L-Soft international, Inc.

List Owner's Manual for LISTSERV^{TM}, version 1.8b

December 12, 1995
Revision 2

The reference number of this document is 9512-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:

`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`**.

Reference Number 9512-UD-02

Table of Contents

(this page left intentionally blank)

L-Soft international, Inc.

List Owner's Manual for LISTSERV™, version 1.8b

August 14, 1995
Revision 1

The reference number of this document is 9505-UD-02.

Preface: 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)
listname	name of an existing list
node	BITNET nodeid or Internet hostname of a BITNET machine which has taken care of supplying an ':internet' tag in its BITEARN NODES entry; or the fully-qualified domain name (FQDN) of an Internet host.
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

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, VMS™, unix® and Windows NT™, and has already been ported to Windows 95™ (formerly Windows 4.0).

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 Documents and Support*, for information on how to get one.

2. Starting a Mailing List – The Basics

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 "List of Lists" maintained by LISTSERV itself. Send the command

```
LIST 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, **LIST GLOBAL IBM** would result in the following being returned to you:

```
Excerpt from the LISTSERV lists known to LISTSERV@PEACH.EASE.LSOFT.COM on 6 Feb
1995 09:57
Search string: IBM

**********************************************************************
* 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
---------------   ---------------------------------
9370-L               9370-L@NIC.SURFNET.NL
                       IBM 9370 and VM/IS specific topics list

AIBIBL               AIBIBL@PLEARN.EDU.PL
                       ACADEMIC INITIATIVE IBM , PROJECT "LIBRARY SYSTEMS" AIBIBL

AIX-L                AIX-L@PUCC.PRINCETON.EDU
                       IBM AIX Discussion List

AIXNEWS              AIXNEWS@PUCC.PRINCETON.EDU
                       IBM AIX News to Mail Distribution
```

Figure 2.1. Sample output of **LIST GLOBAL IBM**

(63 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 archive on NDSUVM1 (or VM1.NODAK.EDU), where the NEW-LIST project at North Dakota State University has been storing list announcements for several years. The NEW-LIST archive contains information about LISTSERV lists as well as about lists running

[1] Parts of this section were adapted from "Some Lists of Lists", compiled by Marty Hoag and available via Gopher to `vm1.nodak.edu` (path: `1/Local LISTSERV Resources/NEW-LIST Project`).

on other types of servers. This information is accessible either via LISTSERV database searches or by Gopher to **`vm1.nodak.edu.`**

- Use a Gopher or World-Wide Web (WWW) client to access the InterNIC database services, where a list of lists is maintained. The InterNIC Gopher is located at **`rs.internic.net`**.

- Get a copy of the Interest Groups List of Lists maintained by SRI on its server at **`sri.com`**. Note that this is a 500KB (or larger) file.

```
ftp sri.com
user:   anonymous
password:   your_user_id
cd netinfo
get interest-groups
```

- Get a copy of the Interest Groups list maintained by David Avery at Dartmouth College. This is a merged list of the LISTSERV list of lists and the Internet Groups list, with one-line entries for each group. Again, this is a large file, so be sure you have room for it. Also available on the **`dartcms1`** server is a Macintosh Hypercard application that formats this file nicely (assuming you have a Macintosh). You may want to do a **`dir`** of the files available in the **`siglists`** directory to see if there is anything else there that might be useful.

```
ftp dartcms1.dartmouth.edu
user:   anonymous
password:   (not required by dartcms1)
cd siglists
get internet.lists
```

  Note that all these files (except the large data stack) can also be retrieved from LISTSERV@DARTCMS1.DARTMOUTH.EDU.

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

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 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.[2] 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.

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.

<u>Access Control Keywords.</u> 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.

<u>Distribution Keywords.</u> 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.

<u>Error Handling Keywords.</u> 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.

<u>List Maintenance and Moderation Keywords.</u> 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 who) 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.

<u>Security Keywords.</u> 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), whether or not the list is protected by a password, and the level of security necessary for changes to the list itself. The "Exit=" keyword is also contained in this group.

<u>Subscription Keywords.</u> 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.

[2] 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.

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. Retrieving and editing the list – some considerations

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 **`GET`** ***`listname`*** command to LISTSERV. This will cause the server to mail you a copy of the entire list (header and subscriber list).

If you want to change header keyword settings only, it is probably advisable to issue the **`GET`** command with the **`(header`** switch:

```
GET listname (header
```

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 **`GET`**s and **`PUT`**s by different list owners. For instance, Owner(A) **`GET`**s the list without locking it. Owner(B) then also **`GET`**s the list. The owners make differing changes to the list header. Owner(B) **`PUT`**s his changes back first. Owner(A) then **`PUT`**s 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: 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).

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".
```

2.6. 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@LSOFT.COM (Nathan Brindle)
* Owner= nathan@linus.dc.lsoft.com
* Owner= Quiet:
* Owner= ncbnet@linus.dc.lsoft.com,cheng@linus.dc.lsoft.com
```

Figure 2.2. 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.dc.lsoft.com,cheng@linus.dc.lsoft.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.

2.7. Adding and changing a list password

When creating the list, the LISTSERV maintainer should assign a password for the list. If this password was not assigned, it is *highly recommended* that you **GET** the list header as described in section 2.5 (use the **(HEADER NOLOCK** options) and add a line to the header as follows:

```
* PW=MYPASSWD
```

Replace "MYPASSWD" with the word you choose. Note that there should *not* be a space between "PW=" and your password. This is the only way to change the list password; for security's sake, there is no LISTSERV command that will change it "on the fly". For additional security, the list password never appears in the list header on subsequent **GET**s; to all intents and purposes it is invisible once it is assigned.

You can change the list password whenever you store the list by assigning a new value to the "PW=" keyword, but when you store the list with the changed password, you must use the old password in the **PUT** command line (because until you actually store the new password, LISTSERV will still be looking for the old one). See section 2.9 (below) for an example of a list header ready to be stored with a new password defined.

2.8. 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 password you have previously defined with the **PW=** list header keyword. 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.

2.9. 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.

2.10. A sample list header file

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. For instance,

```
GET MYLIST PW=MYPASSWD (HEADER
```

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,A,Monthly,Public
* Errors-To= Owner
* Subscription= Open,Confirm
* Ack= Yes                    Confidential= No              Notify= No
* Files= No                  Mail-Via= Distribute          Validate= No
* Reply-to= List,Respect    Review= Public                    Send= Public
* Stats= Normal,Private     X-Tags= Yes
* Default-Options= NoFiles,NoRepro
*
* This list installed on 95/02/02, running under L-Soft's LISTSERV-TCP/IP
* version 1.8b for Windows NT.
*
* Comment lines...
*
```

Figure 2.3. A sample list header file for a list called MYLIST.

Below, we've now edited 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 the password assigned by the LISTSERV maintainer, and note also the PW= keyword in the body of the list header which will define a new password.

```
PUT MYLIST.LIST PW=MYPASSWD
* The Descriptive Title of My List
*
* Owner= NATHAN@LSOFT.COM (Nathan Brindle)
* Owner= Quiet:
* Owner= nathan@linus.dc.lsoft.com
* Owner= ncbnet@linus.dc.lsoft.com
* Notebook= Yes,A,Monthly,Public
* AutoDelete= Yes,Full-Auto
* Errors-To= ncbnet@linus.dc.lsoft.com
* Subscription= Open,Confirm
* Ack= Yes                    Confidential= No                 Notify= No
* Files= No                  Mail-Via= Distribute            Validate= No
* Reply-to= List,Respect    Review= Public                    Send= Public
* Stats= Normal,Private     X-Tags= Yes
* Default-Options= NoFiles,NoRepro
* PW=NEWPASSWD
*
* This list installed on 95/02/02, running under L-Soft's LISTSERV-TCP/IP
* version 1.8b for Windows NT.
*
* Comment lines...
*
```

Figure 2.4. The edited list header file ready to be sent back to the server.

2.11. 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.

2.11.1. 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 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.

The next level is **`VALIDATE= YES`**. At this level, LISTSERV requires a password for all of its "protected" commands. This password can be either the list password or the sender's personal 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 VMS or Unix systems never require validation, as they cannot be forged. See Appendix B for more information on the **`VALIDATE=`** keyword.

2.11.2. 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.

2.11.3. Controlling the service area of your list

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**".

2.11.4 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 allow anyone, including non-subscribers, to review the list, set **REVIEW= PUBLIC** (which is also the default).

To restrict reviews of the list to subscribers only, set **REVIEW= PRIVATE**.

To restrict reviews of the list to list owners only, set **REVIEW= OWNERS**.

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).

2.11.5 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: It is not advised to change the location (second) parameter of the Notebook= keyword without prior approval from the LISTSERV maintainer. Setting this parameter to an illegal value will generate errors that will cause LISTSERV to place your list on hold until the error is corrected.

If enabled, notebook archives are private by default.

2.11.6 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`**. For a list where all posts should be forwarded to a moderator/editor, there are two 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`**:

```
From:
 "L-Soft list server at PEACH.EASE.LSOFT.COM (1.8b)"
 <LISTSERV@PEACH.EASE.LSOFT.COM>
Subject:        ACCESS-L: approval required (701AC4)
To:              Nathan Brindle <NATHAN@LSOFT.COM>

This  message  was  originally  submitted  by  joe@unix1.foo.bar.com  to  the
ACCESS-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 (26 lines) --------------------------
```

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

A final 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[,Hold]`
`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.

2.11.7. 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.

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

- **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 need 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.

Also note that the confirmations must come from the user id that originated the command. You cannot send a command from one account and then approve it from another.

2.11.8 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:

- List Passwords. The list password is coded into the list header (see section 2.7) and is accepted for any list maintenance operations where a password is required. However, note that only a list owner or a LISTSERV maintainer can use this password to make changes.

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

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

```
PW ADD newpassword
```

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

```
PW RESET
```

This command will also require confirmation.

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

```
PW CHANGE newpassword [PW=oldpassword]
```

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).

2.11.9 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 (no wildcards) 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.

3. Advertising Your Public Mailing Lists

3.1. List of Lists

LISTSERV automatically produces a List of Lists that may be reviewed by users anywhere on the Internet via the **`LIST GLOBAL`** 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. The INFO <listname> command and how to implement it

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.

3.3. The NEW-LIST project at North Dakota State

The NEW-LIST project was started in 1989 to promote mailing lists via a mailing list. NEW-LIST@VM1.NODAK.EDU distributes announcements of new and changed mailing lists to over 9500 subscribers every day. 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 can:

- Gopher to vm1.nodak.edu and choose **`"Local LISTSERV Resources"`**, then **`"NEW-LIST Project"`**; or
- Send the command **`GET NEW-LIST PACKAGE`** to **`LISTSERV@VM1.NODAK.EDU`**.

(The NEW-LIST Project also 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.)

3.4. 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 Gopher or WWW client at the INTERNIC gopher, **`rs.internic.net`**.

3.5. 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.

3.6. 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:  Al Gore <vice-president@whitehouse.gov>
```

becomes an **ADD** command as follows:

```
ADD GOV-L Al Gore <vice-president@whitehose.gov>
```

4.1.1 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 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 delete one of these addresses by simply cutting and pasting it into a **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 delete this address, there are two issues.

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

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.

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@LSOFT.COM>
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 mailing lists to LISTSERV from other systems

If you are moving a list from a non-LISTSERV site, you can quickly and easily convert the existing list file 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:

   ```
   QUIET ADD listname DD=X IMPORT
   //X DD *
   user1
   user2
   /*
   ```

   where "***listname***" is the name of the new list, and "***user1***", "***user2***" and other users are the entries from the original list that you want to add to the new list. (You should remove any lines from the original list that do not actually identify subscriber addresses.)

3. Send the job to LISTSERV.

The **IMPORT** option speeds up the operation of adding many subscribers "in bulk" at one time by causing LISTSERV to omit success messages and to relax syntax checking.

4.4. 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
```

4.5. Dealing with bounced mail

4.5.1. 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.

4.5.2. What to do about several types of bounces

Here are a few of the typical mail errors you will have to deal with as a list owner:

1. no such user at *host*
   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.
2. no such host
   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*
   Similar to "no such host". Comes from a different lookup system, and generally means the same thing.
4. Transient failure: cannot deliver for *n* days
   A host is experiencing periodic failures, and the gateway or intermediate host will store the message for *n* days and 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
   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.

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.

4.5.3. 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.

4.6. Automatic and semi-automatic deletion

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 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.

By default, lists with "Validate= All" are set "Auto-Delete= No", while all other lists are set "Auto-Delete= Yes,Semi-Auto,Delay(4),Max(100)".

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

4.6.1. 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.

4.7. 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 to 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.

4.8. 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.

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

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

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 Appendix B for more information on renewal and delay periods.)

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() 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
```

4.9. The SERVE command

If a user sends more than 21 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.

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.

4.10. "Peering" large 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 2 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.

You should ALWAYS contact the LISTSERV maintainer before deciding to link your list to another LISTSERV. Although there is no problem about linking to another L-Soft LISTSERV list, linking to a non-L-Soft mailing list manager is not supported 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.)

4.10.1 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. (For instance, if you are using Notepad in Microsoft Windows, ensure that "Word Wrap" is turned off.) 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.

4.10.1 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.

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.

5.2. How to set personal subscription options for subscribers

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

```
[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, the **MAIL/NOMAIL** option has been 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 your server is not yet running version 1.8b.

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. INDex/NOINDex [VM only]

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.

5.3.4. 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.

5.3.5. 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

Note: In version 1.8b, the obsolete FULLHDR and SHORTHDR options were renamed to FULL822 and SHORT822, while the normal BSMTP-style header options were renamed to FULLHDR and SHORTHDR. The FULL822/SHORT822 options are only required by a small number of ancient BITNET mail systems.

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) 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.

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

5.3.6. 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.

5.3.7. 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`**.

5.3.8. TOPICS

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 6, for more information on this syntax.

5.3.9. POST/NOPOST

This option may be set only by list owners or the LISTSERV maintainer. 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.

5.3.10. EDITOR/NOEDITOR

This option may be set only by list owners or the LISTSERV maintainer, and is effective only on moderated lists. 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`**.

5.3.11. REVIEW/NOREVIEW

This option may be set only by list owners or the LISTSERV maintainer. 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 **`SET`** ***`listname`*** **`NOREVIEW FOR`** **`userid@host`** command to reverse a previously-set **`REVIEW`**.

5.3.12. 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 **`SET`** ***`listname`*** **`RENEW FOR`** **`userid@host`** 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.

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).

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.

Finally, if your mailing list has an international audience, you will need to 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. In most cases, the official language will 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

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= Owner
```

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= Owner,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.

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.

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.*

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. This defaults to the primary list owner if no other editor is defined. 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 6A943C
```

If the message has been in the spool longer than the time-out period (default 48 hours; can be increased with the “Confirm-Delay=“ keyword), you will receive notification that the confirmation number does not match any queued job.

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 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.

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

```
* Topics= topic1,topic2,...topic11
```

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.

Finally, it is important to note that topics are active only when the subscriber's 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.

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.

See Appendix B 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. 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.

3. 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@lsoft.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 is used on ACCESS-L@PEACH.EASE.LSOFT.COM, and is intended as a tool to get feedback from users. ACCESS-L's list owner typically receives 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@ubvm.cc.buffalo.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.[3] 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

[3] 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, given the infrequent (but increasing) incidence of spamming.

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.

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.

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.

7.3. *Database Functions Overview [VM only]*

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.

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

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    *
Select 'Digest=' in LSTOWN-L
Index
/*
```

Figure 7.3. Sample CJLI job.

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:

```
> Select '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    *
Select '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. Narrowing the search

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:

```
Select 'Digest=' in LSTOWN-L since 94/01/01
Select 'Digest=' in LSTOWN-L where sender contains 'Thomas'
Select * in LSTOWN-L where sender is ERIC@SEARN
Select * 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 or from any other LISTSERV host. You can send either a "GET LISTDB MEMO" command or an "INFO DATABASE" command to retrieve the file.

8. Overview of File Archives

There are three file server systems currently in use or under development for LISTSERV:

- The VM (mainframe) version of LISTSERV supports 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 with the "new" file server system, currently under development.

- The workstation and PC versions of LISTSERV support a "temporary" file server system, to provide an interim solution while the new system is being developed. This temporary system only supports a subset of the functions of the traditional system.

- The "new", portable file server system will be a superset of the traditional system, in terms of functionality. 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.

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 two systems are going to be replaced by the third system (projected for the end of 1995), rather than providing an exhaustive chapter detailing all filelist aspects from the list owner side, we have provided only a basic overview of the two systems currently in the field, 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.[4]

On Workstation and PC Systems

With the temporary system, the LISTSERV maintainer stores these definitions in a file called SITE.CATALOG, which should be placed in the same directory with the SYSTEM.CATALOG file.. While files called "*xxxx* FILELIST" *can* be created and users *can* retrieve them, they do not in turn define further files. The new file server system will eliminate this confusion, but in the meantime you should be aware of the differences between VM and workstation file server functions as many list owners use a VM server with different conventions, and may give you incorrect advice. On workstation and PC systems, the LISTSERV maintainer must register files individually in the

[4]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.

site catalog. *Since the current system for workstations and PCs will be replaced, L-Soft does not recommend that separate FILELISTs be created on these systems unless there is a pressing reason to do so.*

8.3. Filelist maintenance (VM systems only)

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 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.

(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. Giving other users access to files on workstation systems

To register a new file to the server on workstation systems, the LISTSERV maintainer adds a line to the SITE.CATALOG file. Here is what a typical SITE.CATALOG entry looks like under Windows NT:

```
MY.FILE       MY.FILE.C:\FILES\XYZ   XXX YYY
```

And the same entry under Unix would look like this:

```
MY.FILE       my.file./files/xyz      XXX YYY
```

(Note that under Unix, LISTSERV does *not* observe case-sensitivity. Therefore you cannot define two different files with the same non-case-sensitive filename. In other words, LISTSERV will not differentiate between MY.FILE and my.file, or even My.File.)

Finally, here is a VMS example:

```
MY.FILE       MY.FILE.FILES:[XYZ]    XXX YYY
```

The first item, **MY.FILE**, is the name by which the file is known to LISTSERV. That is, the users will use **GET MY.FILE** to order a copy of that file. The name should only contain one period. Only the first 8 characters of the name and the first 8 characters of the extension are shown by the **INDEX** command. This restriction will be removed with the new file server system.

The second item, **`MY.FILE.C:\FILES\XYZ`**, is the name LISTSERV will use for the actual disk file: filename, period, extension, period, directory. The strange format is because LISTSERV uses an operating system abstraction layer for file accesses, where all system-dependent attributes are relegated to the last item. Note that the directory must be created before you register the file. For security reasons, LISTSERV will not create the directory (or set the protections) for you. Note that LISTSERV will normally need full access to these files.

The third and fourth items are "File Access Codes" (FACs). The first is for read accesses, and the second for writing. 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 MY.FILE.C:\FILES\XYZ 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.

IMPORTANT: These "file access codes" apply to LISTSERV commands (**`GET, PUT, INDEX`**) only, and not to the workstation or PC's file security system. It is your responsibility to protect the actual disk file by setting the file protections for the directory in which they are created.

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 the site catalog. Then mail the file to LISTSERV with a single line at the top of the document:

1. Edit your file and save it. Add a single line at the top of the file as follows:

   PUT *filename.extension* PW=XXXXXXXX

   (This line will not appear to people who GET the file from LISTSERV.) Replace **`XXXXXXXX`** with your personal password.

2. Be sure that the file has been registered with an entry in a filelist or the site catalog.

3. 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.

4. Send the mail message to LISTSERV.

8.6. Deleting files from the host machine

To delete a registered file on any LISTSERV host:

1. Create a new mail message addressed to LISTSERV. Add a single line at the top of the message as follows:

   ```
   PUT filename.extension PW=XXXXXXXX (DELETE
   ```

   Replace **XXXXXXXX** with your personal password.

2. 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.

3. Send the mail message to LISTSERV.

4. LISTSERV will respond one of two ways:

- On VM: LISTSERV will tell you that the file has been successfully deleted.
- On Other Versions: LISTSERV will tell you that the file has been successfully stored. This is because under the temporary file system you are actually storing a zero-byte file in place of the file on the server rather than actually erasing the file.

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: Notify the LISTSERV postmaster that you have deleted the file so that it can be deleted from **SITE.CATALOG**.

8.7. Automatic File Distribution (AFD) and File Update Information (FUI)

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 has 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.

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 filelist for a file called ***filename* $PACKAGE** , where filename is the name you have chosen for the group of files. Be sure that the filetype is $PACKAGE, with a $ sign, and store your filelist.

Now create a file called ***filename* $PACKAGE** that looks like this:

```
* MYLIST $PACKAGE
* Packing list for MYLIST PACKAGE
*
* You can make other comments here, such as
* the contact email address.
*
* filename filetype filelist
*======================
MYLIST    $PACKAGE MYLIST
INTEREST FILE       MYLIST
NETIQUET FILE       MYLIST
ANOTHER   FILE       MYLIST
```

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.

8.9. Where to find more information on File Archives

A number of guides that refer to File Archive setup and maintenance are referenced in Appendix D, *Related Documentation and Support.*

9. Customizing LISTSERV's Default Mail Templates

9.1. *What LISTSERV uses mail templates for*

Mail 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.

9.2. *The DEFAULT.MAILTPL file and how to get a copy*

LISTSERV stores the default mail template information in a file called DEFAULT.MAILTPL, which can be requested by list owners from LISTSERV with the GET command, just like any other file.

9.3. *Mail template format and embedded formatting commands*

Each template starts with a form name and subject line, such as:

```
>>> EXAMPLE1 This is the subject line
```

The template 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 between '>>>' and the name of the form, or LISTSERV will not recognize the form. Also note that the names of the templates must be typed in **UPPER CASE**.

The template 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. 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 (29 Jul 1993)
`&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).**
&RELEASE	LISTSERV's release number (e.g., "1.8b").
&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 1.1.88", "SunOS 5.4", etc.
&HARDWARE	The type of machine LISTSERV is running on, e.g., "Pentium (32M)".

The following substitutions are also available for templates related to mailing lists:

&LISTNAME	The name of the list per the "List-Address=" keyword or its default value.
&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.

In addition, many templates have their own specific substitutions, meaningful only in their specific context. For instance, a message informing a user that he was removed from a mailing list may have an **&INVOKER** substitution for the address of the person who issued the DELETE command. This is not meaningful for a template informing 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.**"

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**.
.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 to send new subscribers a questionnaire, application form or similar material, you will need to add a '.TO &WHOM' instruction to your modified template, as by default the user will not receive a copy.
.QQ	Cancels the message. LISTSERV stops reading the template 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.

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 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".
.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`**.

`.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.

`.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.

9.4. Creating and editing a <listname>.MAILTPL file for a list

Make a copy of DEFAULT.MAILTPL on your local machine and name it *listname.*MAILTPL.[5] 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 section 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 section but leave the rest of the template untouched, you can delete the rest of the template and store the INFO section 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 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.

9.4.1. The INFO section

The first section of DEFAULT.MAILTPL is called the INFO section, 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 section of DEFAULT.MAILTPL.

[5] 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.

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 mail template 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 mail template (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 section 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 section for a list called MONKEYS:[6]

```
 >>> 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 section for a mail template.

9.4.2. Other useful templates

Version 1.8b introduced many new configurable message templates, and, in particular, two new types of message templates for "linear" and optional messages. 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. The template processor now supports "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. Here is a list of these template messages:

- **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 (see below), 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)**."

[6] Thanks to Marty Hoag of NEW-LIST.

- **`TOP_BANNER`**, **`BOTTOM_BANNER`** (optional): when these templates 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. 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.

- **`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
```

- **`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 using the same procedure as for **`REQACK1`**.

- **`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.

- **`ADDREQ1`** (changed): this message, which was already present in version 1.8a, is sent to the list owner when a user requests to join a list with "Subscription= By owner". In version 1.8a, a copy of the message was sent to the subscriber, to confirm that the request had indeed been forwarded to the list owner. Unfortunately this was confusing to the many novice users who do not understand the difference between primary and secondary message recipients ('To:' vs 'cc:'). In version 1.8b, only the list owner is sent a copy of the **`ADDREQ1`** template. If you were using this template to send new subscribers a questionnaire, application form or similar material, you will need to add a '**`.TO &WHOM`**' instruction to your modified template, as by default the user will no longer receive a copy.

In a linear message, most special instructions are ignored. This is because the contents of the template 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.

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 for this purpose, because it is a linear template and does not give you any control over the 'Reply-To:' field. The **`SUB_OWNER`** template 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. 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 templates, 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 uses 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 templates. 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.

9.6. 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. You use the same formatting commands and replacable parameters in the **DIGEST-H** and **INDEX-H** files as you do in the **MAILTPL** file, and the instructions for storing them on the server are identical.

(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.)

10. Gatewaying to Newsgroups

10.1. Why would I want to?

There are a number of reasons why it might be reasonable to gateway a list. Some users may not be able to reach your LISTSERV host (or vice-versa) via e-mail, but have a good USENET connection. Others may have limited mailbox space and prefer to use a news reader. Still others may have no experience with mailing lists at all before they encounter USENET. In any case, if you are looking for a wider audience for a list, gatewaying it to a newsgroup may be a logical step.

10.2. How to go about it

If you are contemplating gatewaying a list, get the document NETGATE POLICY from LISTSERV@AMERICAN.EDU. This document was written by Jim McIntosh of American University (jim@american.edu), and outlines the procedures you will need to follow in order to gateway a LISTSERV list to USENET.

NETGATE POLICY is also available via anonymous ftp to american.edu (cd netnews). You can also get a package of related files by sending a GET NETGATE PACKAGE command to LISTSERV@AMERICAN.EDU.

10.3 Special considerations and problems with gatewaying

Well-behaved newsgroup gateways will identify themselves as the source of postings. This makes it possible for NetNews postings to come to the list if you have coded your list Send= Private (or "Send= Editor" and "Editor= *userid@host*,(*listname*)" – in any case, any configuration that prevents non-subscribers from posting to the list), since the USENET gateway is subscribed to your list. Misconfigured gateways may not include this information, causing gatewayed mail to bounce.

If your list is coded for automatic subscription renewal with the "Renewal=" keyword, the subscription for a news gateway should always be exempted from the subscription renewal process (**`SET xxxxx@yyyyy NORENEW`**). This will keep LISTSERV from sending the renewal message through the gateway and confusing users who are not subscribed to your list.

Spamming (see Chapter 6.9) was originally created on USENET, and is much more prevalent there than on mailing lists because it is easier to do. If you gateway to NetNews, be forewarned that you will be opening your list up to spamming via that source.

If the topic of your list is particularly controversial, you may want to think twice before gatewaying. Flame wars are much more common on USENET than on mailing lists (although this position could be argued from both sides on certain mailing lists). If you are considering gatewaying to an existing newsgroup, take some time to read the postings there before making a final decision.

Above all, poll your subscribers about the change before making a final decision. Some may have no objections – others may have violent objections. Gatewaying a list can be a touchy subject, particularly if some of your subscribers are ex-USENET users.

11. Solving Problems

11.1. 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.)

11.2. Loop-checking can cause occasional problems with quoted replies

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 11.1. 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 11.2. 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 11.3. 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.

11.3. User can't unsubscribe and/or change personal options

See Chapter 4, section 4.1 where this is discussed in detail.

11.4. 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.

11.5. 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.

There are a couple of ways to get around this problem.

6. 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.

7. 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.

8. 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.

11.6. 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.

Appendix A: System Reference Library for LISTSERV™ version 1.8b

This document is available separately. It can be retrieved in plain text from any server running L-Soft's LISTSERV™ with the command INFO REFCARD.

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.

List subscription commands (from most to least important)

SUBscribe	*listname full_name*	Subscribe to a list, or change your name if already subscribed
SIGNOFF		Remove yourself:
	listname	- From the specified list
	*	- From all lists on that server
	* (NETWIDE	- From all lists in the network
SET	*listname options*	Alter your subscription options:
	ACK/NOACK/MSGack	-> Acknowledgments for postings
	CONCEAL/NOCONCEAL	-> Hide yourself from REVIEW
	Files/NOFiles	-> Toggle receipt of non-mail files from the list
	Mail/NOMail	-> Toggle receipt of mail on/off
	DIGests/NODIGest	-> Toggle digest mode on/off
	INDex/NOINDex	-> Toggle message index mode on/off
	REPro/NOREPro	-> Copy of your own postings?
	TOPICS: ALL	-> Select topics you are subscribed to
	<+/-> topicname	(add/remove one or replace entire list)

Note: The Mail/NOMail toggle now determines whether or not LISTSERV sends mail to the subscriber, be it in the form of individual postings, digests, or indexes. The DIGest and INDex toggles determine whether or not the mail comes in the form of digests or indexes. When the users sends SET *listname* NODIGest or SET *listname* NOINDex, LISTSERV reverts the user to individual postings (formerly the behavior associated with SET *listname* Mail). If the user sends the NOMail command while set to DIGest or INDex, subsequently sending the Mail command will revert the user to DIGest or INDex, depending on which was originally set.

Options for mail headers of incoming postings (choose one):

FULLHdr or FULL822	-> "Full" mail headers (default)
IETFhdr	-> Internet-style headers
SHORTHdr or SHORT822	-> Short headers
DUALhdr	-> Dual headers, useful with PC or Mac mail programs

CONFIRM	*listname1 <listname2* <...>>	Confirm your subscription (when LISTSERV requests it)

Other list-related commands

INDex	*listname*	Sends a directory of available archive files for the list, if postings are archived
Lists	*option*	Send a list of lists as follow:
	(no option)	-> Local lists only, one line per list
	Detailed	-> Local lists, full information returned in a file
	Global	-> All known lists, one line per list, sent as a (large!) file
	Global /xyz	-> Only those whose name or title contains 'xyz'
	SUMmary [*host*]	-> Membership summary for all lists on specified host (if *host* is not specified, the local host is assumed)
	SUMmary ALL	-> For all hosts (long output, send request via mail!)
	SUMmary TOTAL	-> Just the total for all hosts
Query	*listname*	Query your subscription options for a particular list (use the SET command to change them)
	*	-> Query all lists you are subscribed to on that server
REGister	*full_name*	Tell your name to LISTSERV, so that you don't have to specify it on subsequent SUBSCRIBE's
	OFF	Make LISTSERV forget your name
REView	*listname* [(*options*]	Get information about a list
	BY *sort_field*	-> Sort list in a certain order:
	Country	by country of origin
	Name	by name (last, then first)
	NODEid	by hostname/nodeid
	Userid	by userid
	BY (*field1 field2*)	-> You can specify more than one sort field if enclosed in parentheses: BY (NODE NAME)
	Countries	-> Synonym of BY COUNTRY
	LOCal	-> Don't forward request to peers
	Msg	-> Send reply via interactive messages (BITNET users only)
	NOHeader	-> Don't send list header
	Short	-> Don't list subscribers
SCAN	listname text	Scan a list's membership for a name or address
STats	listname [(*options*]	Get statistics about a list
	LOCal	-> Don't forward to peers

Informational commands

Help		Obtain a list of commands
Info	[*topic*]	Order a LISTSERV manual, or get a list of available ones (if no topic was specified)
	[*listname*]	Get information about a list hosted on the server
Query	File *fn ft* [*filelist*] [(*options*]	Get date/time of last update of a file, and GET/PUT file access code

	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
	BITEARN (VM only)	-> Statistics about the BITEARN NODES file
	DISTribute	-> Statistics about DISTRIBUTE
	DPATHs *host1* [*host2* [...]]	-> DISTRIBUTE path from that server to specified host(s)
	DPATHs *	-> Full DISTRIBUTE path tree
	FIXes (VM only)	-> List of fixes installed on the server (non-VM see LICENSE)
	HARDWare or HW	-> Hardware information
	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
	NETwork (VM only)	-> Statistics about the network
	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'
	PATHs *snode node1* [*node2* [...]]	-> BITNET path between 'snode'and the specified node(s)
	STATs	-> Usage statistics (default option)
	VERSion	-> Same as RELEASE command
	(no function)	-> Same as SHOW STATS

Commands related to file server functions

AFD		Automatic File Distribution
	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'
GET	*fn ft* [*filelist*] [(*options*]	Order the specified file or package
	PROLOGtext xxxx	-> Specify a 'prolog text' to be inserted on top of the file
GIVE	*fn ft* [*filelist*] [TO] *user*	Sends a file to someone else

INDex	[*filelist*]	Same as GET xxxx FILELIST (default is LISTSERV FILELIST)
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
	CHange *newpw* [PW=*oldpw*]	-> Change password
	RESET	-> Reset (delete) password
SENDme		Same as GET

<u>Other (advanced) commands</u>

DATAbase	*function*	Access LISTSERV database:
	Search DD=*ddname* [ECHO=NO]	-> Perform database search (see INFO DATABASE 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
DISTribute	[*type*] [*source*] [*dest*] [*options*]	Distribute a file or a mail message to a list of users (see INFO DIST for more details on the syntax)
	Type:	
	MAIL	-> Data is a mail message, and recipients are defined by '<*dest*>'
	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 DDname holding the data to distribute (default: 'DD=DATA')
	Dest:	
	[TO] *user1* [*user2* [...]]	-> List of recipients
	[TO] DD=*ddname*	-> 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
	Options requiring privileges:	
	FROM=*user*	-> File originator
	FROM=DD=*ddname*	-> One line: 'address name'
SERVE	*user*	Restore service to a disabled user

THANKs		Check to see if the server is alive
UDD		Access the User Directory Database (there are 18 functions and many sub-functions, so the syntax is not given here)

File management commands (for file owners only)

AFD/FUI		Automatic File Distribution
	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)
PUT	*fn ft* <*filelist* [NODIST]]	Update a file you own
	[CKDATE=NO]	-> Accept request even if current version of the file is more recent than the version you sent
	[DATE=*yymmddhhmmss*]	-> Set file date/time
	[PW=*password*]	-> Supply your password for command authentication
	[RECFM=F [LRECL=*nnn*]]	-> Select fixed-format file (not to be used for text files)
	[REPLY-TO=*user*]	-> Send reply to another user
	[REPLY-TO=NONE]	-> Don't send any reply
	[REPLY-VIA=MSG]	-> Request reply via interactive messages, not mail (BITNET only)
	["parameters"]	-> Special parameters passed to FAVE routine, if any
	Standard parameters supported for all files:	
	TITLE=*file title*	-> Change file "title" in filelist entry
REFRESH	*filelist* [(options]	Refresh a filelist you own
	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

List management functions
Commands that support the QUIET keyword are marked (*)

ADD(*)	*listname user* [*full_name*]	Add a user to one of your lists, or update his name
	listname DD=*ddname*	-> Add multiple users, one address/ name pair per line
ADDHere(*)		Same as ADD, but never forwards the request to a possibly closer peer
DELete(*)	*listname user* [(*options*]	Remove a user from one of your lists, or from all local lists if listname is '*'
	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)
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 'nodeid'
	PREFer *node*	-> Preferred peer in case of tie (equidistant peers)
	SERVice	-> Check service areas are respected
	With(*node1* [*node2* [...]])	-> Perform analysis as though specified nodes ran a peer
	WITHOut(*node1* [*node2* [...]])	-> Opposite effect
FREE	*listname* [(*options*]	Release a held list
	GLobal	-> Forward request to all peers
GET	*listname* [(*options*]	Get a copy of a list in a form suitable for editing and storing list and lock it
	GLobal	-> Forward request to all peers
	HEADer	-> Send just the header; on the way back, only the header will be updated
	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
	GLobal	-> Forward request to all peers
MOVE(*)	*listname user* [TO] *node*	Move a subscriber 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

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
SET(*)	*listname options* [FOR *user*]	Alter the subscription options of another
	*	user or set of users (when using wildcards)
	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 allowed to post
STats	*listname* (RESET	Resets statistics for the list
UNLOCK	*listname*	Unlock a list after a GET, if you decide not to update it after all

<u>Syntax of parameters</u>

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 Alphabetical Reference for LISTSERV™ version 1.8b

This document is available separately as reference number 9412-UD-01. It can be retrieved in plain text from any server running L-Soft's LISTSERV™ with the command INFO KEYwords.

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, version 1.8b* (L-Soft document reference number 9502-UD-02).

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:

Generic parameters

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, e.g. "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, e.g. EARN, BITNET
- The name of a country, e.g. Germany, Canada
- 'Local', in which case it is equated to the value of the "Local=" keyword (q.q.v.).
- A node name, e.g. 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, e.g. 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".

Keywords fit into several different classifications. These classifications, and the associated keywords, are as follows:

<u>Access Control Keywords (page 76)</u>

Files=	Determines whether non-mail 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 79)

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
Topics=	Defines up to 11 sub-topics for a list

Error Handling Keywords (page 84)

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 86)

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
X-Tags=	Controls whether "X-to:" and "X-cc:" tags are included in list mail headers

Security Keywords (page 91)

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 94)

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 96)

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
Long-Lines=	Controls whether long-lines support is enabled
Translate=	Controls how LISTSERV handles control characters in list mail

Default Values for All Keywords (page 97)

Access Control Keywords

Files=Yes | No

(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.

Filter= Only | Also | Safe,*net-address1,net-address2*,....

"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 word 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 "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. LISTSERV has two built-in filters: a "minimal" one, which is used for safe lists, and 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.

If "Filter= Only" is used, 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.

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.

Note that 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).

Review= *access-level*

This keyword defines the categories of users who are allowed to review the Internet addresses and names of the persons subscribed to a list. The default value is "Public".

Send= *access-level* [,Confirm | Semi-Moderated | Hold]

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, e.g., `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@LSOFT.COM ERIC@LSOFT.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@LSOFT.COM ERIC@LSOFT.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".

Stats= Normal | None,*access-level* [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, i.e. 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.

Yes	A short acknowledgment with statistical information on the mailing will be sent back to you. This is the default.
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.
None	No messages of any kind are sent when your mail file is processed. [same as No for non-BITNET]

Daily-Threshold= *number*

This keyword limits the number of postings that may be processed by the list in a 24 hour period. The default is Daily-Threshold= 50. When the keyword's value 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.

Digest= No
Yes,*where* | Same[,*frequency*][,*when*][,*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.[7]

Digests are larger messages containing all the postings made by list subscribers 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.

[7]The digests conform to RFC1153 with an acceptable deviation from the recommended subject line (verified with the RFC author).

Users sign up for digestified rather than immediate delivery with 'SET listname DIGests', while indexes are selected with 'SET listname INDex'. These two new 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.

- When switching from any option to DIGESTS 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 DIGESTS/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 DIGESTS or INDEX to MAIL, 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 MAIL and then back to DIGESTS. 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 new "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 third parameter is optional and specifies when the digest is to be actually distributed. For daily digests, specify 'hh:ss' or just 'hh' in the usual 00-23 scale (24 is also accepted for midnight). 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. Finally, the last and optional parameter takes the form "Size(number)" and specifies the maximum number of lines the digest is allowed to reach before a "special issue" is cut. 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.

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.

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.

Internet-Via= ***net-address***

There is no default value. This parameter determines whether or not mail bound for Internet addresses is routed through a specific Internet gateway.

Mail-Via= Direct | DISTRIBUTE | DIST2

The default value is **`Mail-Via= DISTRIBUTE`**. DIST2 is functionally equivalent to DISTRIBUTE, and is included for historical reasons. **`Mail-Via= Direct`** causes LISTSERV to ignore the DISTRIBUTE algorithm for subscribers on the local system, but mail to non-local subscribers will still go out on the DISTRIBUTE backbone.

Newsgroups= None | ***usenet_newsgroup1,usenet_newsgroup2...***

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= ***net-address***

There is no default value. This parameter determines whether or not mail bound for NJE addresses is routed through a specific gateway.

Prime= Yes | No | ***when***

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`**. This can be most useful in controlling the load on the machine running LISTSERV. "Prime=" may also be set to an explicit time specification, e.g.,

```
Prime= "MON-FRI: 09:00-17:00; SAT-SUN: -"
```

Reply-To= ***destination*****,Respect | Ignore | Both**

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 "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`**.

Respect:	The original "Reply-to:" tag, if any, is kept.
Ignore:	The original "Reply-to:" tag is ignored and discarded.

Sender= LIST | NONE | "list title <*net-address*>",*ietf-address*

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.

Topics= *topic1*,*topic2*,...*topic11*

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 11 topics for the list. For instance, the list owner could code:

```
Topics= News,Benchmarks,Meetings,Beta-tests
```

WARNING - YOU MUST NEVER REORDER THE TOPICS= KEYWORD

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, 12th 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 | Full-Auto | Manual,Delay(*number*),Max(*number*)

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.

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:

- 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 further examines the "Auto-Delete=" keyword and passes the message to the list owner unless the list is coded "Auto-Delete= Yes,Full-Auto".

- 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 is to be used only when the list owner positively does not have the time to handle the delivery errors LISTSERV sends every day.

- 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.

  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.

- 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= No" for lists with "Validate= All" and "Auto-Delete= Yes,Semi-Auto,Delay(4),Max(100)" for other lists.

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 'Postmaster', and it is recommended that the owners change it to 'Owners' or 'Owners,Postmaster' as soon as they become familiar with LISTSERV.

Loopcheck= Full | None | Noorigin | Nobody | NoCRC | NoSpam
Determines the type of loop checking performed by LISTSERV to avoid perpetuating mail loops. The default is "Loopcheck= Full".

Safe= Yes | No
The list header keyword, "Safe= Yes/No", 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). The use of the "Safe= Yes" option 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.

List Maintenance and Moderation Keywords

Editor= *net-address1,net-address2*,...

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.

Editor-Header= Yes | No

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".

List-Address= *name_info@host_info*

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 configuration file.

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.

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 configuation 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.
- "List-Address= XYZ-L@XYZ.EDU" is not valid. Always specify LISTNAME or LIST-ID for the left-hand part.

List-ID= ***name***

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.

Moderator= <*netaddress1*>,<*netaddress2*>....

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.

New-List= ***netaddress***

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 generally 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:

9. Create the new list
10. Move the subscribers to it
11. **`DELETE`** ***`oldlistname`*** **`*@*`**

12. 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 | (Yes,*where,interval*|Separate,*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.
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).

Note: Notebooks may be retrieved by means of the GET command. A list of all available notebooks can be obtained with a GET NOTEBOOK FILELIST command.

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 "Yes".

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 2.6 of the *List Owner's Manual* for details.

Peers= ***peer1*****,*****peer2*****,...**

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.

Renewal= ***interval1,interval2...,intervalx*****,Delay(*****number*****)**

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).
- An absolute date in the format yy/mm/dd (once on this specific day), the format mm/dd (once yearly on this day), or the format dd (once monthly on this 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,95/07/04,12/25,15,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 1995, a request every year on Christmas Day, and a request every month on the 15th day of the month. 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.

Sizelim= ***number***

If set, causes LISTSERV to reject all messages to the list which exceed the number of 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.)

X-Tags= Comment | Yes | No

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, i.e. "Comment: X-To:" and "Comment: X-cc:".
No:	This information must not appear at all in the mail header.

Security Keywords

Confidential= No | Yes | Service

Indicates whether the list should be hidden from users or not. A confidential list will not appear on the "List" command output. "No" is the default value and indicates that the list is not 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. "Yes" means that the list is unconditionally confidential.

Exit= ***filename***

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.

Starting with version 1.8a, list "exits" are available 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 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,...***

Defines the nodes which are to be considered as 'local nodes' for both service area checking. The local node 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.

PW= ***list-password***

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.

Service= ***area1,area2,...***

Defines the 'service area' outside of which subscription requests must not be accepted. When a SUBSCRIBE command is received, the "Peers=" keyword is checked first to see if there is a nearer peer list in the network. If it is the case, the command is forwarded to this nearer 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, the destination 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, e.g. "Service= Germany", and still have users subscribed to the best possible server.

Validate= No | Yes,Confirm,NoPW

Under L-Soft's LISTSERV, lists are protected by a password which must be specified by the list owner when he sends an updated version of the list back to the server. When "Validate= Yes", password validation applies to ALL the commands that modify the contents of the list, e.g. SIGNOFF, SET, etc. This implies that users cannot use these commands since they do not know the list password. A notable exception is the SUBscribe command, which can still be used (if enabled) to get on the list; however, sending a second SUBscribe command for the same list (to correct a spelling error in your name) would result in the command being forwarded to the list owner and not immediately executed. This is to protect you from network hackers who might issue a command "from" your userid@node to change list settings, such as who has the ability to GET and PUT the list, review concealed subscribers, etc. The default is "No", but it is recommended that "serious" or "important" lists be changed to "Validate= Yes".

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 1.8x, this procedure is also used for authentication purposes. Since the confirmation codes are valid only for a single command, this 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). The following security settings are available:

- "Validate= No" (formerly "Validate= Store only"): all commands except PUT are taken at face value with no validation. While users are not bothered with validation requests, the list is totally unprotected from attacks by hackers. For compatibility reasons, this is the default setting.

- "Validate= Yes" (formerly "Validate= All commands"): "protected" commands, such as DELETE or ADD, require password validation. For list owner commands, both personal and list passwords 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.

- "Validate= Yes,Confirm" (new level): protected commands are validated using the "OK" mechanism by default, although passwords are also accepted where appropriate. This is a good compromise between list security and list owner convenience.

- "Validate= Yes,Confirm,NoPW" (new level): protected commands are 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.

- "Validate= All,Confirm" and "Validate= All,Confirm,NoPW" (new levels): all commands causing a change in state, except the PUT command, are validated using the "OK" mechanism, with or without a password alternative. Commands such as QUERY do not require any validation.

Requests coming from the local system via CP MSG or CP SMSG (on VM systems) or via LCMD (on VMS or Unix systems) never require validation, as they cannot be forged. The PUT command must always be validated with the list password, because there is presently no mechanism to suspend execution of a request to which a file is attached. If the list password is used only for that purpose, the exposure is minimal as PUT operations are not part of everyday list management routine. Note that PUT requests require no validation when submitted via SENDFILE from the machine on which LISTSERV is running, as the operating system then guarantees the authenticity of the transaction.

For lists operating with "Validate= Yes" (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 and DELETE requests from registered node administrators 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.

Subscription Keywords

Confirm-Delay= ***number***

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).

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. Default-Options does not affect existing subscribers.

A typical Default-Options setting might be:

```
* Default-Options=Nofiles,Norepro,Msg
```

Default-Topics= ***topic1,topic2,...***

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 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.

Subscription= By owner | Closed | Open ,Confirm

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:	The users are allowed to subscribe to the list.
By owner:	The users are not allowed to subscribe, but their requests will be forwarded to the list owner. This is the default.
Closed:	The users are not allowed to subscribe, and their requests are not to be forwarded to the list owner.

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).

Other Keywords

Language= ***idiom***

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 to the server, of course. The default language is "English", of course.

Currently only information mail is available in several languages.

Indent= ***number***

Determines the minimum number of columns allowed for list addresses in response to the REVIEW command. The default is Indent= 40.

Long-Lines= Yes | No

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.

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".

Default Values for all keywords

Ack= Yes
Auto-Delete= No if “Validate= Yes”, Yes,Delay(4),Max(100) otherwise
Confidential= No
Confirm-Delay= 48
Daily-Threshold= 50
Default-Options= <none>
Default-Topics= <none>
Digest= Yes,Daily,Same if “Notebook= Yes”, No otherwise
Editor= <none>
Editor-Header= Yes
Errors-To= Owner
Exit= <none>
Files= No
Filter= <built-in>
Indent= 40
Internet-Via= <none>
Language= English
List-Address= <none> (per LIST_ADDRESS system default)
List-ID= <none>
Local= <none>
Long-Lines= Yes
Loopcheck= Full
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= <none>
Renewal= <none>
Reply-To= List,Respect
Review= Public
Safe= Yes
Send= Public
Sender= List
Service= <none>
Sizelim= <none>
Stats= Normal,Private
Subscription= Open
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

Perhaps the best general overview of LISTSERV readily available at this time can be found in a document called "LISTSERV for the non-technical user", which is available from LISTSERV@LISTSERV.NET in several formats:

NSC93.MEMO	(plain ASCII text)
NSC93US.PS	(for US users)
NSC93A4.PS	(for European users and others using A4 paper size)

Subscribers to LISTSERV mailing lists will find answers to many of their questions in the *User's Guide to LISTSERV Version 1.8b*. (The target date for this document is late 1Q1995.)

LISTSERV Maintainers will be interested in the *LISTSERV Maintainer's Manual for LISTSERV 1.8b*. (The target date for this document is late 1Q1995.)

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.

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 LSVOWNER package available from UBVM

The LSVOWNER package contains some handy LISTSERV List Owner files created by Jim Gerland, gerland@ubvm.cc.buffalo.edu. To have LISTSERV send you the files, send a GET LSVOWNER PACKAGE command to LISTSERV@UBVM.CC.BUFFALO.EDU.

Here are the files currently (1 March 1995) available:

LSVOWNER READY	'Your List is Ready' boilerplate
LSVOWNER WELCOME	'Welcome to this list' boilerplate
LSVOWNER TXT	Generic List Owner Instructions
LSVOWNER NEW-LIST	New List Announcement boilerplate
LSVOWNER GATEWAY	USENET News Gateway boilerplate
LSVOWNER HEADERS	List header instructions
LSVOWNER FIL-LIST	FILELIST instructions

Jim adds: "You can subscribe to the package by sending an AFD ADD LSVOWNER PACKAGE command. You will then receive a new version of the filelist as updates are received by the server, with only the updated files being sent to you. You can also subscribe to File Update Information notices for the package by sending a FUI ADD LSVOWNER PACKAGE command."

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.iupui.edu/it/lstsrv/listserv.html`

You can also go directly to LISTSERV TIPS at

`http://cspot11spot1.it.iupui.edu/listserv_tips.html`

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.

D.2.4. NetNews Gatewaying (See also Chapter 10)

The following files are available from LISTSERV@AMERICAN.EDU (and also by anonymous ftp to american.edu (cd netnews)) relating to the NetNews gatewaying service there.

NETGATE	POLICY	Procedure for Establishing a Gateway
NETGATE	GATELIST	List of Gatewayed LISTSERV Lists
NETGATE	MODLIST	Similar to news.admin Post

You can subscribe to the package by sending an AFD ADD NETGATE PACKAGE command. You will then receive a new version of the file as updates are received by the server, with only the updated files being sent to you. You can also subscribe to File Update Information notices for the package by sending a FUI ADD NETGATE PACKAGE command.

Appendix E: Acknowledgments

Acknowledgments
Over the years, a number of people have written valuable documentation of the LISTSERV product. This documentation was constantly consulted by this writer in the course of producing L-Soft's official manuals. In no particular order, I would like to acknowledge the efforts of Lisa Covi, Ben Chi, Jim Gerland, Marty Hoag, and (of course) Eric Thomas in producing the basis for this work. Thanks also go to John Harlan for helping me get my start in owning lists. I wouldn't be writing this today without his initial willingness to assist me. And I promised myself that I wouldn't forget to mention Pete Weiss. Thanks for keeping me on my toes, Pete!

Another invaluable source of information was list owner discussion lists. Heavy use was made of the archives from LSTOWN-L@SEARN in particular. Thanks to the subscribers there who probably know LISTSERV better than anyone alive, aside from Eric.

Thanks also to Susan Lowell, Elena Nolen and Jim Jones of L-Soft for reading and making valuable comments on several drafts.

--Nathan Brindle (nathan@lsoft.com)

Revisions:

950620-001	Added “New-List=“ keyword to Appendix B
950620-002	Documented DIGEST-H and INDEX-H
950808-001	Expanded Section 6.6 to explain the “Editor,Hold” confirmation procedure
950808-002	Added Section 2.10 to document list security options and the “OK” validation procedure
950808-003	Reconciled TOC to actual contents
950808-004	Updated Appendix B, “Send=“
950808-005	Updated Appendix B, “Editor=“
950814-001	Released Revision 1
950816-001	Added Section 2.6 to explain how list owners are defined
950816-002	Renumbered Chapter 2 to reflect new section 2.6
950816-003	Updated Appendix B, “Owner=“
950829-001	Inserted Section 11.5 on list storage problems
950829-002	Fixed TOC
950829-003	Documented new BOTTOM_BANNER parameter for DIGEST keyword
950829-004	Updated Appendix A
950928-001	Clarified “New-List=” issues in Appendix B
950928-002	Retitled Section 2.7 and clarified adding and changing list passwords
950928-003	Added Section 2.11.8 regarding personal passwords
951108-001	Added Section 8.6 to clarify the deletion of files from file archives, and renumbered Chapter 8 to reflect this addition; revised 8.3 to reflect the addition of section 8.6.
951108-002	Revised Section 8.2 to clarify the difference between VM file archives and the “temporary” file archives system for Workstation/PC versions.
951108-003	Significantly reworded “Send=“ in Appendix B to describe the use of the “Semi-Moderated” parameter.
951130-001	Added a “Note:” paragraph in 2.11.5.
951130-002	Corrected “see section 2.5” in 2.11.8 to read correctly (2.7).
951130-003	Added 4.1.1 to discuss X.400 and X.500 addressing problems
951130-004	Corrected syntax of SET TOPICS FOR in 6.7
951130-005	Clarified “Size(number)” parameter of Appendix B, “Digest=”
951130-006	Corrected syntax for “Prime=“ in Appendix B
951130-007	Corrected function description of “Sizelim=“ in Appendix B
951204-001	Clarified “Renewal=“ behavior, Appendix B, “Renewal=“
951206-001	Modified 2.3 on searching for a host site.
951212-001	Added unix and VMS examples in 8.4 for clarity.
951212-002	Noted that template names must be in upper case in 9.3.