISTSERV understands. If the Delay() and Max() parameters are set to non-zero values for a list coded "Auto-Delete= Yes", LISTSERV will not take immediate action on mail delivery errors. You will receive an "auto-deletion monitoring report" daily to show you which subscribers are bouncing mail, what the error is, when it started, when the last error arrived, and how many errors have been received for the subscriber in total. By default, LISTSERV will wait 4 days (or for a maximum of 100 error messages per individual user) before deleting a subscriber.

If you code "Delay(0)", LISTSERV will not wait to take action, but will delete the subscriber at the first error LISTSERV understands. *Note carefully* that LISTSERV will not generate a daily error monitoring report when Delay(0) is used.

The default value is "Auto-Delete= Yes,Semi-Auto,Delay(4),Max(100)" for lists coded "Validate= No" and "Auto-Delete= No" for all other lists.

Under LISTSERV Lite, Auto-Delete= is available but deletes on the first bounce (e.g.,

"Delay(0),Max(1)") regardless of the Delay() and Max() settings.

Implementation of the "Auto-Delete=" keyword is discussed in detail in Appendix B, *List Keyword Alphabetical Reference*, under "Error Handling Keywords."

4.8.1. Auto-Delete considerations for holidays

Making a big increase to the DELAY threshold to provide more leniency during a holiday may not be a good idea. While it will indeed disable the monitor for the duration of the holiday, switching back to the normal threshold when you return will cause the monitor to delete all the users that had been bouncing during the holidays. In general, you should avoid making temporary changes to the DELAY threshold, because it takes the monitor a while to adapt to the new settings.

The best way to relax the rules during a long holiday is to leave the DELAY threshold unchanged but switch the monitor to passive mode ("Auto-Delete= Yes,Manual"). Noone will be deleted over the holidays, but the monitor's cycle will not be perturbed. When you return, you should wait about a week before switching back to automatic mode. This is because, after a long holiday such as Christmas, it usually takes about 2 working days for system administrators to solve all problems. In some cases, the problems will have caused bounces to remain undelivered. So, by fixing the problems, the system administrators may actually send a flood of new bounces corresponding to problems that have now been solved. Unfortunately, since the monitor only receives NON-delivery reports, it has no way to know that these problems have in fact been solved. As a rule of thumb, you will note that your daily delivery error reports are much longer than usual over the vacation. When you return, you should wait until they are back to their normal size before switching back to automatic mode.

4.9. Address probing

This functionality is not available in LISTSERV Lite.

There are two levels of automatic address probing available in LISTSERV.

4.9.1. Active address probing
This functionality is not available in LISTSERV Lite.

Active address probing was introduced in LISTSERV 1.8c, for two reasons: first, to enhance subscription renewal functionality so that no "`CONFIRM listname`" response was required from subscribers in order to stay subscribed, and second, to enhance the ability of the auto-deletion feature to handle bounces that can't be parsed into something LISTSERV can recognize.

"`Renewal= ...,Probe`" activates this enhanced bounce processing feature, whereby subscribers are probed at subscription renewal time using the `PROBE1` mail template. The "Probe" option makes subscription renewal passive rather than reactive; no "`CONFIRM listname`" response is needed from the user. In fact, the desired response from the user is to discard the message and do nothing, making the process very simple. LISTSERV also probes addresses that return mail delivery errors, and probe messages have a special signature in the return address that allows LISTSERV to uniquely identify any bouncing address, without having to understand the bounce itself.

If the probe bounces, LISTSERV first sends the `PROBE2` template with a copy of the bounce, to show the user (if the account actually works in spite of the bounce) what garbage his mail system is sending people. LISTSERV then schedules a new probe for

the next day, or deletes the user immediately, depending on the auto-delete policy. Every failure triggers a new daily probe until the user gets deleted or the problem gets fixed. The user can also save his subscription manually by sending a `CONFIRM` *`listname`* command (this is explained in `PROBE2`). This doesn't solve the underlying problem, so eventually the user should get tired of confirming in an emergency and notify his system administrators that the system is generating bounces saying (for instance) "Your message was registered at the MORONICUS mail gateway. Press F1 for more information" that cause the problem in the first place.

When used together with "`Auto-Delete= ...,Full-Auto`", the probe option deletes all delivery errors from bounced probes, even if LISTSERV can't understand the error. This means the list owner never ever has to see a single bounce from a probed address! Hurray! :-) The list, however, *is* kept clean because bad addresses are *always* detected. In fact, the biggest risk is that the users of the MORONICUS mail gateway will be deleted even though they do get their mail.

This being said, *note carefully* that all errors bounced by non-compliant mail hosts to the wrong address, and non-probe errors that are sent to the owner-listname address but are not in a format that LISTSERV can parse, will still show up in your error mailbox. If the bounce goes to the wrong address, LISTSERV never sees it and cannot probe it. If the error goes to the correct address (owner-listname) but isn't specific enough for LISTSERV to understand, while LISTSERV *will* be able to see it, it still won't be able to probe it. Finally, in some cases where the error is so vague (or constructed in a complicated manner that defies LISTSERV's attempts to parse it) the error may be passed to the LISTSERV postmaster, instead of to the list owner, for disposition, even if it was correctly returned to the owner-listname address.

Yet even with these restrictions, the author saw an error queue of 1300 errors/day shrink to under 50 errors/day by applying the ",Probe" parameter to seven high-volume lists, which in his opinion was *much* more acceptable.

4.9.2. Passive address probing
This functionality is not available in LISTSERV Lite.

Passive address probing is available beginning with LISTSERV 1.8d. In effect passive probing is very similar to active probing, but it is not tied to subscription renewal. Passive probing is enabled by default for small lists (e.g., <1K subscribers) but not for large ones due to the fact that passive probing does cost additional resources and large lists are often used for one-shot mailings where it is simply not effective to use those resources to probe addresses that will not be used a second time.

Passive probing operates by turning a certain percentage of your regular list messages into transparent probes that look like a normal message but also double as a probe, rather than sending out the explicit `PROBE1` template as in active probing. You enable (or tune) passive probing by adding a "`,Probe(xx)`" parameter to the **`Auto-Delete=`** keyword setting. For instance,

`Auto-Delete= Yes,Full-Auto,Probe(30)`

where "30" is the number of days to wait between probes for any given user (the default is `Probe(30)`. Subscribers with working mail systems will not see any difference, subscribers with flaky mail systems will occasionally receive a message showing that their mail bounced and saying that they should report the problem to their ISP, and of course plain bad addresses will go away.

In order to disable passive probing you set the probe parameter to 0, i.e.,

```
Auto-Delete= Yes,Full-Auto,Probe(0)
```

If you have users who for whatever reason should not be probed, you can deactivate passive probing (and any other renewal you have set for the list) with the **SET userid@host NORENEW** command.

4.10. Subscription confirmation

For lists coded "Subscription= Open", you can require confirmation on all new subscription requests, thus ensuring that LISTSERV has a clear mailing path back to the subscriber. In the past, a user could send a subscription for an open subscription list to LISTSERV, which upon acceptance would immediately start sending the user list mail. If the user was located behind a "broken" or one-way gateway, this produced immediate bounced mail until the list owner noticed and deleted the subscription. Note that requiring confirmation at the time of subscription does not guarantee that the clear mailing path will continue to exist permanently.

"Subscription= Open,Confirm" causes LISTSERV to send a Command Confirmation Request to the potential subscriber before actually adding the user to the list. The subscriber is requested to reply to the request by sending a validation "cookie" back to LISTSERV (this "cookie" being the hexidecimal number pulled from the subject line).

The Command Confirmation Request, while straightforward, has the potential to cause confusion if users do not read carefully the instructions that make up the request. LISTSERV expects confirmation codes to be sent in a specific way because some mail gateways add lines to the header of the message that LISTSERV doesn't understand. If a user forwards the request back to LISTSERV, or creates a new mail message to send the 'cookie' back, it usually will not work correctly. The sequence should thus be as follows:

1. SEND the subscription request to LISTSERV.
2. REPLY to the confirmation request ('ok')
3. SEND the confirmation code (if necessary) ('ok 23CBD8', for example)
4. Send mail to the list owner (not the list) if the subscription request fails after step 3.

Note that if a list owner adds a user manually, the confirmation process is bypassed.

4.11. Subscription renewal

You can code subscription renewal into your lists. This is one method to keep lists "pruned down" and avoid having large lists that are actually distributing mail to only a fraction of the users. For instance, you may have a number of subscriptions set to NOMAIL for one reason or another. NOMAIL user(a) may have forgotten that he has a subscription; user(b) may have set NOMAIL instead of unsubscribing; user(c) may no longer exist because she graduated or no longer works for the service provider; you may have set user(d) to NOMAIL because of recurrent mail delivery errors. Requiring a periodic confirmation of subscriptions is therefore a reasonable course of action for large, non-private lists.

Subscription renewal is disabled by default. If you do not want subscription renewal, or if you wish to turn it off, simply do not include a "Renewal=" keyword in your list header.

To add subscription renewal, you add the following keyword to the header of your list:

```
    * Renewal= interval
or
    * Renewal= interval,Delay(number)
or
    * Renewal= interval,Delay(number),Probe
```

where *interval* is a period of time such as Weekly, Yearly, 6-monthly, or something similar, and `Delay(number)` is an integer corresponding to how many days LISTSERV will wait for the renewal confirmation to arrive. (See "Renewal=" in Appendix B for more information on renewal and delay periods; see chapter 4.8.2., above, for more information on the "Probe" parameter.) Note that you can have multiple *interval* parameters; again, see the entry for "Renewal=" in Appendix B for details.

The confirmation request mailing asks the subscriber to send the command `CONFIRM` *listname* back to LISTSERV. If the subscriber does not do so within a certain length of time, LISTSERV automatically deletes the subscription. The default delay time is 7 days. If you wish to use the default delay time, it is not necessary to code ",Delay(7)" into your Renewal parameters.

Note: You may wish to increase the delay time to accommodate users whose subscriptions expire over holidays (such as the Christmas/New Year's week) in order to avoid accidental deletions. Also, be aware that confused subscribers can and will send the `CONFIRM` command back to the list, rather than to LISTSERV. LISTSERV's default filter will catch these commands and forward them to the userid(s) defined by the "Errors-To=" keyword.

It is possible to waive subscription renewal for certain users (such as list owners, editors, redistribution lists, etc.). In order to do this, simply issue the command

> **[QUIET] SET** *listname* **NORENEW FOR** *net-address*

to LISTSERV. *It is most advisable to do this in the case of redistribution lists, as they broadcast the renewal notice to their users, who a) cannot renew the subscription and b) become very confused when they see the notice, often sending "what does this mean?" mail to the list.*

You can also issue the `CONFIRM` command for a subscriber:

> **[QUIET] CONFIRM** *listname* **FOR** *net-address*

Note that "active" users of the list (that is, people who post regularly to the list) will never be required to renew their subscriptions, nor (if subscription "probing" is enabled) will they ever be sent the passive subscription probe. LISTSERV presumes that such users have valid addresses and does not require a renewal confirmation from them.

4.12. Using the SERVE command when a user is "served out"

If a user sends more than 50 consecutive invalid commands to LISTSERV, LISTSERV automatically serves that user off so that further commands from that user will be ignored. Should a user become served off in this fashion, it is possible for the list owner or *any* other user to issue a `SERVE` *net-address* command to restore that user's access. As with all other LISTSERV commands, the `SERVE` command is sent to LISTSERV.

Please note that the number of invalid commands allowed before the user is served off was increased from 20 in 1.8b to 50 in 1.8c and later.

While served off, the user will be unable to set personal options and will be unable to subscribe or unsubscribe to lists on that server. Note that a user will likely be served off of one particular LISTSERV site but not others, and also that the user may not even realize that he has been served off (in spite of the fact that LISTSERV sends notification to the user to that effect).

Note that the `SERVE` command will not restore service to users who have been manually served off by the LISTSERV maintainer.

5. Setting Subscription Options For Subscribers

5.1. How to review current subscription options with QUERY

The syntax is similar to the subscriber's method of reviewing his options, except that the list owner must specify for whom the options are being checked.

```
Query listname FOR userid@host
```

Note that it is possible to use wildcards in the subscriber address. For instance,

```
Q LSTOWN-L FOR J*@UBVM*
```

will return option listings for subscribers such as JIMJ@UBVM, JOHN@UBVMS.CC.BUFFALO.EDU, etc. This can be handy if you are searching the list for someone whose subscription address differs from the address you are given in an error report (see the examples, above, in "Dealing with bounced mail").

Using the **WITH** qualifier, you can also query a list for users who have a specific option set. For instance, you might want to know which users are set to **NOMAIL**. Send the command

```
Q listname WITH NOMAIL FOR *@*
```

and LISTSERV will return a list of those users. It is also possible to query a list for multiple options:

```
Q listname with DIGEST CONCEAL FOR *@*
```

will return a list of those subscribers who have set their subscription to **DIGEST** and also to **CONCEAL**.

Version 1.8c adds the ability to query users by the list topics they are subscribed to. For instance:

```
Q listname WITH TOPICS: ADMIN FORUM FOR *@*
```

shows all members subscribed to both the ADMIN and FORUM topics.

```
Q listname WITH TOPICS: -ADMIN FOR *@*
```

shows all members who are not subscribed to the ADMIN topic.

```
Q listname WITH TOPICS: ADMIN -TEST FOR *@*
```

shows all members who are subscribed to the ADMIN topic but not to the TEST topic.

5.2. How to set personal subscription options for subscribers

Again, the syntax is similar to the subscriber's method.

```
SET listname option FOR userid@host
                    or
QUIET SET listname option FOR userid@host
```

5.3. Options that may be set

5.3.1. Mail/NOMail
Setting this option to `Mail` indicates that the subscriber will receive mail from the list. `NOMail` is the complementary command that stops mail but leaves the user subscribed to the list. (`NOMail` is often a good compromise for users who are leaving the office for vacation or on extended business trips, and who don't want a full mailbox on their return.) The format of the messages received is controlled by the `DIGEST/INDEX/NODIGEST/NOINDEX` options (see below).

5.3.2. DIGest/NODIGest
Causes the subscriber to receive one posting per digest cycle (typically daily) rather than individual messages as they are processed by LISTSERV.

In version 1.8b and later, the `MAIL/NOMAIL` option was isolated from `DIGEST/INDEX`. The `MAIL/NOMAIL` option controls whether messages should be delivered, and the `DIGEST/INDEX/NODIGEST/NOINDEX` option controls the *format* in which messages should be delivered. Thus, switching to `NOMAIL` and back to `MAIL` now preserves the digest/index/normal delivery setting. To provide as much compatibility with the old syntax as possible, the four options operate as follows:

- `DIGEST:` enable digest delivery mode (which negates INDEX), enable mail delivery. No change from version 1.8a.
- `INDEX:` enable index delivery mode (which negates DIGEST), enable mail delivery. No change from version 1.8a.
- `NOMAIL:` disable mail delivery. No change from version 1.8a.
- `MAIL:` restore mail delivery, without altering the digest/index/normal delivery setting (new behavior). For compatibility with 1.8a, if mail delivery was already active, the `MAIL` option negates `INDEX/DIGEST`. Thus, a user going from `NOMAIL` to `MAIL` will keep his previous delivery options, whereas a user going from `DIGEST` or `INDEX` to `MAIL` will in fact deactivate index/digest mode.

To revert from digest/index subscription mode to normal delivery, you can use either the `MAIL` option as before, or the `NODIGEST/NOINDEX` option. The `NODIGEST` and `NOINDEX` options were actually present in versions 1.7f and 1.8a, as synonyms for the `MAIL` option. In other words, you can update your instructions to indicate that the `DIGEST/INDEX` options are negated by the `NODIGEST/NOINDEX` options, even if you are still running an older version of the software.

Note that for backward compatibility, the command `SET` *listname* `MAIL` sent by a user who is set to DIGEST but not also set to NOMAIL will cause the user to be set to NODIGEST (the behaviour is identical for users set to INDEX but not to NOMAIL). `SET` *listname* `MAIL` sent by users set to DIGEST/NOMAIL or INDEX/NOMAIL will simply remove the NOMAIL setting and leave the user set to DIGEST or INDEX as the case may be.

Note that in extreme cases, subscribers using the `DIGEST` option may receive more than one digest per cycle if the digest limit is reached before the end of the cycle.

5.3.3. MIME/NOMIME
Toggles MIME functions on and off. Currently this is only useful if the user has a mail

client that supports MIME digests. Note that users who send their `SUBSCRIBE` command using a MIME-compliant agent will have this option set automatically unless "`Default-Options= NOMIME`" is specified for the list.

In future versions, this toggle may control other MIME functions.

5.3.4. INDex/NOINDex
Causes the subscriber to receive one posting per digest cycle containing only an index of subject topics for all messages during that cycle. See the section on `DIGEST` (above) for further information.

5.3.5. ACK/NOACK/MSGack
These three command words control the level of acknowledgment the subscriber receives when posting to the list. `ACK` causes LISTSERV to send a short confirmation message to the subscriber when the post has been received and distributed. `NOACK` disables the confirmation feature for the subscriber (although BITNET subscribers will receive a short interactive message on their terminal). For BITNET subscribers, `MSGack` provides the same information as `ACK` via interactive messages.

5.3.6. Options for mail headers of incoming postings
By specifying one of the following command words, the subscriber can control the amount of mail header information prepended to list mail. The syntax is `SET listname headertype,` where `headertype` is one of the following:

`FULLhdr`	"Full" mail headers (default) (formerly FULLBSMTP)
`SHORThdr`	Short headers (formerly SHORTBSMTP)
`IETFhdr`	Internet-style headers
`DUALhdr`	Dual headers, useful with PC or Mac mail programs
`FULL822`	"Full" RFC822 mail headers
`SHORT822`	Short RFC822 mail headers
`SUBJecthdr`	"Full" RFC822 mail headers (like the default), except that LISTSERV adds the list's default subject tag to the subject line of mail coming from the list. To turn this off, simply set another mail header option.

Note: do not use `FULL822` or `SHORT822` unless debugging specific problems or unless directed by L-Soft. Use of these options can seriously slow performance as they force LISTSERV to generate a separate "envelope" for each user so set. `FULL822` and `SHORT822` are obsolete but remain available for compatibility with certain older BITNET mailers still in use.

Quite a few non-technical users are relying on non-RFC822 user interfaces for reading their mail. Quite often these user interfaces are user-friendly, quality implementations of a proprietary mail protocol which the users are proficient with, but which happens not to lend itself to bidirectional mapping to RFC822. The users may have a good reason for using this particular program, and they complain that it is not always clear what list the postings come from, or who posted them. Other users have very primitive mail programs which do not preserve the original RFC822 header and may not even have a "message subject" concept. The user knows which list the message came from, but not who posted it, making private replies impossible.

The `DUALHDR` (minimum abbreviation: `DUAL`) header option is provided to help solve this problem. Dual headers are regular short (`SHORThdr`) headers followed by a second header inside the message body. This second header shows what list the message is coming from ('Sender:'), the name and address of the person who posted it ('Poster:'),

the poster's organization, if present, and the message subject. The date is not shown because even the most primitive mail programs appear to supply a usable message date.

The **SUBJECTHDR** (minimum abbreviation: **SUBJ**) header option is provided for users who want to see a "tag" in the subject line of their incoming list mail that indicates where the mail is coming from (e.g., to activate a filter in their mail program to drop the message into a specified notebook). Note that if you have **SHORT** headers (or any other header option) set, setting your option to **SUBJecthdr** will automatically change you to **FULLHdr**, as subject tags require full headers. Also, subject tags are not generated for messages sent without a RFC822 "Subject:" header.

Generally, users will be well-served by the **FULL** header option, which is the default.

Documented Restriction: The use of the **SHORTHDR** or **DUALHDR** options will break messages that depend on MIME encoding, because these options strip the RFC822 headers that identify the message as a MIME message. SHORTHDR and DUALHDR were designed for the non-MIME mail clients which prevailed in LISTSERV's early history. As most mail clients today support MIME, the use of these options is now deprecated.

5.3.7. Putting the list name into the Subject: field

To do this, use the **SUBJecthdr** personal option as explained in the previous section. To set this option by default for new subscribers, include it in the **Default-Options=** keyword setting for your list (see 5.4, below). To set it for existing subscribers, use the **SET** command.

5.3.8. CONCEAL/NOCONCEAL
Occasionally, a subscriber may not want his presence to be known to someone else making a casual **REView** of the list. Subscribers may choose to "hide" their subscription from the **REView** command by using the **CONCEAL** command. Conversely, a subscriber may choose to remove this restriction by issuing the **NOCONCEAL** command. Note that the list owner can always obtain a list of all subscribers, both concealed and unconcealed, by issuing the **GET listname (NOLock)** command, or by issuing a **QUERY *listname* WITH CONCEAL FOR *@*** command.

5.3.9. REPro/NOREPro
This option controls whether or not the subscriber will get a copy of his or her own posts back from the list after they are processed. Generally, if a subscriber's mail program is configured to file copies of the subscriber's outgoing mail, or if the subscriber has one of the acknowledgment options **(ACK/MSGack)** enabled, this option should be set to **NOREPro**. If, on the other hand, the subscriber is set to **NOACK** and doesn't keep a copy of outgoing mail, this option should probably be set to **REPro**.

5.3.10. TOPICS
Topics are not available in LISTSERV Lite.

If list topics are enabled, this option allows the subscriber to specify which topics he or she will receive. The syntax of a **SET TOPICS** statement is significantly different from that of the other options. See Chapter 6, Section 7, for more information on this syntax.

5.3.11. POST/NOPOST
This option may be set only by list owners or the LISTSERV maintainer. It is not available in LISTSERV Lite.

A subscriber set to **NOPOST** may not post to the list. **NOPOST** gives the individual list owner the ability to serve out abusive or obnoxious posters without having to add such users to the list's "Filter=" setting. Subscribers set to **NOPOST** will still receive list mail – they just won't be able to post mail to the list.

The list owner or LISTSERV maintainer may issue the

> **SET *listname* POST FOR *userid@host***

command to reverse a previously-set **NOPOST**.

Note for peered lists: **NOPOST** must be set globally or a user can bypass the setting by simply posting to another peer. Thus you must add the user manually to the other peers and then set the user to **NOMAIL** as well as **NOPOST** on the peers.

Setting **NOPOST** for a user cancels any previous **EDITOR** or **REVIEW** setting for that user.

Note that list editors who are set to **NOPOST** will be able to approve messages but will then be told they cannot post to the list. The **NOPOST** subscriber option does override any **Editor=** or **Moderator=** definition in the list header, so be sure that your editors and moderators are set to **POST**.

5.3.12. EDITOR/NOEDITOR
This option may be set only by list owners or the LISTSERV maintainer, and is effective only on moderated lists. It is not available in LISTSERV Lite.

A subscriber set to **EDITOR** on an edited/moderated list may post directly to the list without a moderator's intervention. It is virtually identical to adding the subscriber's address to the "Editor=" keyword, but easier to manage. The only difference between the **EDITOR** option and the "Editor=" keyword, other than not being visible in the list header, is that the "Editor=" keyword also defines a (seldom used) access level class which can then be used in keywords such as "Review=". Thus, one could have a list with "Review= Editor", indicating that only the users listed in the "Editor=" keyword are allowed to review the list. The **EDITOR** option does not confer this privilege. Note that the **EDITOR** option is only meaningful on moderated lists.

The list owner or LISTSERV maintainer may issue the

> **SET *listname* NOEDITOR FOR *userid@host***

command to reverse a previously-set **EDITOR**.

Setting **EDITOR** for a user cancels any previous **NOPOST** or **REVIEW** setting for that user.

5.3.13. REVIEW/NOREVIEW
This option may be set only by list owners or the LISTSERV maintainer. It is not available in LISTSERV Lite.

When a subscriber is set to **REVIEW**, all postings from that subscriber are forwarded to the list editor or list owner for approval. Approval for these postings is always via the OK

mechanism – there is no need to forward the posting to the list, simply reply to the approval confirmation with "OK".

Note that if a list is unmoderated, it is still possible to direct **REVIEW** postings to a specific person by adding an "Editor=" or "Moderator=" keyword to the list header.

The list owner or LISTSERV maintainer may issue the

<div align="center">

SET *listname* **NOREVIEW FOR** *userid@host*

</div>

command to reverse a previously-set **REVIEW**.

Setting **REVIEW** for a user cancels any previous **NOPOST** or **EDITOR** setting for that user.

5.3.14. RENEW/NORENEW
This option may be set only by list owners or the LISTSERV maintainer.

Enables or disables subscription renewal confirmation on an individual subscriber basis. Setting a subscription to **NORENEW** is particularly useful for exempting list owners, redistribution lists, and other subscriptions which should not or must not receive the confirmation request message from the renewal process.

The list owner or LISTSERV maintainer may issue the

<div align="center">

SET *listname* **RENEW FOR** *userid@host*

</div>

command to reverse a previously-set **NORENEW**.

5.4. Setting original default options with the Default-Options= keyword

The list owner may specify original defaults for many subscriber options by using the "Default-Options=" keyword. This keyword takes regular **SET** options as its parameters. Examples include:

    * Default-Options= DIGEST,NOREPRO,NOACK

    * Default-Options= REPRO,NONE

You may have more than one "Default-Options=" line in your header, as needed.

Note that any default topics are set with the "Default-Topics=" keyword. See Appendix B for details on this keyword.

Also note carefully that your existing subscribers are not affected by any change to the Default-Options= keyword. This keyword sets initial options only for people who subscribe after it is defined. If you want to update your existing subscribers to the Default-Options settings, you must use the **SET** command with a wildcard (i.e., **FOR *@***) to do so.

6. Moderating and Editing Lists

Please note that much of this chapter is subjective, based on personal experiences during several years of list ownership, and may not necessarily match your own philosophy of "the way things ought to be." The following sections are offered as one way to run a list, and the author does not mean to assert that the one way offered – *his* way – is the *only* way. As we seem to say so often, "your mileage may vary."

6.1. List charters, welcome files, and administrative updates

One of the most important things you can do as a list owner is make it clear from the outset what policies are in place and will be enforced if it becomes necessary. Due to a potential for controversy, for instance, some lists may require a formal "list charter" by which all subscribers must agree to abide before they are allowed to subscribe. Other lists may be able to get by with a simple welcome file (see below) that spells out basic netiquette, polices on "flaming" and commercial posts, and anything else that seems appropriate (such as how to get in touch with the list owner in an emergency, where the list archives are located, etc.).

It is particularly important on open subscription lists that you make a concerted effort to remind your subscribers on a regular basis of the policies you have set for your list, as well as any other information they need in order to make best use of your list. If you have a great deal of subscriber turnover, it may be necessary to do this every few weeks. You may decide to put together a quarterly or semi-yearly post for more stable lists. Ensure that the subject line is indicative of what the administrative posting is so that there is no question as to whether or not you posted it (even if subscribers don't read it). (Note that with LISTSERV 1.8c or later you can use the `PROBE1` mail template form and automatic address probing to do this automatically.)

6.2. The role of the list owner as moderator

By default, the list owner becomes a moderator of sorts, even if the list in question is neither edited nor officially moderated. This means that, as a list owner, you must be prepared to maintain order if it becomes necessary. At the same time, you must moderate *yourself* so that you do not alienate users and cause your list and/or host institution to suffer as a result. Thankfully, mailing lists have generally enjoyed relative peace and quiet over the years in comparison to newsgroups, but mailing lists have unique problems of their own.

Lists dedicated to controversial subjects are more likely to become arenas for "flame wars" between subscribers with hard-held and differing opinions than those dedicated to the discussion of popular software packages, but this does not mean that the latter are immune any more than it means that the former are constantly plagued by flames. The example set by you as list owner and as a participating subscriber to the list is perhaps the most important factor in whether or not your list becomes a site known for strife and controversy. In other words, if you appear not to care about whether or not discussion is on topic and/or civilized, no one else will, either. Yet if you become a policeman – the other end of the spectrum – no one will want to subscribe or participate for fear of your wrath. Either way, your list is unlikely to last very long.

The middle ground is, as in most things, the place to be when administering a list. Some call this "firm but fair," letting things go pretty much as they will but stepping in with a wry or gently chiding remark from time to time when exchanges get heated. And they will! Software discussion lists are particularly bad about this when new subscribers ask "frequently-asked questions" (FAQs) and veteran subscribers respond in exasperated

fashion with "RTFM!" (Read The *Fine* Manual) and similar nasty retorts. Good list owner practice at this point is likely to be a good-natured reminder from you that flames belong in private mail, pointing out that new subscribers have no way of knowing that the particular questions they ask have been asked (and answered!) n random times before, and possibly adding a link to the list's archives (if they are available on the web) or instruction on how to use the `SEARCH` command to look for answers before asking.

Finally, if your mailing list has an international audience, you must be careful to account for language problems and cultural differences. You will need to decide which languages are allowed or not allowed on the list; this should be mentioned shortly in the list abstract or welcome message. Unless the list is specific to one country or is explicitly for discussion in a specific language, the official language will probably be English. As your list grows, some subscribers may object to this decision, arguing that people who have trouble expressing themselves in English should be allowed to use their own language, with the understanding that many people will be unable to understand what they are saying. As the list owner, it will be your call. Usually, the best compromise is to start a separate list for discussions in the new language. However, you must be careful in wording your decision. In multi-lingual cultures, it is usually considered a courtesy to use the other person's language. It is certainly considered rude for people to demand that everyone else should speak their language. Thus, if your native language is English, you will be in a delicate position. To avoid a flame war, you will want to make sure that your decision does not come out as a unilateral demand. Politely suggesting a separate list, and tolerating an occasional non-English posting when the poster genuinely cannot speak English, is often the best course of action.

Another possible source of flame wars is unintended rudeness. It is easy to forget that non native speakers are making an effort every time they post something to the list. People will make mistakes, sometimes appearing rude when they did not mean to, simply because they used the wrong word. Another cause of apparent rudeness is cultural difference. Things which are perfectly normal in one culture can be insulting in another. For instance, *ad hominem* attacks are perfectly acceptable in some countries. Conversely, referring to other people by their first name ("As Peter said in his last message, ...") can be downright insulting in some cultures, where anything short of the full title is at best condescending. But, of course, in other countries the use of the full title is considered sarcastic... There is no middle ground here, because there are too many conflicting cultures and too many languages. The only way to successful cross-cultural communication is through the tolerance of other people's cultural habits, in return for their tolerance of yours.

6.3. The role of the list owner as editor

Edited lists are generally used for the purpose of "full moderation" or for refereed electronic journals or the like, for which random postings from subscribers and/or non-subscribers may not be welcome for general distribution. This places the list owner and any editors in the position of being full-time monitors of what is and is not allowed to go through to the list.

A word of warning to potential list editors: Rules on the Internet are not set in stone. Some people will insist on their right to post without what they will term "censorship" by the list editor. Some will become upset to the point of threatening to report you to your local computing center administrators for abridging their freedom of speech, or (in the U.S.) even threatening to sue your institution and you personally for an abridgment of their First Amendment rights. It is therefore vitally important to you that you keep a "paper trail" of such complaints in the event that threats become reality and you are asked about them. This common practice in the business world should be common practice in list

ownership as well.

Freedom of speech and copyright issues on the Internet have not yet been tested in the courts as of this writing. These are both areas in which list editors and list owners in general must tread carefully. *Always* document any problems you may have in these areas.

6.4. Setting up an edited list

Please note that the "Moderator=" keyword and moderation "load-sharing" are not available in LISTSERV Lite.

Note that L-Soft currently recommends that edited lists be coded with the "**,Confirm**" parameter to the "Send=" keyword, in other words:

* Send= Editor,Confirm

or

* Send= Editor,Hold,Confirm

This will help prevent malicious users from forging mail from an editor address in order to get around your moderation settings, by telling LISTSERV to require an "OK" confirmation whenever it receives a posting from an editor address. The "OK" request goes to the editor address, so the forger is stymied.

Note also that some vacation programs and broken mailers have recently been "reflecting" mail back to lists in such a way that the mail appears to be coming from the editor's address (and the mail therefore gets through). Setting the "**,Confirm**" option will stop this phenomenon as well.

When confirmation is required for Editor postings, please note that the confirmation request *always* goes to the Editor who posted, even if you have moderation "load-sharing" configured as noted below. Moderation "load-sharing" applies to postings from general users only.

Should you decide that an edited list is the way to go for your particular situation, you need only add the following lines to your list header file:

> *** Send= Editor**
> *** Editor= userid@some.host.edu**

where "userid@some.host.edu" should be replaced with the network address of the person who will be handling submissions to your list.

There can be multiple editors as well (and multiple Editor= lines, if desirable), and they do not have to be list owners:

> *** Send= Editor**
> *** Editor= alex@reges.org,joe@foo.bar.edu**
> *** Editor= tony@tiger.com**

Normally, LISTSERV forwards submissions only to the first editor defined by the "Editor=" keyword. In the case above, all submissions would go to the primary list owner.

NOTE CAREFULLY that the first editor CANNOT be an *access-level*; e.g., you cannot use the notation "Editor= Owner" to define the first editor. LISTSERV requires that the primary editor of a list must be the e-mail address of a real person.

Note also that this does not apply to second and subsequent editors. For instance, in order to allow subscribers to post directly but have non-subscriber posts sent to an editor for approval, you can code something like:

```
* Send= Editor
* Editor= alex@reges.org,(MYLIST-L)
```

On a high-volume list, LISTSERV allows you to share the editing load via the "Moderator=" keyword. By default, this keyword is set to the same value as the first editor defined by "Editor=". When you define more network addresses with the "Moderator=" keyword, LISTSERV sends submissions to each moderator in sequence. The difference between the "Editor=" and "Moderator=" keywords lies in the fact that while any editor can post directly to the list, only moderators receive the forwarded submissions from non-editors.

Here is an example of a list with both Editor= and Moderator= keywords defined:

```
* Send= Editor
* Editor= joe@foo.bar.edu,tony@tiger.com,kent@net.police.net
* Moderator= kent@net.police.net,joe@foo.bar.edu
```

This list will "load-share" the editing duties between Kent and Joe. Tony is able to post directly to the list, but will not receive forwarded subscriber posts for editing.

Note that whereas an Editor is not required to be a Moderator, a Moderator should *always* be listed as an Editor. LISTSERV currently compares the contents of the "Editor=" and "Moderator=" keywords and consolidates the two sets of parameters if necessary, but coding lists this way is not considered good practice and the "compare/consolidate" feature may be removed in a future upgrade.

Beginning with 1.8c, if the parameter "All" is coded before any moderator addresses, LISTSERV will send copies of all postings to all moderators, any of whom may approve the message. An example of this would be

```
* Moderator= All,kent@net.police.net,joe@foo.bar.edu
```

Assuming "Send= Editor, Hold", once a message is approved by one of the moderators, any other moderator attempting to approve the same message will be told that an identical message has already been posted to the list.

If "Send= Editor" (e.g., without "Hold"), please note that if a note is appended or prepended to the edited post, or if the body of the post itself is edited (that is to say, if the body of the approved message is changed), duplicates are possible. Thus it is important that the moderators of any list set up this way pay close attention to whether or not the posting has already been approved by another moderator.

6.5. Submitting subscriber contributions to an edited list

By default, LISTSERV forwards subscriber contributions to the Moderator/Editor with the following paragraph prepended to the message body:

```
This message  was  originally  submitted  by  JOE@FOO.BAR.COM  to  the  ACCESS-L
list at PEACH.EASE.LSOFT.COM. If you simply  forward it back to the list, using
a mail command that generates "Resent-"  fields (ask your local user support or
consult  the documentation  of  your mail  program  if in  doubt),  it will  be
distributed  and  the  explanations  you   are  now  reading  will  be  removed
automatically. If on the other hand you edit the contributions you receive into
a digest, you will have to  remove this paragraph manually. Finally, you should
be able  to contact  the author  of this  message by  using the  normal "reply"
function of your mail program.

----------------- Message requiring your approval (25 lines) -----------------
[message body]
```

Figure 6.1. The "editor-header" prepended by default to subscriber contributions forwarded to the list moderator.

If you leave this paragraph prepended to the message, LISTSERV will strip it off when it processes the message and to all intents and purposes the message will appear to have come directly from the original sender. *Warning: If your mail program or client does not generate "Resent-" fields, the forwarded postings will appear to be coming from you rather than from the original sender. See Section 6.6 for an alternative if your mail program does not generate these fields.*

(If you leave the editor-header paragraph on the message, make sure that your mail client or mail server does not insert quoting characters (e.g., ">") at the beginning of all of the lines in the message when you use the forwarding function of your mail program. If this happens then the editor-header will not be stripped from the message.)

When you are ready to edit and/or submit the contribution to the list, simply use the "Forward" function of your mail client. You can make any changes you feel are appropriate to the message body, but be sure to read sections 6.2 and 6.3 above before deciding to do so.

6.6. Message Approval with Send= Editor,Hold

LISTSERV includes an optional mechanism allowing you to simply "ok" messages which are then posted with all the correct headers. This option is targeted mainly at list moderators who just approve/reject messages, as opposed to people who actually edit the content of messages. The option is also a good choice if you have a mail client that does not insert "Resent-" header lines into forwarded mail.

To activate this feature, code your list "Send= Editor,Hold" and be sure that you have defined at least one editor who will be in charge of approving the messages. A copy of the message on "hold" is sent to the editor with minimal instructions (in order to avoid adding a long message before the text needing approval each time).

To approve a message forwarded to you with "Send= Editor,Hold", simply reply to the approval request and type "OK" as the body of your reply. LISTSERV will normally pick up the confirmation request number from the subject line. If there is a problem, LISTSERV may ask you to resend the approval confirmation along with the number. For instance,

```
OK 6A943D3C
```

If the message has been in the spool longer than the time-out period (LISTSERV holds these jobs for a minimum of 7 days), you will receive notification that the confirmation number does not match any queued job. If you need to increase the time-out period, you can set a value for the "Confirm-Delay=" list header keyword that is greater than 168h, but please read the section on "Confirm-Delay=" in Appendix B before doing so.

If you do not want the message to be forwarded on to the list, you need not do anything. The message will expire automatically at the end of the time-out period and will be deleted from the queue.

6.7. Using list topics

List topics are not available in LISTSERV Lite.

List topics provide powerful "sub-list" capabilities to a list. When properly set up and used, topics give subscribers the ability to receive list postings in a selective manner, based on the beginning of the "Subject:" line of the mail header. It is important to note the following points about topics:

- Topics are best employed on moderated lists. This makes it possible to review the "Subject:" header line to make sure that it conforms to one or more of the topics defined for the list *before* you forward the post to the list. Not only does this help catch simple errors (such as misspellings of the topic), but it also allows the moderator to add a topic into the subject line if one is not already there.

- If you employ topics on unmoderated lists, your subscribers must be well-educated in their use. Otherwise, there is no point in using them. Messages that do not conform to a specified topic are lumped into the reserved topic "Other" and are distributed only to subscribers who have explicitly defined "Other" as a topic they wish to receive. Therefore some subscribers will receive the message and some won't, and it is problematic as to whether the message will actually reach the entire audience for which it is intended.

- It is important to note that topics are active only when the subscriber's subscription is set to MAIL. All messages posted to the list, regardless of topic, are included in the digest and/or index for the list (if available) because the same digest/index is prepared and sent to all the digest/index subscribers. Similarly, all messages posted to the list are archived in the list's notebook logs (if available), making it possible for subscribers to retrieve postings in topics they are not set to receive normally.

The basic keyword syntax for defining list topics in the list header file is:

```
* Topics= topic1,topic2,...topic21
```

And the basic syntax used to set topics for users once they have been defined is:

```
SET listname TOPICS: xxx yyy zzz for userid@host
```

where *xxx*, *yyy*, and *zzz* can be:

- A list of all the topics the subscriber wishes to receive. In that case these topics replace any other topics the subscriber may have subscribed to before. For instance, after 'SET XYZ-L TOPICS: NEWS BENCH', the subscriber will receive news and benchmarks, and nothing else.

- Updates to the list of topics the subscriber currently receives. A plus sign indicates a topic that should be added, a minus sign requests the removal of a topic. For instance, "SET XYZ-L TOPICS: +NEWS -BENCH" adds news and removes benchmarks. If a topic name is given without a + or - sign, + is assumed: "SET XYZ-L TOPICS: +NEWS BENCH" adds news and benchmarks.

The first topic name must have the plus sign to show that this is an addition, and not a replacement.

- A combination of the above, mostly useful to enable all but a few topics: "SET XYZ-L TOPICS: ALL -MEETINGS".

The colon after the keyword TOPICS: is optional, and TOPICS= is also accepted. The subscriber should not forget to include the special OTHER topic if you want to receive general discussions which were not labeled properly. On the other hand, if the subscriber only wants to receive properly labeled messages it should not be included. ALL does include OTHER.

With the "Default-Topics=" keyword, you can also set default topics for users that will be effective as soon as they subscribe to the list. For instance,

* Default-Topics= NEWS,BENCH,OTHER

would set the new user to receive topics NEWS, BENCHmarks, and any messages that are incorrectly labeled.

In LISTSERV 1.8d and following, you may get a listing of topics with the number of subscribers who have them set by issuing the command

REVIEW *listname* Short TOPics

(if you do not specify **Short** then the topic listing follows the list of subscribers in the review output). The following is a sample output (assuming you actually have topics enabled; if topics are not enabled then the **TOPics** option is ignored):

```
*   Topic                    Subscribers
*   -----                    -----------
*   Apps                          1,411
*   Backup                        1,330
*   Beta                            951
*   Bugs                          1,416
*   Comm                          1,395
*   Desktop                       1,407
*   Hardware                      1,401
*   Install                       1,373
*   Internet                      1,002
*   Network                       1,399
*   Wish                          1,336
*
*   "Other" topic                 1,294
*   Digest/index subscribers      1,384
```

See Appendix B under the **Topics=** keyword for more information on setting up and using list topics.

6.8. The <listname> WELCOME and <listname> FAREWELL files

When a user subscribes and signs off of a list, LISTSERV looks for list owner-supplied files called ***listname*.WELCOME** and ***listname*.FAREWELL**, respectively. If found, it sends the user a copy of the appropriate file in addition to its own administrative message. The WELCOME and FAREWELL files allow the list owner to send a more

personal message to the user that can help set the tone for how the list is used. The WELCOME file may contain information about the list charter and netiquette rules, or be simply a message welcoming the user to the list. The FAREWELL file can be used to gather feedback about how the list is serving users.

6.8.1. Creating and storing the *listname* WELCOME and FAREWELL files

The procedure differs slightly on VM systems, but the following will work for unix, VMS and Windows systems:

1. Create the file. If you place a "Subject:" line at the top of the document, i.e., as the first line, LISTSERV will pick that line up and use it as the RFC822 "Subject:" header line. Otherwise, LISTSERV places a generic subject line in the mail message.

2. If the file contains special characters (i.e., non-7-bit ASCII characters) and you want to specify a character set for LISTSERV to include in the headers of the messsage, place a line such as:

   Character-Set: ISO-8859-7

   at the top of the message (or directly following the "Subject:" line if one is configured). The value "ISO-8859-7" is used here as an example only and should be replaced with the appropriate character set descriptor. If the file does not contain any non-7-bit ASCII characters, this line will be ignored.

3. Be sure that you have defined a "personal password" to LISTSERV with the **PW ADD** command before you **PUT** the welcome file. If you have done this but can't remember the password, send LISTSERV a **PW RESET** command. You will then be able to add a new password with the **PW ADD** command.

4. Send the file to LISTSERV with a **PUT** *listname* **WELCOME PW=XXXXXXXX** command at the top of the file, just as if you were putting the list itself. Replace **XXXXXXXX** with your personal password.

The variation for VM systems is that the LISTSERV maintainer will have to create a fileid for the file before you can **PUT** it on the server. Contact the LISTSERV maintainer before trying to store your WELCOME and/or FAREWELL files.

Here is the format of a very simple WELCOME file. (Note that the FAREWELL file is created and stored in an identical manner.)

```
PUT SONGTALK WELCOME PW=XXXXXXXX
Subject: Welcome to Songtalk!
Welcome to Songtalk, the list for Songwriters talking about their work.
Your list owner is Susan Lowell (susan@example.com).
```
Figure 6.2. Sample WELCOME file.

6.8.2. Using the *listname* WELCOME file as a moderation tool

The WELCOME file should contain information geared toward orienting the new subscriber to several areas. The outline of a *suggested* WELCOME file follows:

1. The revision date for the WELCOME file.

2. A heading including the short and long names of the list, along with the name and network address of the primary list owner (or the list owner who handles subscription

issues/problems).

3. Any warnings about the list that you want people to see immediately. These might include

    -   a notice regarding the volume of mail subscribers can expect from the list
    -   any newsgroups that echo the list
    -   ftp sites for the list
    -   where to send LISTSERV commands
    -   where to find more in-depth information about the list (if you do a quarterly administrative posting or have a FAQ, where can it be found?)

4. A short abstract of what the list is all about. This might be a duplicate of the description you send to NEW-LIST.

5. The author includes the following paragraph at this point:

    > Users new to the use of L-Soft's LISTSERV are encouraged to read the
    > online files LISTSERV REFCARD and LISTSERV GENINTRO, which can be
    > obtained by sending the following commands in the body of a mail
    > message to LISTSERV@LISTSERV.NET:
    >
    > INFO REFCARD
    > INFO GENINTRO

6. Any guidelines for use of the list, including the list charter if you have one.

7. Information about the notebook archives and how to retrieve them.

8. Other list-specific information that might be important to new users.

Naturally, list owners should write WELCOME files based on their own experience of what is needed. A WELCOME file should not be static – review it once in a while to ensure that it continues to meet the needs of new subscribers.

6.8.3. Using the *listname* FAREWELL file as a feedback tool

The following FAREWELL file used to be used on the ACCESS-L list on PEACH.EASE.LSOFT.COM, and was intended as a tool to get feedback from users. When it was in use ACCESS-L's list owner typically received 3-5 responses to this message each week.

```
Subject:  Your ACCESS-L Signoff Request
I'm sorry to see that you're leaving ACCESS-L.  If there is anything you believe
ACCESS-L should have offered but didn't, or there are any other suggestions
you may have for the list, please feel free to write directly to me.

Sincerely,
Nathan Brindle <nathan@indycms.iupui.edu>
ACCESS-L List Owner
```
Figure 6.3. FAREWELL file used as a feedback tool.

6.8.4. The alternative to using WELCOME and FAREWELL files

It is possible to modify LISTSERV's default mail template so that only one message is sent to users when they subscribe and unsubscribe, rather than an administrative message from LISTSERV and a WELCOME or FAREWELL file from the list owner. See Chapter 9 for the details on modifying the default mail templates.

However, it is likely that the average list owner will prefer to use the WELCOME and FAREWELL files over changing the more-complicated templates. Thus both avenues are provided, and may be used depending on the list owner's level of comfort.

6.9. Social conventions (netiquette)

Like so many other things, network users tend to expend a great deal of virtual gunpowder about the subject of etiquette on the network (otherwise known as *netiquette*). Part of the culture of the network is built on the fact that an individual user can put forward any face he or she cares to present. Thus over time, the network has evolved various sets of rules that attempt to govern conduct. To avoid taking up a great deal of space arguing the merits of differing systems of netiquette, the following general pointers that should be accepted by most users are offered for the convenience of the list owner.

Recognize and Accept Cultural and Linguistic Differences

The Internet is international, and while English is generally accepted as the common language of the network, list owners and list subscribers cannot afford to take the position that everyone on the Internet understands English well. In a medium that is invariably connected to language, special understanding is required to deal with questions or statements from people for whom English is not the primary tongue. Often today (at least in the US) a person's first sustained interaction with others on an international basis is via the Internet. It is imperative that this interaction be on the highest level of cordiality and respect from the outset in order for all concerned to benefit.

Additionally, care should be taken when using local idiom and slang. A common word or phrase used by Americans in everyday speech, for instance, might be taken as profanity or insult by those in other English-speaking countries, and may not be understood at all by non-native speakers of English. When a list has a high international readership, it is probably best to avoid non-standard English so as to provide the clearest and least-objectionable exchange of ideas.

Private Mail Should Dictate Private Responses

If someone on a mailing list has sent a private message to you (i.e., not to the list at large) and you have lost that person's address but want to respond, do not post private mail to the list. The `REVIEW` command will give you a copy of the list membership that you can search for the person's address. If this approach does not work, contact the local postmaster or the list owner for help.

Flaming is (Usually) Inappropriate

Flames (insults) belong in private mail, if they belong in mail at all. Discussions will often result in disagreements. Rebuttals to another person's opinions or beliefs should always be made in a rational, logical and mature manner, whether they are made publicly or privately. What is a flame can range from the obvious (ranting and raving, abusive comments, etc.) to the not-so-obvious (comments about how many "newbies" seem to be on the list these days, "RTFM!" exhortations, etc.).

Foul Language

Subscribers should refrain from abusive or derogatory language that might be considered questionable by even the most liberal and open-minded of networkers. If you wouldn't say it in front of your mother, don't say it in electronic mail.

Unsolicited Advertising and Chain Letters

Most of these are contrary to appropriate use policies governing the use of the poster's Internet access provider. Not only that, they are annoying and (in the case of chain letters) often illegal. See Section 6.10 on the subject of "spamming" for more details.

Other Disruptive or Abusive Behavior

Self-explanatory. It is rarely possible to catalog all forms of anti-social network behavior. Be sure that you as a list owner cover as many bases as you think necessary when promulgating a code of netiquette for your list. Then – be sure to adhere to it yourself.

6.10. Spamming: what it is, and what to do about it

"Spamming" is a network term invented to describe the act of cross-posting the same message to as many newsgroups and/or mailing lists as possible, whether or not the message is germane to the stated topic of the newsgroups or mailing lists that are being targeted. A "spam" is defined therefore as either (1) a specific act of spamming, such as the so-called "Green Card Spam", or (2) the message that actually comes to your list as a result of someone initiating a specific act of spamming ("The message you just saw was a spam, and it should be ignored"). Spams are fairly easy to recognize at a quick glance; they often have "To:" fields directed to large numbers of lists, usually in alphabetical order.

If a spam gets through to your list, it will probably engender sarcastic replies (often with the spam quoted in its entirety) – and if your list is coded "Reply-To= List", they will likely come *back to the list*. It is therefore imperative that you make subscribers aware that when a spam occurs:

- The person responsible for the spam is probably not subscribed to the list, and any response back to the list will not reach that person.

- An appropriate response to a spam is to forward a single copy of the spam to the person in charge of the site from which the spam originated ("POSTMASTER", "ROOT", etc.) pointing out that the spammer is probably violating his site's appropriate use policies.

- It is inappropriate to attempt to flood the spammer's mailbox with network mail in response. This is probably in violation of *your* network's appropriate use policies, and it just wastes bandwidth.

Perhaps the best policy an individual subscriber can adopt toward spammers is simply to ignore them and allow list owners and newsgroup moderators to take care of the problem. If this does not work and subscribers send their complaints to the list anyway, it might be a good idea to moderate the list for a few days until the furor dies down.

LISTSERV attempts to detect "spams" using a variety of proprietary methods. When LISTSERV decides that a message is a spam, it locks out the user for 48 hours,

worldwide in the case of backbone servers.[4] While locked the user is still able to use LISTSERV normally and to post to mailing lists, but all messages will be forwarded to the list owners for human verification. The user is informed that this has happened but is not informed of which lists caught the message and which didn't, denying him any idea how successful he has been.

L-Soft will not document how LISTSERV decides a message is a spam because the point has been reached where a number of authors are writing and selling books detailing how to avoid such precautions. If L-Soft were to document its methods, the next editions of these books would simply include updated instructions on how to bypass them.

If you are interested in a discussion of the phenomenon of SPAM, you can join the SPAM-L mailing list on LISTSERV@PEACH.EASE.LSOFT.COM.

6.11. Appropriate use policies: considerations

As a list owner, it is important that you take into consideration any appropriate use policies that might apply to your list. For instance, if your list is hosted by an educational site that has a policy restricting mail with commercial content from being sent out by its users, your list will *technically* be in violation of that policy if it distributes mail from users advertising commercial services. You would be well advised to request a copy of the appropriate use policy (if any) from your host site and make sure that your subscribers are aware of it by including pertinent sections in your WELCOME file and/or your administrative postings.

Host sites are not the only entities that might have appropriate use policies. The network your host is a part of may have such policies as well.

---

[4] In reality this only works with version 1.8b or later servers. List owners running lists on pre-1.8b servers can protect themselves from non-subscriber spams by setting the Editor= parameters in an appropriate manner; however, this introduces new problems that may not be acceptable to the list owner, even given the increasing incidence of spamming.

7. Overview of List Archives

7.1. What is the list archive?

The list archive consists of all of the notebook logs for your list. (If your list is coded "Notebook= No", then it does not have a list archive, of course.) Users can find out what notebook logs are available for a specific list by sending the command **INDex listname** to the appropriate LISTSERV host.

7.2. Setting up and managing archive notebooks

If your list is coded "Notebook= No", you should consult your LISTSERV maintainer before changing the keyword to create list archive notebooks. The LISTSERV maintainer will have to tell you where the notebook should be kept (the second parameter in the "Notebook=" keyword). Also note that depending on local policies, you may or may not be allowed to archive your list, or keep more than a few months' or weeks' worth of archives available at a given time.

7.2.1 Indexing available archive notebooks

To find out what archive notebooks are available for your list, simply send the

INDex listname

command to LISTSERV.

7.2.2. Deleting existing archive notebooks

To delete an existing archive notebook, simply execute a PUT operation for the notebook in question without sending any other text along with the PUT command line. For instance, send an email with the following text as the only content in the body of the message:

PUT MYLIST LOG9607 PW=mypersonalpw

This command without any other additional text would delete **MYLIST LOG9607** from the server.

Two important issues:

1. This command MUST be issued by e-mail. It cannot be issued via the "Execute a LISTSERV command" facility of the web management interface.

2. NOTE CAREFULLY that you MUST turn off your signature file (if one is enabled in your mail client) in order to successfully delete files. If you do not, LISTSERV will store your signature file in place of the file you are trying to delete instead of deleting the file.

7.3. Database Functions Overview

This functionality is not available in LISTSERV Lite.

In this section, we will detail the basics of a LISTSERV command job and show you a sample database query session. Please note that it is not the purpose of this manual to

provide the user with a detailed database function reference. See Section 7.4 for more information.

For information on the new database functions available in 1.8c, please see section 7.3.3, below. For non-VM servers running 1.8c, you can skip sections 7.3.1 and 7.3.2 entirely.

7.3.1. LISTSERV Command Job Language Interpreter

The LISTSERV database command syntax used to access database functions is English-like in structure. This syntax is called *LISTSERV Command Job Language Interpreter,* or *CJLI* for short.

Database commands are sent to LISTSERV in CJLI "batch jobs". When accessing the database in "batch" mode, you must construct a CJLI job which you must then submit to the appropriate server for execution. This means that you must know in advance what you want to do exactly. If you are not familiar with CJLI, you can use the following "job skeleton" to build up your database search job:

```
//       JOB   Echo=No
Database Search DD=Rules
//Rules DD   *
command 1
command 2
...
/*
```
Figure 7.1. Sample database job skeleton

This CJLI job is sent in e-mail to the appropriate LISTSERV host. You will then receive by return e-mail a "DATABASE OUTPUT" file containing the results of your search. This file might look like this:

```
> Select * in TEST-L
--> Database TEST-L, 4 hits.

> Index
Item #   Date   Time   Recs   Subject
------   ----   ----   ----   -------
000001 95/10/18 13:09   12    This is a test looking for upcasing
000002 95/08/24 09:18    9
000003 95/10/18 13:09    8    Test - please acknowledge receipt
000004 95/10/18 13:09    7    Does Reply-To=Both work correctly?
```
Figure 7.2. Sample DATABASE OUTPUT: Each of the commands in the original job is echoed in the output file (unless specifically disabled).

If you realize that the items you were interested in are number 1 and 3, you will have to submit a new job to ask for a copy of them. The new job must include the "Select" command, as LISTSERV does not cache CJLI commands in the expectation that you will send another command job.

7.3.2. A basic database session (VM servers running 1.8b or earlier only)
(See 7.3.3 for VM 1.8c and later, and for non-VM servers)

Let's say that you are looking for messages in the LSTOWN-L mailing list that pertain to the list header keyword "Digest=". You set up a very simple CJLI job as follows and mail it to LISTSERV@SEARN.SUNET.SE:

```
//       JOB   Echo=No
Database Search DD=Rules
```

```
//Rules DD   *
Search 'Digest=' in LSTOWN-L
Index
/*
```

Figure 7.3. Sample CJLI database search job for VM servers

Figure 7.3, when sent to LISTSERV, says: "Look for the string 'Digest=' in all of the archives you have for list LSTOWN-L. Then, send me back an index of all messages in the archives that include that string."

LISTSERV at SEARN obligingly searches the LSTOWN-L archives, finds the following, and sends it back to you in an e-mail message:

```
> Search 'Digest=' in LSTOWN-L
--> Database LSTOWN-L, 37 hits.

> Index
Item #    Date    Time   Recs    Subject
------    ----    ----   ----    -------
001215 93/01/06 21:58    50     New feature in 1.7f - automatic digests
001339 93/01/18 02:46   110     New features for 1.7f - "Filter=" and list keyword+
001375 93/01/28 10:02    19     Initial reports from 1.7f beta tests?
001401 93/02/08 16:39    58     Re: List of LISTSERV header keywords?
001616 93/03/18 13:42    70     DIGEST boilerplate announcement/reference
001727 93/04/04 15:22   916     Changes from release 1.7e to 1.7f
....
```

Figure 7.4. Part of the LISTSERV response to the CJLI job in Figure 7.3.

The next step is to send a CJLI job to request the specific message(s) you are interested in. Let's say that you are interested in changes from one version of LISTSERV to another, and you therefore would like to see messages 1215, 1339, and 1727. You set up the following CJLI framework:

```
//      JOB   Echo=No
Database Search DD=Rules
//Rules DD   *
Search 'Digest=' in LSTOWN-L
Print 1215 1339 1727
/*
```

Figure 7.5. CJLI job instructing LISTSERV to send specific messages to the requestor.

This example says: "Look for the string 'Digest=' in all of the archives you have for list LSTOWN-L. Then, send me back message numbers 1215, 1339 and 1727."

LISTSERV will repeat the search from Figure 7.3 and will package the three messages you have requested into a return mail message and send it back to you.

7.3.3. A basic database session (All servers running 1.8c or later only)

We'll take a similar situation as described in 7.3.2 and apply it to all servers running LISTSERV 1.8c or later. To search for the term "Digest=" in the EASE-HOME list on HOME.EASE.LSOFT.COM, you would make a new mail message and simply type:

Search 'Digest=' in EASE-HOME

No CJLI is necessary (in fact, it should not be used). LISTSERV would respond to you with the following:

```
> Search 'Digest=' in EASE-HOME
-> 10 matches.
```

```
Item #   Date    Time   Recs    Subject
------   ----    ----   ----    -------
000058 96/01/26 14:44    41     What happened
000059 96/01/26 18:14    38     Re: What happened
000066 96/02/02 22:51    31     Digest Problem
000074 96/02/03 15:01    75     Re: Digest Problem
000075 96/02/03 18:52    49     Re: Digest Problem
000076 96/02/03 16:27    52     Re: Digest Problem
000112 96/02/13 23:37    29     not receiving mail
000126 96/02/25 20:20    63     error/bounce msg posted to list How?
000172 96/03/13 09:11    12     Digest Mailing Time
000483 96/06/22 17:36    34     Header Info


To order a copy of these postings, send the following command:

           GETPOST EASE-HOME 58-59 66 74-76 112 126 172 483


>>> Item #58 (26 Jan 1996 14:44) - What happened
   I never touched the Limits= command or the notebook= All I did was
try and add: Digest= Yes,Daily
            ^^^^^^^
I have tryed this several times with the same reply message:

>>> Item #59 (26 Jan 1996 18:14) - Re: What happened
>    I never touched the Limits= command or the notebook= All I did was
>try and add: Digest= Yes,Daily
            ^^^^^^^
(remainder deleted)
```
Figure 7.6. Sample SEARCH output from non-VM servers

Note that LISTSERV includes excerpts from the indexed postings showing the context of the search term(s). We've deleted all but the first 2 in the example above to save space.

You would then use the new **GETPOST** command to order the specific posts you wanted to read. For instance, we want to read posts numbered 66, 74 through 76, and 126. We would make another new message (or reply to the response from LISTSERV) and type in the body:

GETPOST 66 74-76 126

LISTSERV would then respond with the desired postings. For the non-VM servers, **GETPOST** is analogous to the old database command "**PRINT**". There is no corresponding command for the old database command **INDEX**, since the response to a **SEARCH** command includes the index of matching postings.

7.3.4. Narrowing the search

(Works on both the VM and non-VM servers)

It is possible to add further parameters to your search in order to narrow it. You can limit a search by date with a "since. . . " predicate. Likewise, you can limit by sender and/or by the subject line with a "where . . ." predicate. For instance:

```
Search 'Digest=' in LSTOWN-L since 94/01/01
Search 'Digest=' in LSTOWN-L where sender contains 'Thomas'
Search * in LSTOWN-L where sender is ERIC@SEARN
Search * in LSTOWN-L since 94/01/01 where subject contains 'Digest'
```

are all valid search commands that will (hopefully) dramatically reduce the number of index or print entries returned to you.

7.4. Where to find more information on Database Functions

You can get more detailed information on database functions and the database command syntax by requesting the file LISTDB MEMO from LISTSERV@LISTSERV.NET . You can send either a "GET LISTDB MEMO" command or an "INFO DATABASE" command to retrieve the file. There is also more information on the database functions in the *General User's Guide to LISTSERV*, available on L-Soft's WWW site.

8. Overview of File Archives

There are three file server systems currently in use by various versions of LISTSERV:

- The VM (mainframe) version of LISTSERV continues to support the "traditional" file server system. While it is very powerful, this file server system dates back to 1986 and suffers from a few annoying limitations. In addition, it is written in a non portable language. This will be replaced eventually with the "new" file server system, currently under development.

- The non-VM versions of LISTSERV 1.8d enhance further the new file server system introduced in non-VM 1.8c, which includes most of the functionality of the "traditional" file system. Notably, **GIVE** and file "packages" are now available. Most end user commands will continue to work as before. However, there is no guarantee that the internal data files manipulated by the file server functions will remain as before. Note that **SITE.CATALOG** files from versions 1.8a through 1.8c *are* still supported and will not need to be changed in order to work with 1.8d.

- The non-VM versions of LISTSERV 1.8a and 1.8b supported a "temporary" file server system, to provide an interim solution while the new system was being developed. This temporary system supports only a subset of the functions of the traditional system. This system is no longer supported by L-Soft as it has been superseded by the new non-VM file server referenced above.

In general, the three systems are compatible, with the understanding that the temporary system does not include all the possible options. However, the mechanism for registering files (defining them to the file server system) is different.

Since the first and third systems will eventually be replaced by the second system, rather than providing an exhaustive chapter detailing all filelist aspects from the management side, we have provided only a basic overview of the two systems currently in the field with 1.8d, with pointers to where further information may be obtained.

8.1. What is the file archive?

The file archive consists of all files other than notebook logs that have been stored on the LISTSERV host for your list. Users can find out what files are available for a specific list by sending the command **INDex** *listname* to the appropriate LISTSERV host.

8.2. Starting a file archive for your list

On VM Systems ONLY

With the traditional system (running on the VM servers), the LISTSERV maintainer creates files called "*xxxx* FILELIST", which contain definitions for all the files belonging to a particular archive. These FILELIST files must be created by the LISTSERV maintainer at the site before they can be edited by the list owner.[5]

On Workstation and PC Systems

---

[5] If you are interested in the mechanics of starting a VM-type filelist, the best reference is "Setting Up the LISTSERV File Server--A Beginner's Guide" by Ben Chi (bec@albany.edu). This publication is available from LISTSERV@ALBANY.EDU as FSV GUIDE.

With the hierarchical cataloging system first introduced in 1.8c, the LISTSERV maintainer creates a definition for your *listname*.CATALOG in a system-global file called SITE.CATALOG. The list owner then follows the instructions in chapter 8.4, below, to register files and store them on the server.

Please note carefully that the instructions in chapter 8.3 and the instructions in chapter 8.4 are not interchangeable. If you are not sure which system your list is running on, you can send the command **RELEASE** to the server to find out.

8.3. Filelist maintenance (VM systems only)

If your list is running on LISTSERV under unix, Windows, or VMS, please skip this section as it does not pertain in any way to your implementation of LISTSERV.

Maintaining the filelist for your archive is not difficult. It requires only that you have a working knowledge of VM XEDIT (or your local system's editor) and understand how to send files via e-mail.

8.3.1 Retrieving the filelist

To retrieve your filelist in an editable format, send the command

> **GET *listname* FILELIST PW=XXXXXXXX (CTL**

to the LISTSERV host where the filelist is stored. The **(CTL** switch causes LISTSERV to lock the filelist until you store it again or explicitly unlock it with an **UNLOCK *listname* FILELIST** command. (If you don't want to lock the filelist, use **(CTL NOLOCK** instead.) If your mail account is not located on the same host as LISTSERV, you will need to provide your personal password (same as your password for getting and putting your lists).

A filelist retrieved with the (CTL option does not look like the filelist you get with an INDEX command. A sample (CTL option filelist appears below:

```
*  Files associated with MYLIST and available to subscribers:
*                         rec               last - change
* filename filetype   GET PUT -fm lrecl nrecs   date     time    Remarks
* -------- --------   --- --- --- ----- ----- -------- -------- --------
  MYLIST   POLICY     ALL OWN V      79    45 94/03/16 12:04:23 Mission Statement
  MYLIST   BOOKLIST   ALL OWN V      79   177 94/04/19 16:24:57 Books of interest
  MYLIST   QUARTER    ALL OWN V      73   113 95/03/11 08:57:04 Quarterly posting

*  Listowner's files (not public)
  MYLIST   FAREWELL   OWN OWN V      78     9 95/03/11 08:53:41 Goodbye memo
  MYLIST   WELCOME    OWN OWN V      73   105 95/03/11 09:14:38 Hello memo
```
Figure 8.1. Sample filelist retrieved with (CTL option.

Note that the filelist does not include the comment lines you would normally see at the top of an INDEX filelist; nor does it include any notebook archives. LISTSERV creates these lines dynamically at the time the INDEX command is received from a user. If the filelist you have retrieved has any of this kind of material in it, either a) you have not retrieved the filelist correctly, or b) you or someone else has stored the filelist previously with this material included. If you did a GET with (CTL, you should be able to remove these extraneous lines by simply deleting them.

If you do an INDEX of your archive and it has (for instance) two sets of comment lines or duplicate notebook archive listings, then you should GET the filelist with (CTL and edit

out the offending lines. While the extra lines will not affect the operation of the file server, they are a source of potential confusion for your users.

8.3.2 Adding file descriptors to the filelist

"Adding a file to a filelist" is not exactly accurate terminology, although it is a widely-used phrase. Adding files to file archives is a two-step process: First, add a file descriptor to the appropriate filelist and store the filelist on the server. Second, store the file itself on the server.

To add a file descriptor, start a line with a space and then type in your file's name, access codes, five dots (periods) and a short description, each separated by a space. For example:

```
 MYLIST FAQ ALL OWN . . . . . Frequently-Asked Questions for MYLIST
```

Note that the line *must* begin with a space. Also, you *must* place five dots separated by spaces between the PUT file access code (here it is `OWN`) and the short description. These dots are place holders for the record format (recfm), longest record length (lrecl), number of records (nrecs), and the date and time of the last update. If these dots are not present, LISTSERV will return an error message when you try to store the filelist.

You will note that the line you have just added does not look like the other lines in the filelist. Ignore the "pretty" formatting. LISTSERV will reformat the information for you. After adding the line, your filelist should look like this:

```
*  Files associated with MYLIST and available to subscribers:
*                          rec              last - change
* filename filetype   GET PUT -fm lrecl nrecs   date      time    Remarks
* -------- --------   --- --- --- ----- ----- -------- -------- --------
  MYLIST    POLICY    ALL OWN V      79    45 94/03/16 12:04:23 Mission Statement
  MYLIST    BOOKLIST  ALL OWN V      79   177 94/04/19 16:24:57 Books of interest
  MYLIST    QUARTER   ALL OWN V      73   113 95/03/11 08:57:04 Quarterly posting
 MYLIST FAQ ALL OWN . . . . . Frequently-Asked Questions for MYLIST

*  Listowner's files (not public)
  MYLIST    FAREWELL  OWN OWN V      78     9 95/03/11 08:53:41 Goodbye memo
  MYLIST    WELCOME   OWN OWN V      73   105 95/03/11 09:14:38 Hello memo
```
Figure 8.2. Adding a file descriptor to the filelist

Note that you can add comment lines to the filelist by placing an asterisk in the left-most column instead of a space. Comment lines can act as indexes, descriptions, or pointers to other resources.

Once you are finished adding file descriptors, save the filelist to disk.

8.3.3. File Access Codes (FAC) for user access

FACs define which users have access to files in the file archive. The FAC for GET indicates who may retrieve the files, and the FAC for PUT indicates who may store the files on the server. (Note that some special FACs exist for "superusers" such as the LISTSERV maintainer(s) and the LISTSERV Master Coordinator, who may GET and PUT any file regardless of its GET/PUT permissions.)

The basic FAC codes that are always available for the VM server are:

ALL	universal access.
PRV	only members of the associated mailing list have access.

`OWN`   only the owners of the associated mailing list have access.

(The FAC codes PRV and OWN work only on the VM filelist system. They do not work on the non-VM catalog system. See section 8.4 if you are configuring the non-VM systems.)

(Note that this assumes the name of the filelist is identical to the name of the associated mailing list – for instance, MYLIST@FOO.BAR.EDU would have a MYLIST LIST file and a MYLIST FILELIST file. Ask your LISTSERV maintainer for assistance if this is not the case or if you need special FACs added for special user access to files.)

8.3.4 Deleting file descriptors from the filelist

Before you delete file descriptors from the filelist, you should delete the files themselves from LISTSERV's archive disk. See section 8.6, below, for instructions.

If this step is not followed, LISTSERV may not be able to find the file you want to delete after you edit the filelist and store it.

8.3.5. Storing the filelist

1. Create a mail message to LISTSERV at the appropriate host. (Sending a filelist to LISTSERV@LISTSERV.NET will not work. The filelist must be sent to the host it resides on.)

2. Include the filelist file as plain text in the body of the mail message. Do not attach it with MIME or another encoding scheme, as LISTSERV does not translate encoded messages.

3. Make sure that your mail client does not automatically add a signature file to the bottom of your mail. If it does, your signature file will be treated as part of the filelist and will be stored along with it.

4. At the top of the filelist, add a single line as follows:

   PUT *filename* FILELIST PW=XXXXXXXX

   where **XXXXXXXX** is your personal password for LISTSERV on that host. Note that this is similar to the PUT command used when storing the list file.

5. Send the filelist to LISTSERV.

Once LISTSERV acknowledges the receipt and storage of the filelist, you can send the files that correspond to the file descriptors in your filelist. See section 8.5, below, for instructions.

8.4. The listname.CATALOG system on non-VM systems (1.8c and later)

NOTE: If your list is running LISTSERV 1.8b, please refer to the List Owner's Manual for LISTSERV 1.8b for information regarding the file server. The information below is specific to 1.8c and will not work on pre-1.8c servers.

Please note that list-level file catalogs are not available in LISTSERV LITE; you must register files in the SITE.CATALOG file per the instructions in the installation guide.

LISTSERV version 1.8c introduces a new file archive registration system similar to (but differing in important respects from) the old VM FILELIST system. This new system is available on the VMS, unix, and Windows ports only. VM sites will continue to use the old FILELIST system indefinitely as it still offers more functionality than the new system.

This "sub-catalog" enhancement allows the LISTSERV administrator to delegate file management authority in a controlled and secure manner. Multiple list owners can be given the authority to maintain their own sub-catalog in a predefined directory.

From a list owner's point of view, the procedure works as follows:

1. Ask the LISTSERV administrator to create the sub-catalog for your list. You will need to provide the e-mail addresses of the person(s) who will be in charge of managing it ("catalog owners").

2. The catalog owners use the GET and PUT commands to update their catalog and register new files in their directory. Each file has the usual GET and PUT file access codes, allowing the catalog owners to further delegate the management of individual files to third parties ("file owners").

3. The file owners manage the files in question using the GET and PUT commands. Authorized users can retrieve the files using the GET command.

If your list is being migrated from VM to one of the non-VM versions of LISTSERV, please note that it is not necessary to create entries in your sub-catalog for WELCOME, FAREWELL and MAILTPL files. If entries for these files are not created, they simply do not appear in the output of an INDEX command. However, if desired, you can force them to appear by defining them in your sub-catalog.

8.4.1. Updating the sub-catalog

Once the sub-catalog is created, the catalog owner(s) can register new files using the following procedure (in this example, it will be assumed that the sub-catalog is called MY.CATALOG):

1. Send a GET MY.CATALOG command to LISTSERV (or, if the catalog is brand new, start from an empty file).

2. Register new file(s) in the catalog (see below).

3. Use the PUT MY.CATALOG PW=XXXXX command to store the updated catalog.

Alternatively, if the catalog owner has an account on the LISTSERV host system and write access to the directory associated with the sub-catalog, the file can be edited directly. Note however that, in that case, the LISTSERV-ISP quota system will be inoperative as it has no control over disk accesses which do not go through LISTSERV itself.

The format of sub-catalogs is similar to that of SITE.CATALOG:

```
MY.FILE          my.file                          ALL JOE@XYZ.COM
(1)              (2)                              (3) (4)
```

Notes:

(1) This defines the name of the file as seen by LISTSERV users. That is, the command to retrieve the file will be GET MY.FILE.

(2) This defines the name of the actual disk file where the contents of MY.FILE will be stored. Normally, you should specify the same as (1), or just an equal sign (LISTSERV will then substitute the name you provided for (1)). However, in some cases you may want to make a particular file available under multiple names. This can be done by registering multiple files (ie multiple values for (1)), and using the same (2) value every time.

(3) This file access code determines who can order the file through a GET command. The following file access codes are available:

```
ALL              universal access.
PRIVATE(xxx)     only members of the xxx list have access.
OWNER(xxx)       only the owners of the xxx list have access.
SERVICE(xxx)     only users in the service area of the xxx list have access.
NOTEBOOK(xxx)    same access as the archives of the xxx list.
user@host        the user in question is granted access.
```

Except for ALL, which must occur on its own, multiple file access code entries can be specified, separated by a comma with no intervening space. For instance:

```
MY.FILE C:\FILES\XYZ\MY.FILE JOE@XYZ.EDU,JACK@XYZ.EDU,PRIVATE(XYZ-L) CTL
```

defines a file that Joe, Jack and the subscribers of the XYZ-L list can order via the **GET** command, but that only the LISTSERV administrator can update.

(4) This file access code determines who can update the file with the PUT command. See note (3), above, for more information on FAC codes.

Note: (2) defaults to the value of (1), and (3) and (4) default to the GET and PUT access codes of the sub-catalog itself, respectively. So, in most cases a sub-catalog entry will be as simple as:

MY.FILE

Additionally, comment lines (starting with an asterisk) or blank lines can be interspersed with file definitions. These comments will be echoed when the sub-catalog is indexed (see below), in sequence with the file definitions. For instance, your catalog could read:

```
*
* Files for the XYZ sub-project
*
XYZ.AGENDA
XYZ.BUDGET
XYZ.PROPOSAL-1
XYZ.PROPOSAL-2
```

8.4.2. Indexing the sub-catalog

If MY.CATALOG is defined as:

```
MY.CATALOG     /home/lists/xyz/my.catalog   xxx JOE@XYZ.COM
```

then any user who matches the 'xxx' file access code is authorized to issue an **INDEX MY** command to get a formatted version of the catalog. For compatibility with older versions of LISTSERV, GET MY.FILELIST will produce the same results. If there is a mailing list called MY, a list of the archive files will be appended automatically.

8.5. Storing files on the host machine

To store a file on any LISTSERV host, first ensure that it has been registered with an entry in a filelist or catalog. Then follow these instructions:

1. Be sure that you have defined a "personal password" to LISTSERV with the **PW ADD** command before you **PUT** the new or edited file. If you have done this but can't remember the password, send a **PW RESET** command to LISTSERV, then a new **PW ADD** command.

2. Edit your file and save it. Add a single line at the top of the file as follows (square brackets indicate optional parameters):

```
PUT filename extension [filelist|catalogname] PW=XXXXXXXX
```

(This line will not appear to people who GET the file from LISTSERV.) Replace **XXXXXXXX** with your personal password. If you specify the filelist or catalog name, do not put the square brackets around the name.

There are a couple of issues that need to be noted here:

- If the file you are going to store is registered in the sitewide catalog or filelist, do not specify the name of the catalog or filelist.

- If the file you are going to store is registered in a sub-catalog or filelist other than the sitewide one, you may have to specify the name of the sub-catalog or filelist in order to be able to store the file. This is because it is entirely possible that two lower-level filelists or catalogs may have files registered with the same name (for instance, README TXT). If LISTSERV has two sub-catalogs registered (for instance, MYLIST CATALOG and HISLIST CATALOG) that both have a file called README TXT registered, then a PUT README TXT command will tell LISTSERV to try and store the file in the first catalog it comes to in the hierarchy. If MYLIST CATALOG is registered before HISLIST CATALOG in SITE CATALOG, LISTSERV will try to store the file as if it belonged to MYLIST (which we assume is what you want). However, if HISLIST CATALOG is registered before MYLIST CATALOG (and many sites like to keep things in alphabetical order, so this is a most likely scenario), LISTSERV will try to store the file as if it belonged to HISLIST, and you will get an error stating that you aren't allowed to store the file.

1. Be sure that the file has been registered with an entry in a filelist or the site catalog.

2. Send the mail message to LISTSERV.

8.6. Deleting files from the host machine

To delete a registered file on any LISTSERV host:

1. Be sure that you have defined a "personal password" to LISTSERV with the **PW ADD** command before you **PUT** the delete job. If you have done this but can't remember the password, send a **PW RESET** command to LISTSERV, then a new **PW ADD** command.

2. Create a new mail message addressed to LISTSERV. Add a single line at the top of the message as follows:

PUT *filename extension* **[***filelist***|***catalogname***] PW=XXXXXXXX**

(Replace **XXXXXXXX** with your personal password.) The same issues noted in 8.5 regarding the filelist/catalog name are operative here.

3. Send the mail message to LISTSERV.

4. LISTSERV will tell you that the file has been successfully deleted.

5. For VM Systems ONLY: **GET** the *listname* **FILELIST** for your list and delete the line for the file you've just deleted. **PUT** the *listname* **FILELIST** back on the server.

6. For Workstation and PC Systems ONLY: Get the *listname***.CATALOG** for your list and delete the line for the file you've just deleted. PUT the *listname***.CATALOG** back on the server. Note that this is not necessarily required since under non-VM, if the physical file does not exist, LISTSERV will not include it in the output of an INDEX command. This is primarily a housekeeping measure.

Note that #5 and #6 are not necessary when you are deleting notebook archives. LISTSERV generates the notebook archives index "on the fly" when needed.

8.7. Automatic File Distribution (AFD) and File Update Information (FUI)

If your list is running on LISTSERV under unix, Windows, or VMS, please skip the rest of this section as it does not currently pertain in any way to your implementation of LISTSERV.

AFD and FUI have not yet been ported to the workstation and PC environments. However, this feature is supported on VM and will be supported in the near future on the other platforms.

These two features are similar in their command syntax, but do different things. AFD provides a method whereby users may subscribe to specific files, which will be sent to them any time the files are updated. For instance, if you have a FAQ file that is updated monthly, a user could send an AFD subscription to that FAQ file and LISTSERV would send it to the user every time you updated and stored the FAQ.

FUI, on the other hand, is a method whereby a user subscribes to a file but receives only a notification that the file has been updated. The user can then GET the file at his own discretion.

AFD and FUI can be password-protected to protect users from network hackers who might forge mail from the user subscribing him to large or frequently-updated files. If a password is not provided in an AFD or FUI ADD command, LISTSERV warns the user that it would be a good idea to password protect the subscription.

8.8. File "Packages"

This feature is available for VM (all versions) and non-VM (beginning with 1.8d).

You can define a group of files as a "package" that can be retrieved by users with a single **GET** command. First, ensure that all the files in the package are defined in the appropriate filelist and stored on the server as detailed above.

Next, create a file descriptor in the appropriate filelist or catalog for a file called *filename* **$PACKAGE** (or *filename.***$PACKAGE** for non-VM), where *filename* is the name you have chosen for the group of files. Be sure that the filetype or extension is $PACKAGE, with a leading $ sign, and store your filelist.

Now create the actual *filename* **$PACKAGE** file. At the top of the file you can insert comment lines beginning with asterisks, e.g.:

```
* MYLIST $PACKAGE
* Packing list for MYLIST PACKAGE
*
* You can make other comments here, such as
* the contact email address.
*
* filename filetype filelist
*=========================
```

Following these comment lines, you insert lines for each of the files contained in the package. There are two ways to format entries in your $PACKAGE file:

- A "compatibility" mode that works on all platforms, and which is identical to the original method used on VM (and which VM servers still must use). In the compatibility mode the basic format for the entries is

  filename filetype filelist <optional_comments>

  for example,

  ```
  MYLIST    $PACKAGE MYLIST The packing list
  INTEREST FILE      MYLIST Interest groups
  NETIQUET FILE      MYLIST How to behave
  ANOTHER  FILE      MYLIST No comment
  ```

- In the second (new) mode for non-VM servers only, the entries are formatted like this:

  filename.extension <optional_comments>

  for example,

  ```
  MYLIST.$PACKAGE The packing list
  INTEREST.FILE    Interest groups
  NETIQUET.FILE    How to behave
  ANOTHER.FILE     No comment
  ```

Note that anything that is not the name of a file in the package must be commented out

with an asterisk in the leftmost column of the line. It is possible to create a package file without any comment lines at all, but this is not preferable in practice. Often users will get the package file itself just to see what is in it. You should include a reference to the package file itself so that the user will get a copy of the "packing list" to check against the files he receives from LISTSERV.

The final step is to send the package file to LISTSERV like any other file.

Now users can do one of two things:

1. They may get the entire package of files sent to them by sending LISTSERV the command **GET** *filename* **PACKAGE** (without the $ sign); or

2. They may request that LISTSERV send only the package file itself by sending LISTSERV the command **GET** *filename* **$PACKAGE** (with the $ sign).

Packages may be subscribed to with the **AFD** and **FUI** commands (VM only).

8.9. Where to find more information on File Archives

Guides that refer to File Archive setup and maintenance **for VM systems only** are referenced in Appendix D, *Related Documentation and Support*. LISTSERV maintainers can also find more information in the *Site Manager's Operations Manual for LISTSERV*.

9. Creating and Editing LISTSERV's Mail and Web Templates

9.1. What LISTSERV uses templates for

Templates are used to generate some of the mail LISTSERV sends to users in response to commands it receives. Among these are the "You are now subscribed . . ." message, the message sent to users when LISTSERV cannot find a subscription for them in a specified list, and others. Note that certain administrative mail (for instance, the response to the STATS and RELEASE commands) is hard-coded into LISTSERV and cannot be changed.

Other templates are used to generate the HTML code used by the web archive and administration interfaces.

A word about nomenclature: When we talk about "templates" we are talking about "files that contain one or more template forms", in other words, files like DEFAULT MAILTPL or DEFAULT WWWTPL. A "template form" is an individual section of a template which begins with a title line (three ">" symbols followed by a space, the name of the template form, and (optionally) a short description of the template, which for some template forms is also used as the subject of the mail LISTSERV constructs with the template form), followed by one or more lines of copy and/or imbedded commands, and ends at the next title line or the end of the file, whichever is reached first. A template may contain one or more template forms.

9.2. The default template files and how to get copies

LISTSERV stores its default mail template forms in a file called DEFAULT.MAILTPL, which can be requested by list owners from LISTSERV with the GET command, just like any other file. Note that DEFAULT MAILTPL contains some (but not all) of the web interface template forms.

LISTSERV stores the rest of its default web interface template forms in a file called DEFAULT WWWTPL, which can be retrieved in a manner identical to that for DEFAULT MAILTPL.

Under 1.8d and following, all template forms may be edited using the web administration interface described in chapter 11. Edited template forms are placed in template files that will not be overwritten by software upgrades.

9.3. Mail template format and embedded formatting commands

Each individual template form starts with a form name and subject line, such as:

```
>>> EXAMPLE1 This is the subject line
```

Please note carefully the following instructions for the form name and subject line:

- The template form starts with the line containing the form name and subject, and ends with the next line starting with '>>>', or at the end of the file.
- The subject line may contain substitutions (such as "&LISTNAME: &WHOM requested to join").
- Ensure that there is a blank space (ASCII 0x20) between '>>>' and the name of the form, or LISTSERV will not recognize the form.

- Also note that the names of the template forms must be typed in UPPER CASE.

A template form contains text and, optionally, formatting/editing commands, which start with a period in column 1. All other lines are treated as normal text: sequences starting with an **&** sign are substituted, then lines are joined together to form a paragraph, which is finally formatted like with any non-WYSIWYG text processor. You can suspend formatting with **.FO OFF** and resume it with **.FO ON**; when formatting is suspended, LISTSERV no longer joins lines to form a paragraph, but simply writes one line of text to the message for each line read from the template form. This makes it possible to include tables or a text-mode logo, but can create seriously imbalanced text if substitutions are used. For instance, a typical **&WHOM** substitution can range from a dozen characters to 60 or more, even though it only takes up 5 characters on your screen when you enter it.

The following substitutions are always available:

&DATE	Long-style date (04 Jan 1998)
&TIME	hh:mm:ss
&WEEKDAY	Three-letter day of the week, in English

&MYNAMES The substitution you will use most of the time when you need to refer to LISTSERV. For Internet-only or BITNET-only servers, this will display LISTSERV's only e-mail address. For servers with both Internet and BITNET connectivity, it will say "LISTSERV@*hostname* (or LISTSERV@*nodeid*.BITNET)".

&MYSELF LISTSERV's address, in the form LISTSERV@XYZ.EDU or, if no Internet hostname is available, LISTSERV@XYZVM1.BITNET.

&MYNODE LISTSERV's BITNET nodeid, without the '**.BITNET**', or its Internet hostname if no NJE address is available.

&MYHOST LISTSERV's Internet hostname or, if none is available, its NJE address (with '**.BITNET**').

&MBX(*addr*) Looks up the specified address in LISTSERV's signup file and displays "name <addr>" if a name is available, or just the original address otherwise. This is typically used to give the name of the command originator or target, along with his e-mail address: **&MBX(&WHOM)** or **&MBX(&INVOKER)**. Please note however that **&WHOM** and **&INVOKER** are not always available in every template.

&RELEASE LISTSERV's release number (e.g., "1.8d").

&OSTYPE The operating system under which LISTSERV is running, e.g., VM/VMS/unix/Windows.

&OSNAME The full operating system name including the version number, e.g., "VM/ESA 1.2.3", "Windows NT 3.51", "Linux 2.0.27", "SunOS 5.4", etc.

&HARDWARE The type of machine LISTSERV is running on, e.g., "Pentium (128M)".

The following substitutions are also available for templates related to mailing lists:

&LISTNAME Either the short or long name of the list based on the value of "List-Address=" and/or its system default. By default the long ("List-ID=")

	name is used if present.
&TITLE	Title of the list, or empty string.

&KWD(*kwd*)   Value of the specified keyword for the list. You do not need to specify the name of the list - it is implicit. You need not put quotes around the keyword names either, although quotes will be accepted if present. Optionally, you can specify a second numeric argument to extract just one of the terms of a list header keyword; for instance, if the list header contains "Notebook= Yes,L1,Monthly,Private", **&KWD(NOTEBOOK,4)** has the value "Private". A third argument, also optional, specifies the default value for the keyword in case it was not initialized. It is meant to be used for conditional formatting in the default templates and list owners should not worry about it.

&LITE   (1.8c and following) Has the value 1 when running the LISTSERV Lite product, and 0 otherwise. This variable can be used to write generic templates that account for the differences between the two products.

&ISODATE   (1.8c and following) Returns today's date in ISO format, i.e., yyyy-mm-dd.

&DAYSEQ(*n*)   (1.8c and following) Used to create FAQ templates with rotating topics. May also be used to create bottom banners with rotating text (e.g., for lists with multiple commercial sponsors who get "ad space" in the banner on a rotating basis).

In addition, many template forms have their own specific substitutions, meaningful only in their specific context. For instance, a message informing a user that he was added to a mailing list may have an **&INVOKER** substitution for the address of the person who issued the ADD command. This is not meaningful for a template form intended to inform a user that he must confirm his subscription to a list within 10 days, so it is not generally available. If you attempt to use a substitution which is not available, the template processor writes an error message to the mail message it is generating, but sends it anyway, in the hope that the recipient will be able to figure out the meaning of the message in spite of the error. If you need to include a sentence with an ampersand character, you will have to double it to bypass the substitution process, as in "**XYZ &&co.**"

Starting with 1.8c, the mail template processor supports HTML-like variable closure, in addition to the traditional LISTSERV closure (both methods are supported concurrently; there is no need to select one over the other). For example:

Traditional: **For more information, please send mail to &EMAIL or call &PHONE.**

HTML: **For more information, please send mail to &EMAIL; or call &PHONE;.**

Previously, HTML writers who used HTML closure conventions would not get the expected results. This change makes it easier for webmasters to get the desired results the first time.

Any line starting with a period in column 1 is processed as a formatting command. Note that neither substitutions nor formatting commands are case sensitive. Here is a list of the formatting commands list owners may need to use:

`.*`	Comment: anything on this line is simply ignored. This is useful for recording changes to template files when there are multiple owners. Just add a comment line with the date and your initials every time you make a change, for the benefit of the other owners.
`.FO OFF`	Turns off formatting: one template line = one line in the final message. You can resume formatting with `.FO ON` or `.FO RAGGed`. (`.FO RAGGed` requires LISTSERV 1.8e-2002a or later, that is, build date of 31 October 2002 or later)
`.FO RAGGed`	Changes right-justified text formatting to left justified text formatting. You can resume right-justified formatting with `.FO ON`. (`.FO RAGGed` requires LISTSERV 1.8e-2002a or later, that is, build date of 31 October 2002 or later)
`.CE text`	Centers the text you specify (just the text you typed on the same line as the `.CE` command). This can be useful to highlight the syntax of a command.
`.RE OWNERS`	Adds a 'Reply-To:' field pointing to the list owners in the header of the generated message. Use this command when you think users are likely to want to reply with a question. You can also use `.RE POSTMASTER` to direct replies to the LISTSERV administrator, if this is more appropriate.
`.CC OFF`	Removes all "cc:" message recipients, if any. You can also add message recipients by specifying a series of e-mail addresses after the `.CC` statement, as in `.CC JOE@XYZ.EDU`. PC mail users should note that in this context "cc:" is a RFC822 term that stands for "carbon copy". RFC822 messages may have "cc:" recipients in addition to their "primary" recipients. There is no real technical difference between the two, the "cc:" indicator just denotes a message that is being sent for your information. Some administrative messages sent to list owners are copied to the user for their information, and vice-versa; this behavior can be disabled by adding a `.CC OFF` statement to the template.
`.TO`	Replaces the default recipients of a message with the value specified. For instance, if you use the ADDREQ1 template form to send new subscribers a questionnaire, application form or similar material, you will need to add a '.TO &WHOM' instruction to your modified template form, as by default the user will not receive a copy.
`.QQ`	Cancels the message. LISTSERV stops reading the template form and does not send anything. This is useful if you want to completely remove a particular message; note however that this can be confusing with certain commands, as LISTSERV may say "Notification is being sent to the list owners" when in fact nothing will be sent because of the `.QQ` command in the template form.
`.QU`	(Starting with 1.8e) Ends processing of the current template as if you had reached the end, but *without* cancelling the message. The main purpose is to avoid multi-level nested .BB/.EB conditional blocks (see below) that are hard to keep track of.

A number of more advanced commands are available to list owners with more sophisticated needs and some programming experience. If you encounter one of these

commands in a template, you will probably want to leave it alone.

.IM name   Imbeds (inserts) another template form at this point in the message. This is used to avoid duplicating large pieces of text which are mostly identical, such as the templates for "you have been added to list X by Y" and "your subscription to list X has been accepted".

As noted above, LISTSERV will not pick up an "imbedded" template form from $SITE$.MAILTPL. If you wish to include an "imbedded" template form (e.g., **$SIGNUP**) in $SITE$.MAILTPL, you must also include the template form that calls it with the **.im** command.

.DD ddname   Copies the contents of the specified DD into the message. This is meaningful only if a DD has been set up by LISTSERV for this purpose. As a rule of thumb, you should either leave these statements unchanged or remove them.

.BB cond   Begin conditional block. The boolean expression following the keyword is evaluated and, if false, all the text between the **.BB** and **.EB** delimiters is skipped. Conditional blocks nest to an arbitrary depth. The expression evaluator is recursive but not very sophisticated; the restriction you are most likely to encounter is that all sub-expressions have to be enclosed in parentheses if you are using boolean operators. That is, "**.BB &X = 3**" is valid but "**.BB &X = 3 and &Y = 4**" is not. String literals do not require quoting unless they contain blanks, but quotes are accepted if supplied. Comparison operators are **= <> ^= IN** and **NOT IN** (the last two look for a word in a blank-separated list of options, such as a keyword value). These operators are not case-sensitive; **==** and **^==** are available when case must be respected. Boolean operators are **AND** and **OR**. Note that a conditional block must be contained on one physical line and may not wrap, so be careful when sending MAILTPL files back to LISTSERV that you do not accidentally wrap long **.BB** lines.

Starting with LISTSERV 1.8d the operators **=*** and **^=*** are available to perform wildcard matches in conditional blocks. For instance **JOHN_DOE@UNIX.EXAMPLE.COM =* J*DOE@*EXAMPLE.COM** is a true statement. The wildcard specification is on the right-hand side whereas the actual text (or variable) you are evaluating is on the left.

.EB   End conditional block (see **.BB**).

.QU   Stop (, ie, QUit) processing of the current template as if you had reached the end, but without cancelling the message. The main purpose is to avoid multi-level nested **.BB/.EB** conditional blocks that are hard to keep track of. Available in 1.8e and following.

.SE var text   Defines or redefines a substitution variable. This is convenient for storing temporary (text) expression results which need to be used several times. Even standard variables such as **&LISTNAME** can be redefined - at your own risk. You must enclose the text expression in single quotes if you want leading or trailing blanks.

.CS text   Define a (non standard) character set for the template in question, i.e.,

```
.CS ISO-8559-7
```

This setting is ignored if the template does not actually contain special characters (for instance, if the template is written in 7-bit ASCII). Otherwise the appropriate headers are created for the message in question when it is sent out, i.e.,

```
Content-Type: text/plain; charset=ISO-8859-7
Content-Transfer-Encoding: quoted-printable
```

.TY text   Types one line of text on the LISTSERV console log. This can be useful to the LISTSERV maintainer for debugging, and also to record information in the console log.

.ASIS text   Tells LISTSERV to leave the text immediately following the .ASIS directive alone, ie, don't convert "<" and ">" characters into HTML < and > when creating pages. This is specifically for use in HTML templates where it is important not to convert parts of a URL reference. For instance,

```
.ASIS Click <a href="http://some.host.com/some-doc.html">here</a>.
```

As with the **.CE** directive, the text you intend to affect with the **.ASIS** directive must not wrap. The **.ASIS** directive will only work on text it finds on the same physical line into which it is coded.

9.3.1. 8-bit characters in templates

Starting with 1.8d, if you include 8-bit characters (e.g., accented or national language characters) in templates, LISTSERV will automatically encode the templates on-the-fly in MIME quoted-printable encoding. While there is no guarantee that every mail program will be able to properly display 8-bit characters, those mail programs that understand MIME quoted-printable encoding should have no trouble doing so.

9.4. Creating and editing a <listname>.MAILTPL file for a list

Please note that list-level mail templates are not available in LISTSERV Lite.

Make a copy of DEFAULT.MAILTPL on your local machine and name it *listname*.MAILTPL.[6] Keep the original DEFAULT.MAILTPL around in case you make a mistake and need to start over.

At this point, you could theoretically store the *listname*.MAILTPL back on the LISTSERV host. However, without making any changes that would be somewhat pointless. At the very least you should edit the INFO template form before storing the template. Note also that you need only store the sections of the template that you have changed. For instance, if you edit the INFO template form but leave the rest of the template untouched, you can delete the rest of the template and store the INFO template form alone as *listname*.MAILTPL. The benefit to this approach is that any administrative changes to the rest of the default template are automatically applicable to your list as soon as they are

---

[6] If your local machine is running MS-DOS and/or Windows 3.x, obviously this will not work--you will have to conform to the 8.3 naming convention. Probably the best thing to do in this case is simply name the file *listname*.mai, then rename it when you upload it to a mainframe or network workstation account (or use the web administration interface described in chapter 11, instead).

made, rather than requiring that you edit your mail template individually to reflect such changes. L-Soft recommends that this approach be followed as the default.

Under LISTSERV 1.8d and following it is not necessary to do the GET and PUT; you can edit individual template forms by using the web administration interface (described in chapter 11) instead.

9.4.1. The INFO template form

The first section of DEFAULT.MAILTPL is called the INFO template form, and it is LISTSERV's response to the command **INFO** *listname*. By default, it contains the following:

```
>>> INFO Information about the &LISTNAME list
There is no information file for the &LISTNAME list. Here is a copy of
the list "header", which usually contains a short description of the
purpose of the list, although its main purpose is to define various
list configuration options, also called
"keywords". If you have any question about the &LISTNAME list, write to
the list owners at the generic address:

.ce &LISTNAME-Request@&MYHOST

.dd &LISTHDR
```

Figure 9.1. The default contents of the INFO template form of DEFAULT.MAILTPL.

Note the replaceable parameters **&LISTNAME** and **&MYHOST**. Don't change **&MYHOST**; LISTSERV replaces it with the correct value for the name of the host site. **&LISTNAME** automatically inserts the name of the list. It's probably best to use **&LISTNAME** to refer to the list throughout the document rather than to replace it with something like "MYLIST-L". This ensures that the template form will be consistent with the default and will be simpler to debug should a problem arise. Also, in the event the name of the list changes, it will be unnecessary to edit the template form (although it would have to be renamed to match the new name of the list, of course).

Should it be desirable to replace the default INFO template form with information about the list, it is probably best to remove the **.dd &LISTHDR** line. This line instructs LISTSERV to read in the header of the list and add it to the response in lieu of any other data about the list. Many list owners add descriptive comment lines to their list headers, thus this default.

Here is a minimally-edited sample INFO template form for a list called MONKEYS:[7]

```
>>> INFO Information about the &LISTNAME list
&LISTNAME is an open, unmoderated discussion list featuring
monkeys.  Things such as how to care for a pet monkey, monkey
diseases, monkey lore, endangered species of monkeys, and
monkey psychology are likely to be discussed.  The list is
NOT intended for discussion of Darwinism and/or theories of
evolution.

If you have any question about the &LISTNAME list, write to
the list owners at the generic address:

.ce &LISTNAME-Request@&MYHOST
```

Figure 9.2. Sample edited INFO template form.

---

[7] Thanks to Marty Hoag of NEW-LIST.

9.4.2. Other useful template forms

Traditionally, message templates have contained the text of "long" administrative messages, such as messages informing subscribers that they have been removed from a mailing list. These notices were sent unconditionally, as a separate message. Since version 1.8b, the template processor has supported "linear" messages, which are sent as a normal command reply and allow the list owner to modify the replies from selected commands, and "optional" messages, which are only sent if a template for this action has been specifically provided by the list owner.

In a linear message, most special instructions are ignored. This is because the contents of the template form are just a few lines out of a larger message that is being prepared by LISTSERV to contain the reply to the user's command(s). For instance, you do not have any control over the "Reply-To:" field of the message, because the message in question is shared with other commands and, in fact, may not be a mail message at all but an interactive message to the user's terminal, a GUI request, etc. Generally speaking, with a linear message you are providing the TEXT of the reply to be shown to the user, but you do not have any control over the methods used for delivering this information.

Here is a list of all of the template forms (other than `INFO`, described above) available in DEFAULT.MAILTPL, in the order in which they appear and with a short description for each. Linear and optional template forms are noted where applicable.

- **`MOVE1:`** Usually active only for peered lists. This message is sent to the subscriber when the list owner or LISTSERV maintainer changes which peer the subscriber receives his or her mail from.

- **`SIGNOFF1`**: a notification to the list owner that someone has unsubscribed from the list. Whether or not the list owner receives this notification is controlled by the "Notify=" list header keyword.

- **`SIGNOFF2`**: this message is sent to any user who attempts to unsubscribe from a list to which he or she is not subscribed under the userid from which the unsubscribe command has been sent. For instance, joe@unix1.somehost.com may be subscribed to list MYLIST-L. If his Pine client is set so that his mail comes from his root domain (e.g., joe@somehost.com), he will get this message if he tries to unsubscribe from MYLIST-L.

- **`DELETE1`**: the message sent when a list owner or the LISTSERV maintainer deletes a user from a list. You can suppress the sending of this message by prepending "QUIET" to your "DELETE" command.

- **`AUTODEL1`**: this is the message that is sent to users who are deleted by the delivery error monitor. You can customize it to fit your needs, or suppress it for your list by simply redefining it in the 'listname.MAILTPL' and using the **`.QQ`** instruction:

    ```
    >>> AUTODEL1 This message is not wanted for our list
    .QQ
    ```

    Note that L-Soft does not generally recommend suppressing this message, as it may indicate a serious problem for the deleted subscriber.

- **`ADD1`**: the message sent when a list owner or a LISTSERV maintainer manually adds

a subscriber to a list.

- **ADD2**: the message sent to the list owner(s) when someone subscribes to their list. As with **SIGNOFF1**, whether or not the list owner(s) receive this message is controlled by the "Notify=" list header keyword.

- **ADDREQ1**: this message is sent to the list owner when a user requests to join a list with "Subscription= By_owner". Only the list owner is sent a copy of the **ADDREQ1** message. If you use this template form to send new subscribers a questionnaire, application form or similar material, you will need to add a '**.TO &WHOM**' instruction to your modified template form, as by default the user does not receive a copy.

- **SETINFO**: the message sent to the subscriber when the list owner or LISTSERV maintainer changes their personal subscription options. Can be suppressed by the invoker with the use of the "QUIET" command modifier.

- **CHANGE1**: the message sent when a list owner or LISTSERV maintainer uses the CHANGE command to change a subcriber's address.

- **ADDMOD2**: the message sent to the subscriber when the list owner or LISTSERV maintainer changes the subscriber's "real name" field in the SIGNUP database.

- **ADDPW1:** the message sent to the user when a LISTSERV maintainer adds a personal password for that user. List owners should not change this template form.

- **ADDPW2:** an informational message sent to the LISTSERV maintainer when a user adds or changes his password, but only if an LSV$PW exit has been enabled to do so. Most installations will never use this template form, but it should not be deleted from DEFAULT.MAILTPL in any case. List owners should not change this template form.

- **ADDPW3:** an information message sent to the LISTSERV maintainer when a user tries to add or change his password, but only if an LSV$PW exit has been enabled to do so. Most installations will never use this template form, but it should not be deleted from DEFAULT.MAILTPL in any case. List owners should not change this template form.

- **DELPW:** the message sent to the user when a LISTSERV maintainer deletes that user's personal password. List owners should not change this template form.

- **RENEW1:** this message is sent to subscribers whose subscriptions are due for renewal (see the Renewal= list header keyword for more information).

- **RENEW2:** this message is sent to subscribers who did not renew their subscriptions within the grace period after being notified that their subscription was due for renewal.

- **SIGNUP1:** the basic "Your subscription request has been accepted" message.

- **$SIGNUP:** a template form included with **SIGNUP1** and **ADD1** (assuming that **SIGNUP1** and **ADD1** templates include an ".im $SIGNUP" line, which by default they do) which gives the subscriber a basic outline of how to use the list, how various options are set, and where to get more information on using LISTSERV. You can use this template in lieu of a WELCOME file for your list if you don't want two messages

to go to the user at subscription time.

- **SUB_CLOSED** (linear): this is the message that is sent to a subscriber attempting to join a list with "Subscription= Closed". The default is "Sorry, the **&LISTNAME** list is closed. Contact the list owner (**&OWNER**) for more information."

- **SUB_OWNER** (linear): this message is sent to a subscriber attempting to join a list with "Subscription= By_owner". The default is "Your request to join the **&LISTNAME** list has been forwarded to the list owner for approval. If you have any question about the list, you can reach the list owner at **&OWNER**." Because this is a linear template form (see above), it is not the best place to put long questionnaires, application forms, terms and conditions, or other material that the subscriber should be required to review prior to joining the list. See the "Tips" section below.

- **POST_EDITOR** (linear): this is the message LISTSERV sends to people attempting to post to the list, if it is moderated. The default is "Your **&MESSAGE** has been submitted to the moderator of the **&LISTNAME** list: **&MBX(&MODERATOR)**."

- **REQACK1**: this message is sent automatically in reply to any message sent to the xxx-request address. The message acknowledges receipt, explains the difference between the LISTSERV and xxx-request addresses, and contains instructions for joining and leaving the list. To suppress this message for your list, simply redefine it in the '*listname*.MAILTPL' and use the **.QQ** instruction:

  ```
  >>> REQACK1 This message is not wanted for our list
  .QQ
  ```

- **REQNAK1**: (1.8e and following): this message is sent automatically in reply to any message sent to the special ALL-REQUEST address by a user who is not authorized to post to that address.

- **CONFIRM1**: The template form sent whenever an "OK" confirmation is required.

- **WWW_INDEX**: this template form is used by sites which have implemented LISTSERV's WWW archive interface. It includes the HTML code for the main archive access screen. You probably should leave this alone unless you know exactly what you are doing.

- **PROBE1**: this template form is sent as part of LISTSERV's new bounce processing feature if this feature is activated for your list. The desired response from the user is to discard the message and do nothing. See chapter 4.6.2 of the *List Owner's Manual* or chapter 13.5 of the *Site Manager's Operations Manual* for details on the "Probe" option.

- **PROBE2**: If the mail containing the **PROBE1** message bounces, this template form is sent along with a copy of the bouncing mail. See chapter 4.6.2 of the *List Owner's Manual* or chapter 13.5 of the *Site Manager's Operations Manual* for details on the "Probe" option. (If you have **Auto-Delete= ...,Delay(0)**, **PROBE2** is *not* sent, rather the bouncing user is deleted immediately.)

Several template forms for the WWW archive interface follow **PROBE1**. L-Soft does not recommend that list owners modify these templates. Please contact your LISTSERV maintainer for details.

- **BAD_CONTENT** (linear, 1.8e): If a posting to a list violates one of the content rules defined in the optional **CONTENT_FILTER** mail template form (see 7.18, above) and is rejected, this template form is sent back to the poster.

- **BAD_ATTACHMENT** (linear): If a posting to a list contains an attachment of a type not allowed by the "Attachments=" setting for the list, this template form is sent back to the poster.

- **DIST_VIRUS** (linear, 1.8e): Assuming that the anti-virus feature introduced in 1.8e is enabled, if LISTSERV detects a virus in a DISTRIBUTE job, the message is returned to sender along with this template text.

- **SIZELIM_EXCEEDED** (linear, 1.8e): This template form is used if a posting to a list exceeds the limit set by the Sizelim= list header keyword.

The following are template forms that can be defined, but which are not present in DEFAULT.MAILTPL. Note carefully that these are templates which are defined as part of the *listname***.MAILTPL** file; they are *not* stored as separate files. If you attempt to store a file such as *listname***.BOTTOM_BANNER**, your PUT operation will be rejected.

- **POSTACK1** (optional): when present, this message is sent in reply to any message posted to the list. This is very useful for creating "infobots", or just for returning a standard acknowledgement to contributors. The **&SUBJECT** variable contains the subject of the original message, and naturally the usual substitutions (**&LISTNAME**, **&DATE**, **&TIME**) are available.

- **TOP_BANNER**, **BOTTOM_BANNER** (optional): when these template forms are present, their contents are automatically inserted at the top (respectively bottom) of each and every message posted to the list. Typically, the top banner would be used for a copyright or short legal warning which absolutely has to be seen by each and every reader. The bottom banner could contain instructions for signing off the list, a disclaimer, an acknowledgement of a sponsor's contribution, a "tip of the week", etc.

> **Documented Restriction:** *The use in banners of substitutions which do not yield a constant result (e.g.,* **&TIME***) will defeat the duplicate mail detection part of LISTSERV's loop-checking heuristics in any case where a subscriber is forwarding all mail back to the list. L-Soft advises that such substitutions never be used in a* **TOP_BANNER** *or* **BOTTOM_BANNER***.*

Prior to 1.8e, for digests, note that the **BOTTOM_BANNER** is printed only once, at the top of the digest, directly following the table of contents. This avoids having the banner repeat after every message in the digest. The default behavior can be overridden if preferred by adding the "BOTTOM_BANNER" parameter to the **Digest=** list header keyword.

In LISTSERV 1.8e, a major change to the way LISTSERV attaches banner messages to postings changed the digest behavior described above. The change lets LISTSERV correctly insert banners in MIME messages, and solves certain quoted-printable error and message rejection problems observed in the previous versions. However, LISTSERV no longer attempts to remove bottom banners from individual messages in digests. In 1.8d, new banners were not actually inserted into the digest, so they appeared to have been successfully removed. While banner removal did work with single-part, unencoded messages, this was the only case when it worked. In 1.8e banners are inserted in a different way which precludes

attempting to remove existing banners when the digest is generated.

Note that the **Digest=** keyword's "BOTTOM_BANNER" override parameter still works in 1.8e, insofar as it will prevent the banner from being printed at the top of the digest.

- **TOP_BANNER_HTML**, **BOTTOM_BANNER_HTML** (optional, 1.8e and following): When these template forms are present, they will be used "as is" for HTML message parts. If absent, the regular banner is used for HTML, probably with less than 100% satisfaction.

> **Documented Restriction:** *The use in banners of substitutions which do not yield a constant result (e.g.,* **&TIME***) will defeat the duplicate mail detection part of LISTSERV's loop-checking heuristics in any case where a subscriber is forwarding all mail back to the list. L-Soft advises that such substitutions never be used in a* **TOP_BANNER_HTML** *or* **BOTTOM_BANNER_HTML**.

- **CONTENT_FILTER** (optional, 1.8e and following): When present, provides LISTSERV with a ruleset for message content filtering that can be configured at the list level. See chapter 2.16, above, for more information on how to use content filtering.

9.4.3. Tips for using templates

- Many list owners require prospective subscribers to fill in a little questionnaire before being added to the list, or to explicitly state that they have read the list charter and agree to follow all rules or be removed from the list. The most convenient method, for both list owner and subscriber, is to have the SUBSCRIBE command return a copy of the questionnaire (or list charter, etc), and not forward the request to the owner. The user answers the questions and returns them directly to the list owner, who then adds the subscriber manually. Naturally, it is more convenient for the user if this information arrives in a separate message, with a 'Reply-To:' field pointing to the list owner's address. Thus, you should not use the **SUB_OWNER** template form for this purpose, because it is a linear template form and does not give you any control over the 'Reply-To:' field. The **SUB_OWNER** template form could be modified to read "A copy of the list charter is being sent to you, please read it carefully and follow the instructions to confirm your acceptance of our terms and conditions." The list charter would then be sent separately, through the **ADDREQ1** template form. You would use the **.RE OWNERS** command to instruct LISTSERV to point the 'Reply-To:' field to the list owners, and **.TO &WHOM** to change the destination from list owner to subscriber. If you want to receive a copy of the message, you can use **.TO &WHOM cc: xxx@yyy**.

- When writing template forms, it is a good idea to use substitutions (**&XXXX**) for information which may change in the future. In particular, it is not uncommon for lists to have to be moved from one host to another, and this will be a lot easier if the template forms use substitutions for the list address and list host. The **&LISTADDR** substitution translates the full address of the list (XYZ-L@XYZ.COM), whereas **&LISTNAME** is just the name (XYZ-L). For references to the server and host, use **&MYHOST** for the Internet hostname, **&MYSELF** for the server address (normally **LISTSERV@&MYHOST**), and **&OWNER** for the xxx-request mailbox address. These substitutions are "universal" and can be used in all template forms. For instance, if you decide to make a bottom banner with instructions for leaving the list, the text could read: "To leave the list, send a SIGNOFF **&LISTNAME** command to **&MYSELF** or, if you experience difficulties, write to **&OWNER**."

9.5. Storing the <listname>.MAILTPL file on the host machine

The procedure differs slightly on VM systems, but the following will work for unix, VMS and Windows systems:

1. Get a copy of DEFAULT.MAILTPL and edit it.

2. Be sure that you have defined a "personal password" to LISTSERV with the `PW ADD` command before you `PUT` the template file. If you have done this but can't remember the password, send a `PW RESET` command to LISTSERV, then a new `PW ADD` command.
.
3. Send the file to LISTSERV with a `PUT listname MAILTPL PW=XXXXXXXX` command at the top of the file, just as if you were storing the list itself. Replace `XXXXXXXX` with your personal password.

The variation for VM systems is that the LISTSERV maintainer will have to create a fileid for the file before you can `PUT` it on the server. Contact the LISTSERV maintainer before trying to store your template file.

. Other template files: DIGEST-H and INDEX-H

Two other template files that are available pertain to the automatic digestification feature. You may create and store files called *listname* `DIGEST-H` and *listname* `INDEX-H`. These files define custom digest headers and custom index headers, respectively. The DIGEST-H and INDEX-H files are plain text files, like the WELCOME and FAREWELL files, and the instructions for storing them on the server are identical. Note that, as with the WELCOME and FAREWELL files, you cannot use the template formatting commands and replaceable parameters discussed above.

A typical DIGEST-H or INDEX-H file for a list called MYLIST might contain:

```
The MYLIST list is sponsored by ABig Corporation.

See http://www.abig.com for information on ABig Corporation's products.
```
Figure 9.3. Typical contents of a DIGEST-H or INDEX-H file.

The contents of DIGEST-H and INDEX-H are appended to the digest or index, respectively, immediately following the list of topics. For instance,

```
Date: Tue, 11 Jun 2001 11:52:41 -0500
From: Automatic digest processor <LISTSERV@MYHOST.COM>
Reply-To: My test list <MYLIST@MYHOST.COM>
To: Recipients of MYLIST digests <MYLIST@MYHOST.COM>
Subject: MYLIST Digest - 10 Jun 2001 to 11 Jun 2001

There is one message totalling 10 lines in this issue.

Topics in this issue:

  1. Testing 125...3 sir!

The MYLIST list is sponsored by ABig Corporation.

See http://www.abig.com for information on ABig Corporation's products.
```
Figure 9.4. Sample DIGEST output for a list with a DIGEST-H file. The INDEX-H output would be

similar, following the list of postings.

(Note that you can't add a digest or index "footer" because anything after the end of the digest text is supposed to be discarded.)

9.7. Templates and template forms for the WWW interface

The following describes the available template files and their respective template forms for the WWW archive and administration interface. ***L-Soft does not advise modifying these templates unless you know exactly what you are doing. If you modify the templates it is strongly recommended that you keep copies of the originals in a safe location for fall-back.***

9.7.1. Forms contained in DEFAULT MAILTPL

Note that, although these template forms are available in DEFAULT MAILTPL (and thus theoretically available for list owners to modify), individual list owners cannot tamper with them. If the LISTSERV maintainer desires to change the "look" of the site, it is preferable to create a file called **www_archive.mailtpl** (see the *Site Manager's Operations Manual*, chapter 5.4.5 and below) rather than to edit the forms in DEFAULT MAILTPL, since DEFAULT MAILTPL will be overwritten during a software upgrade.

- **WWW_ARCHIVE_INDEX**: The basic INDEX.HTML page for the WWW archive interface. While this template form is available in DEFAULT MAILTPL, it cannot be changed by list owners.

- **WWW_ARCHIVE_USER_FORMS**: Tells LISTSERV which additional "user" forms to format for the list.

- **WWW_ARCHIVE_TRAILER**: The page trailer file included by the WWW interface's CGI script. When this template is included in a *listname*.**MAILTPL** file it controls ONLY the trailer for the *listname*.**html** main index page. See **WWW_LIST_TRAILER**, below.

- **WWW_ARCHIVE_HEADER**: The page header file included by the WWW interface's CGI script. When this template is included in a *listname*.**MAILTPL** file it controls ONLY the trailer for the *listname*.**html** main index page. See **WWW_LIST_HEADER**, below.

- **$WWW_IMAGES_URLDEF**: Default URLs for standard images, do not change

- **$WWW_IMAGES_URL**: URLs for standard images. If these images are stored in non-standard locations you put the URLs for those locations here. Otherwise LISTSERV uses the defaults in **$WWW_IMAGES_URLDEF** .

- **$WWW_ARCHIVE_HEADER**: Contains the header text for the WWW archive interface, e.g., what prints at the top of the page. This template form is included by default in the **WWW_ARCHIVE_HEADER** template form.

- **$WWW_ARCHIVE_TRAILER**: Contains trailer text for the WWW archive interface, e.g., what prints at the bottom of the page. This template form is included by default in the **WWW_ARCHIVE_TRAILER** template form.

- **XHTML_LISTSERV_REPLY_TRAILER**: Contains a trailer used for HTML digests

(technically a mail template rather than an HTML template)

- **DIRECTORY**: Template directory for **X-GETTPL** (overrides only). You probably do not want to change this template form unless advised to do so by L-Soft.

- **WWW_ARCHIVE_DIRECTORY**: Template directory for **X-GETTPL** (WWW_ARCHIVE only). You probably should not change this template form unless advised to do so by L-Soft.

- **WWW_LIST_HEADER**: A second-level header file that can be defined by the list owner for all pages other than the main *listname*.html page (see **WWW_ARCHIVE_HEADER** for the main page). If defined, this header appears after the **WWW_ARCHIVE_HEADER** and before the rest of the page's content on all pages below *listname*.html.

- **WWW_LIST_TRAILER**: A second-level trailer file that can be defined by the list owner for all pages other than the main *listname*.html page (see **WWW_ARCHIVE_TRAILER** for the main page). If defined, this trailer appears before the **WWW_ARCHIVE_TRAILER** and after the rest of the page's content on all pages below *listname*.html.

The following will help clarify the page placement of **WWW_LIST_HEADER** and **WWW_LIST_TRAILER** when they are defined in *listname*.**mailtpl**:

```
WWW_ARCHIVE_HEADER
WWW_LIST_HEADER
[page content]
WWW_LIST_TRAILER
WWW_ARCHIVE_TRAILER
```

Except as noted for the main list archive page (*listname*.html), list owners may not override **WWW_ARCHIVE_HEADER** or **WWW_ARCHIVE_TRAILER** as they are defined on a site-wide basis.

9.7.2. The www_archive.mailtpl file

Rather than changing DEFAULT MAILTPL to customize your site's "look", it is recommended that the LISTSERV maintainer(s) place modified templates from DEFAULT MAILTPL in a file called **www_archive.mailtpl** , which must be located in the same directory as DEFAULT MAILTPL and which will not be overwritten by a software update. (List owners cannot change this file but some of its templates may be overridden in a *listname*.**mailtpl** file.)

9.7.3. The default.wwwtpl file

The DEFAULT WWWPTL file contains the default templates for the parts of the WWW archive interface that are not defined in DEFAULT MAILTPL. This file should not be edited, as DEFAULT WWWTPL will be overwritten by a software update. Any site-wide emendations should be made in SITE WWWTPL (editable by LISTSERV maintainers only; see the next section) and list-level emendations should be made in *listname* WWWTPL. Both of these files can be edited via the web administration interface (see chapter 11).

When editing these templates please note two fundamental differences between them and the templates in DEFAULT MAILTPL:

1. Any substitution variable that you use (for instance, `&LISTNAME`) must be escaped with a "+" symbol between the ampersand and the name of the variable, thus: `&+LISTNAME` . (Note that, as with the regular mail template forms, not all substitution variables are available in every HTML template form.)

2. Any dot-formatting command you use (for instance, `.CC` , `.BB` , etc.) must have a "+" symbol rather than the dot, thus: `+CC +BB`

The templates currently included in DEFAULT WWWTPL are:

- `Style-sheet:` Style sheet for dynamic web templates. You will find this template form imbedded in most other web template forms; it makes it easier to change the overall "look" of the pages.

- `A1-main`: Second-level archive page (one month/week)

- `A1-def`: Special options for second-level archive page (see A1-MAIN)

- `A2-main`: Archive browsing (one message)

- `S1-main`: Main search page

- `S2-main`: Search results

- `S2-missing`: Search function, missing argument

- `OPEN-error`: Error message, can't access files (generic, returns error code)

- `OPEN-bad-index`: Error message, invalid index file

- `LOGIN-main`: Login screen

- `LOGIN-cookie`: Login confirmation after password saved in cookie

- `LOGIN-cookie-reset`: Confirmation after resetting login cookie

- `CHPW-main`: Change password screen

- `NEWPW-main`: Main password registration screen

- `LOGIN-CHECK-COOKIE`: Offer to reset cookie if authentication failed (imbedded template)

- `LOGIN-BROWSE-NOTAUTH`: Error screen, not authorized to browse

- `LOGIN-SEARCH-NOTAUTH`: Error screen, not authorized to search

- `LOGIN-ADMIN-NOTAUTH`: Error screen, not a LISTSERV administrator

- **LOGIN-MANAGE-NOTAUTH**: Error screen, not a list owner

- **LOGIN-MANAGE-NOPW**: Error screen, list cannot be managed via WWW

- **POST-NOTAUTH**: Error screen, not authorized to post

- **MM-DBMS-NOTAUTH**: Error screen, not authorized for DBMS mail-merge jobs

- **MM-LIST-NOTAUTH**: Error screen, not authorized for list mail-merge jobs

- **LITE-NOTSUPP**: Error screen, not supported in Lite.

- **NEWPW-mailed**: Awaiting mailed confirmation of new password screen.

- **ACTMGR-main**: Account management functions, main screen

- **ACTMGR-usersel**: Account management functions, user selection screen

- **ACTMGR-subopt**: Account management functions, view/update subscription options

- **ACTMGR-subopt-msglib**: Account management functions, text for subscription options

- **HDREDIT-main**: Edit list header, main screen

- **TPLMGR-formsel**: Template management, form selection screen

- **TPLMGR-formedit**: Template management, form edit screen

- **LMGT-main**: List management, main page

- **P1-QUOTE**: Reply function, text to prepend when including original message

- **P1-main**: Post/reply function

- **SUBEDIT-main**: Authenticated subscribe/leave, main page

- **BULKOP-main**: Bulk operations, main page

- **LAYOUT-data**: Layout customization, data page

- **LAYOUT-SYSTEM-data**: Layout customization, mandatory data (DO NOT EDIT!)

- **LAYOUT-data-wrapper**: Wrapper for layout data, sets useful variables

- **LAYOUT-main**: Layout customization, main page

- **LCMD-main**: Execute an arbitrary LISTSERV command (invoke "wa" with the parameter "**?LCMD1**")

- **MM1-main**: Mail-merge (DBMS based)

- **MM2-main**: Mail-merge (list based)

- **NEWLIST-main**: List creation, main page

- **LIST-LIBRARY**: Library of list header templates

- **LIST-LIBRARY-INIT**: Initialization sequence for library of list header templates

- **SEARCH-HELP**: Help page for the WWW archive interface's search functions

- **SETTINGS-HELP**: Help page for subscription settings

- **LIST-select**: Form to access archives of confidential (unlisted) list. Reached by invoking "wa" with the parameter "**?LIST**" (or by clicking on the link on the first-level archive page).
.
- **LOGIN-MSGLIB**: Miscellaneous error messages (for translation purposes)

9.7.4. The `site.wwwtpl` file (optional)

If desired, the LISTSERV maintainer(s) can override the `default.wwwtpl` file by providing a customized `site.wwwtpl` file in the same directory. This will prevent site-wide definitions being overwritten in an upgrade (i.e., when `default.wwwtpl` will normally be overwritten). The `site.wwwtpl` file takes precedence over `default.wwwtpl` but (for list-level templates only) will itself be overridden by definitions in any *`listname.wwwtpl`* files you have installed.

9.7.5. National language template files (*`idiom.mailtpl`*) (optional)

National language templates can be written and used with LISTSERV (L-Soft does not provide them). The use of such templates is governed by two settings:

- Site-wide: The **DEFAULT_LANGUAGE=** site configuration variable allows the LISTSERV maintainer to set the site-wide national language template for use by all lists on the server. By default this variable is unset and DEFAULT MAILTPL is used.

- List-level: The **Language=** list header keyword can be used to specify a national language template to be used for a particular list, for instance a Spanish-language list on an otherwise English-language server. The language template must already be present on the server in order for the list owner to be able to specify it in this keyword setting.

Since national language templates can be created only by the LISTSERV maintainer(s), further information on creating them can be found in the *Site Manger's Operations Manual*.

9.7.6. Template precedence

For template forms found in DEFAULT MAILTPL, the following precedence is used when LISTSERV searches for a given template form:

listname MAILTPL
idiom MAILTPL

```
WWW_ARCHIVE MAILTPL8
DEFAULT MAILTPL
```

That is to say, if LISTSERV needs a copy of the ADD1 mail template form, it will look first in the **listname.mailtpl** file for the list in question. If no such file exists, or if ADD1 is not present in **listname.mailtpl**, LISTSERV will look in **idiom.MAILTPL** (if **Language=** or **DEFAULT_LANGUAGE=** is set to *idiom*). Again, if the ADD1 form is not present in **idiom.mailtpl**, or if **idiom.mailtpl** does not exist, LISTSERV will then look in **default.mailtpl** (**www_archive.mailtpl** is skipped because ADD1 is not a web template form) and pull out the default ADD1 template form.

For template forms found in DEFAULT WWWTPL the precedence is:

```
listname WWWTPL
idiom WWWTPL
SITE WWWTPL
DEFAULT WWWTPL
```

The same sequence of events applies as for the MAILTPL files, except that **SITE WWWTPL** is never skipped (all template forms in the WWWTPL files are web forms).

9.8. Using the DAYSEQ(n) function

The **DAYSEQ(n)** function is quite powerful. This function allows the list owner to code template forms (such as the **PROBE1** or **BOTTOM_BANNER** messages) that change or "rotate" automatically.

The **DAYSEQ(n)** function is invoked in a **.BB - .EB** conditional block, and n corresponds to the number of days in the rotation period, i.e., to the number of variations that you want to make to the text of the message. **&DAYSEQ(n)** returns a number from 1 to n which increases by 1 every day, with no special regard for weekends. That is, if the rotation period is to last for a week, you code **DAYSEQ(7).** If the rotation period is 15 days, you code **DAYSEQ(15).** Two examples follow:

9.8.1. Rotating bottom banner

To create a rotating bottom banner, follow this example. A list has three commercial sponsors, each of whom are provided with an advertisement every three days. (Note that this doesn't take weekends into account; in this example, if company A is featured in the banner on Monday, it will be featured again on Thursday and then again on Sunday. However, in the following week it will be featured on Wednesday, Saturday, and Tuesday, so it will actually get rather good coverage.) Our **BOTTOM_BANNER** template form would look like this:

```
>>> BOTTOM_BANNER
.BB &DAYSEQ(3) = 1
Today's copy of the &LISTNAME newsletter has been brought to you
by Company A.
.EB
.BB &DAYSEQ(3) = 2
```

---

8 WWW_ARCHIVE MAILTPL is searched only for web-related template forms and is bypassed for mail template forms. For instance WWW_ARCHIVE MAILTPL will not be searched for ADD1 but will be searched for WWW_INDEX.

```
Today's copy of the &LISTNAME newsletter has been brought to you
by Company B.
.EB
.BB &DAYSEQ(3) = 3
Today's copy of the &LISTNAME newsletter has been brought to you
by Company C.
.EB
```

(Naturally you can feel free to be more florid with your prose :)

If a company needs to get a higher percentage of "air" time than another, you can simply assign it more than one of the possible n values of **&DAYSEQ(n)**. For instance, if you have two companies but one should get twice as many days of "air" time, you might code something like this:

```
>>> BOTTOM_BANNER
.BB (&DAYSEQ(3) = 1) OR (&DAYSEQ(3) = 3)
Today's copy of the &LISTNAME newsletter has been brought to you
by Company A.
.EB
.BB &DAYSEQ(3) = 2
Today's copy of the &LISTNAME newsletter has been brought to you
by Company B.
.EB
```

This would cause Company A's message to appear on days 1 and 3 of the rotation period and Company B's message to appear on day 2 only.

9.8.2. Rotating FAQ via the PROBE1 template and "Renewal= xx-Daily"

Subscription renewal can be coded with daily granularity (however, please note that it is and remains inadvisable to use renewal intervals of less than a week). If you further code subscription probing into the "Renewal=" keyword with the ",Probe" parameter, you open up the possibility of turning the standard **PROBE1** template form into a periodic FAQ. Here's how:

We'll assume to start that you will code "Renewal= 15-Daily,Probe" in your list header. (You can experiment with other numbers, but since we have two messages and will be using **&DAYSEQ(2)**, we need an odd renewal period.) We'll also assume that you want to send two versions of your FAQ each month; the first, a complete FAQ document, and the second, an abbreviated "reminder" version that just contains information about how to sign off, how to post to the list, and so forth. The basic algorithm is therefore:

When **&DAYSEQ(2) = 1**, send the full FAQ.
When **&DAYSEQ(2) = 2**, as it will 15 days later, send the abbreviated FAQ.

Your **PROBE1** template form would thus look like this:

```
>>> PROBE1 Periodic FAQ posting for &LISTNAME
&WEEKDAY, &DATE &TIME
.BB &DAYSEQ(2) = 1
This is the complete FAQ for &LISTNAME.  Please read it and keep a copy
for future reference. A FAQ document for &LISTNAME is distributed every
15 days, the full FAQ alternating with a shorter "reminder" FAQ.

<body of the full FAQ document>
```

```
.EB
.BB &DAYSEQ(2) = 2
This is the abbreviated FAQ for &LISTNAME. Please read it and keep a copy
for future reference. A FAQ document for &LISTNAME is distributed every
15 days, the full FAQ alternating with a shorter "reminder" FAQ.

<body of the abbreviated FAQ document>
.EB
```

9.8.3. Calculating the value for DAYSEQ()

When you first start using a rotating banner with the &DAYSEQ variable, the
`&DAYSEQ(n)= 1` period begins based on the number of days elapsed since a baseline.
On VM (and in REXX generally) you can calculate today's value easily with:

```
/* */
say Date('B') + 1
```

If you do not have access to a REXX interpreter, `Date('B')` is described as "the
number of complete days (that is, not including the current day) since and including the
base date, 1 Jan 0001, in the format '`dddddd`' (no leading zeros or blanks)."[9] It also is
equal to the C language expression time(0)/86400 + 719162 or, for OpenVMS users, to
the Smithsonian base date plus 678575.

For example, for Friday 21 May 1999, the value of Date('B') is 729895. This value
increases by one every day at midnight.

9.9. Serving up custom web pages for your list

This feature is not available in LISTSERV Lite.

Originally in order to serve up custom or special web pages for a list it was necessary to
construct those pages as HTML files and place them either into the /archives directory or
link them from somewhere else. This was sometimes impossible for list owners who had
no administrative access to the server's web directories or who had no other place from
which to serve web pages.

In 1.8d and later it is possible to add ad-hoc web page templates by creating new (non-
standard) template forms and enabling their display by setting a special variable value,
SHOWTPL_ALLOWED, in the template form. For instance, one could set up a page with
special administrative information, the list charter, netiquette information, or the like, and
serve it and maintain it directly from the LISTSERV web template interface without need
for any other access to the server.

9.9.1. A practical example: ADMIN_POST

The author used to serve up an administrative posting via FTP back in the days when his
lists lived on a server that had FTP access to the archive notebooks. When FTP access
to the server was cut off due to security concerns, he had to find another way to serve
the information via the web. Here is how it was done:

First, log into the web administration interface. Choose the list for which you will be
making a new page and click the "Templates" button to enter the mail and web template

---

[9] Cowlishaw, Michael: *The REXX Language: A Practical Approach to Programming*, 2nd ed., p.92.
Englewood Cliffs, NJ: Prentiss-Hall, Inc., 1990.

editing area. Since the template you will be creating is a web template, click the "Switch to WWW templates" button to change modes.

Hopefully everything between the two lines of asterisks below will be changed in a future version...

****
**

Currently there is no simple method to create a new (ie previously non-existent) template form from the web interface. The workaround is to open one of the existing template forms and add the template delimiter line at the bottom, then save the existing template form. For instance, we'll choose the first template form that presents itself, A1-DEF. Open this template by clicking on the "Edit form" button. If this template has not been changed from its default (and it probably will not have been), it will look like this:

```
+* You should now set these options using the layout customization
+* screen, except for the special F and S options, which are much too
+* advanced for the layout screen and must still be set in this template.
+IM LAYOUT-data-wrapper
+SE D &+LYT_DVIEW_D;
+SE FP &+LYT_DVIEW_FP;
+SE H &+LYT_DVIEW_H;
+SE O &+LYT_DVIEW_O;
+SE T &+LYT_DVIEW_T;
```

At the bottom of this form, directly after the `+SE T &+LYT_DVIEW_T;` line, type

```
>>> ADMIN_POST Administrative information page
```

(Be sure that there is a space between ">>>" and "ADMIN_POST". This is required.) Next, click on the "Update template" button and the template will update. You will see the message at the top of the page:

The A1-DEF form has been successfully stored in the LISTNAME template library.

(LISTNAME of course will be the name of the list you are working with.)

Next, back out of the preceding pages and return to the main web template editing page, and reload that page. In the drop-down list box you will see a new line that says

(*) Administrative information page [ADMIN_POST]

You now have one final cleanup action to take before you edit the new template form. Go back into the A1-DEF form editing screen (you'll note that it also has a "(*)" next to it, indicating that it has been modified). You'll note when you go into the editing screen that it says at the top,

This form is defined in the LISTNAME template.

Clear the text out of the edit box, and also clear the "Description:" text box (this is important, otherwise you will leave a blank A1-DEF template form in the list's template library, and that is not good). Update the template again. The interface will again tell you that

The A1-DEF form has been successfully stored in the LISTNAME template library.

In actuality what has happened is you have removed the copy of A1-DEF that was placed into listname.WWWTPL when you saved the modifed copy that contained the template

form delimiter line for your ADMIN_POST template form. If you back out to the template management page again you will see that the "(*)" indicator is no longer present at the start of the A1-DEF line. Now you can open the ADMIN_POST template form and edit it.
****
**

Next, choose the "(*) Administrative information page [ADMIN_POST]" line, and click the "Edit form" button. You will be presented with an editing page with a blank text box into which you can place your HTML code for the page. The first thing you need to do, however, is to enable the template form so that it can be served by LISTSERV's web interface. This is done by placing the following line at the very top of the template form:

+SE SHOWTPL_ALLOWED 1

Following this line you can start adding your HTML. However note carefully that you cannot override the default headers and footers that have already been defined by other template forms in the library. You can start with a <title></title> block but it will be followed by the pre-defined header and *then* by your HTML.

The URL for the page you are defining in this example is then

http://*your_server_hostname*/*path_to_wa*?SHOWTPL=ADMIN_POST&L=LISTNAME

(the parameters for 'wa' are case-sensitive and must be sent in upper case). For instance, the author's version of the ADMIN_POST template form can be viewed at

(The following example assumes that you have the 1.8e web administration interface installed. New template forms cannot be created this way in previous versions.)

First, log into the web administration interface. Choose the list for which you will be making a new page and click the "Templates" button to enter the mail and web template editing area. Since the template you will be creating is a web template, click the "Switch to WWW templates" button to change modes.

Next, type the name of the new template form into the box provided, and click "Create". A page entitled "Edit List Template" will come up, with the command response

The ADMIN_POST form has been successfully stored in the TEST template library.

In the "Description:" box, type a description of the template, for example, "Administrative information page".

In the large text box provided for the template text, first type the following line:

+SE SHOWTPL_ALLOWED 1

This line tells LISTSERV that it is allowed to serve the page on the web. If the line is not found, the template will not be available.

Following this line you can start adding your HTML. However note carefully that you cannot override the default headers and footers that have already been defined by other template forms in the library. You can start with a <title></title> block but it will be followed by the pre-defined header and *then* by your HTML.

After adding your HTML, click "Update", and the template form will be stored. The URL

for the page you are defining in this example will be

http://*your_server_hostname*/*path_to_wa*?SHOWTPL=ADMIN_POST&L=*listname*

(the parameters for 'wa' are case-sensitive and must be sent in upper case). For instance, the author's version of the ADMIN_POST template form can be viewed at

http://peach.ease.lsoft.com/scripts/wa.exe?SHOWTPL=ADMIN_POST&L=VISBAS-L

Documented Restriction: For LISTSERV 1.8d (or LISTSERV 1.8e running with the 1.8d web interface) there is no simple method to create a new (ie previously non-existent) template form from the web interface. If you are not comfortable with the GET and PUT method of updating list-level template forms, a web-based workaround is to open one of the existing template forms and add the template delimiter line at the bottom, then save the existing template form. For instance, we'll choose the first template form that presents itself, A1-DEF. Open this template by clicking on the "Edit form" button. If this template has not been changed from its default (and it probably will not have been), it will look like this:

```
+* You should now set these options using the layout customization
+* screen, except for the special F and S options, which are much too
+* advanced for the layout screen and must still be set in this template.
+IM LAYOUT-data-wrapper
+SE D &+LYT_DVIEW_D;
+SE FP &+LYT_DVIEW_FP;
+SE H &+LYT_DVIEW_H;
+SE O &+LYT_DVIEW_O;
+SE T &+LYT_DVIEW_T;
```

At the bottom of this form, directly after the `+SE T &+LYT_DVIEW_T;` line, type

```
>>> ADMIN_POST Administrative information page
```

(Be sure that there is a space between ">>>" and "ADMIN_POST". This is required.) Next, click on the "Update template" button and the template will update. You will see the message at the top of the page:

The A1-DEF form has been successfully stored in the LISTNAME template library.

(LISTNAME of course will be the name of the list you are working with.)

Next, back out of the preceding pages and return to the main web template editing page, and reload that page. In the drop-down list box you will see a new line that says

(*) Administrative information page [ADMIN_POST]

You now have one final cleanup action to take before you edit the new template form. Go back into the A1-DEF form editing screen (you'll note that it also has a "(*)" next to it, indicating that it has been modified). You'll note when you go into the editing screen that it says at the top,

This form is defined in the LISTNAME template.

Clear the text out of the edit box, and also clear the "Description:" text box (this is important, otherwise you will leave a blank A1-DEF template form in the list's template library, and that is not good). Update the template again. The interface will again tell you that

---

The A1-DEF form has been successfully stored in the LISTNAME template library.

In actuality what has happened is you have removed the copy of A1-DEF that was placed into *listname*.WWWTPL when you saved the modifed copy that contained the template form delimiter line for your ADMIN_POST template form. If you back out to the template management page again you will see that the "(*)" indicator is no longer present at the start of the A1-DEF line. Now you can open the ADMIN_POST template form and edit it.

10. Solving Problems

10.1. Helping subscribers figure out the answers

As the saying goes: "Give a man a fish, feed him for a day; teach a man to fish, feed him for life." The analogy can and should be extended to all Internet users, not the least of whom are your own subscribers.

Depending on your own preferences, some requests from subscribers for operations that they can perform for themselves can be fulfilled by you as the list owner, or by the subscribers with some coaching from you. While it is a negative approach, the list owner can never assume that the subscriber reads or saves the materials sent to him at the time of subscription. Thus you will have to deal on a regular basis with users who ask how to unsubscribe, or how to get archive files, or how to set their subscription to DIGEST or NOMAIL.

Often these requests for help are posted directly to the list. The proactive approach to this problem is to do one or both of two things:

- Respond to the list with the answer so that all can benefit
- Respond privately to the subscriber with the answer if it has been posted repeatedly

If a user asks a question about a topic that has been discussed previously, you might suggest in a tactful way that the answer can be found in the archives. If your host server supports the LISTSERV database functions, you might even include a sample DATABASE JOB that the user can "clip and send" to LISTSERV.

Often it is tempting to simply "get things over with" and take care of the user's request in many cases – the user wants to be set to NOMAIL because he's going on vacation, the user wants off the list, etc. – but while this solves the short-term problem, it doesn't teach the user anything. Naturally it takes more time to be a coach than it does to be the all-powerful list administrator, but the goodwill you can create by being proactive rather than reactive outweighs the convenience of simply sending the command yourself. You will find that many subscribers appreciate the fact that someone takes the time to explain the complexities of LISTSERV to them.

In order to cut down on the time it takes to respond in "coaching" situations, many list owners prepare "boilerplate" files with the answers to common questions that they can simply "cut and paste" into return mail. (Several such "boilerplate" files are included in Appendix C.)

10.2. Loop-checking can cause occasional problems with quoted replies

(See also 4.7.5.)

By default, LISTSERV's internal loop-checking routines look for anything in the body of a mail message that looks like a header line – specifically anything that looks like a "To:", "Sender:", or "Reply-To:" header line. If it finds anything like this, LISTSERV intercepts the message and sends it to the list owner (or the person(s) designated by the "Errors-To=" keyword) as an error.

Often a user who replies to list mail includes all or part of the message he is replying to as part of his reply ("quoting"). While this is a questionable practice to begin with, unfortunately a number of popular mail programs make it worse by including the quoted message in its entirety (including header lines) in the body of the reply. For instance, the

following message ended up in the author's error mailbox:

```
The enclosed message, found in the ACCESS-L mailbox and shown under the spool
ID 6305 in the  system log, has been identified as  a possible delivery error
notice  for the  following reason:  "Sender:", "From:"  or "Reply-To:"  field
pointing to the list has been found in mail body.

------------------------ Message in error (42 lines) -------------------------
Received: by access.mbnet.mb.ca id AA05697
  (5.67b/IDA-1.4.4 for Microsoft Access Database Discussion List
<ACCESS-L@peach.ease.lsoft.com>); Wed, 1 Mar 1995 10:26:29 -0600
Date: Wed, 1 Mar 1995 10:26:29 -0600
From: xxxxxx xxxxxxxxx <xxx@MBNET.MB.CA>
Message-Id: <199503011626.AA05697@access.mbnet.mb.ca>
To: Microsoft Access Database Discussion List
Message-Id: <199503011626.AA05697@access.mbnet.mb.ca>
To: Microsoft Access Database Discussion List
<ACCESS-L@PEACH.EASE.LSOFT.COM>
Subject: Re:      Re: Foxpro listserv address
X-Mailer: AIR Mail 3.X (SPRY, Inc.)


<---- Begin Included Message ---->
Date:         Thu, 23 Feb 1995 01:17:36 -0500
From: xxxxxxx@xxx.com
Sender: Microsoft Access Database Discussion List
              <ACCESS-L@peach.ease.lsoft.com>
Subject:      Re: Foxpro listserv address
To: Microsoft Access Database Discussion List
              <ACCESS-L@peach.ease.lsoft.com>

>BTW, I don't know why she is still on Foxpro, I thought they went out
>into the desert??

<---- End Included Message ---->

(subscriber's reply deleted)
```

Figure 10.1. Sample error message with included headers.

The   problem with this reply was two-fold, from a list owner's standpoint. First (a netiquette issue), the sender didn't bother to remove unnecessary header lines from his reply. If properly formatted, however, this would not normally cause an error.

Second, the mail software he was using didn't include ">" characters at the beginning of every line of the included message. Had it done so, the message would have passed through LISTSERV unhindered.

One variation on this error is mail software that quotes messages by adding the ">" character followed by a space for esthetic reasons. For instance, using the above error as an example:

```
> Date:         Thu, 23 Feb 1995 01:17:36 -0500
> From: xxxxxxx@xxx.com
> Sender: Microsoft Access Database Discussion List
              <ACCESS-L@peach.ease.lsoft.com>
> Subject:      Re: Foxpro listserv address
> To: Microsoft Access Database Discussion List
              <ACCESS-L@peach.ease.lsoft.com>

> BTW, I don't know why she is still on Foxpro, I thought they went out
> into the desert??
```

Figure 10.2. A slightly different sample error message with included headers.

This won't work either. Generally this is a client configuration problem and it can be fixed by setting the quoting character in the client's configuration file.

On the other hand, the following quote *would* have worked:

```
>Date:          Thu, 23 Feb 1995 01:17:36 -0500
>From: xxxxxxx@xxx.com
>Sender: Microsoft Access Database Discussion List
               <ACCESS-L@peach.ease.lsoft.com>
>Subject:       Re: Foxpro listserv address
>To: Microsoft Access Database Discussion List
               <ACCESS-L@peach.ease.lsoft.com>

>BTW, I don't know why she is still on Foxpro, I thought they went out
>into the desert??
```
Figure 10.3. A correctly-formatted message with included headers.

The ultimate solution to the problem is to warn subscribers to limit their quoting to a minimum, and in any case to be sure to delete anything that looks like a header line in the body of their reply.

10.3. User can't unsubscribe and/or change personal options

See Chapter 4, section 4.2 where this is discussed in detail.

10.4. Firewalls

Firewalls on the Internet are set up for essentially the same reason firewalls are designed into buildings and automobiles – to keep dangerous things (in this case, hackers, viruses, and similar undesirable intruders) from getting in and wreaking havoc with sensitive data. Unfortunately, they don't always keep people from behind them from sending mail out, and this can cause problems when users from such sites attempt to subscribe to lists.

If your list is set to confirm all subscriptions with the "magic cookie" method ("Subscription= Open,Confirm"), you will receive an error message any time a user from a firewalled site attempts to subscribe, since the "cookie" confirmation message will bounce off the firewall. If your list is not set to confirm subscriptions, the same user will be able to subscribe to your list but all mail sent to him will bounce.

Some firewalls reportedly can recognize "friendly" LISTSERV mail and let it through, but because of security considerations, it is unlikely that this problem will ever completely go away. Thankfully it does not seem to be a major cause of mailing list errors.

10.5. What to do if LISTSERV won't store your list

LISTSERV expects list files to be delivered to it without any formatting characters (excluding, of course, the carriage return-line feed at the end of each line). This can cause a problem if you try to store the entire list (header and subscribers) using a mail client that inserts line-wrap characters into text longer than 80 columns. Specifically, one client that does this is Pine; others that can cause problem are cc:Mail and just about any Windows POP client.

There are a couple of ways to get around this problem.

1. Don't get the entire list if all you're going to do is edit the header. Use the **GET listname (HEADER** syntax to get the header only, and use **ADD** and **DELETE** commands to manipulate the subscriber list. This is the preferred method.

2. If you have to get the entire list, e.g., in order to delete a subscriber manually, use a

client that does not wrap text (or turn off line wrap if possible). If you are on a unix system that has mailx installed, you can store a list from the command line with the command syntax

```
mailx listserv@host < listfile
```

Note that L-Soft does not recommend hand-editing the subscriber list; it is preferable to use wildcards to delete problem addresses, and using an editor to do this should *always* be the last resort.

3. If all else fails, you can use a public-domain utility called LB64 to convert the list file into a base-64 command JOB that LISTSERV will understand. This utility is generally available from the VM LISTSERV sites; send a **GET LB64 C** command to **LISTSERV@LISTSERV.NET** if you can't find it anywhere else. Note that this is an unsupported utility. You will need to compile it with a C compiler (not supplied). The utility is primarily for users on unix systems, although with two minor modifications it can also be used on 32-bit Windows systems.

10.6. If I can't find the answer, where do I turn?

Two LISTSERV lists exist for list owner and LISTSERV maintainer questions.

LSTSRV-L is the LISTSERV give-and-take forum. Its primary mission is to provide assistance to LISTSERV maintainers, but it can also be of interest to list owners who desire a more in-depth knowledge of the workings of the system. To subscribe to LSTSRV-L, send your subscription request to LISTSERV@LISTSERV.NET.

LSTOWN-L is the LISTSERV list owners' discussion list, where list owners can get assistance on list maintenance and other aspects of list ownership. To subscribe to LSTOWN-L, send your subscription request to LISTSERV@LISTSERV.NET.

11. Using the Web Adminstration Interface

[Information on the new 1.8e web interface was not available in time for the release of the public beta. This section will be rewritten prior to the 1.8e product release.]

LISTSERV 1.8d introduces a powerful web-based interface for the management of existing mailing lists (currently it is not possible to create a mailing list with the interface). Virtually all list management operations can be accomplished via this interface, which is tied into LISTSERV's own password manager for security.

Please note carefully that this interface *cannot* be used to manage lists that are coded `Validate= Yes,Confirm,NoPW` or `Validate= All,Confirm,NoPW` , because passwords are not accepted for validation in those cases.

11.1. Default LISTSERV Home Page

Starting with LISTSERV 1.8d the interface includes a default home page for LISTSERV. Typically this is reached by using the URL:

On unix: `http://yourhost.domain/cgi-bin/wa`
On VMS: `http://yourhost.domain/htbin/wa`
On Windows: `http://yourhost.domain/scripts/wa.exe`
     or   `http://yourhost.domain/cgi-bin/wa.exe`

Of course this is not standardized; the location of the 'wa' script is determined by the value of WWW_ARCHIVE_CGI in LISTSERV's site configuration file. In any case, invoking 'wa' without any parameters returns the default home page, which looks like this:

Welcome to LISTSERV!

From this page, you can access the following services:

- Online mailing list archives.

- CataList, the official catalog of public LISTSERV® lists.

- Online documentation in HTML format:
    - LISTSERV user's guide.
    - LISTSERV list owner's quick start.
    - LISTSERV list owner's guide.
    - LISTSERV site manager's guide.

- Mailing list management interface (list owners only).

- Server management interface (LISTSERV administrator only).

This page can be modified by site managers so it may not look exactly like this.

11.2. Logging in

You can log into the list administration interface from any list's main web archive index page (assuming that this link has not been removed by the list owner; it exists in the WWW_INDEX mail template by default). The interface may also be reached by a link from the default LISTSERV home page mentioned in 11.1, above.

To access the list administration interface without a link, you point your web browser to the "wa" script (just as you do to access the web archive interface) with a value of "LMGT1". This value must be in CAPS. Typically the interface is accessed as follows:

On unix: `http://yourhost.domain/cgi-bin/wa?LMGT1`
On VMS: `http://yourhost.domain/htbin/wa?LMGT1`
On Windows: `http://yourhost.domain/scripts/wa.exe?LMGT1`
        or   `http://yourhost.domain/cgi-bin/wa.exe?LMGT1`

but this is not always standardized. Note that the `LMGT1` parameter passed to "wa" MUST be in CAPS. Additionally, under Windows NT and Windows 95, certain web servers may insist that "wa.exe" be in CAPS as well.

In any case, once you link to this URL, you will be asked to log in:

Login required

The function you have requested requires authentication. Please enter your e-mail address and your LISTSERV password (*not* the password you use to login to your computer or read your mail), and click on the "Login" button. If this is the first time you see this dialog, or if you have forgotten your password, you will need to get a new LISTSERV password first.

E-mail address: [_____]
Password: [_____]

                    [Login]   [Change password]
                [Login and save my password in a cookie]

If you already have a personal LISTSERV password, simply log in with your userid and password. Please note that the userid you use here must be associated with the personal password you have from LISTSERV. If you have registered a password as joe@unix.host.com and try to log in here as joe@host.com with that password, LISTSERV will reject your login attempt.

If you login with the "save my password in a cookie" method, LISTSERV will issue you a cookie that allows you to bypass this login screen (and incidentally to stay logged into the interface for longer than 15 minutes without having to log in again when your session expires). This option is, however, only recommended for people who have physically secure machines (for instance, on your machine at home or in your office) or who are able to otherwise keep unauthorized users from logging in, since LISTSERV cannot tell who is using the cookie. Specifically if your browser does not support separate configurations or bookmark files for different users, you should not use the cookie method in a workplace environment.

Please note that there is a known bug in Netscape prior to version 4.0 which allows you to see the userid and password typed into the text boxes if you back up to the login page using the "Back" arrow.

11.3. Setting a LISTSERV password

If you do not already have a personal LISTSERV password (set with the `PW ADD` command or via the web interface) or cannot remember your password, you need to

define one now. If you choose to do this via the web interface, simply click the hyperlink and you will get the following page:

Registering your LISTSERV password

Please enter your e-mail address and the desired password, then click on the "Register password" button. If you already had a LISTSERV password, but cannot remember what it was, this procedure will automatically replace your existing password with the new one you will be entering below.

E-mail address: [_____]
Password: [_____]
Password (again):   [_____] *(verification)*

[Register password]

When you hit the "Register password" button, you will be transferred to this page:

Confirmation e-mailed

Your password registration request has been accepted. For your protection, the password will not be activated just yet (anyone could have completed this form using your e-mail address).

To activate your password, simply follow the instructions which have been e-mailed to you at *joe@unix.host.com*. Please wait until you receive an e-mail message from LISTSERV saying *"Your new password was registered successfully"* before trying to use it with the WWW interface.

Then you simply need to "OK" the password confirmation message that was mailed to you by following the instructions in that message, and your password will be registered.

11.4. The List Management main page

Once you get logged in, you will see the main List Management page.

List management – main page

This screen allows you to manage your mailing list using LISTSERV's WWW interface. First, select the list you would like to work with:

List name: [_____]

[Subscribers][Configuration][Layout][Templates][Bulk op.][Mail merge]

[Manage subscribers]
These menus allow you to add or delete subscribers, change a subscriber's e-mail address or subscription options, see whether someone is still subscribed to the list, etc. If you have a lot of subscribers to add or delete, see bulk operations.

[Edit list configuration]
This is the place to go if you want to change the configuration options for your list (also known as the *list header*).

<div style="border: 1px solid black; padding: 10px;">

[Customize layout]

     The layout editor allows you to customize the WWW interface using a simple graphical interface. You can switch between text and graphical (icon-based) layout for the archive pages, disable functions which are not useful or not wanted for your particular list, or even translate the archive pages.

[Edit mail and web templates]

     The template editor allows you to customize the administrative messages sent by LISTSERV in response to most commands (known as *mail templates*). You can also use it to exercise finer control on the layout of the WWW interface than is possible through the graphical layout editor. Note that the banners at the very top and bottom of WWW archive pages are under the LISTSERV administrator's control. You can, however, add your own top and bottom banners in addition to the site-wide ones imposed by the administrator.

[Bulk operations]

     This screen allows you to add or delete large numbers of subscribers from a text file that will be downloaded through your browser.

[Command]

     This screen allows you to execute an arbitrary LISTSERV command and see the results immediately, in your browser window.

[Mail merge]

     If enabled by the administrator, this screen allows you to send customized mail-merge messages to your subscribers. You can choose which subscribers should receive the message (for instance, all AOL subscribers who are set to NOMAIL), and you can include customized substitutions or conditional blocks in the message (this is particularly useful when coupled to a DBMS back-end).

Enter the <u>server administration area</u> (LISTSERV administrator only).

</div>

List owners who have only one or just a few lists running on the server will be presented with a drop-down list box from which they can choose the list they want to work on (only their own lists will be displayed). Site maintainers or list owners who own many lists on the server will either see the drop-down list box (on servers with up to 51 lists) or a plain text box (on servers with 52 or more lists). Either type or choose the name of the list you want to work on, and click the button for the operation you want to perform.

Note that if you have saved your password in a cookie, the note about bookmarking not memorizing your password is still correct, but when you do click the bookmark in the future, LISTSERV will request the cookie from your browser and the net effect is the same.

For the sake of argument, let's say that we have a list called CAT-FANCY that we want to administer. The following four sections explain what is back of the four function buttons.

11.5. Maintaining subcriptions via the web

Clicking on the "Manage subscribers" button will bring up the following screen:

<div style="border: 1px solid black; padding: 10px; text-align: center;">

Account management – CAT-FANCY

Examine or delete a subscription

</div>

```
┌─────────────────────────────────────────────────────────────────────┐
│                                                                       │
│  Name or address:      [_____]     │
│                        henry@somewhere.com                            │
│                        Henry Brown                                    │
│                        s*lvia                                         │
│                                                                       │
│                        [Search in CAT-FANCY]        [Clear]           │
│                                                                       │
│                        Add a new user to the list                     │
│                                                                       │
│  Name & address:       [_____]     │
│                        henry@somewhere.com Henry Brown                │
│                        Henry Brown <henry@somewhere.com>              │
│                        [•] Send welcome message                       │
│                        [o] Do not notify the user in any way          │
│                                                                       │
│                        [Add to CAT-FANCY]   [Clear]                   │
│                                                                       │
├───────────────────────────────────────────────────────────────────── │
│ Back to the list management page                                      │
└─────────────────────────────────────────────────────────────────────┘
```

11.5.1. Examine or delete a subscription

This works very much like the "SCAN" command. Simply enter your criteria in the text box and click the "Search in ..." button. If there is no match for your entry, you will get back the same page as above, but with a

```
┌─────────────────────────────────────────────────────────────────────┐
│ Scan: No match.                                                       │
└─────────────────────────────────────────────────────────────────────┘
```

message at the top. If on the other hand your search *is* successful, one of two things will happen. If there are multiple matches for your criteria, the following screen will be displayed, with a drop-down list box containing all of the matches:

```
┌─────────────────────────────────────────────────────────────────────┐
│                                                                       │
│                  Account management – CAT-FANCY                       │
│                                                                       │
│                        Select a subscriber                            │
│                                                                       │
│                  ["Joe User" <joe@host.com> [V]                       │
│                                                                       │
│                  When deleting someone from the list:                 │
│                        [•] Notify the user by e-mail                   │
│                  [o] Do not notify the user in any way                 │
│                                                                       │
│       [Examine]   [Delete]   [New search]   [Delete from all lists]   │
│                                                                       │
├───────────────────────────────────────────────────────────────────── │
│ Back to the list management page.                                     │
└─────────────────────────────────────────────────────────────────────┘
```

You now simply choose the user you want to examine or delete and click on the appropriate button. If there was only a single match to your query, the preceding screen will be bypassed and you will go directly to the next screen. If you didn't find what you were looking for, you can press the "New search" button to get a new search screen.

```
┌─────────────────────────────────────────────────────────────────────┐
│                                                                       │
│                  Account management – CAT-FANCY                       │
│                                                                       │
│                  View or set subscription options                     │
│                                                                       │
```

```
┌─────────────────────────────────────────────────────────────────────┐
│                          joe@host.com                                 │
│                                                                       │
│  Notification options:  [●] Notify the user by e-mail                 │
│                         [o] Do not notify the user in any way         │
│                         [Update]   [Delete]   [New search]   [Delete from all lists] │
│                                                                       │
│  Name:                  [Joe User                ]                    │
│  Address:               [joe@host.com            ]                    │
│                         Subscribed on 7 Jan 1998                      │
│                                                                       │
│  Subscription type:                                                   │
│               [●] Regular                              [NODIGEST]      │
│               [o] Digest (traditional)            [NOMIME DIGEST]      │
│               [o] Digest (MIME format)      [NOHTML MIME DIGEST]      │
│               [o] Digest (HTML format)               [HTML DIGEST]    │
│               [o] Index (traditional)              [NOHTML INDEX]      │
│               [o] Index (HTML format)                [HTML INDEX]      │
│                                                                       │
│  Mail header style:                                                   │
│               [o] Normal LISTSERV-style header         [FULLHDR]      │
│               [●] LISTSERV-style, with list name in subject  [SUBJECTHDR] │
│               [o] LISTSERV-style, short              [SHORTHDR]        │
│               [o] "Dual" (second header in mail body)  [DUALHDR]       │
│               [o] sendmail-style                       [IETFHDR]       │
│                                                                       │
│  Acknowledgements:                                                    │
│               [o] No acknowledgements          [NOACK NOREPRO]        │
│               [●] Short message confirming receipt  [ACK NOREPRO]     │
│               [o] Receive copy of own postings   [NOACK REPRO]        │
│                                                                       │
│  Miscellaneous:                                                       │
│               [o] Mail delivery disabled temporarily    [NOMAIL]      │
│               [●] Address concealed from REVIEW listing [CONCEAL]     │
│               [o] User is exempt from renewal/probing  [NORENEW]      │
│               [o] User may bypass moderation           [EDITOR]       │
│               [o] All postings sent to list owner for review [REVIEW] │
│               [o] User may not post to list            [NOPOST]       │
│                                                                       │
│                   [Update]   [Delete]   [New search]                  │
│                                                                       │
│  Back to the list management page.                                    │
└─────────────────────────────────────────────────────────────────────┘
```

If you are deleting someone or changing/updating their options, the two radio-button options allow you to choose whether or not the operation will be non-"quiet" (where notification is sent), or "quiet" (where no notification is sent). The two options when used with the "Delete" button are therefore strictly equivalent to "**DELETE** *listname userid@host*" and "**QUIET DELETE** *listname userid@host*", respectively, and the other equivalent commands are formatted identically. "Notify the user by e-mail" is the default.

You will note that this page allows you to set virtually every user option available, excepting only a couple of obsolete header style settings that are still retained in the command set for backwards compatibility. (Should a user be set to one of these obsolete settings, it will show up as "Special or obsolete header style". Specifically this will happen

if a user is set to the `FULL822` or `SHORT822` header options.)

Note also that the subscription date is displayed for the user. If there is no subscription date, then this indicates a user who has been on the list since before your server was upgraded to version 1.8c (and thus prior to the time when the subscription date was tracked by LISTSERV).

"Delete from all lists" is strictly equivalent to the command "`DELETE * userid@host`" and is used to delete the user from all lists on the local server (for site managers) or from all lists on the local server which are owned by the invoker (for list owners).

If you are making changes to the user's name field, address, or user options, use the "Update" button to commit the changes. If you make changes to both the options and the identification fields, user option settings are updated first, and then changes are made to the name and address fields.

Following both a "Delete" and an "Update" operation, the main Account Management screen is displayed along with a message indicating the success or failure of your operations.

11.5.2. Add a new user to the list

To add a new user to the list, simply type the user's address and full name into the bottom section of the page shown at the beginning of 11.4. (The full name is optional; if omitted the user will be added anonymously to the list.) Then choose whether or not to notify the user that he has been added and click on the "Add to *listname*" button.

11.6. Maintaining the list header via the web

From the List Management Main Page (shown in 11.4, above), type in the name of the list for which you want to display and/or modify the header and click on the second button ("Edit list header").

You then get a screen as shown on the next page. The list header appears in a multi-line text box which can be scrolled both up and down and left and right. At first glance you will appear to have a standard header, but note carefully one apparently missing element: There are no asterisks in column 1 of the header lines.

For the purpose of this interface, the asterisks denoting that header lines are, in fact, header lines, are not required. You simply type in the changes or added lines just as if you were using a regular text editor. When you are finished, you click the "Update" button to submit the changes. If you make a mistake in the editing or simply want to start over, you can click the "Reload" button to reload the header information from the server.

Edit list header – CAT-FANCY

```
+------------------------------------------------------------------------+
|The Cat Fancier's List                                                  |
|                                                                        |
| Review= Owners                                                         |
| Send= Public                                                           |
| Notify= No                                                             |
| Reply-to= List,Respect                                                 |
| Validate= No                                                           |
| Notebook= Yes,E:\LISTS\CAT-FANCY,Weekly,Public                         |
| Subscription= Open,Confirm                                             |
```

```
| Confidential= No                                                      |
| Ack= Yes                                                              |
| Renewal= 2/1,5/1,8/1,11/1,Probe                                       |
| Digest= Yes,A,Daily,23:00,Size(1000)                                  |
| Mail-Via= Distribute                                                  |
+-----------------------------------------------------------------------+
```

[Update] [Reload]

Back to the <u>list management page</u>.

When you submit your changes, you will get the same kind of feedback from LISTSERV as you would if you sent a **PUT** operation by mail. The next screen will either say that the header of the list has been successfully updated, or it will indicate that it has found errors and that the header has not been stored. The feedback page also has a text box containing the header information you've just stored (or tried to store) so if you need to make further emendations to the header, you don't have to back up and start over.

11.7. Customizing how a list's pages look

By clicking the "Customize layout" button on the main list management page, you can pull up a page that allows you to customize how a list's pages look by simply answering a few questions. This is much simpler than editing the raw templates and can help avoid formatting or syntax problems that might be difficult to fix.

The page is fully self-documented.

11.8. Maintaining mail and WWW templates via the web

From the List Management Main Page (shown in 11.4, above), type in the name of the list for which you want to display and/or modify the header and click on the third button ("Edit mail and WWW templates").

You then get the following screen:

Template management – CAT-FANCY

Select a form to view or edit

[<u>Welcome message</u>   **[V]**

[Edit form]   [Switch to WWW templates]

Back to the list management page.

From this page you can either:

- edit the forms from your *listname*.**MAILTPL** file as well as your *listname*.**WELCOME** and *listname*.**FAREWELL** files (or add them if they do not already exist); or

- edit the special WWW templates.

In order to edit templates or your WELCOME or FAREWELL file, you simply choose the form you want to work with from the drop-down list box and click on the "Edit form" button. To switch to WWW templates, click on the "Switch to WWW templates" button; if

working on WWW templates and you wish to return to mail templates, click on the corresponding "Switch to mail templates" button on that page.

(Note: To customize the "look" of a list's archive pages, you can alternately use the questionnaire-driven functionality described in 11.7, above.)

If you do not have a WELCOME or FAREWELL file, the response will be a page with the message "The TEST list has no WELCOME message" or "The TEST list has no FAREWELL message". You can then create the message you want by typing it into the multi-line text box (this is very similar to the list header editing interface). There is also a separate box to type the subject line into.

If you choose to edit one of the standard forms found in DEFAULT MAILTPL for your list, simply choose the template form from the drop-down list box and click "Edit form". You will get a page with the message "This form is defined in the DEFAULT template" along with the appropriate subject line and template text which you can then edit and submit. If your listname.MAILTPL already contains the template form you want to edit, the message will say "This form is defined in the *listname* template" and there will be an asterisk on the entry for the form in the drop-down list, indicating that it has been altered from the standard text.

11.9. Bulk operations via the web

Please note carefully that your browser MUST support the RFC1867 file upload extension or you will not be able to use the bulk operations page. Most current browsers do support this extension, including but not limited to Netscape 3.x and later, and Internet Explorer 4.x and later.

From the List Management Main Page (shown in 11.4, above), type in the name of the list for which you want to display and/or modify the header and click on the last button ("Bulk Operations"). You will get the following page:

Bulk operations – CAT-FANCY

Caution: some of the functions offered through this page will ***remove all subscribers*** from CAT-FANCY. Double-check your selection before submitting!

 Input file: [_____] **[Browse...]**

 Function: [•] **Add** the imported addresses to CAT-FANCY; do not remove any subscribers
            [o] **Remove all subscribers** from CAT-FANCY, then **add** the imported addresses (to remove all subscribers, select this option and omit the input file)
            [o] **Remove** the imported addresses from CAT-FANCY; do not add any subscribers
            [o] **Remove** the imported addresses from **all lists**

            [Import]

Note:

> - The input file must be a *plain text* file (not a word processor document or spreadsheet) and must contain one address per line, optionally followed with a space (or TAB) and the subscriber's name.
>
> - The subscribers being added or deleted will not be notified.
>
> - These functions require a browser supporting the "file upload" extension (RFC1867). Current versions of Netscape and Internet Explorer both support this operation, but you should try other browsers carefully on a test list.
>
> Back to the list management page.

(If you get an error 2 when you click on the "Import" button, this means that the "upload" directory has not been created. If you get an error 13 when you click on the "Import" button, this means that the "upload" directory has been created but the CGI program user does not have write permission in that directory.)

The input file is created on your own machine with an ASCII text editor (as noted in the instructions on the page). If you have the "upload" directory correctly configured, the next page you will see after clicking the "Import" button will have a command response like the following:

If the first radio button is set (Add but do not delete any existing subscribers):

> **ADD: no error, 202 recipients added, no entry changed, no duplicate, none forwarded.**

If the second radio button is set (Delete *@*, then add from the file if specified):

> **DELETE: 14 subscribers removed.**
>
> **ADD: no error, 38 recipients added, no entry changed, no duplicate, none forwarded.**

(If the second button is set and no input file is specified, you will only get the DELETE: message.)

If the third radio button is set (Delete the users in the uploaded file):

> **DELETE: 93 subscribers removed.**

If the fourth radio button is set (Delete the users in the uploaded file from all local lists to which they are subscribed):

> **DELETE: 243 subscribers removed.**
>
> **DELETE: 109 subscribers removed.**
>
> **Global deletion process complete, 352 entries removed.**

If you do not supply an upload file where required, or if your browser does not support the RFC1867 file upload extension, you get the following message:

> **Your browser did not upload any file during the transfer. Assuming you did fill in**

the file input box, the most likely cause is that your browser does not support the file upload extension (RFC1867).

11.10. Sending interactive commands via the web

Clicking on the "Command" button brings up a simple, single-line interface where LISTSERV commands may be typed in for interactive execution (similar to the `LCMD` command line utility).

11.11. Mail merge

Starting with LISTSERV 1.8d, advanced mail-merge features are available and can be accessed either by sending specially-formatted DISTRIBUTE jobs to LISTSERV or by using the web administration interface. The web interface is not a "wizard" but simply an interface that allows you to "cut and paste" a mail merge message and select different standardized groups of list subscribers to whom the message is to be sent.

Note: The use of mail-merge functions by list owners must be enabled by the server administrator.

Mail merge functions are documented fully in the *Developer's Guide for LISTSERV*, available separately.

Appendix A: LISTSERV Command Reference for LISTSERV® version 1.8e

(For a quick reference card of LISTSERV commands, send the command INFO REFCARD to LISTSERV.)

This appendix is divided into two parts, corresponding to those commands available for use by the general user and for use by list owners and file owners. Non-privileged users can send commands by mail or by interactive commands. (Note that interactive commands can only be sent if a two-way NJE or MSGD connection exists.) Privileged users can send commands by mail, interactive commands (subject to the same restriction previously noted) or via the console (VM) or the LCMD utility (non-VM).

Unless otherwise noted, commands are listed in alphabetical order, with the minimum acceptable abbreviation in capital letters. Angle brackets are used to indicate optional parameters. All commands which return a file accept an optional '**F=fformat**' keyword (without the quotes) that lets you select the format in which you want the file sent; the default format is normally appropriate in all cases. Some esoteric, historical or seldom-used commands and options have been omitted.

Note that some commands are not available on all platforms; these commands are marked appropriately.

Continuation cards can be used to split long commands (for instance, ADD commands for users with long X.500 addresses) into two or more 80-character cards. In that case you must insert "// " (two slashes followed by a space) before the command text and a comma at the end of each line of the command so that LISTSERV considers it as a control card and performs the required concatenation. For instance,

```
// QUIET ADD MYLIST someone.with.a.real.long.userid.that.wraps@hishost.com ,
His Name
```

or, for instance, for a large **GETPOST** job,

```
// GETPOST MYLIST 10769-10770 10772 11079 11086 11095 11099-11100 11104 ,
11111 11115 11118 11121 11124 11131 11144 11147 11153 11158 11166 11168
```

Be sure to put a space before the comma at the end of the first line, as LISTSERV will not add the space for you.

Where mutually-exclusive command options are accepted, they will be separated by a logical OR sign (|). Non-required command options are indicated by enclosure in square brackets (**[]**); for instance, in the case of

```
SUBscribe listname [full_name│ANONYMOUS] [WITH options]
```

only the first token (the name of the mailing list) is required. Do not type the brackets when using the non-required options! Where the use of square brackets and logical OR signs together could be confusing, we have shown each of the alternate configurations on separate lines.

A.1. General Commands

A.1.1. List subscription commands (from most to least important)

SUBscribe *listname* **[***full_name* **|** **ANONYMOUS] [WITH** *options***]**

> The **SUBscribe** command is LISTSERV's basic command, issued by users to join mailing lists. This command can also be used to change one's "full_name" field in LISTSERV's SIGNUP database (simply reissue the command with the changed name). Note that the *full_name* is not required if the user has previously signed up to lists on the same LISTSERV server, or if the user has previously registered in LISTSERV's SIGNUP database by using the **REGISTER** (q.q.v.) command.
>
> LISTSERV 1.8c and later supports the following syntax:

> **SUBSCRIBE** *listname* **ANONYMOUS**

> This indicates that the user wishes to join the list anonymously, that is, without specifying a name. The **CONCEAL** subscription option is automatically set, granting the subscriber the maximal level of protection available.
>
> LISTSERV 1.8d and later supports the following additional syntax:

> **SUBSCRIBE** *listname* *full_name* **WITH** *option1* *option2* **...**

> This syntax allows you to "preset" subscription options at subscribe time. For instance, you might want to subscribe to MYLIST-L in order to be able to search its archives, but don't want to receive postings. You would use the command

> **SUBSCRIBE MYLIST-L Joe User WITH NOMAIL**

> Or you might want to receive individual postings with the **SUBJecthdr** option and receive copies of your own postings instead of the standard acknowledgement that your message was distributed to the list:

> **SUBSCRIBE MYLIST-L Joe User WITH SUBJecthdr REPRO NOACK**

JOIN *listname* **[***full_name* **|** **ANONYMOUS]**

> **JOIN** is a synonym for **SUBscribe**.

SIGNOFF *listname***|*|* [(NETWIDE]**

> The **SIGNOFF** command allows the user to cancel his or her subscription to lists. **SIGNOFF** requires a qualifying parameter, as follows:

> | | |
> |---|---|
> | *listname* | Sign off of the specified list |
> | ***** | Sign off of all lists on that server |
> | *** (NETWIDE** | Sign off of all lists in the LISTSERV network |

> The "*** (NETWIDE**" parameter causes the LISTSERV server to forward a copy of the signoff request to all other registered LISTSERV servers. This is a good option for a user who is changing service providers or otherwise losing a specific address that will not be forwarded. Please note that this parameter will *not*

remove the user from non-LISTSERV lists or from LISTSERV lists running on non-registered sites.

LISTSERV will attempt to sign off the address it finds in the RFC822 "From:" line and will not "fuzzy match" for "similar" addresses.

UNSUBscribe *listname*|*|* **[(NETWIDE]**

UNSUBscribe is a synonym for **SIGNOFF**.

CHANGE *listname*|* *newaddr*

This form of the **CHANGE** command can be used by any subscriber. It must be sent from the currently-subscribed address and results in an OK confirmation request being sent back to that address. This cookie then MUST be confirmed by the currently-subscribed address, exactly as it was entered, or the command will fail. This is the only case where a 1.8d cookie must be confirmed by a specific address. Note that this assumes that the user still has login access to both addresses, or at least the ability to send mail from the old address.

SET *listname option1* **[***option2 ...***]**

Allows the user to change his or her subscription options without administrative intervention. The options available to be changed are as follows:

ACK	A mail message acknowledging the receipt and distribution of the user's posting is sent back to the user.
NOACK	No posting acknowledgement is sent. In general, this setting should only be used if the user has also set himself to REPRO, as it is desirable in most cases that some indication of whether or not the posting was received by LISTSERV be sent.
MSGack	An interactive message is sent to acknowledge receipt and distribution. Note that this works only if both the machine running LISTSERV and the user's machine have NJE connectivity (e.g., BITNET). If NJE connectivity is not available on both ends, this option is effectively the same as NOACK.
CONCEAL	Allows the user to be concealed from the REVIEW command. Note that the list owner or LISTSERV maintainer can always get the complete list of subscribers, regardless of this setting.
NOCONCEAL	"Unhides" the user
Files/NOFiles	These options toggle the receipt of non-mail files from the list. Note that this is NJE-specific, and thus obsolete for systems without NJE connectivity, but retained for compatibility.
Mail/NOMail	These options toggle the receipt of mail from the list. Users who will be away from their mail for an extended

period of time may prefer to simply turn the mail off rather than to unsubscribe, particularly if subscription to the list is restricted in some way.

Note that for backward compatibility, the command **SET** *listname* **MAIL** sent by a user who is set to DIGEST but not also set to NOMAIL will cause the user to be set to NODIGEST (the behaviour is identical for users set to INDEX but not to NOMAIL). **SET** *listname* **MAIL** sent by users set to DIGEST/NOMAIL or INDEX/NOMAIL will simply remove the NOMAIL setting and leave the user set to DIGEST or INDEX as the case may be.

DIGests/INDex/NODIGests/NOINDex
These options change the format in which list mail is received by the subscriber. **DIGEST** turns on digest mode, in which the subscriber receives a digest of postings at set times dependent on how the "Digest=" keyword of the list is set. **INDEX** turns on index mode, in which the subscriber receives a daily listing of subjects posted to the list, from which he or she may order postings of interest. **NODIGEST** and **NOINDEX** toggle the mode back to individual postings sent as received by LISTSERV. Note that these options are interrelated; setting one will negate another.

REPro/NOREPro   Causes LISTSERV to send you a copy of your own postings as they are distributed. Some users may prefer this behavior to the **ACK** option (see above).

MIME/NOMIME   Toggles MIME options on and off. Currently only digests are available in MIME format. If **DIGEST** mode is set, the user will receive a MIME digest instead of the regular plain-text digest. Note that you must have a mail client that supports MIME digests (Pegasus is one that does) or this setting will do you little good. This option is automatically set at subscribe time for users who send their subscription command using a MIME-compliant agent, unless "Default-Options= NOMIME" is specified for the list.

HTML/NOHTML   Toggle the HTML function for digests and indexes on and off. New in 1.8d.

TOPICS: ALL | [+/-]*topicname*
For lists with topics enabled (see the Topics= list header keyword), subscribe or unsubscribe to topics. For instance, if a list has topics SUPPORT and CHAT, a user could subscribe to CHAT by sending **SET TOPICS +CHAT** . Or the user could unsubscribe to SUPPORT by sending **SET TOPICS -SUPPORT** . Finally, the user can subscribe to all available topics by sending **SET TOPICS ALL** .

Options for mail headers of incoming postings (choose one):

FULLhdr	"Full" mail headers, (default) containing all of the routing information.
IETFhdr	Internet-style headers.
SHORThdr	Short headers (no routing information).
DUALhdr	Dual headers, useful with PC or Mac mail programs which do not preserve the RFC822 return email address.
SUBJecthdr	"Full" mail headers (like the default) except that setting this option tells LISTSERV to add the list's default subject tag to the subject line of mail coming from the list. (See the listing in Appendix B for "Subject-Tag=" for more information.) Note that if the user is set to SHORThdr (or any other header option other than FULLhdr), LISTSERV will automatically switch the user to FULLhdr, as subject tags require full headers. Under 1.8c subject tags are not generated for messages sent without an RFC822 "Subject:" header; starting with 1.8d a subject tag is generated (for subscribers with the **SUBJecthdr** option set) even if the original message had no "Subject:" header. To turn the subject tagging off, the user simply sends a new **SET** command with any of the other header options (e.g., **SET listname FULLhdr**) and the **SUBJecthdr** option is reset.
FULL822	Essentially the same as "full" mail headers, but with the important difference that the recipient's email address is specified in the "To:" line rather than the address of the list. "**FULL822**" headers should be used with extreme caution, as they cause LISTSERV to create a separate mail envelope with a single RFC821 RCPT TO: for each address so set. *This behavior can significantly affect the performance of both LISTSERV and of your external mail system.*
SHORT822	Essentially the same as "short" mail headers, with the same caveats as noted for **FULL822**.

Note that **FULL822** and **SHORT822** headers should only be used if a specific problem indicates that they might solve the problem. One possible use would be to determine which subscriber from a specific site is causing the site to throw back delivery errors if that site does not specify which RCPT TO: is generating the error. These headers should *never* be used by default.

> **Documented Restriction:** The use of the **SHORTHDR** or **DUALHDR** options will break messages that depend on MIME encoding, because these options strip the RFC822 headers that identify the message as a MIME message. SHORTHDR and DUALHDR were designed for the non-MIME mail clients which prevailed in LISTSERV's early history. As most mail clients today support MIME, the use of these options is now deprecated.

CONFIRM *listname1* [*listname2*]...]]

The **CONFIRM** command should be issued when LISTSERV requests it. A request for **CONFIRM** should not be confused with a "command confirmation

request" which requires an "OK" response. The **CONFIRM** command is used in two cases:

- When the list in question requires periodic subscription renewals (controlled by the **Renewal=** keyword). In this case, the amount of time between the request for confirmation and termination of the subscription is controlled by the **Delay()** parameter of the **Renewal=** keyword; the default is seven days.

- When LISTSERV's automatic address probing function fails and you receive a message to that effect. The response time is controlled by the settings of the **Auto-Delete=** keyword for the list in question.

A.1.2. Other list-related commands

INDex [*listname*]

The **INDEX** command sent to LISTSERV without further qualification sends back the contents of the "root" level archive filelist on VM systems (LISTSERV FILELIST) or archive catalog on non-VM systems (SITE.CATALOG plus the contents of SYSTEM.CATALOG).

If the **INDEX** command is sent with the name of a list (e.g., **INDEX MYLIST**) or the name of a special filelist or catalog file (e.g., **INDEX TOOLS** , if TOOLS FILELIST on VM or TOOLS.CATALOG on non-VM exists), LISTSERV sends back the contents of the specified filelist or catalog. Several possibilities exist:

- For mailing lists without an associated filelist or catalog, LISTSERV creates an index "on the fly" containing entries for the accumulated notebook archives for that list. If notebook archives are not enabled for the list, LISTSERV will respond, "This server does not have any file by the name '*listname*.filelist'."

- For mailing lists with an associated filelist or catalog, LISTSERV will append the "on the fly" index of notebook archives to the entries in the associated filelist or catalog. For instance, for a list called MYLIST with associated catalog MYLIST.CATALOG, **INDEX MYLIST** might return:

```
*
* MYLIST FILELIST from LISTSERV@LISTSERV.MYCORP.COM
*
* :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
*
*  The GET/PUT authorization codes shown with each file entry describe
*  who is authorized to GET or PUT the file:
*
*     ALL = Everybody
*     CTL = LISTSERV administrators
*     OWN = List owners
*     PRV = Private, ie list members
*     LMC = LISTSERV master coordinator
*     N/A = Not applicable - file is internally maintained by LISTSERV
*     MSC = Miscellaneous - contact administrator for more information
*
* :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
*
*
* Information files for MYLIST
*
* filename      filetype        GET PUT size (bytes) date       time
```

```
* --------      --------       --- --- ------------ ---------- --------
  MYLIST        FAQ            ALL MSC       22,528 1996-02-09 21:30:10
  MYLIST        WELCOME        ALL MSC          279 1998-02-02 09:59:44
  MYLIST        FAREWELL       ALL MSC           92 1998-02-05 11:06:14

*
* Archive files for the MYLIST list at LISTSERV.MYCORP.COM
* (monthly logs)
*
* filename      filetype       GET PUT size (bytes) date       time
* --------      --------       --- --- ------------ ---------- --------
  MYLIST        LOG9603        LOG OWN        8,668 1998-05-27 15:29:57
  MYLIST        LOG9605        LOG OWN        7,865 1998-06-29 08:43:26
  MYLIST        LOG9606        LOG OWN       17,298 1998-07-23 12:46:20
```
Figure 6.1. Sample output of an **INDEX** *listname* command.

- Lastly, for catalogs or filelists without an associated list, the INDEX command returns only the entries in the catalog or filelist, since there are no associated list archives to be indexed.

Under VM, instead of the size in bytes, three separate VM-specific columns are used. Please note the following definitions for them:

rec -fm	Indicates whether the file is in a fixed or variable record format
lrecl	Logical record length. For a file with fixed record format (F), this is the length of each record. For a file with variable record format (V), this is the maximum record length.
nrecs	Number of records (lines) in the file

Lists [*option*]

Access the global list of lists maintained by LISTSERV. If no options are specified, then LISTSERV returns only local lists, one line per list. The available options are:

Detailed	All local lists, complete with full header information.
Global *xyz*	Only those whose name or title contains 'xyz'
SUMmary [*host*]	Membership summary for all lists on specified host, or the host to which the command is sent if no host is specified
SUMmary ALL	Membership summary for all hosts (long output, send request via mail!)
SUMmary TOTAL	Membership totals only

"Lists Global" without a search string, which returns the entire list of lists, may no longer be issued by general users. If you are asked about this, you should advise users to use the "Lists Global /xyz" format to search the list of lists, or use L-Soft's CataList service at **http://www.lsoft.com/catalist.html**.

"Lists SUMmary", when issued to an unregistered host or to a host running in STANDALONE mode will generate the response "No information available yet -

please try again later." because the file required for this function does not exist.

Query *listname*

Query your subscription options for a particular list (use the SET command to change them). Using the "*" wildcard in place of the name of a single list queries subscription options on all lists on the server.

REGister *full_name* **| OFF**

Register's the user's full name field in LISTSERV's SIGNUP files, or changes the current value of that field. When a user's name is registered, he or she can omit the full name field from subsequent **SUBscribe** requests to that server. Sending "REGISTER OFF" to LISTSERV deletes the user's entry from the SIGNUP file.

REView *listname* **[(***options***]**

Get information about a list, assuming the list header keyword "Review=" is set appropriately. For general users, **REVIEW** does not return address information about subscribers who are set to **CONCEAL**. The options are:

BY *sort_field*	Sort list in a certain order:
Country	by country of origin (ISO country codes)
Date	by subscription date (newest to oldest)
Name	by user name (last, then first)
NODEid	by hostname/nodeid
Userid	by userid
BY (*sort_field1 sort_field2***)**	
	You can specify more than one sort field if enclosed in parentheses. For instance: **BY (NODE NAME)**
Countries	Synonym of **BY COUNTRY**
Topics	(1.8d and later) Adds a breakdown of subscribers per topic (if **Topics=** is defined in the list header) at the end of the subscriber list. If you just want the breakdown, use **REVIEW** *listname* **SHORT TOPICS** . This does *not* show topics by individual subscribers (see the **QUERY** command instead). If **Topics=** is not enabled for a given list then this option is ignored.
LOCal	Don't forward request to peers. This is only useful if the list is peered; normally it should not be necessary to issue this option.
Msg	Send reply via interactive messages (BITNET users only)
NOHeader	Don't send list header, just send the subscriber list
Short	Don't list subscribers, just send the header and the membership summary for the list.

Note that you can get a quick read of the number of subscribers on the list by sending the command **REVIEW** *listname* **SHORT NOHEADER**.

In 1.8d and later list owners and site maintainers may also use the additional option:

ALL	List concealed members (who will show up with

"[concealed]" next to their entry) as well as non-concealed members. (NOT available to general users even if **Review= Public**.)

SCAN *listname text*

Scan a list's membership for a name or address. Helpful if a user attempts to send a SET command or an UNSUB command and is informed that their address is not subscribed to the list. At the non-privileged level, this command will show all non-concealed entries that match the search text, assuming that the list is set to "Review= Public".

The following command is available on VM servers only:

Stats *listname* **[(***options***]**

Get statistics about a list. **NOT AVAILABLE ON NON-VM SERVERS.** This command is VM-specific, and was originally intended to return BITNET data. The single option is:

　　　　LOCal　　　　　　　Don't forward to peers

A.1.3. Informational commands

Help

Obtain a list of commonly-used LISTSERV commands. Also explains how to get the comprehensive reference card and tells who the (non-hidden) server manager(s) are.

Info [*topic***|***listname***]**

Order a LISTSERV manual, or get a list of available ones (if no topic was specified); or get information about a list. For **Info** *listname*, the text in the INFO template form of *listname*.MAILTPL is used; however, if *listname*.MAILTPL does not exist or does not contain an INFO template form, the INFO template form of DEFAULT.MAILTPL is used.

Query File *fn ft* **[***filelist***] [(***options***]**

(*Available only on VM*) Get date/time of last update of a file, and GET/PUT file access code. The single option is:

　　　　Flags　　　　　　　Get additional technical data (useful when reporting problems to experts)

RELEASE

Find out who maintains the server and the version of the software and network data files.

SHOW [*function***]**

Display information as follows:

```
ALIAS node1 [node2 [...]]
                    BITNET nodeid to Internet hostname mapping
DISTribute          Statistics about DISTRIBUTE
HARDWare or HW      Hardware information; what kind of machine
                    is LISTSERV running on?
LICense             License/capacity information and software
                    build date
LINKs [node1 [node2 [...]]
                    Network links at the BITNET node(s) in
                    question
NADs [node1 [node2 [...]]
                    Addresses LISTSERV recognizes as node
                    administrators for the specified site(s)
NODEntry [node1 [node2 [...]]
                    BITEARN NODES entry for the specified
                    node(s)
NODEntry node1 /abc*/xyz
                    Just the ':xyz.' tag and all tags whose name
                    starts with 'abc'
POINTs [ALL | list1 [list2...]]
                    Graduated (LISTSERV Classic) license point
                    information. This information can help you
                    plan orderly expansion of your site if you are
                    running with a graduated LISTSERV Classic
                    license. Under Lite this command shows
                    Classic point usage.
STATs               Usage statistics for the server (this is the
                    default option)
VERSion             Same output as RELEASE command
```

If no function is specified, the output is per **SHOW STATS**.

The following options are available for VM servers only:

```
BITEARN                 Statistics about the BITEARN NODES file
DPATHs host1 [host2 [...]]
                        DISTRIBUTE path from that server to
                        specified host(s)
DPATHs *                Full DISTRIBUTE path tree
FIXes                   List of fixes installed on the server (non-
                        VM see SHOW LICENSE)
NETwork                 Statistics about the NJE network
PATHs snode node1 [node2 [...]]
                        BITNET path between 'snode' and the
                        specified node(s)
```

6.1.4. Commands related to file server and web functions

GET

VM Syntax:
GET fn ft [filelist] [(options] [F=fformat] [SPLIT=integer]
Non-VM Syntax:
GET fn ft [catalogname] [([F=fformat] [SPLIT=integer]]

For non-VM see also below for special TCPGUI parameters.

Order the specified file or package from LISTSERV. The single option is for VM servers only and is:

`PROLOGtext xxxx`   Specify a 'prolog text' to be inserted on top of the file

Examples:

```
GET MYFILE TEXT
GET MYFILE TEXT MYLIST-L
```

Typically the filelist name or catalog name is the same as the name of the list to which the files belong. A password (PW=xxxxx at the end of the command) is required to retrieve any file that does not have a GET FAC of ALL. For more information on FAC (File Access Control) codes, see 8.3.5 of this manual (for VM) or 8.4.1 (for non-VM).

Do not use dots (periods) in the file specification when specifying the filelist or catalog name, as this will result in an error. For instance

```
GET MYFILE.TEXT MYLIST-L
```

will result in an error, whereas

```
GET MYFILE TEXT MYLIST-L
```

will not.

To control the format in which LISTSERV returns the file(s) to you, you can specify the `F=fformat` parameter. Supported formats are `Netdata, Card, Disk, Punch, LPunch, UUencode, XXencode, VMSdump, MIME/text, MIME/Appl, Mail`

To split very large files into manageable chunks, you can specify the `SPLIT=integer` parameter. The integer value is the size you want the chunks to be generated, in kilobytes. For instance if you were ordering a 2MB notebook log and wanted to break it into 100KB chunks, you would specify `SPLIT=100`. This is handy for people whose mail systems place a limit on the size of an individual mail message that may be received by a given user.

TCPGUI parameters for Change-Logs: Under non-VM LISTSERV 1.8e and later, the following syntax is accepted:

```
GET listname CHANGELOG (MSG [COLUMNS(colspec1 colfilter1
[colspec2 colfilter2[...]])
```

The design goal was to provide access via the TCPGUI (and thus the web interface) for change-log data. Further information can be found in the *Developer's Guide to LISTSERV.*

`GIVE`

VM Syntax:	`GIVE fn ft [filelist] [TO] userid@host`
Non-VM Syntax:	`GIVE fn.ft [TO] userid@host`
	`GIVE fn ft catalogname [TO] userid@host`

(Note: Prior to 1.8d this command is not available on non-VM servers.)

Sends a file stored in a LISTSERV file archive to someone else. For instance, you may want to send LISTSERV REFCARD to a new user. Rather than retrieving LISTSERV REFCARD and then forwarding it to the user, you simply issue a GIVE command to tell LISTSERV to send it directly. Note that the token "TO" is optional. Examples:

For LISTSERV running under VM:

```
GIVE LISTSERV REFCARD joenewuser@hishost.com
GIVE LISTSERV REFCARD TO joenewuser@hishost.com

GIVE README TEXT MYLIST-L joenewuser@hishost.com
GIVE README TEXT MYLIST-L TO joenewuser@hishost.com
```

For LISTSERV running on non-VM hosts there are two syntaxes, depending on whether or not you need to specify a catalog name for the file in question. Note that the only real difference is whether or not you are required to specify a dot between the filename and the extension. Examples are:

```
GIVE LISTSERV.REFCARD joenewuser@hishost.com
GIVE LISTSERV.REFCARD TO joenewuser@hishost.com

GIVE README TXT MYLIST-L joenewuser@hishost.com
GIVE README TXT MYLIST-L TO joenewuser@hishost.com
```

INDex [*filelist|catalog*]

Same as **GET xxxx FILELIST**. If no filelist is specified, the default is LISTSERV FILELIST (on non-VM, SITE CATALOG is returned as LISTSERV FILELIST in this case).

PW *function*

Define/change a "personal password" for protecting AFD/FUI subcriptions, authenticating PUT commands, and so on.

ADD *firstpw*	Define a password for the first time, or after a **PW RESET**. Requires confirmation via the "OK" confirmation method.
CHange *newpw* [PW=*oldpw*]	Change your existing password. If you do not include your old password for authentication, LISTSERV will require confirmation via the "OK" confirmation method.
REP *password*	Starting with 1.8d, this function is a hybrid of "**ADD**" and "**CHange**". If a password does not exist for the user, one will be added. If a password does exist for the user, it will be changed (with confirmation required via the "OK" confirmation method). "**REP**" was added primarily for use by the web archive and

administration interface but can be used in e-mailed **PW** commands as well.

    RESET   Reset (delete) your password. This function always requires confirmation via the "OK" confirmation method.

SENDme

Same as **GET**

The following commands are available on VM servers only:

AFD

Automatic File Distribution. The functions are as follows:

    ADD *fn ft* [*filelist* [*prolog*]]
        Add file or generic entry to your AFD list

    DELete *fn ft* [*filelist*]
        Delete file(s) from your AFD list (wildcards are supported)

    List   Displays your AFD list

For node administrators:

    FOR *user* ADD/DEL/LIST etc
        Perform requested function on behalf of a user you have control over (wildcards are supported for DEL and LIST)

FUI

File Update Information: same syntax as **AFD**, except that **FUI ADD** accepts no 'prolog text'

A.1.5. Other (advanced) commands

DISTribute *type source dest* [*options*]

Note: Starting with 1.8d, the ability to send DISTRIBUTE jobs is limited to LISTSERV Maintainers by default, and requires a password. This section is retained for compatibility with 1.8c and earlier, and for 1.8d and later servers which have the DISTRIBUTE security feature turned off.

Distribute a file or a mail message to a list of users (see the *Developer's Guide for LISTSERV* for more details on the syntax). The various parameters are, briefly:

Type:
    MAIL   Data is a mail message, and recipients are defined by '<dest>'

MAIL-MERGE	Data is a mail-merge message. See the *Developer's Guide for LISTSERV* for specifics.
FILE	Data is not mail, recipients are defined by '<dest>'
RFC822	Data is mail and recipients are defined by the RFC822 'To:'/'cc:' fields

Source:

DD=*ddname*	Name of DD holding the data to distribute (default: '**DD=DATA**')

Dest:

<TO> *user1 <user2 <...>>*	
	List of recipients
<TO> DD=*ddname*	
	Use a DD called *ddname* for the destination addresses, one recipient per line

Options for the general user:

ACK=NOne/MAIL/MSG	
	Acknowledgement level (default: **ACK=NONE**)
CANON=YES	'TO' list in 'canonical' form (**uid1 host1 uid2 host2...**)
DEBUG=YES	Do not actually perform the distribution; returns debug path information
INFORM=MAIL	Send file delivery message to recipients via mail
TRACE=YES	Same as **DEBUG=YES**, but file is actually distributed
AV=YES[,FORCE]	(1.8e Classic and later) Check the message for viruses. See the *Developer's Guide for LISTSERV* for specifics.

Options requiring privileges:

FROM=*user*	File originator
FROM=DD=*ddname*	One line: 'address name'

GETPost *listname post_number* [*post_number* [...]] [NOMIME]

GETPost is used after receiving the output of a **SEARch** command to retrieve the postings you want from the **SEARch** output. For instance, if you want postings numbered 1730, 1731, 1732, and 1840 from the MYLIST list, send the command

GETPost MYLIST 1730-1732 1840

GETPost is analogous to the VM database command **PRINT**.

In previous versions, the **GETPost** command returned messages that contained MIME attachments in their "raw" form, which could not be extracted automatically by MIME-aware mail clients. Customers who wished to use list notebooks to archive word-processing documents (for instance) found this to be a problem. From LISTSERV 1.8e, attachments returned in messages by way of the **GETPost** command will now display as inline clickable links in the individual messages.

Users of certain email clients (specifically Pine, which handles attachments in a secondary viewing area) may find the new format difficult to use. If preferred, the

pre-1.8e behavior may be reverted to by specifying "NOMIME" as the last parameter of the `GETPost` command.

Search

For lists running on VM servers, see also below at `DATABASE`.

The `Search` command syntax is similar to that of the `SEARCH/SELECT` commands in the "old" database functions. A very basic `Search` command for list MYLIST would look like this:

`Search` *search_string* `IN MYLIST`

You can also restrict your search by date, sender, or other criteria, for example,

`Search` *search_string* `IN MYLIST SINCE 96/01/01`
`Search` *search_string* `IN MYLIST WHERE SENDER CONTAINS ERIC`

etc. The specific syntax is outlined in LISTDB MEMO (available from LISTSERV with the command "INFO DATABASE") and in the *Developer's Guide for LISTSERV*. Note that the new `Search` command does not require a CJLI job framework to operate; simply send the `Search` command in the body of an email message to the appropriate server. LISTSERV will respond with an index of the postings matching your criteria and instructions on how to use the `GETPost` command to retrieve the posts you want.

SERVE *user*

Restore service to a user whose access to LISTSERV has been disabled. This generally occurs when a user has sent 51 incorrect commands in a row to LISTSERV, which LISTSERV interprets as a possible mail loop. (Note also that certain mail packages that send "Read:/Not Read:" notifications back to LISTSERV will trigger this scenario after 51 iterations. The best solution would be for the user to disable receipt notifications.) The user in question cannot restore his or her own service; this command must be issued from another userid. Note that if the user has been manually served out by privileged user (a LISTSERV maintainer), the `SERVE` command must be issued by a similarly-privileged user (who must also be a LISTSERV maintainer, although naturally the same user who issued the `SERVE OFF` command can issue the `SERVE` command). For 1.8d and later please note that the `THANKs` command will *not* reset the serve-off counter (so vacation messages or auto-replies that contain a sentence starting with something like "Thanks for writing" will not defeat the system and users sending them will eventually be served off instead of continuing to loop ad infinitum).

THANKs

You can send this command to check to see if the server is alive. If it is, the server politely responds, "You're welcome!".

The following commands are available only on VM servers:

DATAbase *function*

Access LISTSERV database(s). The functions are explained in detail in the

version of LISTDB MEMO available from VM servers, but the basic syntax is:

```
Search DD=ddname <ECHO=NO>
```
Perform database search (see the VM version of LISTDB MEMO for more information on this)

`List` Get a list of databases available from that server

`REFRESH dbname` Refresh database index, if suitably privileged

```
Dbase
```

Same as `DATABASE`

A.2. List Owner and File Owner Commands

A.2.1. File management commands (for file owners only)

`PUT fn ft <filelist <NODIST>>`

Update a file you own. Options are:

`<PW=password>` Supply your password for command authentication

The following options are VM-specific and will not work on the non-VM servers.

The "NODIST" option prevents AFD and FUI distributions when the file is updated. Other available VM only options include:

`<CKDATE=NO>` Accept request even if the current version of the file is more recent than the version you sent

`<DATE=yymmddhhmmss>` Set file date/time

`<RECFM=F <LRECL=nnn>>` Select fixed-format file (not to be used for text files).

`<REPLY-TO=userid>` Send reply to another user

`<REPLY-TO=NONE>` Don't send any reply

`<REPLY-VIA=MSG>` Request reply via interactive messages, not mail (Requires NJE connectivity)

`<"parameters">` Special parameters passed to FAVE routine, if any

Standard parameters supported for all files:

`TITLE=file title` Change file "title" in filelist entry

The following commands are available on VM servers only:

`AFD/FUI`

Automatic File Distribution privileged commands. In addition to the AFD/FUI functions listed above, a file owner may use the following function:

`GET fn ft <filelist>`
Get a list of people subscribed to a file you own

`GET fn FILELIST <(options>`

Special options for filelists:

CTL	Return filelist in a format suitable for editing and storing back
NOLock	Don't lock filelist (use in conjunction with CTL)

REFRESH *filelist* **<(***options***>**

Refresh a filelist you own. The single option is:

NOFLAG	Don't flag files which have changed since last time as updated (for **AFD/FUI**)

UNLOCK *fn* **FILELIST**

Unlock filelist after a **GET** with the **CTL** option if you decide not to update it after all

A.2.2. List management functions
Commands that support the QUIET keyword are marked (*)

ADD(*) *listname user* **[***full_name***]**
ADD(*) *listname* **DD=***ddname* **[IMPORT [PRELOAD]]**

The first syntax is used to add an individual user to one of your lists, or update his name field. Note that you can substitute an asterisk ("*****") for *full_name* and LISTSERV will substitute "<No name available>" in the list.

The second syntax is used for bulk ADD operations where a dataset (DD=*ddname*) is used add multiple users, one address/name pair per line. For bulk operations you may also use the **IMPORT** option, which implies a **QUIET ADD** (in other words you do not need to specify **QUIET** if you use **IMPORT**) and otherwise vastly speeds up the **ADD** process by loosening syntax checking and omitting success messages. The **IMPORT PRELOAD** option first appeared in 1.8d and is used to direct LISTSERV to preload the existing e-mail keys in memory before starting the transaction, which speeds the operation up considerably. This option is used primarily with DBMS lists to speed up bulk adds. **PRELOAD** is not necessary for traditional LISTSERV lists and does not normally lead to a significant performance improvement. However, when importing a new list (no existing subscribers), it does reduce CPU usage somewhat.

For a bulk **ADD** operation, the users are defined in a separate dataset beginning on the line following the **ADD** command. For instance,

```
ADD listname DD=ddname IMPORT
//ddname DD *
userid@host.com User Name
userid2@host.com User2 Name
... more address/name pairs, one per line ...
/*
```

Please see chapter 7.17, below, for specific instructions for bulk **ADD** operations.

ADDHere(*)

Same as **ADD**, but means "add the user on this peer, do not forward this request to a (possibly) closer peer". For non-peered lists, is functionally identical to **ADD**.

```
CHANGE(*) listname|* newaddr
CHANGE(*) listname|* oldaddr|pattern newaddr|*@newhost
```

The first form can be used by any subscriber and results in a cookie being sent to the new address. This cookie MUST be confirmed by the new address, exactly as it was entered, or the command will fail. This is the only case in 1.8d and later where a cookie must be confirmed by a specific address.

The list owner form does not use cookies but simply applies the standard "Validate=" rules (as for a **DELETE** command). You can specify a wildcard pattern for the old address and *@newhost for the new address to rename certain addresses to a new hostname. The CHANGE1 template is sent unless you specify **QUIET**.

Change log entries are made (**CHANGE** *oldaddr newaddr*) and there is a CHG_REQ exit point which allows you to reject the operation.

```
DELete(*) listname user [(options]
DELete(*) listname DD=ddname [BRIEF]
```

The first syntax is used to remove a single user from one of your lists, or from all local lists if *listname* is '*****' The available options are:

Global	Forward request to all peers
LOCal	Don't try to forward request to closest peer if not found locally
TEST	Do not actually perform any deletion (useful to test wildcard patterns)

The second syntax is used for bulk **DELETE** operations (similar to a bulk **ADD** operation). See chapter 7.17 of this manual for details. The single available option is:

BRIEF	Good for deleting wildcard patterns (such as ***@***) when you don't want a "userid@host has been deleted from list xxxx" for each user deleted. Returns instead only a count of the users that were deleted.

```
FREE listname <(options>
```

Release a held list. The single option is:

Global	Forward request to all peers

```
GET listname <(options>
```

Get a copy of a list in a form suitable for editing and storing the list and lock it so that other list owners can't modify it until you store it back (or until you or they issue an **UNLOCK** command). The options are:

Global	Forward request to all peers
HEADer	Send just the header; on the way back, only the header

	will be updated. This is the recommended way to modify your list header.
NOLock	Do not lock the list
OLD	Recover the "old" copy of the list (before the last **PUT**)

HOLD *listname* **<(***options***>**

Hold a list, preventing new postings from being processed until a **FREE** command is sent. The single option is:

Global	Forward request to all peers

Lists [*option***]**

Additional options available for list owners and moderators:

OWNed	Returns a list of local lists owned by the invoker.
MODerated	Returns a list of local lists that are moderated by the invoker.

MOVE(*) *listname user* **<TO>** *node*

Move a subscriber to another peer. Do NOT use this command to move users from one list host site to another during migration. It is strictly for moving subscribers from one peer to another peer.

listname **DD=***ddname*	Move several subscribers to various peers

PUT *listname* **LIST**

Update a list from the file returned by a GET command. This is the standard "PUT command" or "list PUT" referred to throughout this document.

Starting with LISTSERV 1.8d, use of the PUT command to store a list header with new subscriber data at the bottom (e.g., an attempt to add subscribers "on the fly") will result in only the header of the list being stored, and in the generation of the following warning:

```
WARNING: new  subscriber data was found  in the replacement list  you sent,
possibly due to the use of a signature file with an unusual separator line.
If  you really  meant to  update the  subscriber data,  please resend  your
request with the word "PUT" replaced  with "PUTALL". For now, only the list
header will be updated.
```

PUTALL *listname* **LIST**

Starting with 1.8d, this command allows you to **PUT** an entire list file, that is, the list header followed by the list of subscribers.

Documented restriction: **PUTALL** does NOT work with DBMS lists; only the header information is replaced. Subscriber information in the DBMS table is not changed. For DBMS lists where the subscriber information needs to be replaced *in toto*, either the DBMS should be manipulated with your regular DBMS tools or you should use **ADD IMPORT** .

Query *listname* **<WITH** *options***> FOR** *user*

Query the subscription options of another user (wildcards are supported).

> *** <WITH *options*> FOR *user***   Searches all the lists you own for the specified user(s) with the specified option(s).

SET(*) *listname options* <FOR *user*>

Alter the subscription options for other users (wildcards are supported when setting options for another user or set of users).

Additional options for list owners:

NORENEW/RENEW	Waive subscription confirmation for this user
NOPOST/POST	Prevent user from posting to list
EDITor/NOEDITor	User may post without going through moderator
REView/NOREView	Postings from user go to list owner or moderator even if user is otherwise allowed to post

UNLOCK *listname*

Unlock a list after a GET, if you decide not to update it after all, or unlock a list if it has been locked by another list owner or by the LISTSERV maintainer. Note that if you are not the person who originally locked the list, it is considered good practice to ask the person who originally locked the list whether or not they are done with the list before you unlock it.

The following commands are available only on VM servers:

EXPLODE *listname* <(*options*>

Examine list and suggest better placement of recipients, returning a ready-to-submit **MOVE** job.

BESTpeers *n*	Suggest the N best possible peers to add
Detailed	More detailed analysis
FOR *node*	Perform analysis as though local node were '*node*'
PREFer *node*	Preferred peer in case of tie (equidistant peers)
SERVice	Check to see that service areas are respected
With(*node1* <*node2* <...>>>)	
	Perform analysis as though specified nodes ran a peer
WITHOut(*node1* <*node2* <...>>>)	
	Opposite effect

Stats *listname* (RESET

Resets (BITNET) statistics for the list.

Syntax of parameters

filelist	=	1 to 8 characters from the following set: **A-Z 0-9 $#@+-_:**
fformat	=	Netdata, Card, Disk, Punch, LPunch, UUencode, XXencode, VMSdump, MIME/text, MIME/Appl, Mail
fn	=	(filename) same syntax as 'filelist'
ft	=	(VM "filetype" or VMS/unix/DOS "extension") same syntax as 'filelist'
full_name	=	firstname <middle_initial> surname (*not* your e-mail address); sometimes referred to as "your real name"

host   = Internet hostname
listname = name of an existing list
node   = BITNET nodeid or Internet hostname of a BITNET machine which has taken care of supplying a ':internet.' tag in its BITEARN NODES entry
pw   = 1 to 8 characters from the set: `A-Z 0-9 $#@_-?!|%`
user   = Any valid RFC822 network address not longer than 80 characters; if omitted, the 'hostname' part defaults to that of the command originator

Appendix B: List Keyword Reference for LISTSERV® version 1.8e

This document (reference number 0205-UD-01) is available separately. It can be retrieved in plain text from any server running L-Soft's LISTSERV® with the command INFO KEYwords, or may be downloaded from the URL

ftp://ftp.lsoft.com/documents/listkeyw.memo

The List Header

The list header contains configuration information for the list. To edit it, use the `GET` `listname (HEADER` command, edit the header, and send it back to LISTSERV with the `PUT listname PW=XXXXXXXX` command. For more details on this procedure, consult the *List Owner's Manual for LISTSERV*. If you have the web archive and administration interface installed, you can also do this via the web (see chapter 11 of this manual).

Each line of the header must begin with an asterisk ("*"). The first line of the header must contain the list title, which must fit on a single line and not exceed 40-50 characters. Succeeding lines hold list control keywords and their values. Any words in the list header followed by the "=" character are assumed to be keywords. Following the list of keywords, you may add a few lines containing a brief description of the purpose of the list. These lines must also begin with an asterisk ("*").

This document is a description of the list control keywords that appear in the header of each list. Whenever default values are supplied for the keywords, they are listed first in the description. Words in *italics* are "generic parameters" which define a set of possible values for a keyword operand, as described below:

Format of List Header Keyword Settings

List header keyword settings can be defined in any of the following formats (we are using three parameters for our examples; note that some keyword settings have more then three parameters and adjust accordingly):

```
* Keyword= parameter1,parameter2,parameter3

* Keyword=parameter1,parameter2,parameter3

* Keyword= parameter1, parameter2, parameter3

* Keyword = parameter1,parameter2,parameter3

* Keyword= parameter1
* Keyword= parameter2
* Keyword= parameter3
```

The last example above is useful when defining a keyword setting (for instance, Notebook=) which may contain a long directory path. When using this format, note carefully that the last parameter on the line MUST NOT be followed by a comma. LISTSERV will properly concatentate the parameters internally as if they had commas. For instance,

```
* Keyword= parameter1,parameter2
* Keyword= parameter3
```

and variations are also supported; for instance,

```
* Notebook= Yes
* Notebook= /home/listserv/archives/english101-fall2002
* Notebook= Weekly,Private
```

is perfectly legal.

LISTSERV reads list headers one line at a time, assuming that any word followed by an equal sign is a keyword, and then, starting at the equal sign, reads keyword parameter settings either until it encounters a space that is not preceded by a comma (the first parameter excepted), or until it reaches the next word in the line which it evaluates as being a keyword. Thus LISTSERV will completely ignore a keyword setting coded as follows:

```
* Keyword parameter1, parameter2, parameter3
```

because there is no equal sign following the keyword. (This is one way to comment out a keyword setting if you do not want to completely remove it from the list header.) Likewise, while LISTSERV will recognize the following as a keyword setting:

```
* Keyword= parameter1, parameter2 parameter3
```

it will read only to the second parameter because the second parameter is not followed by a comma.

It is also possible to define multiple keywords on a single line, so long as the line does not wrap and does not exceed 100 characters. For instance,

```
* Send= Private     Confidential= Yes     Review= Owners
```

is a legal LISTSERV header line containing settings for the Send=, Confidential=, and Review= list header keywords.

Keyword settings are always evaluated in a case-insensitive manner. Under unix, where case sensitivity is an important issue, file and directory paths defined in the list header are always converted to lower-case for LISTSERV's use. Thus all data files written to or read from by LISTSERV under unix must be named in lower case, regardless of the case used in the list header keyword setting.

Hiding header lines

Starting with LISTSERV 1.8d, it is possible to hide part or all of a list header (except for the list title) from users who send the REVIEW command or who try to view the list's configuration via the CataList. The following syntax is used:

```
* My very own list
*
* blah blah blah
*.HH ON
* This line is hidden
* This line is also hidden
```

```
*.HH OFF
* This line is not hidden
```

The sequence can be repeated as many times as required. **GET** will return the unedited header with the **.HH** sequences, **REVIEW** will replace hidden lines with a note saying that lines were hidden. You can't hide the fact that some lines were hidden because it would lead to people spending hours trying to figure out problems which only appear to be problems because some of the keywords are not visible. Please note that L-Soft will not field support inquiries with hidden headers; you must send the entire raw header (including the **.HH** lines) when requesting support.

Generic parameters

Note: Special parameters used by only one or two keywords are defined along with the keyword and do not appear in this listing.

net-address   Describes an Internet address, such as JACK@XYZ.COM.

access-level   Controls which category of users has access to the information or service to which this parameter applies. *access-level* can be either:

       Public   Everybody has access to the information.
       Postmaster Only the postmaster (i.e. LISTSERV operations staff) has access to the information.
       A1,A2,... with Ai being either:

           Private   Only users subscribed to the list have access to the information.
           (***listname***)   Only the subscribers of the named list have access to the information.
           Owner   Only the list owner can access the information.
           Owner(*list***)** Only the owner of the named list can access the information.
           Service   Only people in the service area of the list can see the information.
           Service(*list***)** Only subscribers of the named list's service area can see the information.

destination   Indicates the destination of a piece of mail, message or reply.

       List   The reply message is sent to the list.
       Sender   The reply message is sent to the sender of the original piece of mail.
       Both   The reply message is sent both to the list and to the original sender.
       None   No reply message is sent at all.
       "*address***"**   The reply message is sent to the specified network address if enclosed in double quotes

interval   Is a time interval that indicates how frequently an operation is to be renewed. Note that depending on the operation being performed, some of the options may not be available. For example, "Notebook= Yes,A,Daily" is not available.

       Yearly   }

Monthly	}	
Weekly	}	Self-explanatory
Daily	}	
Hourly	}	
Single	The operation is to be done only a single time.	

peer   Is the node-id or network address of a peer list. If the name of the peer list is the same as the name of the local list (which will usually be the case), only the node name needs be given. If the list names are different, the full list network address must be given, for example "REXX-L@UIUCVMD".

area   Is a means whereby a node or list of nodes can be identified. An area can be either:

- The name of a network, for example EARN, BITNET
- The name of a country, for example Germany, Canada
- 'Local', in which case it is equated to the value of the "Local=" keyword (q.q.v.).
- A node name, for example SEARN
- A simple wildcard nodename pattern such as FR*, *11, *ESA*, D*ESA*, etc.

mon-address   Is a means whereby 'list monitors' can be identified (the term 'list monitor' refers to a human person who monitors the activity of a list). A 'mon-address' can be:

- A single network address, for example INFO@TCSVM
- 'Postmaster', which indicates the "main" postmaster
- 'Postmasters', which indicates ALL the postmasters, main and alternate
- 'Owner', which indicates the "main" list owner (the first to be listed in the "Owner=" keyword)
- 'Owners', which indicates ALL list owners

Some keywords can take more than one parameter. Where multiple parameters are accepted, they will be separated by a logical OR sign (|). Unless specified otherwise, commas have "higher priority" than OR signs, that is to say, "Public|Private, Open|Closed" means "(Public|Private), (Open|Closed)", not "Public|(Private,Open)|Closed".

Optional parameters are indicated by enclosure in square brackets (**[ ]**); for instance, in the case of

Send= *access-level* **[,Semi-Moderated][,Hold][,Confirm]**

the only required parameter is the first one ("*access-level*"), and each of the following three parameters is optional. Do not type the brackets when using the optional parameters! Where the use of square brackets and logical OR signs together could be confusing, we have shown each of the alternate configurations on separate lines.

Keyword Classifications

Keywords fit into several different classifications. These classifications, and the associated keywords, are as follows:

Access Control Keywords (page 172)
Attachments=	Determines whether MIME attachments may be distributed to the list
Files=	Determines whether non-mail (NJE) files may be distributed by the list
Filter=	Gives list owners control over problem users and/or gateways
Review=	Restricts who may review the list of subscribers
Send=	Restricts who may send postings to the list
Stats=	Determines whether or not list statistics are available, and to whom

Distribution Keywords (page 179)
Ack=	Controls the level of acknowledgement messages to those posting messages
Daily-Threshold=	Limits the total number of messages that will be processed by the list per day before the list is held
Digest=	Controls the automatic digestification option
Internet-Via=	Determines through which gateway Internet mail will be sent
Mail-Via=	Determines how LISTSERV distributes list mail
Newsgroups=	Defines USENET newsgroups linked to the list
NJE-Via=	Determines through which gateway NJE mail will be sent
Prime=	Controls whether or not mail will be processed during "prime time"
Reply-To=	Sets a default for the "Reply-To:" field in the header of list mail
Sender=	Defines the value LISTSERV places in the "Sender:" header field of list mail
Sub-lists=	Defines sub-lists of a "container" or "super-" list.
Topics=	Defines up to 23 sub-topics for a list

Error Handling Keywords (page 189)
Auto-Delete=	Sets parameters for the auto-deletion feature
Errors-To=	Determines the network address to whom mail delivery errors are directed
Loopcheck=	Defines the type of mailing loop checking performed by LISTSERV
Safe=	Determines which built-in address filter is used by LISTSERV

List Maintenance and Moderation Keywords (page 194)
Editor=	Defines an editor or editors for moderated lists
Editor-Header=	Controls if an explanatory mail header is added to list messages forwarded to the list editor (if one is defined)
List-Address=	Determines how the list address is announced in message headers
List-ID=	Defines a long listname alias for the list
Moderator=	Defines the editors on moderated lists who will receive postings for approval.
New-List=	Sets forwarding when a list is moved to a different LISTSERV host
Notebook=	Controls the notebook archive for a list
Notebook-Header=	Determines the type of header information included in the notebook archive
Notify=	Defines whether or not (or to whom) subscription notification is sent
Owner=	Defines the owner (or owners) of the list
Peers=	Defines peers for the list
Renewal=	Controls whether or not subscription renewal is implemented, and how
Sizelim=	Controls the maximum size of any single message posted to the list
Subject-Tag=	Controls the "subject tag" text for messages coming from the list
X-Tags=	Controls whether "X-to:" and "X-cc:" tags are included in list mail headers

Security Keywords (page 204)
Change-Log=	Enable logging (in *listname*.CHANGELOG) of all subscription changes
Confidential=	Determines whether or not an entry for the list appears in the List of Lists
Exit=	Defines a list "exit" which modifies the default behavior of LISTSERV
Local=	Defines which nodes are considered "local" for this list
PW=	Sets a password used for validation of list maintenance commands
Service=	Defines an area or areas outside which subscription requests are not accepted
Validate=	Determines whether or not list commands must be validated with a password or

the "OK" mechanism

Subscription Keywords (page 212)

Confirm-Delay=   Defines a default number of hours LISTSERV holds jobs requiring confirmation
Default-Options= Defines what options should be set by default for new subscribers
Default-Topics=   Defines what topics should be set by default for new subscribers
Subscription=   Defines how new subscriptions are handled, and if confirmation is required

Other Keywords (page 216)

Categories=   Defines search categories for the CataList service
DBMS=   Controls DBMS features
Indent=   Defines the minimum number of columns allowed for list addresses in a REVIEW
Language=   Defines the language in which information mail and messages are sent
Limits=   Defines certain limits (no. of subscribers, etc.) for a list (ISP scope option only)
Long-Lines=   Controls whether long-lines support is enabled
Mail-Merge=   Controls whether or not list postings are treated as mail-merge jobs
Translate=   Controls how LISTSERV handles control characters in list mail

Default Values for All Keywords (page 220)

Access Control Keywords

Attachments= No[,Filter]
Attachments= Yes[,*allowed_content_types*[,Filter]]
Attachments= All[,*allowed_content_types*[,Filter]]

LISTSERV 1.8d 2000b "level set" and LISTSERV 1.8e and later include a list-owner-configurable message attachment filter. This feature allows you to control the posting of various types of MIME attachments (images, audio, etc.) to your lists. LISTSERV 1.8e adds the ability to control the posting of inline uuencoded files to your lists on an on/off basis (off being the default if attachment control is enabled).

NOTE: The ability of LISTSERV to filter or reject messages that contain MIME attachments is completely dependent on the ability of the poster's mail client to properly identify the MIME attachment when the mail is originally sent. Filtering/rejection is done based on the Content-Type headers found in the message--NOT by evaluation of the actual contents of the attachment. If for instance an executable binary (normally Content-Type: application/octet-stream) is sent by the client with a Content-Type of "text/plain", it will not be filtered or rejected by LISTSERV since (as noted below) text attachments are not covered by this keyword setting.

A registry of allowable MIME types for attachments, maintained by the Internet Assigned Numbers Authority (IANA) per RFC2048, can be found at

http://www.isi.edu/in-notes/iana/assignments/media-types/media-types

The options are:

`Attachments= Yes`   All types of attachments are allowed to be posted to the list (the default). Note however that other configuration options may still disallow the posting of certain attachments, and that "Attachments= Yes" does not override them.   For instance, if you have "Language= NoHTML", setting "Attachments= Yes" does not override the Language= setting. Or if you have "Sizelim=" set to a value that precludes a file of x number of lines from being posted to the list, setting "Attachments= Yes" will not override the Sizelim= setting if the message with its attachment exceeds the number of lines specified by Sizelim=.

`Attachments= No`   All types of attachments are disallowed, other than plain text (always allowed) and HTML text (which is controlled exculsively by the "Language= NoHTML" keyword setting). With "Attachments= No", LISTSERV rejects messages containing attachments and bounces them back to the poster.

`Attachments= No,Filter` Same as "Attachments= No", except that LISTSERV simply removes the unwanted material from the message and processes it instead of rejecting it out of hand. The removal of material is a silent operation and the poster is not notified that the attachment was discarded.

In LISTSERV 1.8e and later, note that in all three of the above cases, when a message containing one or more uuencoded files is posted to the list, the encoded file(s) is/are stripped from the body of the message and the remainder

of the message is passed through to the list. LISTSERV 1.8d was unable to recognize or strip inline uuencoded files.

`Attachments= All`   (Requires 1.8e or later) This setting tells LISTSERV to *allow* inline, uuencoded files, such as are generated by Microsoft Outlook, overriding the default.

> *One important restriction:* UUencode filtering is strictly on/off. There is no attempt on the part of LISTSERV to guess file types when filtering is enabled (the default). This would be hazardous to begin with as support for these attachments is usually provided on a legacy basis in mail clients; that is, client A and client B could have a very different opinion on the type of the attachment.

It is also possible to allow certain MIME types to be passed through to the list while rejecting or filtering all others. For instance,

    Attachments= Yes,image,application/*msword

allows only the specified attachment types and rejects everything else. If you don't want to reject messages that contain other types of attachments, but just want to remove all other types of attachments, you add the ",Filter" parameter at the end of the line--ie,

    Attachments= Yes,image,application/*msword,Filter

This means, "Allow all image and application/*msword attachments, and strip all other attachments".   Again, note that plain text ("Content-Type: text/plain") is always allowed and does not need to be included in the list of allowed attachment types.   Likewise, HTML text is controlled exclusively by the "Language= NoHTML" keyword setting.   Other text subtypes are, however, controlled by "Attachments=", so they need to be listed if you intend to allow them.

Additionally, should it be desired to allow all inline uuencoded files but restrict the list to certain MIME types, you can specify, similar to the above, something like

    Attachments= All,image,application/*msword

or

    Attachments= All,image,application/*msword,Filter

(In the preceding examples note carefully that "image" by itself is equivalent to "image/*"--in other words, when you code "Attachments= image", you are saying that all MIME image sub-types, for example, "image/jpeg", "image/gif", and so forth, are to be accepted. If only certain sub-types are acceptable, for instance if you want to accept only JPEG graphics and ensure that others don't go through, you must specify the types explicitly--eg "Attachments= image/jpeg".)

Note carefully that simply coding something like "Attachments= image" will not necessarily allow all image files through. This is highly dependent on the client being used by the poster. For instance, if your client attaches all binary files as "Content-Type= application/octet-stream", regardless of whether a given binary is (for instance) an executable image, a Word file, or a compressed archive, and you send a JPEG to a list with "Attachments= image" set in the header, it will be

rejected since the image does not have a "Content-Type: image" tag. Specifically this appears to be the case with Eudora 3.x but may not be limited to that particular client.

In the 1.8d version of the attachment filter, attachments sent by Microsoft Outlook cannot be blocked by LISTSERV as they do not follow MIME standards (at least not up to and including Outlook 97; this writer has not installed Outlook 2000). Outlook sends attachments as imbedded uuencoded files and does not use the MIME Content-Type: header. LISTSERV 1.8e is able to recognize and filter inline uuencoded "attachments" as noted above.

The rejection message sent by LISTSERV when ",Filter" is not specified is found in the **BAD_ATTACHMENT** mail template form (see chapter 9 for information on LISTSERV's mail templates). Note that the **BAD_ATTACHMENT** template form is a linear template and as such does not allow text formatting commands to be used.

The reason HTML text is not subject to "Attachments=" filtering is to allow you to reject (bounce) messages with attachments, while silently suppressing HTML text in multi-part messages which also contain a plain-text alternative. Some mail programs send both HTML and plain-text versions of messages, and, even if you do not want HTML text on your list, there is little point in keeping out people who use it (who are often new to the Internet and aren't aware that their mail programs are sending HTML text) when you can simply remove the HTML part. At the same time, you may want to reject postings containing images out of hand, rather than removing the images and continuing. The same applies to Exchange attachments, which are filtered by default (see "Language= Exchange").

Files=Yes | No

This keyword is not available in LISTSERV Lite.

(NJE only; obsolete in other versions) Indicates whether NJE files can be sent to the list or not. The default value is "No". "**Files= No**" may prevent some non-RFC822 mailer users from posting to lists.

Files= has absolutely no effect under the non-NJE versions of LISTSERV. Specifically it will not prevent users from sending "attached" (MIME-encoded) files to lists. It is provided under all versions for backwards compatibility only (i.e., for lists being migrated from NJE servers). See the **Attachments=** keyword for attachment blocking.

Filter= Only,net-address1,net-address2,...[Allow],net-addressn,... |
Filter= Also,net-address1,net-address2,...[Allow],net-addressn,... |
Filter= Safe,net-address1,net-address2,...[Allow],net-addressn,... |

"**Filter=**" is checked when a user attempts to post or subscribe to a list (but not when the list owner issues an **ADD** command). The first parameter of this keyword is either "Only", "Also" or "Safe", and it is followed by a list of patterns such as 'X400MAIL@*' or '*@*.XYZ.EDU' (without the quotes).

- If "Filter= Also..." is specified, your filter is used in addition to the standard LISTSERV filter; this is useful to register additional looping mailers, to prevent users behind broken gateways from subscribing until the problem is addressed, or to ban anonymous posters.

- If "Filter= Only..." is specified, the addresses you specify are the only ones which LISTSERV prevents from posting to the list. CAUTION: You should not use this option unless you also code "Safe= Yes", and even then you will want to ask your LISTSERV maintainer for permission. This option has been added mostly for LISTSERV maintainers with very specific problems to solve. The minimal filter is very small and you should never need to override it.

- If "Filter= Safe" is specified, LISTSERV uses the "more safe" of its two built-in filters. These built-in filters are (1) a "minimal" one, which is used for safe lists; and (2) a "safe" one which is used for lists running with "Safe= No". That is, the unsafe lists need a safe filter to avoid mailing loops; safe lists only need the minimal filter, but can be made even safer by selecting "Filter= Safe". This, however, prevents usernames such as 'root' from posting to the list, because they are included in the safe filter.

  Messages sent to the LISTSERV userid for execution are always checked with the minimal filter, as people with userids such as 'root' would otherwise not be allowed to subscribe to lists which were set up to allow them.

In all cases, LISTSERV extracts as many e-mail addresses as it can from the userid being checked and runs them all through the filter. For instance if your list receives mail from 'searn.sunet.se!mailer@xyz.edu', LISTSERV will check 'searn.sunet.se!mailer@xyz.edu', 'mailer@searn.sunet.se' and 'mailer@searn' (via the '`:internet.`' tag in `BITEARN NODES`).

Version 1.8d adds the ability to "exempt" certain addresses (or wildcards) from the default filters. You can combine "`Filter= ...,Allow,...`" with the forms documented above as long as you put the "allow" information last. That is, the first word must be `ONLY`, `SAFE` or `ALSO` as before, and you can then have `ALLOW` anywhere (including as the second word) followed by a list of addresses that should be allowed even if present in the filter. The default for `ALLOW` is the empty string. Examples of `ALLOW` usage follow:

```
Filter= Also,userid@host1.com,*@host2.edu
Filter= Allow,niceguy@host2.edu
```

The first example stops everyone from host2.edu from posting to the list, but since we've determined that niceguy@host2.edu is a considerate human being and should be allowed to post, we've defined him as an exception to the general rule by including him in the "Allow" part of the filter.

```
Filter= Safe,Allow,root@host1.edu
```

The second example would invoke the "safe" filter mentioned above, but would allow root@host1.edu to subscribe to and post to the list, instead of bouncing his mail because it matches one of the entries in the "safe" filter. All other "root" users' mail will be caught by the "safe" filter and transferred to the list owner as an error.

See also `Safe=` and `Loopcheck=`.

Review= *access-level*
This keyword defines the categories of users who are allowed to review the (non-

concealed) Internet addresses and names of the persons subscribed to a list. Beginning with version 1.8c, the default value is "**Review= Private**".

Send= *access-level* **[,Semi-Moderated][,Hold][,Confirm][,NoMIME]**

Defines the categories of users who can mail or send files to the list. Possibly puts the list under control of an editor. The default value is "Public". Other *access-levels* for use with Send= would include "Private", "Editor", "Owner", etc. (see the beginning of this document for the definition of an *access-level*). A literal Internet e-mail address may also be used in place of the *access-level*, for example, **Send=joe@foo.bar.com**. Using a literal address is one way to ensure that only an authorized person can post to the list, for instance, if the list is an "announce-only" list rather than a discussion list.

When the list is controlled by an editor (**Send= Editor**), any file or piece of mail sent to the list is forwarded to the editor, who is the only person (with the list owner) to be able to actually mail or send files to the list. The network address of the editor is defined by the "**Editor=**" keyword (see below under "List Maintenance and Moderation").

When the "Semi-Moderated" option is enabled (**Send= Editor,Semi-Moderated**), mail sent to the list will be treated in one of two different ways, depending on the contents of its "Subject:" field. If the subject starts with "Urgent:" (case-independent), the list is treated as a non-moderated one, which means that the message will be immediately distributed provided that the sender matches the access-level description. If the subject does not start with "Urgent:", the message is forwarded to the primary list editor (unless it came from someone defined as an editor). A "Subject:" field beginning with "Re: Urgent:" is treated identically, so that replies to urgent messages are by default considered urgent.

Note that

* Send= Public,Semi-Moderated

is a contradiction. If **Send= Public**, no Editor is involved and anyone can post to the list, so Semi-Moderated is ignored.

An example:

* Send= Private,Semi-Moderated
* Editor=NATHAN@EXAMPLE.COM,ERIC@EXAMPLE.COM

In this example, a message sent to the list would be:

- Discarded, if the sender was not subscribed to the list, regardless of the subject
- Processed, if the sender was subscribed and used the "Urgent:" subject
- Forwarded to the moderator if the sender was subscribed but didn't use the "Urgent:" subject.

Another example:

* Send= Editor,Semi-Moderated
* Editor=NATHAN@EXAMPLE.COM,ERIC@EXAMPLE.COM

In this example, a message sent to the list would be:

- Processed, if the sender used the "Urgent:" subject
- Forwarded to the moderator if the sender didn't use the "Urgent:" subject.

Note that in the above example, messages don't get discarded if the sender isn't subscribed.

When the "Hold" option is enabled (`Send= Editor,Hold`), the moderator(s) may approve postings using the "OK" mechanism rather than forwarding the posts back to the list. "Hold" is valid only with "Editor".

When the "Confirm" option is enabled (e.g., `Send=Editor,Confirm`), mail sent by any editor or moderator requires confirmation by that editor or moderator. This is to prevent a user from forging mail to the list under an editor's or moderator's return address. The confirmation request is validated with the "OK" mechanism. Please note that this does not mean that a double validation is required when an editor approves other peoples' postings, but only that the editor's own postings to the list and any reposts he does on others' behalf require a confirmation from him that he actually submitted them. In other words if the editor simply "OK"s a posting, he doesn't have to "OK" his own "OK".

It is also possible to set a list to

`* Send= Editor,Hold,Confirm`

This allows you to "OK" both subscriber submissions and editor/moderator approvals, as described above.

Moderation "OK" requests and MIME attachment display: In versions previous to LISTSERV 1.8e, an OK confirmation request for a message coming to a moderated list displayed the message to be approved in its "raw" format, i.e., there was no attempt made to display/decode MIME attachments that might be present in the message to be approved. LISTSERV 1.8e addresses the problem by including a copy of the first text/plain part (if one exists in the message) for the purpose of quick screening. The following restrictions apply:

1. This is only done for MIME messages (even simple single-part ones, but they must have MIME headers).

2. The text part in question is sent pretty much 'as is', that is, as an extra text/plain part in the message, with all the options and encoding and what not supplied in the original message. The reason is quite simply that it would be a lot of work and, in some extreme cases (incompatible code page, etc), completely impossible, to embed it into the first text/plain part with the LISTSERV message. The drawback is that some mail agents might conceivably only show the first part until you take some kind of clicking action.

It is important to understand that only the first text/plain part is extracted in this fashion. The goal was to make it easier to approve or reject simple text messages, not to build a factory around a simple problem. The ENTIRE message is available at an extra click.

Where security is a concern, it is important to review the ENTIRE original message and not just the plain text part. There could be an obscene GIF or another text part or a text/html part not matching the contents of the text/plain

part or whatever. This is why, again, you are given the ENTIRE original message.

List owners using certain email clients (specifically Pine, which handles attachments in a secondary viewing area) may find the new format difficult to use. If preferred, the pre-1.8e behavior may be reverted to by specifying "NOMIME" in the Send= list header keyword; for instance,

```
* Send= Editor,Hold,NoMIME
```

Stats= Normal | None,*access-level*

This keyword is not available in LISTSERV Lite.

This keyword is obsolete and has absolutely no effect on all ports of the software except for VM. On non-VM servers it is provided for backwards compatibility only (i.e., for lists being migrated from VM) in order that any existing "Stats=" keyword setting in a migrated list header does not trigger a command parser error.

UNDER VM ONLY, indicates whether or not statistics are to be maintained for the list and if yes, which level of statistics is desired and who is able to retrieve the statistics reports. The default value is "Normal,Private".

Normal statistics include number of mailings for each user on the list, and similar information for file distribution.

Distribution Keywords

Ack= Yes | Msg | No | None

Defines the default value of the "ACK/NOACK" distribution option for the corresponding list, that is, the value assigned to new users when they subscribe to the list. This value can be altered by subscribers ("`SET`" command), but not by users who are not signed on to the list. This means that this option will always be in effect when distributing mail from people who are not on the distribution list. The default is "`Ack= No`".

Yes	A short acknowledgment with statistical information on the mailing will be sent back to you.
Msg	Messages will be sent when your mail file is being processed. Statistical information will be sent via messages, but no acknowledgment mail will be sent. [BITNET only]
No	For Internet users, no acknowledgement will be sent. For BITNET users, a single interactive message will be sent as the message is processed. This is the default value.
None	No messages of any kind are sent when your mail file is processed. [same as No for non-BITNET]

Daily-Threshold= *nnn1[,nnn2]*

This keyword limits the number of postings that may be processed by the list in a calendar day (midnight to midnight, server time), and, with the addition of a (new) optional second parameter, limits the number of postings that may be accepted from any individual user per calendar day (midnight to midnight in the server's local time zone).

The default is `Daily-Threshold= 50`. When the value of the first parameter is reached, the list is automatically placed on hold, and the list owner or LISTSERV maintainer must issue the `FREE` *listname* command. Note that it may or may not be advisable to increase this parameter for higher-volume lists – individual list owners should study the issue carefully before increasing the daily threshold of their high-volume lists.

When the value of the optional second parameter is reached by an individual user, the user is told that their posting will not be processed and that they should resend it later if they still want it to be posted. The list itself is not held in this situation. The default is to have no such limit, in which case the second parameter is not defined. Note that list owners and list editors are exempt from the individual daily limit. There is no command to reset the limit for an individual user, although the list owner may update the header to increase the value.

Digest= No
Digest= Yes,*where* | Same[,*frequency*][,*when*][,Size(*maxsize*)][,BOTTOM_BANNER]

This keyword controls the automatic digestification function allowing subscribers who do not have the time to read large numbers of messages as they arrive to subscribe to a digestified or indexed version of the list. The list owner decides whether digests are available or not, the frequency at which they are issued and the day of week or time of day when the digest should be distributed.[10]

Digests are larger messages containing all the postings made by list subscribers

---

[10]The digests conform to RFC1153 with an acceptable deviation from the recommended subject line (verified with the RFC author).

over a certain period of time. Unlike real-world digests, LISTSERV digests are not edited; what you see is exactly what was posted to the list. The only difference is that you get all the messages for a given day, week or month in a single batch. This is mostly useful if you are just "listening in" to the list and prefer to read the postings at your leisure. Digests are kept separately from list archives and can be made available for mailing lists which do not archive postings (i.e. which run with "`Notebook= No`").

Indexes, on the other hand, only provide a few lines of information for each posting: date and time, number of lines, name and address of poster, subject. The actual text is not included. You select just the messages you are interested in, and order them from the server. This is useful for mailing lists where most messages really don't interest you at all, or as an alternative to **SET NOMAIL**: when you come back from vacations, you can quickly order the messages you are most interested in. Note that, since indexes are not useful without the ability to order a copy of the messages you do want to read, they are not made available unless the list is archived and digests are enabled.

Users sign up for digestified rather than immediate delivery with '**SET** *listname* **DIGests**', while indexes are selected with '**SET** *listname* **INDex**'. These two options are alternatives to **MAIL** and **NOMAIL**. When switching around between these delivery options, users will observe the following behavior (digests will be assumed to be daily for the sake of clarity):

- When switching to **NOMAIL**: delivery stops immediately. The day's digest is not sent, as the user is assumed to desire immediate termination of traffic from the list. (Note that any mail already "in the pipeline" to the subscriber will still be delivered.)

- When switching from any option to **DIGEST** or **INDEX**: mail delivery stops immediately, and the first index or digest may contain some items the user has already seen (if switching from **MAIL** to **DIGEST/INDEX**). This is because the digests and indexes are global to the list - they are the same for everyone, just like regular issues of newspapers.

- When switching from **DIGEST** or **INDEX** to **NODIGEST** or **NOINDEX**, the current, unfinished digest or index is immediately mailed to the user. New messages are delivered normally, as they arrive. Thus, a "trick" to get a copy of the current digest is to switch to **NODIGEST** and then back to **DIGEST**. You can send both commands in the same mail message to make sure they are executed together.

The list owner controls the availability and frequency of digests through the "`Digest=`" list header keyword, which defaults to "`Digest= No`" for lists without an archive and "`Digest= Yes,Same,Daily`" for archived lists. Again, it is not necessary for the list to be archived to keep a digest; LISTSERV just attempts to avoid having to store large amounts of digest data on its private area for lists which, lacking a "`Notebook= Yes,xxx`" keyword, do not specify any suitable directory for the digest data. Conversely, having daily as the default frequency keeps the additional cost in disk space to a minimum.

The syntax of the keyword is "`Digest= Yes,where,frequency,when,maxsize`" when digests are enabled, or then "`Digest= No`". The second parameter is a disk or directory specification, just as with the "`Notebook=`" keyword, or "`Same`", which means that the digest must be

stored on the same disk as the list archives.

The third parameter is either "**Daily**" (the default), "**Weekly**" or "**Monthly**".

The fourth parameter is optional and specifies when the digest is to be actually distributed. For daily digests, specify 'hh:mm' or just 'hh' in the usual 00-23 scale (24 is also accepted for midnight). Note that daily digests are cut on the hour, regardless of whether or not you have provided ":mm" in your setting. For weekly digests, specify a weekday such as "Tuesday". For monthly digests, you may specify a number from 1 to 31 corresponding to the day of the month when the digest will be distributed, although this is not recommended. The purpose here is to make it possible for digests to be issued at mid-month rather than on the first of the month - if you code a number larger than 28, you may not get a digest every month.

The fifth parameter is also optional. It takes the form "**Size(*number*)**" and specifies the maximum number of lines the digest is allowed to reach before a "special issue" is cut. (Note that your digests may run over the limit set in "**Size(*number*)**". This is because LISTSERV will never truncate a message in order to meet the digest size limit. Thus, if you've reached 950 lines of your 1000 line setting and the next message is 100 lines long, your digest will cut at 1050 lines.) Bear in mind that most unix systems do not accept messages larger than 100 kilobytes, so values larger than 1500 should be avoided. The default is to have virtually no limit - 10,000 lines.

The list owner must take special care when disabling digests for a list, as LISTSERV does not presently have any facility which would allow it to alter subscription options automatically on the basis of changes to the list header. Subscribers who had opted for digests would continue not to receive mail as it arrives, but would not get the digests either. The best way to solve this problem is to announce the change long enough in advance, so that people can switch back before digests are suspended. The reason nothing has been done to remove this limitation is that it is not expected to be a frequent condition. Daily digests take up very little disk space and there is no reason to disable them for a typical list.

(1.8c and 1.8d only) The default behavior of a list with a **BOTTOM_BANNER** template defined in *listname*.**MAILTPL** is to suppress the banner throughout the digest and print it only once at the beginning, between the list of topics and the first message in the digest. This behavior can be disabled so that the banner is printed in its normal position at the end of each message in the digest by adding the **BOTTOM_BANNER** parameter to the **Digest=** keyword. Evaluators should note that this behavior is also standard on evaluation copies, with the difference that the evaluation kit banner cannot be turned off. L-Soft does not expect that this parameter will be much used, but it is included for the sake of completeness.

(1.8e and following) Bottom banners are no longer suppressed in individual messages when **BOTTOM_BANNER** is specified. The only use of the **BOTTOM_BANNER** parameter in 1.8e and following is to force a copy of the banner to be printed at the top of the digest as in previous versions.

Note that **TOP_BANNER**s are always included at the top of each message in the digest. Generally, **TOP_BANNER** contains copyright or other important information that should be included with each message, and therefore it is not suppressed.

The second parameter of the "`Digest=`" keyword ("*where*") may only be changed by the LISTSERV maintainer. A list owner is allowed to change "`Digest= No`" to "`Digest= Yes,Same....`", but any other specification for the digest file location will cause an error. A list owner is also allowed to change "`Digest= Yes...`" to "`Digest= No`" without the intervention of the LISTSERV maintainer. Note that if the list is not archived ("`Notebook= No`"), changing "`Digest= No`" to "`Digest= Yes,Same`" will cause the digest files to be written to LISTSERV's A disk (or equivalent specification on the workstation systems). Since the overhead for a typical digest is small, it is not expected that this will cause any problem for the LISTSERV maintainer.

Internet-Via= *internet-hostname*

This keyword is not available in LISTSERV Lite.

There is no default value. This parameter determines whether or not mail bound for Internet addresses is routed through a specific Internet gateway. In principle this keyword should never need to be set on non-BITNET hosts.

Mail-Via= DISTRIBUTE | DIST2 | Direct

This keyword is not available in LISTSERV Lite.

The default value is **`Mail-Via= DISTRIBUTE`**. DIST2 is functionally equivalent to DISTRIBUTE, and is included for historical reasons.

The Mail-Via keyword should generally not be set, except by the site administrator for troubleshooting mail delivery problems.

On "Networked" and "Tableless" LISTSERV sites, **`Mail-Via= DISTRIBUTE`** causes mail to go out over the DISTRIBUTE backbone to spread the delivery load over the LISTSERV Network. **`Mail-Via= Direct`** causes LISTSERV to send all mail through the local SMTP server.

On "Standalone" LISTSERV sites, it has no practical use.

Newsgroups= None | *usenet_newsgroup1,usenet_newsgroup2...*

This keyword is not available in LISTSERV Lite.

This keyword defines the RFC822 "Newsgroups:" header for a list. This field may be required by certain news gatewaying software and should only be defined if the list is gatewayed to usenet and if the gatewaying software does require it. The default is **`Newsgroups= None`**.

A typical setting for this keyword might be:

`* Newsgroups= bit.listserv.lstown-l`

NJE-Via= *internet-hostname*

This keyword is not available in LISTSERV Lite.

There is no default value. This parameter determines whether or not mail bound for NJE addresses is routed through a specific gateway. In principle this keyword should never need to be set on non-BITNET hosts.

Prime= Yes | No | *when*

This keyword is not available in LISTSERV Lite.

Determines whether or not mail for the list is processed during "prime time", a value that is determined by the LISTSERV maintainer and is kept in the system configuration file. The default is "`Prime= Yes`", which means that LISTSERV will allow postings to be processed during prime time. You can also explicitly code "`Prime= No`", which means that LISTSERV will use the value in its `PRIMETIME` site configuration variable to determine "prime time" for the list.

This keyword can be most useful in controlling the load on the machine running LISTSERV. "Prime=" may also be set to an explicit time specification, for example,

`Prime= "MON-FRI: 09:00-17:00; SAT-SUN: -"`

or (for a once-weekly newsletter issued on Wednesday, perhaps)

`Prime= "MON-TUE: 00:00-23:59; WED: -; THU-SUN: 00:00-23:59"`

Note that the time specification for Prime= must *always* be surrounded by double quotes (`""`). Otherwise LISTSERV will stop reading the specification at the first space (ASCII 32) it encounters. For example, the above example coded without quotes would be interpreted as `Prime= MON-FRI:` with the balance of the string ignored. LISTSERV will not issue an error if you omit the quotes.

Note also that when you are coding a prime time specification that LISTSERV's week starts on Monday and runs through Sunday. Thus something like the following examples:

`Prime= "MON-TUE: 00:00-23:59; WED: -; THU-SUN: 00:00-23:59"`

`Prime= "TUE: 01:00-4:59; THU-SUN: 00:00-23:59"`

are correct syntax, whereas

`Prime= "WED: -; THU-SUN: 00:00-23:59; MON-TUE: 00:00-23:59"`

is not. Furthermore note carefully the weekdays must be specified in their correct order, that is,

`Prime= "THU-FRI: 00:00-23:59; SAT-MON: 21:00-23:59"`

is not correct because it starts on Thursday and ends on Monday The correct specification in this case would be

`Prime= "MON: 21:00-23:59; THU-FRI: 00:00-23:59; SAT-SUN: 21:00-23:59"`

IMPORTANT: When you set a "prime time" either for a list or globally for the entire server ("`PRIMETIME=`" in the site configuration file), you are setting the

time during which LISTSERV does **not** process postings. It is "prime time" for the machine when it should be doing other things, for example, number crunching, daily backups, or any other function during which LISTSERV should not be using cycles.

Note also that the minutes specification is cosmetic only. LISTSERV checks on jobs held awaiting non-prime time only once each hour, on the hour. Thus if you have

```
* Prime= "MON-SUN: 06:00-21:00"
```

then jobs awaiting non-prime time will be executed at 22:00, not 21:00 as you might otherwise expect. On the other hand, if you code

```
* Prime= "MON-SUN: 06:00-20:xx"
```

where "**xx**" is any two-digit integer between 01 and 59, then jobs awaiting non-prime time will be executed when LISTSERV runs its hourly check of PRIME jobs at 21:00.

If you need to open only one short window during one or more days, you can do this by coding something like:

```
* Prime= "MON-FRI: 00:00-02:59 04:00-23:59; SAT-SUN: -"
```

This example allows LISTSERV to process mail for the list only between 2 AM and 4 AM Monday through Friday, and at any time on Saturday and Sunday. Note that there is no punctuation--just a space--between the time settings for a given day or day sequence.

Reply-To= List|Sender|Both|None|*net-address*,[Respect|Ignore]
Indicates whether the "Reply-to:" tag supplied by the sender of the mail file is to be preserved or discarded (if present), and, if discarded or omitted, what should be placed in the new "Reply-to:" generated by the server. The default value is "`Reply-To= List,Respect`".

Note that some mailing systems are unable to process a "Reply-To:" field with multiple addresses correctly and may therefore disregard the `Reply-to= Both` option and treat it as `Reply-to= List`.

PLEASE NOTE CAREFULLY: Setting this parameter guarantees only one thing -- that LISTSERV will generate an appropriate RFC822 Reply-To: header in the mail it distributes to subscribers. THERE IS UNFORTUNATELY NO GUARANTEE THAT THE MAIL TRANSFER AGENT (MTA) OR MAIL CLIENT ON THE RECEIVING END WILL HONOR THE Reply-To: HEADER. This is because some mail clients, out-of-office robots, and Internet MTAs either simply do not recognize the existence of Reply-To: or do not implement it properly. Specifically RFC2076 "Common Internet Message Headers" reports that the use of Reply-To: is "controversial", that is, "The meaning and usage of this header is controversial, meaning that different implementors have chosen to implement the header in different ways. Because of this, such headers should be handled with caution and understanding of the different possible interpretations." (RFC2076, page 4). While L-Soft recognizes that it is sometimes important to provide an explicit Reply-To: header to indicate a response path, L-Soft cannot be held responsible for problems arising from the inability of a remote server to properly

process Reply-To: headers.

The two parameters are specified as follows:

First parameter:

List	Replies are directed to the list address.
Sender	Replies are directed to the original sender.
Both	Reply to both the original sender *and* to the list (see note regarding this above)
None	No Reply-to: header is generated unless ",Respect" is specified and a Reply-to: header is present in the original posting, in which case replies are directed to wherever the Reply-To: tag points.
net-address	Replies are directed to the specified internet address

Second parameter:

Respect	The original "Reply-to:" tag on the posting, if any, is kept. If no valid Reply-To: tag exists in the posting, the value defined in the first parameter of this keyword is used.
Ignore	The original "Reply-to:" tag on the posting, if any, is ignored and discarded, and the value defined in the first parameter of this keyword is used instead. If **Reply-To= None,Ignore**, then a Reply-to: tag will never be generated by LISTSERV.

Sender= List | None
Sender= "list title *<net-address>*",*ietf-address*

This keyword is not available in LISTSERV Lite.

NOTE CAREFULLY: Setting this value DOES NOT change the RFC822 "From:" header. Per standard, LISTSERV is not allowed to change the From: header, but must pass it through unchanged.

Used to define the value LISTSERV will place in the RFC822 "Sender:" field. The second parameter is optional, and is included to allow the specification of a second mailbox for use with IETF headers. The first value is used for non-IETF headers and is expected to contain the name and address of the list, or the keywords LIST or NONE. The second mailbox is used for IETF headers; if it is omitted, the generic "owner-listname" mailbox is substituted. Example:

```
* Sender= "Test List <TEST@LISTSERV.X.EDU>",owner-test@listserv.x.edu
```

Note that the first address must be contained in quotes.

Sub-lists= *sublist1,sublist2...sublistn*

This feature and keyword are not available in LISTSERV Lite.

This keyword first appeared in 1.8c and makes it possible to define a "super-list" (as in opposite of sub-list), that is, a "container" list that includes all the subscribers in a predefined set of sub-lists. This can be done recursively to any depth. Only the maintainer can create a super-list, for security reasons. Concretely, the "Sub-lists=" keyword is protected from owner tampering in the same fashion as "Notebook=". The value is a comma separated list of all the sub-lists, which must all be on the same (local) machine. For instance:

```
* Sub-lists= MYLIST-L,MYOTHERLIST-L
```

or (if you want to put each sublist on a separate line):

```
* Sub-lists= MYLIST-L
* Sub-lists= MYOTHERLIST-L
```

The default value for this keyword is null, that is, to have no sublists. Please note that the super-list and all of its sublists must reside on the same LISTSERV server.

The only difference between a normal list and a super-list is what happens when you post to it. With the super-list, the membership of all the sub-lists is added (recursively) and duplicates are suppressed. Other than that, the super-list is a normal list with its own archives, access control, etc. You can even subscribe to it, and this is actually an important aspect of the operation of super-lists. If you are subscribed to the super-list itself, the subscription options used to deliver super-messages to you are taken from your subscription to the super-list, just like with any other list. All combinations are allowed, and in particular NOMAIL is allowed, meaning you don't want to get messages posted to the super-list. When you are subscribed to multiple sub-lists, on the other hand, things work differently:

1. NOMAIL subscriptions are ignored. You will get the super-message if you have an active (not NOMAIL) subscription to at least one sub-list. The idea is that the super-message must be equivalent to posting to all the sub-lists, without the duplicates. Since all it takes to get a message posted to all the sub-lists is a single non-NOMAIL subscription, this is how the super-list works. The only way not to get the super-messages is to subscribe to the super-list directly and set yourself to NOMAIL.

2. The DIGEST and INDEX options are ignored and internally converted to MAIL. The first reason is that, since in most cases the user will be on multiple sub-lists (otherwise you don't need a super-list in the first place), the only safe method to set subscription options for super-messages is by subscribing to the super-list so that there is no ambiguity. The second reason is that, in most cases, super-lists will be used for out of band administrative messages rather than for large volume discussions, so it is actually preferable to have the message sent directly. The third reason is that the super-list and sub-lists may not necessarily offer the same options (DIGEST and INDEX). In particular it is expected that many super-lists will not have archives. If you want a DIGEST or INDEX for the super-messages, you must subscribe to the super-list directly.

3. In LISTSERV 1.8c and 1.8d, the REPRO option is NOT inherited by sub-lists. That is to say, even if the sub-list subscriber is set to REPRO on the sub-list AND the super-list is set up such that sub-list subscribers may post directly to it, he will NOT receive a copy of his own posting. REPRO is effective only for users who are directly subscribed to the super-list. This restriction has been removed in LISTSERV 1.8e.

Topics, if defined, are evaluated on a per-list basis. That is, for every sub-list (and for the super-list), LISTSERV determines whether the topic of the message is one that you want to see. If not, it acts as if you were not subscribed to this particular list. Roughly speaking, this works very well if all the sub-lists have the

same set of topics (or a well-defined set of common topics), and doesn't work well at all if every list has its own set of topics.

Topics= *topic1,topic2,...topic23*

This feature and keyword are not available in LISTSERV Lite.

List topics provide a way to run a mailing list (preferably moderated) where several sub-topics are being discussed in parallel but some subscribers are only interested in a subset of the topics. For instance, a working group might have general discussions, decisions, and messages related to meetings. People who cannot attend the meetings can then opt out of last calls for hotel reservations and discussions about seafood restaurants, whereas people who have no time to follow the discussions can elect to get just the decisions. At any rate, such a compartmented list requires a certain discipline in order to be successful, as the posters must label their messages to indicate which topic(s) they belong to.

Through the `Topics=` keyword, the list owner can define up to 23 topics for the list (note that 1.8c and earlier are limited to 11 topics). For instance, the list owner could code:

`Topics= News,Benchmarks,Meetings,Beta-tests`

If necessary, you may use multiple `Topics=` lines in your header in order to fit all of your topics in.

WARNING - YOU MUST NEVER REORDER THE TOPICS= KEYWORD(S)

To save disk space, LISTSERV remembers which topics users have selected through their ordering in the "Topics=" keyword. That is, "News" is "topic number 1" for LISTSERV, "Benchmarks" is "topic number 2", and so on. This means you can change the name of a topic without requiring users to alter their subscriptions (for instance, you could decide that "Tests" is a better name than "Beta-tests" and just make the change). However, you must never change the order of the topics in the "Topics=" keyword. If you want to remove a topic, replace it with a comma. For instance, to remove the "Meetings" topic, you would change the keyword to:

`* Topics= News,Benchmarks,,Beta-tests`

This restriction might be removed in a future release.

Topic names can contain any character except space, colon and comma; the use of double quotes or equal signs is discouraged, as they require special attention when coding list header keywords. In addition, topic names may not start with a plus or minus sign, and the words ALL, NONE, RE, OTHER and OTHERS are reserved.

Posters label their messages through the subject field. LISTSERV first skips any possible sequence of 'Re:' keywords, and takes anything to the left of a colon as a list of topics, separated by commas. The posting is considered to belong to all the topics listed before the colon. If none of these topics is valid for the list, it is classified in a special topic, "Other". If some of the topics are valid but others are undefined, the invalid ones are ignored. At any rate the subject field is left unchanged. Here is an example:

Subject: Benchmarks,News: Benchmarks for XYZ now available!

Messages which should be read by everyone can be posted to the special topic "All". Topic names can be shortened to any unambiguous abbreviation. In our example, "Be" is ambiguous because it could be either "Beta-tests" or "Benchmarks", but "Bench" is acceptable.

Subscribers select the topics they wish to receive with the SET command. The syntax is 'SET listname TOPICS: xxx' where 'xxx' can be:

- A list of all the topics the user wishes to receive. In that case these topics replace any other topics the user may have subscribed to before. For instance, after 'SET XYZ-L TOPICS: NEWS BENCH', the user will receive news and benchmarks, and nothing else.

- Updates to the list of topics the user currently receives. A plus sign indicates a topic that should be added, a minus sign requests the removal of a topic. For instance, 'SET XYZ-L TOPICS: +NEWS -BENCH' adds news and removes benchmarks. If a topic name is given without a + or - sign, + is assumed: 'SET XYZ-L TOPICS: +NEWS BENCH' adds news and benchmarks. The first topic name must have the plus sign to show that this is an addition, and not a replacement.

- A combination of the above, mostly useful to enable all but a few topics: 'SET XYZ-L TOPICS: ALL -MEETINGS'.

The colon after the keyword TOPICS: is optional, and TOPICS= is also accepted. Do not forget to include the special OTHER topic if you want to receive general discussions which were not labeled properly. On the other hand, if you only want to receive properly labeled messages you should not include it. ALL does include OTHER.

Finally, it is important to note that topics are active only when your subscription is set to MAIL. Digests are indexes always contain all the postings that were made, because the same digest is prepared and sent to all the subscribers.

(See also **Default-Topics**.)

Error Handling Keywords

Auto-Delete=   **No**
          Yes,Semi-Auto[,Delay(*number*)][,Max(*number*)][,Probe(*number*)]
          Yes,Full-Auto[,Delay(*number*)][,Max(*number*)][,Probe(*number*)]
          Yes,Manual[,Delay(*number*)][,Max(*number*)][,Probe(*number*)]

This keyword is available in LISTSERV Lite, but is not full-featured. The behavior in LISTSERV Lite with Auto-Delete= Yes is Auto-Delete= Yes,Semi-Auto,Delay(0),Max(1). Any other settings are ignored. Specifically the passive probing option available in Classic is disabled in Lite.

LISTSERV includes support for automatic deletion of users whose account has expired or whose system has permanently disconnected. When the delivery error is generated by LMail (any version), MX V3.2 or higher, PMDF V4.2 or higher, or LSMTP(TM) , which all implement the same delivery error format, LISTSERV may be able to automatically process the delivery error and take action based on the value of the "Auto-Delete=" list header keyword. The unix versions of LISTSERV also support sendmail's delivery error format. The auto-deletion code is also fully compatible with RFC1893 "Notary" format error codes

If the list has been coded "Auto-Delete= No", or if the delivery error is not in LMail format and LISTSERV cannot understand it, LISTSERV simply passes it to the list owner. Otherwise LISTSERV processes the message automatically. The algorithm may be refined in a future version, but at present the following steps are taken whenever the auto-deletion feature is enabled:

When auto-deletion is set to "Full-Auto" or "Semi-Auto":

- LISTSERV looks for "permanent" errors (no such user, no such host, and so on). If the failing recipients are subscribed to the list, LISTSERV removes them and notifies you. No action is required from the list owner.

- If there are permanent errors for users LISTSERV could not find on the list (for instance because the account subscribed to the list is a totally different one which forwards mail to a dead account), or if there are only "temporary" errors (host unreachable for 3 days, system error, disk quota exceeded, and so on), LISTSERV passes the actual error message to the list owner for further disposition if running in semi-auto mode. If running in full-auto mode, the error messages themselves are discarded and the errors only show up as entries in the daily auto-deletion monitoring report.

- When running in full-auto mode, LISTSERV never passes back a delivery error unless it took action on it. This means that certain errors may remain unsolved, as LISTSERV presently ignores temporary errors and some of them are virtually permanent (if the owner of the account has left but for some reason his account was not closed, his disk quota is bound to remain exceeded until someone takes action). Full-auto mode should be used only when the list owner positively does not have the time to handle the delivery errors LISTSERV sends every day.

When auto-deletion is set to "Manual":

- When running in manual mode, the auto-delete monitor informs the list owner of the error(s) and takes no further action on delivery errors.

Some considerations for configuring the auto-delete monitor parameters:

- Setting the Delay(number) option. The default is 4. This is the number of days that a subscriber's mail needs to bounce before he's automatically deleted. If "Delay(0)" is coded, LISTSERV won't wait, but will delete on the first bounce.

  Most delivery errors occur on weekends when systems are taken down for maintenance, system administrators are not around to reboot after crashes, and the like. Because of this, most delivery errors only last for 2-3 days and may not be "permanent" even if they seem to be at first.

  The nature of delivery errors is such that LISTSERV has no way to establish that a problem has been fixed because it receives only negative acknowledgements when a message bounces. This taken together with the transient, "weekend" nature of most delivery errors indicates that it is not a good idea to set Delay() to a value close to 7. For instance, if Delay(7) and a subscriber's mail regularly bounces on the weekend, LISTSERV will wait until the next weekend to decide whether or not to delete him, at which point the subscriber will bounce mail again and start the process all over. The bottom line is that LISTSERV might as well have gone ahead and deleted the subscriber as soon as the first bounce occurred.

- Setting the Max(number) option. To prevent auto-deletion monitoring from getting out of hand, subscribers are deleted after a specified number errors regardless of how many days it has been going on. The default is Max(100). This is so LISTSERV won't spend its life monitoring 50 bogus users x 100 messages = 5000 a day. Note that if Delay(0), the setting for Max() is ignored (in effect it is set to Max(1)).

- Setting the Probe(number) option. This parameter tunes the "passive probing" option (beginning with 1.8d). Passive probing operates by turning a certain percentage of your regular list messages into transparent probes that look like a normal message but also double as a probe, rather than sending out the explicit **PROBE1** template as in active probing. You enable (or tune) passive probing by adding a "**,Probe(xx)**" parameter to the **Auto-Delete=** keyword setting. For instance,

  Auto-Delete= Yes,Full-Auto,Probe(30)

  where "30" is the number of days to wait between probes for any given user. Subscribers with working mail systems will not see any difference, subscribers with flaky mail systems will occasionally receive a message showing that their mail bounced and saying that they should report the problem to their ISP, and of course plain bad addresses will go away.

  In order to disable passive probing you set the probe parameter to 0, i.e.,

  Auto-Delete= Yes,Full-Auto,Probe(0)

  If you have users who for whatever reason should not be probed, you can deactivate passive probing (and any other renewal you have set for the list) with the **SET *userid@host* NORENEW** command. The default for this parameter is Probe(30) for lists up to ~2K subscribers, and Probe(0) for

larger lists (because by its nature, probing can be a non-inconsiderable performance hit).

For more information on passive probes, see chapter 13.5.2 of the *Site Manager's Operations Manual*.

- When you take a vacation, note that it is best to switch auto-delete to MANUAL. Then do not restore to auto on the day you come back, because you will have a number of subscribers on file ready to be deleted. Wait DELAY+n days before changing back to Full-Auto or Semi-Auto, where n is an adjustment to account for the fact that people don't fix all problems right away at 09.00 on the day your vacation ends. n=2 is a reasonable choice.

The default value is "Auto-Delete= Yes,Semi-Auto,Delay(4),Max(100)" for lists coded "Validate= No" and "Auto-Delete= No" for all other lists.

Note that if you have coded "Delay(0)" and/or "Max(0)", LISTSERV simply deletes any error-generating subscriber it can (generally 95-98%), discards any further errors it does not understand, and does not generate daily monitoring reports. If you want the daily monitoring reports you must code at least "Delay(1),Max(1)".

Errors-To= *mon-address1,mon-address2,...*
Defines the person or list of persons that are to receive rejection mail for the list. The default value is 'Owners'.

In LISTSERV 1.8e and following, the internet address of the list is explicitly disallowed as an error-receiving address, and attempting to set Errors-To= to the internet address of the list will raise an error. The list should never be configured to receive its own errors as this is guaranteed to cause looping.

Loopcheck= Full
Loopcheck= None
Loopcheck= *option1*[,*option2*][,*optionn*]

This keyword is not available in LISTSERV Lite.

Determines the type of loop checking performed by LISTSERV to avoid perpetuating mail loops. The default is "`Loopcheck= Full`". Loop checking is configured on a list by list basis only.

ALWAYS USE THIS KEYWORD WITH CAUTION!
Misuse of this keyword can and will allow mailing loops onto your list!

The various `Loopcheck=` parameters are defined as follows. "Full" and "None" must be specified alone, whereas the other options may be specified together as required.

Full   LISTSERV uses its full suite of loop checking heuristics to check incoming mail for loops. This is the default, and should not be changed without good reason.

None   All LISTSERV loop checking and "command to list" checking is disabled for this list. WARNING: "None" tells LISTSERV that, by definition, anything that reaches its reader is NOT a delivery error. It is

never a good idea to use this parameter except in special cases where a bug is suspected in the loop checking heuristics. Generally this parameter should not be used without checking with L-Soft first, and only for the diagnosis of an existing problem.

Noorigin Allows the list owner to disable the check for "known mailer origins" such as MAILER, POSTMASTER, ROOT, UUCP, et al. Mail whose 'From:' field is the address of the local mailer is still trapped, but wildcard checks on the mail origin are disabled.

Nobody Allows the list owner to disable the check for identical text in the body of incoming mail only. LISTSERV relies only on the Subject: field of the mail message to determine whether or not mail is looping. This is a very dangerous option: it means that any mailer not using one of the "standard" subjects known to LISTSERV will cause a loop.

NoCRC Allows the list owner to disable CRC (cyclical redundancy check) check of incoming mail. CRC loop checking calculates a "checksum" based on the contents of the mail message and compares it to other incoming mail to spot duplicates.

NoSpam Allows the list owner to disable the anti-spamming filters to incoming mail.

Spam-Delay(n) Allows the list owner to modify the number of minutes LISTSERV holds mail from non-subscribers before releasing it to the list. The assumption is that, within n minutes, a spam alert may or may not arrive regarding non-subscriber mail. The list owner can disable this function for his list by coding "`Loopcheck= Spam-Delay(0)`", or can tune it to his preference by simply specifying the number of minutes for LISTSERV to hold the mail. The default is 10 minutes, or "`Spam-Delay(10)`".

Please note carefully that L-Soft does not recommend changing `Loopcheck=` from its default of "Full" unless you are prepared to accept the very likely possibility of a mail loop occuring on your list; a situation for which L-Soft would not and could not be held responsible for. The only exception would be the "`Loopcheck= NoSpam`" (which might be necessary to keep adminstrative mail to multiple lists on a single host from triggering the anti-spamming filter) or "`Loopcheck= Spam-Delay(n)`" options, neither of which stops canonical mail loops *per se*.

See also `Filter=` and `Safe=`.

Safe= Yes | No

The list header keyword, "`Safe=`", controls the e-mail address LISTSERV places in the SMTP MAIL FROM: field, which is where well-behaved mailers will return delivery errors. With "`Safe= No`", these errors are sent to the list address as before, hopefully to be intercepted by the loop detector and passed on to the list owner. With "`Safe= Yes`", the error address is set to 'owner-listname', and delivery errors sent to that address are passed on to the list owner without the risk of creating a mailing loop. The default is "`Safe= Yes`".

IMPORTANT: The use of "`Safe= Yes`" does not guarantee that all errors will go to the 'owner-listname' mailbox. Unfortunately, there are many non-compliant

mailers which will continue to send the error back to the list (usually because it is listed in the 'Reply-To:' or 'Sender:' field). Using "`Safe= Yes`" significantly decreases the potential for mailing loops, but not enough to actually code "`Loopcheck= No`", unless you are sure that all your subscribers have compliant mailers.

See also `Filter=` and `Loopcheck=`.

List Maintenance and Moderation Keywords

Editor= *net-address1,net-address2|access-level1,...*

Defines the list editor(s). When used in conjunction with the "`Send=Editor`" option, it causes all mail sent to the list to be automatically forwarded to the first person listed in the "`Editor=`" keyword, who will then send it back to the list at his discretion. The editors are the only persons (with the list owners) who are allowed to mail directly to the list. Note that ANY editor can send mail to the list while only the FIRST one will receive copies of mail sent to the list (but see also `Moderator=`).

The file will be forwarded to the editor 'as is', without being included in a mail envelope. This method makes sure that the original "Resent-" tags (if any) and "To:" keyword are preserved.

Note that the first editor *must* be a network address (e.g., `someuser@foo.bar.com`) and not an *access-level*. Subsequent editors may be *access-levels*. For instance, you can code

`* Editor= joe@baz.net,(MYLIST-L)`

which allows all subscribers from the MYLIST-L list to post without going through the editor, and diverts all non-subscriber mail to joe@baz.net for approval.

IMPORTANT NOTE: The first editor MUST be a human person, not a file server, list server, mailer, or suchlike. Specifying a program's mailbox as the primary editor could result in a mailing loop for which L-Soft international, Inc., could not be held responsible.

Finally, please note that the `NOPOST` subscriber option will take precedence over both `Editor=` and `Moderator=`, if set for someone so defined. This means that if you have "`Default-Options= NOPOST`" for your list and you add an editor or a moderator as a subscriber, you will have to manually reset the editor to `POST` (with "`SET listname POST FOR userid@host`") before things will work properly. You will know that this is necessary if your editor or moderator can successfully approve postings but is then told that he or she cannot post to the list.

Editor-Header= Yes | No

This keyword is not available in LISTSERV Lite.

If an editor is defined (see `Editor=`), this keyword determines whether or not special header information is prepended to list messages forwarded to the editor. The default (for lists configured with an Editor) is "`Editor-Header= Yes`".

Note that `Editor-Header= No` is ignored if you have `Send= Editor,Hold` or `Send= Editor,Hold,Confirm` . In these cases the editor-header information is required so as to provide the confirmation code for the OK command.

List-Address= *name_info[@host_info]*

This keyword is not available in LISTSERV Lite.

This keyword determines how LISTSERV announces its list address in the header of messages delivered to the list: NJE vs. Internet address, short vs. long list name, etc. The default options (when neither "**List-Address=**" or **LIST_ADDRESS** are defined) are long list name and Internet address. A corresponding **LIST_ADDRESS** configuration option must be added to the LISTSERV site configuration file.

It is important to note that the only effect of the "**List-Address=**" keyword is to change the way the list identifies itself in list postings, command replies, etc. It does not instruct the mail system to create forwarding entries to support the new name, nor does it establish the specified name as an alias for the list (use "**List-ID=**" for this purpose). In general list owners should not use this keyword without first consulting with the LISTSERV maintainer.

In 1.8b and earlier versions, the first token (*name_info*) can be either **LISTNAME** or **LIST-ID**. Do not attempt to specify the actual list name. Use **LISTNAME** if you want LISTSERV to use the "short" list name (always available), and **LIST-ID** if you would rather see the "long" list name ("**List-ID=**" keyword). If there is no "long" name, the short name is substituted.

Version 1.8c introduced the ability to specify the name of the list in the first token (i.e., you may now specify something like "**List-Address= XYZ-L@XYZ.EDU**").

The second token (*host_info*) can be either **NJE**, **FQDN**, or the fully qualified domain name of your choice. That is, you may type the actual hostname that you want LISTSERV to use, which may be useful if the machine on which LISTSERV is running is known under several hostnames.

If you only want to override one of the two parts of the list address, you do not need to specify the other. For instance, if you only want to change the hostname, you can enter "**List-Address= XYZ.EDU**" in the list header and let the left-hand part default from the value of the system default in the LISTSERV configuration file. Similarly, "**List-Address= List-ID**" takes the right-hand part from the system default. To avoid bad surprises, LISTSERV will also accept a comma in lieu of @-sign in the list header, or a blank in the LISTSERV configuration file. Here are a few examples:

- "**List-Address= FQDN**" announces the list under the Internet address for the LISTSERV host, if one is available (for BITNET-only sites this setting has no effect).

- "**List-Address= List-ID@FQDN**" uses the long list name and the Internet hostname.

- "**List-Address= Listname@XYZ.EDU**" uses the short list name and the hostname XYZ.EDU.

- Starting with version 1.8c, "**List-Address= XYZ-L@XYZ.EDU**" is also valid. You no longer are restricted to specifying **LISTNAME** or **LIST-ID** for the left-hand (username) part.

List-ID= *longlistname*

This keyword is not available in LISTSERV Lite, and is technically obsolete on all ports of the software except for VM.

On VM systems, this keyword allows the list owner to specify a long list ID in addition to the normal 8-character list name. This is particularly useful for peered or gatewayed lists that have names longer than 8 characters. On non-VM systems, if the normal list name is longer than 8 characters and the list is being migrated from a VM system, it may be a good idea to specify the first 8 characters of the list name in this keyword, at least temporarily. This way subscribers who were used to the old 8-character name can continue to use it on the new system.

Non-VM systems may use this keyword for aliasing. However, today there is really no good reason to use this keyword on non-VM systems, as it is possible to define lists on such systems with native file system names longer than 8 characters. L-Soft's recommendation is that this keyword be used only if you are migrating a list from VM that was known by both its "short" name and its "long" `List-ID=` name. (On unix you can avoid this by simply specifying an extra set of aliases in `/etc/aliases` for the "long" name that point to the same places as do the ones for the "short" name.)

In any case a list owner should not set a value for `List-ID=` without first consulting with the LISTSERV maintainer, since it will be necessary to add appropriate system mailer aliases before the name specified in `List-ID=` will work.

`List-ID=` will not work properly on NT systems running with the SMTPL "listener" because the "listener" has no way to know that the list ID specified in this parameter is a valid local address.

`List-ID=` *will* work on NT and VMS systems running LSMTP, but you must first configure a route in LSMTP for the `List-ID=` name so that LSMTP will know to deliver mail addressed to the `List-ID=` address to LISTSERV (as opposed to POP or SMTP, etc.).

Under VMS and unix, it is necessary to add the appropriate aliases to the mailer's aliases file in order for `List-ID=` to work, since mailers such as sendmail and PMDF otherwise have no way to know that the name specified in `List-ID=` is a valid address. This means that lists that have the `List-ID=` keyword specified need two complete sets of aliases defined (unless `List-ID=` is identical to the list name, in which case it should not be implemented to begin with).

Starting with LISTSERV 1.8d, if you do use `List-ID=` to specify a "long" name for a list with web archives, LISTSERV will create an HTML page for both the long and short names. Here again, however, on non-VM systems L-Soft does not recommend the use of `List-ID=` .

Moderator= [All,]*netaddress1*[,*netaddress2*]...

This keyword is not available in LISTSERV Lite.

This keyword defines which editors of a moderated list receive postings for forwarding to the list. The default is the first editor as defined by the "`Editor=`"

keyword. If multiple moderators are defined, the load is spread across them.

Note that all editors may still post directly to the list, but only those editors defined by "**Moderator=**" will have messages from non-editors forwarded to them.

Beginning with 1.8c, if the parameter "All" is coded before the list of moderator addresses, LISTSERV will send copies of all postings to all moderators, any of whom may approve the message. An example of this would be

* Moderator= All,joe@somehost.com,jill@someplace.net

Note that this could also be coded as:

* Moderator= All,joe@somehost.com
* Moderator= jill@someplace.net

Assuming "**Send= Editor,Hold**", once a message is approved by one of the moderators, any other moderator attempting to approve the same message will be told that an identical message has already been posted to the list.

If "**Send= Editor**" (i.e., without "Hold"), please note that if a note is appended or prepended to the edited post, or if the body of the post itself is edited (that is to say, if the body of the approved message is changed), duplicates are possible. Thus it is important that the moderators of any list set up this way pay close attention to whether or not the posting has already been approved by another moderator.

Finally, please note that the **NOPOST** subscriber option will take precedence over both **Editor=** and **Moderator=**, if set for someone so defined. This means that if you have "**Default-Options= NOPOST**" for your list and you add an editor or a moderator as a subscriber, you will have to manually reset the editor to **POST** (with "**SET** *listname* **POST FOR** *userid@host*") before things will work properly. You will know that this is necessary if your editor or moderator can successfully approve postings but is then told that he or she cannot post to the list.

New-List= *net-address*

This keyword is not available in LISTSERV Lite.

When a list is moved to a different LISTSERV host, this keyword can be added to the list header left on the original host. This facilitates forwarding of administrative commands and postings from the original host to the new host. Users posting to the old address will also receive a short note in return listing the new address.

Note that success in setting the "**New-List=**" keyword is dependent on whether or not you have deleted practically every other keyword other than "**Owner=**" from the list header. Since the use of "**New-List=**" is intended to be an automatic pointer to the new host and/or new list name, no other keywords should be defined. Keywords that would cause a problem will therefore generate fatal errors on a list **PUT** operation and the list header will not be updated.

Further note that the old list's subscriber list cannot be modified once the "**New-**

`List=`" parameter is defined. The appropriate sequence of operations is:

1. Create the new list

2. Move the subscribers to it

3. `DELETE oldlistname *@*`

4. Modify the header of the old list by deleting unneeded keywords and adding the "`New-List=`" keyword with its pointer to the new list.

Should this sequence not be followed, note that by removing the "`New-List=`" keyword, the old list will be unlocked and the subscriber list can then be deleted if desired.

Notebook= No
Notebook= Yes,*where,interval*|Separate,*access-level*[,*access-level,...*]
Indicates whether or not an automatic log of every piece of mail sent to the list is to be kept, and defines at which interval of time its file name must be changed and who is allowed to retrieve it from the server. The default values are "`Notebook= No,A,Single,Private`".

where is the filemode of the minidisk (VM) or the disk and directory (non-VM) on which the notebook is to be kept. The default value of "`A`" is equivalent to LISTSERV's main working directory. On VM servers, this is LISTSERV's A disk; on VMS and Windows servers, this is LISTSERV's `MAIN` directory, and on Unix servers it is `~listserv/home` (or whatever value has been used in the Makefile for `$LSVROOT/home`). Naturally, you may change this value to any directory you wish, provided that a) the directory exists (for security reasons, LISTSERV will not make it for you) and b) LISTSERV has read-write access to that directory. Rather than use the "A" directory, L-Soft strongly recommends that you create a separate directory structure with subdirectories for each list and use a full path spec for this parameter. This is important for security purposes related to the file server functions (see chapter 8 for details).

Note that under unix this parameter MUST IMPERATIVELY point to a directory specification that is all lower-case. LISTSERV for unix cannot write archives to directories named in upper- or mixed-case.

If your server is running the Web Archive Interface, L-Soft *does not* recommend that this parameter be pointed to the web archive index directory.

interval Defines the filetype or extension of the "notebook" file for the list, as indicated below (the filename will always be the same as the list name):

Single: A single file with the extension "NOTEBOOK" is created.
Yearly: A new file is started each yearly, extension is "LOGyy"

Monthly:	The extension is "LOGyymm"
Weekly:	The extension is "LOGyymmw" (w in "A"-"E")
Separate:	A separate file is kept for each mailing (e.g. announcements, newsletters). The extension is "yy-nnnnn" (sequential counter).

While you may change the notebook interval at any time, LISTSERV will not convert existing notebooks into the new interval format. For instance, if you convert from Monthly to Weekly notebooks, LISTSERV will continue to maintain your original notebooks in their monthly format, while writing any new postings into weekly notebooks. This is perfectly normal and does not affect the proper operation of your list (in particular it does not cause any breakage to the archive search feature).

For 1.8c servers with the WWW archive interface interface installed, please note that in order for archives to appear in the interface, the following requirements must be met:

1. Notebooks must be "Public"
2. The notebook interval must be "Monthly", "Weekly", or "Yearly" ("Yearly" is not recommended).
3. The LISTSERV maintainer *must* create an index directory for your list per the instructions in the *Site Manager's Operations Manual*.

Note further that lists that meet the above three requirements will show up in the WWW archive interface *even if the list is set "Confidential= Yes"*. See chapter 5.4.6 of the *Site Manager's Operations Manual* for details.

For 1.8d and later servers with the WWW archive interface interface installed, please note that in order for archives to appear in the interface, the following requirements must be met:

1. Notebooks can be "Public" or "Private" (or any other *access-level*)
2. The notebook interval *must* be "Monthly", "Weekly", or "Yearly" ("Yearly" is not recommended).
3. The "Confidential=" keyword *must* be set either to "No" or "Service"
4. The LISTSERV maintainer *must* create an index directory for your list per the instructions in the *Site Manager's Operations Manual*.

See chapter 5.4.6 of the *Site Manager's Operations Manual* for further details.

Note: Notebooks may be retrieved by means of the `GET` command. On VM only, a list of all available notebooks can be obtained with a `GET NOTEBOOK FILELIST` command.

The first two parameters of the "`Notebook=`" keyword may only be changed by the LISTSERV postmaster.

If necessary, you may break the "Notebook=" keyword into multiple lines in order to avoid running up against the 100-character header line limit. For instance

`* Notebook= Yes,/home/listserv/lists/mylist-l,Monthly,Private`

is strictly equivalent to

```
* Notebook= Yes
* Notebook= /home/listserv/lists/mylist-l
* Notebook= Monthly,Private
```

This can be particularly important if it is necessary to specify multiple *access-levels* for the notebooks (for instance if you have many sub-lists and want the subscribers to the sub-lists to be able to access the super-list's notebooks), for example,

```
* Notebook= Yes,C:\LISTS\SUPER,Monthly,Private,(SUB-A),(SUB-B)
* Notebook= (SUB-C),(SUB-D),(SUB-E),(SUB-F)
```

Notebook-Header= Short | Full

Determines whether or not individual message in notebook archives are stored with full Internet header information or with "short" headers. The default is "`Notebook-Header= Short`".

Notify= Yes | No | *mon-address*

Defines whether the list owner (or the person indicated by "`Notify= mon-address`") is to receive notification of new subscriptions and deletions, etc. The default is "`Notify= Yes`", meaning that non-quiet list owners will be notified.

Owner= *net-address1* | *mon-address1*,[Quiet:,]*net-address2* | *mon-address2*,...

Defines the person or list of persons who "own" the list. They are responsible for controlling access to the list and defining the list control keywords which are best suited to the purpose of the list. The default value for this keyword which should ALWAYS appear in the list header is the list of the userids of the postmasters. Any combination of explicit network addresses and complex access-levels is acceptable, for example: `Owner= BIG@BLUE,(STAFF-L),Owner(MAIN-L)`

An interesting application is to create a STAFF-L list containing the userids of all the local LISTSERV staff members and set the "`Owner=`" keyword of all local lists to "`Owner= (STAFF-L)`". This way when there is a change in the local LISTSERV management it is not necessary to modify the headers of all the lists – just modify the STAFF-L list.

The use of the "`Quiet:`" parameter causes all subsequently-defined list owners to be excluded from receiving any delivery error messages or other administrative mail from LISTSERV.

List owners may be defined on a single line or on multiple lines. See Chapter 2.7 of the *List Owner's Manual* for details.

Peers= *peer1,peer2,...*

This keyword is not available in LISTSERV Lite.

Defines the (global) list of all the servers in the world that are peer-linked to the list, either directly or via one or more other peer servers. This information is used by the various list management commands to determine the "nearest" peer list to a given user. For example, when a `SUBSCRIBE` command is received from a user and it is determined that there is a better (nearer) peer list for him, the subscription request is automatically forwarded to the appropriate LISTSERV.

Be sure to read the appropriate sections of the LISTSERV *List Owner's Manual* before peering any list. Note that peers must have the same `PW=` keyword setting.

Renewal= *interval1,interval2...,intervalx*,**Delay(***number***)[,Probe]**

This keyword is not available in LISTSERV Lite.

This keyword controls whether or not subscribers are required to renew their subscriptions on a regular basis, and what the subscription period is. Multiple intervals can be set, each interval being one of several things:

- Monthly, Yearly, Weekly, or a numeric variation such as 3-Monthly (meaning, quarterly). Note also that 1.8c introduces the ability to code "`Renewal= xx-Daily`", for instance, "`Renewal= 15-Daily`". While the use of intervals of less than a week is and remains inadvisable, FAQ templates with rotating topics may require the selection of a very precise renewal interval (for congruence purposes), which was not possible with "`xx-Weekly`" granularity. Please refer to chapter 9.9 of either the list owner's manual or the site manager's manual for a discussion of rotating FAQ support.
- An absolute date in the format `yyyy/mm/dd` (once on this specific day), or the format `mm/dd` (once yearly on this month/day).
- The confirmation delay, in the format `Delay(n)`, where `(n)`=the number of days between the time the subscriber is asked to confirm the subscription and the day the user is removed from the list. This default is Delay(7), or seven days.

A typical `Renewal=` configuration might be:

`* Renewal= 6-Monthly,Delay(14)`

Conceivably `Renewal=` could also be set to something like:

`* Renewal= 6-Monthly,1998/07/04,12/25,Delay(14)`

which would cause LISTSERV to send renewal requests once every six months on the anniversary date of the user's original subscription, a specific request on 4 July 1998, and a request every year on Christmas Day. Note that this is provided ONLY as an example. L-Soft does not recommend using a renewal scheme of this sort.

Note: When setting up `Renewal=` for the first time on an older, established list, you may find that a substantial number of subscribers are prompted for confirmation immediately even though you may have set `Renewal=` to a value that might not be expected to cause such behavior. This is because LISTSERV uses the last activity date (which may or may not be the same as the subscription anniversary date) for the purpose of subscription renewal. The last activity date may be one of the following: The subscription anniversary date; the last date the subscriber posted to the list; or the last date the subscriber changed personal options.

Note also that if you code a specific date without specifying a year field (e.g., `Renewal= 6/1`), LISTSERV will immediately request a renewal from any subscriber whose last activity date is prior to that date in the *current* year.

The "**Probe**" parameter, introduced in Version 1.8c (but disabled in LISTSERV Lite) activates LISTSERV's "active probing" bounce processing feature, whereby the users are "probed" regularly using the **PROBE1** mail template. The desired response from the user is to discard the message and do nothing. If the probe bounces, LISTSERV first sends the **PROBE2** template with a copy of the bounce (assuming that the address actually works regardless of the bounce), and then schedules a new probe for the next day or deletes the user immediately, depending on the list's "**Auto-Delete=**" policy. For more information see chapter 4.6 of the list owner's manual.

Subscription renewal is disabled by default. To turn it off for a specific list, simply remove the "**Renewal=**" keyword from the list header.

See also **Auto-Delete=**.

Sizelim= *number | number***K** *| number***M**

This keyword is not available in LISTSERV Lite.

In 1.8d and earlier, if set, causes LISTSERV to reject all messages to the list which exceed the number of lines (including all Internet header lines) indicated. This can be helpful in discouraging subscribers from posting long screeds or uuencoded files to your lists. It can also be set higher than the LISTSERV default if desired; check with your LISTSERV maintainer before changing this upward. (Generally "**sizelim= 250**" is large enough for long posts but short enough to discourage postings of uuencoded binaries, but of course, your mileage may vary.)

The Sizelim= list header keyword has been enhanced in 1.8e to allow list owners to specify a maximum message size in either kilobytes or megabytes, rather than in lines, if preferred. For instance:

```
Sizelim= 100K        Reject messages over 100Kbytes
Sizelim= 1M          Reject messages over 1Mbyte
```

As before, the limit operates against the entire message file, including all Internet header lines.

Subject-Tag= *text*
LISTSERV 1.8c and higher supports "subject tags", i.e., the ability to insert a predefined text tag into the subject line of mail coming from a list. For instance, your subscribers might want the subject lines of mail coming from your list to contain the name of your mailing list. Whereas the RFC822 subject line of a typical list posting without a "subject tag" would look like this:

```
Subject: I think ID4 is a great movie, don't you?
```

if you were to define

```
* Subject-Tag= SCI-FI
```

the subject would look like this for all users who are set to the new "SUBJheader" personal option:

```
Subject: [SCI-FI] I think ID4 is a great movie, don't you?
```

Note that this option may be toggled on and off by the user by use of the new "**SET** *listname* **SUBJecthdr**" option. It is turned off by default.

The normal default for "**Subject-Tag=**" is the name of the list, for example, **SCIFI-L**. If "**List-Address=**" is defined for your list, the default is either the name of the list or the list ID, whichever is listed in "**List-Address=**". A subject tag can be only a single word; in other words, you cannot define a sentence to be used as a subject tag.

Starting with LISTSERV 1.8d, if a user sends a message with a blank RFC822 "Subject:" header, LISTSERV will create a "Subject:" header and place the subject tag into it (but only for subscribers with the **SUBJecthdr** option set.) Under 1.8c, subject tags worked only when posters defined a subject for their message.

Setting this keyword does *not* automatically reset users to the **SUBJecthdr** option. This must be done manually for existing users and may be specified by default for new subscribers by use of the **Default-Options=** keyword.

X-Tags= Comment | Yes | No

This keyword is not available in LISTSERV Lite.

Indicates whether "X-To:" and "X-cc:" tags are to be included in the output mail files to list recipients of the original mail file (other than the list userid) or not, and how they should appear in the RFC822 header.

Yes: This information must be provided in the form of "X-To:" and "X-cc:" tags in the RFC822 header (similar to the "To:" and "cc:" tags). This is the default.

Comment: This information must be provided in the form of "Comment:" tags, for example, "Comment: X-To:" and "Comment: X-cc:".

No: This information must not appear at all in the mail header.

Security Keywords

Change-Log= No | Yes[,Yearly|Monthly|Weekly|Daily|Single]

This keyword is not available in LISTSERV Lite.

When set to YES, causes LISTSERV to write a *listname.***CHANGELOG** file (*listname* **CHANGLG** on VM) in the "A" disk or directory which contains information about all changes made to individual subscriptions. Commands tracked include **SUBscribe/JOIN**, **SIGNOFF/UNSUBscribe**, auto-deletions, and all changes to users' personal options. A CHANGELOG file can be retrieved by list owners and site maintainers with the **GET** command and deleted with the **PUT** command like any other file (it is not necessary to make catalog entries for CHANGELOG files).

In 1.8d, change-logs could be only enabled or disabled. There was no facility to rotate change-logs at all, so once enabled, a change-log simply kept growing until it was deleted. Since it can grow quite large, it was recommended that this option be enabled under 1.8d only if the list owner(s) kept it pruned on a regular basis. Alternately the option could be enabled for problem resolution, and disabled after debugging.

Non-VM LISTSERV 1.8e has an enhanced **Change-Log=** list header keyword that allows list owners to rotate change-logs along the same lines as they rotate list notebook logs.

Rotated logs are renamed to *listname.***CHANGELOG-***yyyy***[***mm***[***dd***|***w***]]** and can be retrieved as usual with the GET command, which was changed to recognize these as change-logs.

When the feature is introduced (ie when upgrading to 1.8e), old logs will be renamed based on the date in the last entry. If the last entry is compatible with the newly introduced period (eg logs are YEARLY and the last entry is 2001), the log will continue, even if there were also entries for 2000. The definition of the 2001 log is that it contains all the entries for 2001, not that it is the first log started in 2001. Logs are not split as the feature is introduced.

If the rename fails (eg because you keep changing the period back and forth), the current log continues.

While LISTSERV on VM will accept all of the above settings without complaint, everything will behave as if you had specified SINGLE. This is because filetypes are limited to 8 characters on VM and it would be difficult to support a different naming convention just for VM.

The default is "Change-Log= No". For backward compatibility, "Change-Log= Yes" is equivalent to "Change-Log= Yes,Single".

Confidential= No | Yes | Service

Indicates whether the list should be hidden from users or not. A confidential list will not appear on the "**Lists**" command output. "**Confidential= No**" is the default value and indicates that the list is not confidential. "**Confidential= Service**" indicates that the list is to be hidden from users who are not in the list's service area (see "**Service=**" keyword) but not from other users.

"**`Confidential= Yes`**" means that the list is unconditionally confidential.

Please note that in LISTSERV 1.8c and following, the local list of (public) lists can be retrieved only by those users who are considered local, per the setting of the server-wide **`LOCAL=`** variable in LISTSERV's site configuration file. All other users will be told that none of the lists on the server are visible via the LISTS command, and will be referred to the use of the **`LISTS GLOBAL`** *`search-text`* command or to the CataList. This is regardless of the setting of **`Confidential=`** as outlined below.

Exit= *filename*

This feature and keyword are not available in LISTSERV Lite.

Background for non-technical users: an "exit" is a program supplied by the customer to modify the behavior of a product (such as LISTSERV) in ways that the supplier of the product could not anticipate, or could not afford to support via standard commands or options. The product checks for the presence of the "exit" program and calls it on a number of occasions, called "exit points". In some cases, the "exit" program supplies an answer ("return code") to the main program, which adjusts its behavior accordingly. For instance, LISTSERV may ask an exit program "Is it OK to add JOE@XYZ.EDU to the ABC-L list?", and the program will answer yes or no, and possibly send a message to the user explaining why his subscription was accepted or rejected. In other cases, the "exit point" call is purely informative: the exit program gets a chance to do something, such as sending an informational message to a user, but does not return any answer. Because this "exit" is a computer program, it must be prepared by a technical person and installed by the LISTSERV maintainer.

LISTSERV version 1.8a and higher support list "exits". List "exits" allow you to control the major events associated with list maintenance. This makes it easier to tailor the behavior of LISTSERV to local requirements that are too specific to be addressed through standard facilities.

An exit is enabled by adding "**`Exit=`** *`filename`*" to the list header. For security reasons, all exits must be explicitly declared in the **`LIST_EXITS`** configuration variable (in the LISTSERV site configuration file). This prevents list owners from causing the invocation of arbitrary executable files through the use of the "**`Exit=`**" keyword.

This keyword is not generally usable by list owners without specific intervention by the LISTSERV maintainer, and thus is not otherwise documented here.

Local= *node1,node2,...*

This keyword is not available in LISTSERV Lite.

Defines the nodes which are to be considered as 'local nodes' for service area checking. The LISTSERV machine is automatically considered as a 'local node' and does not have to appear in the list. Subscribers from any of the local nodes will receive separate pieces of mail with a single recipient in the "To:" field – in other words, they will never receive a grouped piece of mail as non-local recipients would if there are more than one recipient in their node. Note that 'node' is a generic term that means "anything after the '@' sign in the network address". For instance, "SEARN" and "SEARN.SUNET.SE" are both valid node

names.

By default, this keyword takes its value from the **LOCAL** variable in LISTSERV's site configuration file.

PW= *list-password*

(Obsolete since version 1.8c [except for peered lists]; included for backwards compatibility)

Defines the list password. When sending the list back to the server, the password is prefixed to the list file for validation (see the **Validate=** command for more specifics). The **PW=** parameter is "invisible" once it is defined; that is, for security reasons, it does not appear either when the list is reviewed or when it is retrieved with a **GET** command by the list owner.

LISTSERV version 1.8c and higher generate a 16-character random password for lists at list creation time if this keyword is not explicitly defined, making such lists more secure from random hackers. List owners are now encouraged to use personal passwords (defined with the **PW ADD** command, q.q.v.) in preference to list passwords for this reason.

The one exception to this keyword's obsolescence is when you are creating peer lists. Peers must have the same **PW=** setting, so you cannot use the LISTSERV-generated random password when creating peers.

Service= *area1,area2,...*

This keyword is not available in LISTSERV Lite.

Defines the 'service area' outside of which subscription requests must not be accepted. When a **SUBSCRIBE** command is received, the "**Peers=**" keyword (if set) is checked first to see if there is a nearer peer list in the network. If this is the case, the command is forwarded to this nearer peer server. If not, the service area is checked to ensure that the recipient is acceptable; if it is not, the subscription request is denied. When the command is forwarded to a peer, the destination peer server might still deny access to the list if the subscriber is outside its own service area, if any.

It is important to note that the service area check is made only after the "best placement" check. This allows several servers in the same country to share an identical service area, for example "**Service= Germany**", and still have users subscribed to the best possible server.

For lists running the web archive interface: Starting with LISTSERV 1.8d it is possible to define "Service=" in terms of IP address blocks in order to limit access to list archive notebooks via the web archive interface. This is implemented as follows:

1. **Notebook= ...,Service**

2. "Service=" can contain entries of the form:

[^]IP(a.b.c.d[/e])

3. For any other keyword in "Service=" which contains neither period, wildcard

nor parentheses, if a site configuration variable called `IP_xxx` is defined, it is processed using the syntax in #2, except that the IP() is implicit, i.e., the syntax would be (for unix; no quotes for NT as usual):

`IP_LOCAL="192.36.125.0/24 ^192.36.125.199"`

If both #2 and #3 are present, they are combined. Likewise, you can have multiple occurrences of #2 or #3 and they will just be combined.

Access will be granted if the IP address matches at least one of the entries that do not begin with a `^` (you can also use a minus sign if you prefer) AND the IP address does not match any of the negative entries. Otherwise you get a normal login request without any further comment.

Note carefully that LISTSERV does not do a reverse lookup on the IP addresses you code into the Service= keyword! When coding IPs into `Service=` you must also code in FQDN values for allowed hostnames. Thus if you have a list that should be restricted to the 192.36.0.0/16 subnet, which belongs to a domain called FOO.COM, you really have to code something like

`* Service= FOO.COM,*.FOO.COM,IP(192.36.0.0/16)`

in order for everyone in the FOO.COM domain who needs access to be able to have it.

The default value is "`Service= *`" (e.g., any host).

Validate= No | Yes[,Confirm[,NoPW]] | All,Confirm[,NoPW]]

The `Validate=` keyword determines what level of validation (if any) is performed for various LISTSERV commands that apply to individual lists. There are six different settings ranging from very basic to very strict. The two most common settings are (arguably) "`Validate= No`" and "`Validate= Yes`".

Lists are protected from hackers at the most basic level by the fact that a list `PUT` operation always requires validation, regardless of the setting of the `Validate=` keyword. In other words, the list owner or LISTSERV postmaster must *always* use a personal password (set with the `PW ADD` command, q.q.v.) when he sends an updated version either of the list header or of the entire list back to the server, even if "`Validate= No`". This is to protect you from network hackers who might issue a command "from" your userid@host address to change list settings, such as who has the ability to `GET` and `PUT` the list, review concealed subscribers, etc. The default for this keyword is "`Validate= No`", but it is recommended that "serious" or "important" lists be changed to at least "`Validate= Yes`".

When "`Validate= Yes`", password validation applies to so-called "protected" commands (all of the commands that modify the contents of the list, for example `ADD`, `DELETE`, `SIGNOFF`, etc.--but *not* `SUBscribe` or `SET`). This implies that hackers cannot use these "protected" commands since they do not know the list owner's or LISTSERV postmaster's personal password. While at first glance this would also seem to mean that legitimate subscribers cannot use the `SIGNOFF` command, that is not the case, because for lists operating with "`Validate= Yes`" (i.e., without the "Confirm" option), LISTSERV may still use the "OK" mechanism in certain cases if it is deemed appropriate. LISTSERV's rationale is that the "`Validate=`" keyword describes the desired behavior for interaction

with the list owner and people who can be expected to use the list on a regular basis. **SIGNOFF** requests from legitimate subscribers and **DELETE** requests from registered node administrators (NADs) on behalf of a user on their machine, for instance, may be validated using the "OK" mechanism even though that was not requested, because users and node administrators are not generally expected to have a password with which to validate such requests.

A notable exception to the list of "protected" commands is the **SUBscribe** command, which can still be used (if enabled, for example, if "**Subscription= Open**") to get on the list; however, when "**Validate= Yes**", sending a second **SUBscribe** command for the same list (for instance, to correct a spelling error in your name) would result in the command being forwarded to the list owner and not immediately executed. Also note that the **SET** command used to set various personal subscription options is not a "protected" command and may be issued without need for validation even when "**Validate= Yes**".

A rundown of the six different settings and what they mean follows:

- "Validate= No": all commands except **PUT** are taken at face value with no validation. While users are not bothered with validation requests, the list is almost totally unprotected from attacks by hackers. For compatibility reasons, this is the default setting.

- "Validate= Yes": "protected" commands, such as **DELETE** or **ADD**, require password validation. For list owner commands, personal passwords set with the **PW ADD** command are accepted. Some user commands may accept a personal password, while others will cause the request to be forwarded to the list owners for verification. Other "protected" commands include **GET**, but do not include **SUB** or **SET**.

- "Validate= Yes,Confirm": protected commands are validated using the "OK" mechanism by default, although personal passwords are also accepted where appropriate. This is a good compromise between list security and list owner convenience.

- "Validate= Yes,Confirm,NoPW": protected commands are always validated using the "OK" mechanism. Passwords are not accepted, as they are not as safe as "cookies". This is the recommended setting for secure lists. Note that lists with this setting cannot be managed via the Web Management Interface.

- "Validate= All,Confirm": all commands causing a change in state, except the **PUT** command (which is always password-validated), are validated using the "OK" mechanism by default, although personal passwords are also accepted where appropriate. "Protected" commands (see above) are included in the class of commands that cause a change of state. Non-"protected" commands that cause a change in state include **SUB** and **SET**.

- "Validate= All,Confirm,NoPW": all commands causing a change in state (except **PUT**, as noted above) are always validated using the "OK" mechanism; passwords are not accepted, as with "Validate= Yes,Confirm,NoPW". Note that lists with this setting cannot be managed via the Web Management Interface.

Warning regarding obsolete values: Under Revised LISTSERV (that is, LISTSERV for VM prior to version 1.8a), "Validate= All commands" (or "Validate=

All") and "Validate= Store only" (or "Validate= Store") were the only acceptable values for this keyword. These old values are still accepted for compatibility reasons, but generate a warning with an explanatory message when you update the list header. Since these values are obsolete and may not be supported in the future, you should change any instance of these settings in your lists to the current equivalent values (or to other currently-acceptable values as you see fit):

"Validate= Store only" is now "Validate= No"
"Validate= All commands" is now "Validate= Yes"

Informational commands such as `QUERY`, `SHOW`, `INDEX` and `REVIEW` do not require any validation, regardless of the setting of `Validate=`.

Requests originating on the local machine via CP MSG or CP SMSG (on VM systems) or originating on the local machine via LCMD (on VMS, Unix, and Windows 95/NT systems) never require validation, as they cannot be forged.

In all cases save one, the `PUT` command must always be validated with the personal password of the list owner or LISTSERV postmaster who is executing the `PUT` operation. This is because LISTSERV is not currently able to (1) suspend the execution of your `PUT` command, (2) store your list header or other file in a temporary file, and (3) wait for your "OK" before executing the `PUT`. If your password is used only for the purpose of validating `PUT` commands, any password exposure is minimal as `PUT` operations are not part of everyday list management routine. VM users should note that `PUT` requests require no validation when submitted via CMS SENDFILE from the machine on which LISTSERV is running, as the operating system itself guarantees the authenticity of the transaction (and thus there is no need to store the file you are `PUT`ting and wait for an "OK"). This is the only case in which a `PUT` operation does not require a password.

Table B.1 shows how LISTSERV commands are influenced by the `Validate=` keyword under different settings. Some redundant commands (e.g., `JOIN`, `LEAVE`, and `UNSUBscribe`) are not documented in the table, but behave exactly as do their "official" counterparts (e.g., `JOIN` behaves exactly as does `SUBSCRIBE`). Some commands never require validation but are included for completeness because they are specifically "list-level" commands. If a command is not otherwise listed below, it is not influenced in any way by the `Validate=` keyword.

```
NONE = does not require any validation
PW = requires password validation
OK = requires OK validation, will not accept PW validation
OK/PW = requires OK validation by default but will also
accept PW validation
```

	Validate= setting is:					
Command	No	Yes	Yes, Confirm	Yes, Confirm, NoPW	All, Confirm	All, Confirm, NoPW
ADD(*)	NONE	PW	OK/PW	OK	OK/PW	OK
CHANGE(**)	NONE	PW	OK/PW	OK	OK/PW	OK
CONFIRM	NONE	NONE	NONE	NONE	NONE	NONE
DELETE(*)	NONE	PW	OK/PW	OK	OK/PW	OK
FREE(*)	NONE	PW	OK/PW	OK	OK/PW	OK
GET(***)	NONE	PW	OK/PW	OK	OK/PW	OK

GETPOST	NONE	NONE	NONE	NONE	NONE	NONE
HOLD(*)	NONE	PW	OK/PW	OK	OK/PW	OK
INFO	NONE	NONE	NONE	NONE	NONE	NONE
PUT(*)	PW	PW	PW	PW	PW	PW
QUERY	NONE	NONE	NONE	NONE	NONE	NONE
REVIEW	NONE	NONE	NONE	NONE	OK/PW	OK
SCAN	NONE	NONE	NONE	NONE	NONE	NONE
SEARCH	NONE	NONE	NONE	NONE	NONE	NONE
SET	NONE	PW	OK/PW	OK	OK/PW	OK
SIGNOFF	NONE	NONE	NONE	NONE	OK/PW	OK
SUBSCRIBE	NONE	NONE	NONE	NONE	OK/PW	OK
UNLOCK(*)	NONE	PW	OK/PW	OK	OK/PW	OK

Table B.1. LISTSERV list-level commands and how they are affected by Validate=.

(*)   All commands so marked may be issued only by a list owner or LISTSERV postmaster.

(**)   The **CHANGE** command has two syntaxes, one for general users, one for list owners/postmasters. General users will always be required to use "OK" confirmation, regardless of the **Validate=** setting. The values in the table above are for the syntax issued by list owners or the LISTSERV postmaster(s).

(***) '**GET** *listname*' may be issued only by a list owner or LISTSERV postmaster. General users may issue **GET** commands for notebook archives and/or files listed in the list's file catalog and with appropriate **GET** FACs only.

Please note carefully that Table B.1 assumes that you have defined default values for most keywords. For instance, the **SUBSCRIBE** command will require an "OK" confirmation if you have "**Subscription= Open,Confirm**" and "**Validate= No**"; **REVIEW** will only be available depending on how you have the "**Review=**" keyword set; **GETPOST** and **SEARCH** may require passwords depending on the "**Notebook=**" setting; etc. However none of these conditions are influenced directly by "**Validate=**" except as noted in the table.

In some cases, "**Validate= Yes**" will cause an "OK" request to go out instead of requiring a password (i.e., if no password was included with the command). This is to avoid confusion on the part of a subscriber who may or may not have a LISTSERV password and who may not understand why he is being asked to provide a password before a given command will work.

Note also that even with "**Validate= No**" some users may be required to confirm commands with the "OK" method if they are sending commands via a web browser and **WEB_BROWSER_CONFIRM=** is set to 1 (the 1.8c default; under 1.8d the default is 0, or disabled) in the site configuration file.

History: This keyword was revised substantially in versions 1.7f and 1.8a. The "OK" command confirmation mechanism was introduced in version 1.7f, where it was used to implement the "Subscription= Open,Confirm" address verification mechanism. When a user tries to subscribe to a mailing list with that setting, he is mailed a confirmation request with a randomly generated confirmation key, also known as "magic cookie". The user replies to the message, types "OK" in the message body, and the command is confirmed. If for any reason the user's address cannot be replied to, the confirmation request is never received (or the "OK" message never arrives) and the user is not added. In versions 1.8b and following, this procedure is also used for authentication purposes. Since the confirmation codes are valid only for a single command, this actually provides

better security than personal passwords, while simplifying book-keeping.

As before, the security level of the mailing list is controlled through the "Validate=" keyword. The contents of this keyword, however, have changed from earlier versions (the old values are still accepted for compatibility reasons, but generate a warning with an explanatory message when you update the list header. This may change in subsequent versions, so it is advisable to use the new values).

Subscription Keywords

Confirm-Delay= *number*

This keyword is not available in LISTSERV Lite.

This parameter is an integer representing the number of hours LISTSERV will hold subscription jobs requiring confirmation before flushing them from its queue. For instance, if **Subscription= Open,Confirm** and **Confirm-Delay= 72**, LISTSERV will accept a subscription request pending confirmation, send the "cookie" command confirmation request, and will wait 3 days (72 hours) for that confirmation to be received. If the period expires before the "cookie" is received, the subscription request is deleted and the subscriber must resubmit his or her request. The default setting is 48 hours (2 days).

Many unreliable gateways have a turnaround time of several days, and this is another way to filter them: if the confirmation delay is long enough, they will never manage to subscribe and you will not have to put up with gateways that take a week to realize that the subscriber's account has expired and return a week's worth of delivery errors. On the other hand, if you do want to let these people in, you will have to increase the confirmation delay to a week or so (1 week=168 hours).

In LISTSERV 1.8b and later, this keyword can also extend the period of time during which postings to a list coded "**Send= Editor,Hold**" are held before they are flushed. The default (and minimum) for holding such postings is 7 days (168 hours). Note that you can only increase this period with "**Confirm-Delay=**", not decrease it. Thus for a list with "**Send= Editor,Hold**" and "**Confirm-Delay=48**", the holding period would still be 7 days. But for a list coded "**Send= Editor,Hold**" and "**Confirm-Delay=240**", the holding period would be 10 days (240 hours).

Please inform the LISTSERV maintainer before any significant increase to the value of "**Confirm-Delay=**", particularly if your list is coded "**Send= Editor,Hold**" or "**Send= Editor,Hold,Confirm**", as the increased delay could cause a problem with disk space availability.

Note that if you increase "**Confirm-Delay=**" to extend the holding period for postings, you also are increasing the period during which LISTSERV will hold subscription jobs requiring confirmation.

See also **Subscription=**.

Default-Options= *option1,option2,...*
A "Default-Options" keyword is available to define initial personal options for new subscribers. The syntax is the same as for the **SET** command, except that options are separated by commas in the usual fashion. Setting **Default-Options=** does not affect existing subscribers. If you want existing subscribers to have these settings, you must update them manually with a **SET** *listname* *options* **FOR *@*** command.

A typical **Default-Options=** setting might be:

* Default-Options=Nofiles,Norepro,Msg

Note that if you have "**Default-Options= NOPOST**" for your list and you add an editor or a moderator as a subscriber, you will have to manually reset the editor to **POST** (with "**SET** *listname* **POST FOR** *userid@host*") before things will work properly. You will know that this is necessary if your editor or moderator can successfully approve postings but is then told that he or she cannot post to the list.

Starting with LISTSERV 1.8d, all default options are applied to non-subscribers, so it is possible to force even non-subscribers to post through a moderator by simply setting "**Default-Options= REVIEW**", or lock them out altogether by setting "**Default-Options= NOPOST**". This works even if your list is set "**Send= Public**", in which case there is a side benefit: the setting will stop people whom you have set to REVIEW or NOPOST from signing off the list and being able to post.

Two caveats regarding the use of this keyword under 1.8d:

Default-Options= REVIEW is overridden for addresses defined in **Editor=** or **Moderator=**.

Default-Options= NOPOST in conjunction with **Send= Editor** causes mail from non-subscribers to be forwarded to the appropriate Editor for approval rather than simply rejecting it with a "you are not allowed to post" message.

Default-Topics= *topic1,topic2,...*

This keyword is not available in LISTSERV Lite.

A "**Default-Topics=**" list header keyword is available to define the initial topics for new subscribers. The syntax is the same as for the **SET TOPICS** command, except that topic names are separated by commas in the usual fashion and that the first topic may not start with a + or - sign (there is nothing to add to, as this is a new subscription). This is similar to "**Default-Options=**" in that it does not affect existing subscribers. Users who signed up before topics were enabled on the list are automatically subscribed to all topics.

As with **Default-Options=**, setting this keyword affects only subscribers who sign up after you set it. If you want existing subscribers to be set to these topics, you must update them manually with a **SET** *listname* **TOPICS:** *topics* **FOR *@*** command.

Subscription= Open [,Confirm]
Subscription= By_Owner[,Confirm]
Subscription= Closed
This keyword defines whether or not new users are allowed to subscribe to the list, and if not, whether their subscription requests are to be forwarded to the list owner or not.

Open:	New users are allowed to subscribe to the list.
Open,Confirm:	Before new users are allowed to subscribe to the list they must confirm their address with an "OK" response. No list owner intervention is required.
By_Owner:	New users are not allowed to subscribe, but their requests will be forwarded to the list owner. This is the default.

By_Owner,Confirm: (1.8e) Similar to "By_Owner", but before the request is forwarded to the list owner, the would-be subscriber must first respond to an "OK" confirmation request. NOTE CAREFULLY that you MUST specify "By_Owner" rather than "By Owner" -- the underscore is required for this syntax.

Closed: New users are not allowed to subscribe, and their requests are not to be forwarded to the list owner.

Note that "Subscription= By Owner" is still supported but is deprecated, and as noted, the underscore must be supplied if ",Confirm" is used.

(The ",Confirm" option is used in conjunction with "Subscription= Open" and "Subscription= By_Owner" only. It has no effect with "Subscription= Closed".)

One problem plaguing some mailing lists is one-way or non-repliable addresses. Most of the time this is due to a small number of faulty gateways, which one can just ban from the list until their maintainers address the problem. But sometimes the very topic of the list is bound to attract a large number of people connected through such gateways, and the amount of domains to filter out becomes unmanageable. This is particularly problematic when the gateway keeps quiet for a few days, and suddenly returns hundreds of delivery errors with a promise to keep doing so every day for another 6 days.

This problem can be avoided by probing the return address before accepting the subscription. If the address cannot be replied to, only one delivery error will be bounced to the list owner (perhaps for several days, but there will be a single undeliverable message). With a gateway that waits 3 days before sending its first delivery error, however, there can be hundreds of messages "in the pipe" if the subscription is accepted directly. This probing is activated by specifying "**Subscription= Open,Confirm**" in the list header. When a user attempts to subscribe to the list, he is mailed a confirmation request with a randomly generated "confirmation code". The procedure for confirming the subscription is simple - you just reply to the message, type "OK", and send. If the return address does not work, the request will never be confirmed and the user will not be subscribed. And since the user cannot guess the confirmation code he will be assigned, he cannot "cheat" by sending the confirmation along with his request.

The subscription request also expires after a certain amount of time, as determined by the "**Confirm-Delay=**" keyword (the default is 48h).

Similarly, starting with LISTSERV 1.8e, it is possible to code "**Subscription= By_Owner,Confirm**". This adds an address probe into the usual procedure for subscriptions that must be approved by the list owner. The behaviour with "**Subscription= By_Owner**" is

1. User sends SUBSCRIBE command.
2. LISTSERV forwards to list owner.

and no check is done to verify that the user's address is viable. Adding the "**,Confirm**" parameter changes the behaviour to

1. User sends SUBSCRIBE command.
2. LISTSERV asks for OK confirmation (new).
3. User confirms.

4. LISTSERV forwards to list owner.

With this setting in place, list owners who run restricted-subscription lists will no longer have to discover for themselves whether or not a potential subscriber's address has been spoofed or is otherwise non-viable, since they will never see subscription requests that have not been confirmed by the subscriber.

Other Keywords

Categories= *category1,category2,...categoryn*

> Note: the full list of categories may not be available when version 1.8d is released.

> Sets search categories for this list (by default, none are defined) for the CataList service (see Chapter 3.3 of the *List Owner's Manual* for details). For instance, you might have a list on the topic of great opera tenors of the 20th century, and want anyone searching the CataList based on certain topics to find your list. You might therefore code:

> ```
> * Categories= Arts:Music:Opera,Arts:Music:Opera:Singers
> * Categories= Arts:Music:Opera:Pavarotti
> ```

> and so forth.

DBMS= No
DBMS=Yes[,Table(*xxx***)][,Email(***xxx***)][,Name(***xxx***)][,Uemail(***xxx***)][,Options(***xxx***)]**

> *This keyword is not available in LISTSERV Lite.*

> *This functionality is not available under VM. Under non-VM it requires an appropriate DBMS application (not provided by L-Soft) be installed on the LISTSERV machine.*

> "DBMS=" (introduced in 1.8d) is used to tell LISTSERV that the list of subscribers is kept in an ODBC or OCI database rather than in the traditional `*.LIST` file. In order to set this keyword to anything but the default, DBMS support must be installed on the LISTSERV server. Please see the *Developer's Guide for LISTSERV* (available separately) for more information on installing support for and using the DBMS and Mail Merge functions.

> *Warning: List owners should NOT add this keyword to their list header as the use of the DBMS= keyword presupposes existing DBMS support that has been configured to work with LISTSERV. Further, list owners should NOT change the value of this keyword if it exists in their list header, as changing any of the parameters could lead to unexpected results.*

> **Windows OCI Support Withdrawn:** In LISTSERV 1.8e, it is possible to define multiple simultaneous connections to DBMS tables, and as a result, previous OCI support for Windows NT/2000 (which was provided to work around the earlier limitation of only a single ODBC connection at a time) has been withdrawn. OCI databases may still be accessed via ODBC.

> When migrating an existing list to use a DBMS, you are responsible for migrating the subscriber data to the DBMS, if necessary (in many cases, the subscriber data will already be in the DBMS, possibly in a slightly different format). Once you add the "DBMS= Yes" keyword, LISTSERV stops accessing subscriber data from the xxx.LIST file and uses the DBMS instead.

> The default is "DBMS= No", i.e., keep subscriber information in a traditional `*.LIST` file.

Indent= *number*

This keyword is not available in LISTSERV Lite.

Determines the minimum number of columns allowed for list addresses in response to the **REVIEW** command. The default is **Indent= 40**.

Language= *idiom*/*option*[,*option*]

This keyword is not available in LISTSERV Lite, except that it can be set to "Language= Exchange" to avoid suppression of application/ms-tnef attachments as noted below.

Defines the language in which information mail and messages are to be sent to subscribers of the list. The postmaster must have provided the required data file (called ***idiom*.MAILTPL**, where ***idiom*** is the name of the language specified by this keyword) to the server. The default is "**Language= English**", which uses **DEFAULT.MAILTPL**.

L-Soft does not provide non-English templates.

Starting with 1.8d, this keyword was expanded to cover the following functionality:

Language= HTML: This setting allows you to specify that LISTSERV's administrative messages (i.e., those specified in the MAILTPL) be sent out in HTML format. You specify either **Language= HTML** or **Language= *idiom*,HTML** to enable this feature. *Note carefully that this setting does not affect the WELCOME or FAREWELL files.*

Language= NOHTML: This setting allows you to specify that LISTSERV strip any HTML attachments from postings (while retaining HTML tags sent in the body of plain text messages). You specify either **Language= NOHTML** or **Language= *idiom*,NOHTML** to enable this feature.

The actual function of this setting is to remove the attachment that contains the HTML mail from the message. It does not remove HTML tags from plain text messages. This means that setting this option will not suppress HTML in messages sent from (for instance) Eudora Pro 3.x (since Eudora Pro 3.x does not send the HTML message as a MIME attachment with a plain text alternative).

In 1.8e and following, if an HTML message does not contain a plain text alternative, and **Language= NoHTML** is set, the message will be rejected. This is a change from the behavior in 1.8d, which would allow such messages through.

Language= EXCHANGE: This setting allows you to specify that LISTSERV keep Microsoft Exchange attachments in postings (the default is to remove them). You specify either **Language= EXCHANGE** or **Language= *idiom*,EXCHANGE** to enable this feature. Note that this affects "application/ms-tnef" attachments only-- LISTSERV does not currently strip WINMAIL.DAT attachments.

These three formatting options are not mutually exclusive and may be defined in any grouping (in other words, **Language= HTML,NOHTML,EXCHANGE** is legal although it is unlikely anyone would want to use it).

Limits= Sub(*number*),...

This keyword is not available in LISTSERV Lite.

This keyword is available only with the ISP add-on, and may only be added or changed by the LISTSERV Maintainer. Defines specific limits for a list. Currently only the number of subscribers can be limited, for example,

```
* Limits= Sub(100)
```

This keyword may only be added or changed by the LISTSERV postmaster, and the list creation password is required for the list `PUT` operation when the keyword is added or changed. The list owner may execute a `PUT` operation with this keyword defined in the header as long as the values for the keyword are not changed.

Long-Lines= Yes | No

This keyword is not available in LISTSERV Lite.

Enables or disables "long-lines" support. This keyword was added to maintain compatibility with LISTEARN and will be removed in a future version of LISTSERV. The default is "`Long-Lines= Yes`". It is unlikely that this keyword will need to be set for any list.

Mail-Merge= Yes | No

This keyword is not available in LISTSERV Lite.

> **Documented Restriction:** Note that LISTSERV's mail merge functionality REQUIRES the use of LSMTP Classic as the outgoing MTA. Mail merge does not work with sendmail, qmail, Post.Office, Netscape Mail Server, Microsoft Exchange, PMDF, MX, or any other MTA except L-Soft's LSMTP Classic mailer.
>
> Under unixes not supported by LSMTP Classic this may require that you set `SMTP_FORWARD=` accordingly in `go.user`, to point to a separate machine running LSMTP (for instance, a dedicated Windows NT LSMTP machine). Under OpenVMS or Windows NT can run LSMTP Classic either on the same machine (the preferred method), or on a separate machine if desired. The main point is that the outgoing mail-merge postings MUST be handled by LSMTP Classic. (LSMTP Lite does not support mail-merge.)
>
> Mail merge functions are documented fully in the *Developer's Guide for LISTSERV*, available separately.

`Mail-Merge= Yes` makes every posting to the list a mail-merge operation (see the *Developer's Guide to LISTSERV* for more information about formatting mail merge jobs). Digests and indexes also become mail-merge jobs. Header modifications are done in the same way as with a normal list, you confirm with OK as usual, etc. The text will then follow the same rules as when entered in a mail-merge job using the web interface--ASSUMING that your mail program did not do anything fancy to the text, such as replace all ampersands with =xx or rewrite everything in rich-text format. It is fundamentally difficult and unreliable to use a mail program as the client for mail-merge since mail programs are engineered to encode information in fancy and unpredictable ways that were not

used in the previous version, whereas mail servers are engineered to only look at the information they positively need to look at to get the job done, so that they do not inadvertently get in the way of new, fancier encoding methods. Therefore it is important when using a mail program to send mail-merge jobs that you ensure that the mail program sends plain text rather than any kind of encoded text.

This keyword should be used only on lists that are set up as announce-only mailing lists (that is to say, posting should be restricted to the list owner(s) only for security purposes). Because this keyword alone does not restrict the ability to post, the list owner(s) should ensure that the "**Send=**" keyword is set appropriately. Note that if you do not restrict the ability to post, anyone who is otherwise permitted to post to the list will also be able to send mail-merge jobs to it.

This method can be used to attach files unconditionally. It cannot be used to send file 1 to some people and file 2 to others because the mail program knows nothing about mail-merge and, as such, does not mark them as conditional blocks. Any markings you provide are part of the text attachment you are working in.

The default is to send mail normally, in other words, **Mail-Merge= No** . As noted above, if your LISTSERV installation does not use LSMTP as its outbound mail server, you should never change this setting from the default.

Translate= Yes | No

Determines whether LISTSERV keeps or removes control characters from files which it distributes. "**Translate= Yes**" removes control characters; "**Translate= No**" keeps them. The default setting is "**Translate= Yes**".

Setting "**Translate= No**" may be required for accurately passing messages written using double-byte character systems such as those available for the Japanese language.

Default Values for all keywords

Ack= No   Attachments= Yes
Auto-Delete= No if "Validate= Yes", Yes,Semi-Auto,Delay(4),Max(100) otherwise
Categories= <none>   Change-Log= No
Confidential= No   Confirm-Delay= 48
Daily-Threshold= 50   DBMS= No
Default-Options= <none>   Default-Topics= <none>
Digest= Yes,Same,Daily if "Notebook= Yes", No otherwise
Editor= <none>   Editor-Header= Yes
Errors-To= Owners   Exit= <none>
Files= No   Filter= <built-in>
Indent= 40   Internet-Via= <none>
Language= English   Limits= <not set, ISP option only>
List-Address= <none> (per LIST_ADDRESS system default)
List-ID= <none>   Local= <none> (per LOCAL system
                                                        default)
Long-Lines= Yes   Loopcheck= Full
Mail-Merge= No   Mail-Via= DISTRIBUTE
Moderator= <none> (defaults to first Editor if "Editor=" is defined))
New-List= <none>   Newsgroups= <none>
NJE-Via= <none>   Notebook= No,A,Single,Private
Notebook-Header= Short   Notify= Yes
Owner=   (This is a mandatory parameter which must be filled with at least one
          person's network address in userid@node or userid@fqdn format)
Peers= <none>   Prime= Yes
PW=   (randomly generated at list creation time if not specifically defined)
Renewal= <none, disabled>   Reply-To= List,Respect
Review= Private   Safe= Yes
Send= Public   Sender= List
Service= *   Sizelim= <none>
Stats= Normal,Private   Sub-lists= <none>
Subject-Tag= short name of the list, for example, MYLIST-L
Subscription= By_Owner   Topics= <none>
Translate= Yes   Validate= No
X-Tags= Yes

Appendix C: Sample Boilerplate Files

So-called "boilerplate" files are handy for list owners who find themselves answering the same questions over and over again. Usually these questions refer to basic LISTSERV usage. You can save yourself a lot of time by keeping files on-line such as the ones below to cut and paste into replies. Feel free to edit these to suit your own tastes (or compose your own!).

(Be sure to insert the appropriate list names and LISTSERV hosts as required.)

C.1. Subscription requests sent to the list

LISTSERV subscription requests need to be sent to the LISTSERV address rather than to the list itself. You do this by sending mail to LISTSERV@*host* with the command

SUB *listname* Your Name

as the body of the message. If you are unfamiliar with LISTSERV and its associated commands, I suggest that you add the commands

INFO GENINTRO
INFO REFCARD

as additional lines of your message. LISTSERV will then send you a file containing a General Introduction to Revised LISTSERV that will give you some instruction on the service and a Quick Reference Card of the various commands.

Thanks for your interest. If you have trouble subscribing with this method, please let me know and I will attempt to help.

[if you have `Subscription= Open,Confirm` you might want to add the following:]

Because LISTSERV verifies mailing paths for new subscribers (a process not implemented when the list administrator adds people manually), it is preferred that users subscribe themselves by the method outlined above.

C.2. User is sending other commands to the list, or to the *-request address for the list

On Sun, 20 Mar 1994 22:44:25 -0800 (PST) you said:
>"INFO REFCARD"

You need to redirect LISTSERV commands like the above (minus the double quotes by the way :)), to <listserv@*host*>. The *-request type addresses are for reaching the person that run the list.

[another version:]

You've sent mail that appears intended for a mailing list to one of the addresses used to reach the list owner. That is, rather than sending your mail to *listname@host* you've sent the note to OWNER-*listname@host* or *listname*-REQUEST@*host.* Please re-send the appended note to the list address if you haven't done so already.

----------- original message follows:

C.3. User isn't subscribed but complains that he's getting mail anyway

[Use this one after you have done an exhaustive search of the list and determined that the person simply isn't on the list. Typically the user is subscribed to a redistribution list and doesn't realize it.]

Unfortunately I can't unsubscribe you from *listname* because you aren't subscribed to *listname@host*. I have run a check to see if you might be subscribed under a slightly different network address and have not found anything.

There are a few possibilities you should look into. Are you getting a digest? Are you perhaps getting a redistributed copy of postings, possibly from a redistribution list? If you look at the mail headers, and there is an indication that you may be getting the postings from another source, you will have to ask the people that run the other source to remove you from their list.

C.3. User unsubscribed successfully but is still getting list mail

I've done a search of *listname* for a possible duplicate subscription for you and have not found anything. It's possible that the mail you are receiving was actually sent from *listname* before your unsubscribe request was processed. Depending on the routing, it could take anywhere from 24 to 48 hours for all such messages to get through the network, so please be patient.

C.4. Quoted replies from user's mail client includes message headers in the mail body, causing them to be bounced to the list owner

[If you forward such messages to the list, or back to the sender, you can add the following at the beginning. I ran across this one in the CBAY-L mailing list archives, and edited it slightly.]

This message was sent to me from LISTSERV instead of the list. The original message included the entire message being replied to, including the mail headers. These headers in the body pointed to the list itself. LISTSERV has mail-loop avoidance code and when it sees headers that it thinks it generated itself, it bounces the message to the list owner. If your mail client does this, please remember to delete such "included header lines" from the body of your list replies.

------original message------

C.5. Asking a postmaster for help on a bounced address you've set to NOMAIL, with a cc: to the bounced address

Postmaster(s),

Can you shed any light on the following error message? Please let me know what you find as I have removed the e-mail address from the mailing list in question and would like to restore service as soon as is feasible. Thanks.

Aside to user: Should this note reach you (meaning that the mail delivery problems have been resolved), you can re-enable your mail service by sending mail to listserv@*host* with the command,

      SET listname MAIL

C.6. You get a delivery error that doesn't specify which user account is causing the bounce

Postmaster,

I received the appended mail delivery report from your system and need help isolating the e-mail address that is causing the error. That is, there are multiple recipients from your system on the list but the delivery error doesn't explicitly mention any of the users on the list. I'm including a list of subscribers from your system. If any of them are no longer valid, or aren't usable address for some other reason, please let me know.

---- list of e-mail address on the indicated list follows:

C.7. You've set a user to DIGEST because of bouncing mail and the user is asking why he is now getting the digest

I received a mail delivery error for your address and issued a

> SET listname DIGEST

on your behalf to minimize the number of bounce messages. I also sent a copy of the error I received to your postmaster (or the postmaster of the mail gateway that generated the error), asking for help. And since such delivery problems are often transient, I CC'd a copy of that note to your address, and included instructions for turning your mail back on. Apparently I didn't hear anything from your postmaster, or he/she said not to turn your mail back on until the problem was resolved. If they had responded and said the problem was resolved, I would have set you back to MAIL..

The other possibility is that I received a mail message indicating that there was some temporary problem with your account. In that case, for example if you had exceeded your disk quota and couldn't receive any new mail, I would not have bothered your postmaster. I have a different form letter that I send when that happens. Again it explains what has occurred and includes instructions for re-enabling your mailing list subscription. But I only send that one to the address the list member. Either way, whatever was wrong has been corrected, and you'd probably like to start receiving mail again. So, here's how you can restore your mail service. If you have any problems doing so, please let me know and I'll help. But since I don't know which of the three mail service options you had chosen before, I can't do it for you without guessing. You can re-enable your mail service by sending mail to listserv@*host* with one of the following commands

> SET listname MAIL
> SET listname DIGEST (if you want digest-format mail)
> SET listname INDEX (if you want digest-index-format mail)

in the *body* of the mail message. Please note that these settings are mutually exclusive, you can't choose more than one. :)

Appendix D: Related Documentation and Support

D.1. Official L-Soft Documentation

Subscribers to LISTSERV mailing lists will find answers to many of their questions in the

User's Guide to LISTSERV Version 1.8d.

LISTSERV maintainers will be interested in the *Site Manager's Operations Manual for LISTSERV 1.8d*.

All of L-Soft's manuals for LISTSERV are available in ascii-text format via LISTSERV and in popular word-processing formats via **ftp.lsoft.com**. They are also available on the World Wide Web at the following URL:

URL: http://www.lsoft.com/manuals/index.html

L-Soft invites comment on its manuals. Please feel free to send your comments via e-mail to MANUALS@LSOFT.COM, and mention which manual you are commenting on.

D.2. User-Created Documentation

LISTSERV began as a non-commercial product, with fairly-complete but terse documentation. Over time, list owners and LISTSERV maintainers found that it was necessary to amplify some of the documentation, and wrote their own manuals. Some of these manuals, while they may be dated, are still of value and are still available.

D.2.1. LISTSERV List Owner Information Area on LISTSERV.BUFFALO.EDU

This site replaces the old LSVOWNER package mentioned in earlier versions of this manual. Here are the files currently (18 September 1998) available:

How to Maintain your Class E-mail Distribution List
 Guide from Julia Cohan, School of Management
LISTSERV "Ins & Outs"
 Guide from Hugh Jarvis, Faculty of Social Sciences
LISTSERV list header keywords information
LISTSERV list management commands
Explanation of LISTSERV List Columns 81-92
List Owner USENET News Gateway Information
List Owner New List Announcement
List Owner Welcome to the List Message
List Owner General Information
List Owner How To Maintain A List Header
WWW Gateway to LISTSERV(tm) Lists

These files are found at the URL

http://listserv.buffalo.edu/owner/

(Please note that any links found there to list management or list creation are for local use only and remote users should avoid trying to use them :)

D.2.2. LISTSERV TIPS

LISTSERV TIPS is a document with the formal title "List Management Tips for LISTSERV Postmasters and List Owners". It was compiled in 1991 by Lisa Covi. LISTSERV TIPS can be retrieved from the LISTSERV hosts at SEARN and UBVM (among others) with the usual GET command.

The basic document has never been updated and is very BITNET-oriented, but is still

quite useful as a basic information source on running a list. To view LISTSERV TIPS on the World Wide Web, see Holly Stowe's LISTSERV FAQ at

`http://www.listserv.iupui.edu/`

You can also go directly to LISTSERV TIPS at

`http://www.iupui.edu/~lstmaint/listserv_tips.shtml`

Please note that LISTSERV TIPS is not an official L-Soft publication and is therefore not maintained by L-Soft.

D.2.3. FSV GUIDE

FSV GUIDE (formal title: "Setting Up the LISTSERV File Server – A Beginner's Guide") is really aimed at LISTSERV maintainers, but it is a handy guide for list owners who have filelists on VM systems as well. Ben Chi (bec@albany.edu) is the author of this fine document about the VM LISTSERV file server system. You can get a copy from LISTSERV@ALBANY.EDU.

Appendix E: Revision History

Note carefully that these revision notes are cumulative. For instance, if a revision note says that chapter x was renumbered to chapter y, or that a change was made to chapter y, and a later revision note says that chapter y was moved to another manual, obviously the later revision supersedes the earlier one (which is nonetheless retained for reference). So before complaining that you can't find a certain revision in the manual, please read the entire list of revisions to make sure something else didn't change in the meantime :) Revision notes from the release of the 1.8e manual through the beta stages will be removed when the manual for the next version is released and this process will begin again.

011214-001   **Initial beta of 1.8e List Owner's Manual.**
020523-001   **Initial release of 1.8e List Owner's Manual.**
20020801-001 2.3.1, clarified that it was 17.3 of the site manager's manual that was being pointed to.
20021022-001 2.16 did not have the complete documentation for the CONTENT_FILTER template form as released.
20021031-001 .FO RAGGed documented in 9.3.
20030404-001 4.8, default for Auto-Delete was wrong.

Index