L-Soft international, Inc.

Site Manager's Operations Manual for LISTSERV®, version 14.5

23 February 2006
Initial Release

The reference number of this document is 0603-MD-01.

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

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

L-Soft invites comment on its manuals. Please feel free to send your comments via e-mail to **MANUALS@LSOFT.COM**, and mention which manual you are commenting on. (However, please do not send support questions to this address. See chapter 19 of this manual for appropriate support addresses.)

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

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

A word about formatting: This manual was written in Microsoft Word 2000, and originally formatted to be printed on 8-1/2"x11" paper on an HP LaserJet 1000 series printer. When printing the manual on a different type of printer, or converting to a different word-processing program, it is highly likely that the formatting and pagination will change and it will be necessary to update the tables of contents and figures as well as the index prior to printing. The author has taken great pains to ensure that the pagination and formatting works properly with the particular printer mentioned above, and cannot be held responsible for what is, in the end, a limitation of the software used to produce the manual.

Reference Number 0603-MD-01

Table of Contents

List of Tables and Figures

Preface: LISTSERV Command Syntax and Other Conventions

Editorial Note – New Version Numbering

With this release, L-Soft is aligning LISTSERV's version numbering with the rest of the e-mail industry. There have been 50 released versions of LISTSERV since 1986 – 14 major upgrades and 36 minor releases. Version 1.8e in the "traditional" numbering system corresponds to 14.0, and the present update to 14.5.

Because the old nomenclature is more familiar to our users, in this version of the documentation we will continue to refer to versions of LISTSERV inferior to version 14.4 by the old versioning system.

LISTSERV Command Syntax Conventions

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

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

Deviations from this include:

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

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

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

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

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

1. Who should read this book

This manual makes the following assumptions:

- You are a system administrator of a VM, VMS, unix, Windows NT/2000/XP/2003 or Windows 95/98/Me system (or in any case, a person with root- or system-level administrative privileges) whose assignment it is to be the LISTSERV maintainer;
- You have already installed the current version of L-Soft's LISTSERV on your system in accordance with the installation instructions that come with the package, and have it running;
- You have sufficient knowledge (or know where to find it) of your system mailer to fine-tune it without needing instructions from this manual.

In other words, we expect you already to be knowledgeable about the system on which you plan to install and run LISTSERV. *This manual does not contain installation instructions*; individual installation guides for the four general types of operating systems supported by L-Soft can be found at **http://www.lsoft.com/manuals** .

L-Soft international's LISTSERV software is designed to run on various platforms that have widely-differing configurations. Therefore it is not within the scope of this manual to describe in detail (for instance) how you can tune sendmail 8.7.3 under Linux for optimum performance with LISTSERV. However, general tips that could work on *all* systems will be offered within these pages.

Overall you will find that LISTSERV works much the same way on a unix workstation or a VMS minicomputer or an Intel Pentium machine running Windows 2000 as it has since 1986 on VM mainframes. Where LISTSERV procedures do differ between platforms, we will detail those differences in order to minimize confusion.

1.1. Changes and updates to the manual

When we find a mistake in the manual, or when between-release features are added, we normally report changes to the manual to the announce-only mailing list **LISTSERV-DEVELOPERS@PEACH.EASE.LSOFT.COM** , provided as a free service for our customers.

Other changes are documented in the Revision History, found in Appendix F.

1.2. New documentation is coming!

L-Soft has committed to the production of completely new documentation for the LISTSERV product. The new documentation is targeted to become available with LISTSERV 15 (no release date has been set).

2. Differences Between Architectures and Implementations

This chapter outlines differences between how LISTSERV is implemented on VM and non-VM machines, and the differences between LISTSERV and LISTSERV Lite.

2.1. Differences between architectures

In version 14, LISTSERV running under VM continues to differ in some regards from its counterparts on the other architectures. Here is a short list of these differences:

- VM: The web interface is not available.
- VM: Rotating change-logs are not available.

- Non-VM: Only a subset of the VM file server functions are available
- Non-VM: Certain rarely-used commands (e.g., STATS *listname*) are not available
- Non-VM: FUI (File Update Information) and AFD (Automatic File Distribution) are not available

Note that LISTSERV 14 running on non-VM systems actually has about 98% of the functionality of the VM version, and nearly 100% of the functionality that people actually use day-to-day.

The File Server

There are actually two different file server systems in operation across the LISTSERV network. One is the original version running on VM, which includes the ability to create "filelists" (indexes) which point in turn to more files which can be stored on the server, and the AFD and FUI functions mentioned above. This file server system, while still quite powerful and easy to use, is unfortunately written in a non-portable language, making a complete rewrite from the beginning a necessity. There has been no change in the VM file server from 1.8b through 1.8e (and subsequently 14.x).

The second file server system currently in operation runs on the VMS, unix, and Windows ports of LISTSERV. This is in essence still a subset of the old system in which the LISTSERV maintainer creates entries in a SITE.CATALOG file for each file that will be made available to users. With the release of 1.8c, it became possible for the LISTSERV maintainer to create sub-catalogs, which can be maintained by list owners or other responsible people. 1.8d added the **GIVE** command and the ability to create file "packages" to the non-VM versions. For more information, please see chapter 8 of this manual.

L-Soft is still developing LISTSERV's file server, which will eventually include a super-set of the original VM file server command set. Complete details are not available as of this writing, but pains are being taken to ensure that the most common commands will not change along the development path. This will help to keep a great deal of existing documentation that has been passed along the Internet from becoming obsolete overnight. The fully-developed file server is scheduled to include AFD (Automatic File Distribution) and FUI (File Update Information) in addition to other new functionality.

The WWW List Archive and List Management Interface

In Version 1.8c, a web-enabled List Archive Interface was introduced. In 1.8d, the interface was expanded to include list and site management features. The interface as been significantly rewritten for version 1.8e and is fully documented in chapter 11 of this manual. The web interface continues to be unavailable on VM but is available on all other

platforms on which the software runs.

Year 2000 Compliance

Year 2000 compliance is addressed in L-Soft's Year 2000 Compliance FAQ, which can be viewed at **http://www.lsoft.com/corporate/default.asp?item=y2k** .

2.2. Differences between LISTSERV and LISTSERV Lite

LISTSERV Lite is LISTSERV running with a special license activation key (LAK) which limits what you can do with the software. With the Free Edition of LISTSERV Lite (activated by a LAK which is both free and perpetual), you can run up to 10 mailing lists as long as you do not derive a profit from this activity. You can also purchase LISTSERV Lite LAKs that allow more (or unlimited) lists.

However, note carefully that LISTSERV Lite does not have all of the functionality of the full, Classic version--a list of the keywords and functions disabled in LISTSERV Lite follows this paragraph. For more information on the exact terms and conditions under which you may run LISTSERV Lite, please see L-Soft's World Wide Web site or contact L-Soft's sales department.

LISTSERV Classic Keywords disabled in LISTSERV Lite

Change-Log	**Confirm-Delay**	**DBMS**	**Default-Topics**
Editor-Header	**Exit**	**Files**	**Indent**
Internet-Via	**Language**	**List-Address**	**List-ID**
Local	**Long-Lines**	**Loopcheck**	**Mail-Merge**
Mail-Via	**Misc-Options**	**Moderator**	**New-List**
Newsgroups	**NJE-Via**	**Peers**	**Prime**
Renewal	**Sender**	**Service**	**Sizelim**
Stats	**Sub-Lists**	**Topics**	**X-Tags**

Note: the fact that the keyword is disabled only means that the default value cannot be changed. For instance, loop checking is still present, you just cannot control the details of its operation. On the other hand, if the default value is that the function in question is disabled (as is the case with "Peers="), then the function is actually gone. See Appendix B for more information on keyword defaults.

A feature comparison chart follows on the next page.

Comparison chart: LISTSERV Lite vs. LISTSERV Classic

(Version numbers in parenthesis indicate version in which the feature first became available, starting with LISTSERV 1.8d)

Feature	LISTSERV Classic	LISTSERV Lite
Moderated lists	Yes	Yes
Moderation sharing	Yes	No
DISTRIBUTE	Yes	No
Peered lists	Yes	No
Topics (up to 23 different topics per list)	Yes	No
Validate keyword (provides security)	Yes	Yes
Filter keyword (screens mail)	Yes	Yes
Spam detector	Yes	Yes
Spam filter	Yes	No
Customization of mail templates	List based	Site based
Auto-delete	Yes, full featured	Yes, not full featured
Probe (never see a bounce again!)	Yes(*)	No
List exits	Yes	No
Networked mode	Yes	(**)
Subscription options: • RENEW • EDITOR • REVIEW • NOPOST All other LISTSERV subscription options	 Yes Yes Yes Yes Yes	 No No No No Yes
File server functions	Yes, hierarchical	Yes, non-hierarchical
Database (archive search) functions	Yes	No
WWW archive interface	Yes, with search interface	Yes, but no search functions
WWW administration interface (1.8d)	Yes	Yes
DBMS/Mail Merge functions (1.8d)	Yes	No
Anti-Virus scanning feature (Windows NT/2000 and Linux only) (1.8e)	Yes (requires special LAK and special AV software package; contact your sales representative for details)	No
Message content filtering (1.8e)	Yes	No

(*) The probe feature does not work with all MTAs (mail servers), or may only work with recent enough versions.

(**) Networked and Standalone RUNMODEs are not available in the Free Edition of LISTSERV Lite, but *are* available in the commercial version of LISTSERV Lite.

2.3. Operating Systems and Architectures Supported

LISTSERV 1.8e (14) is the last version for several operating systems which have become obsolescent over the life of this product cycle. The operating systems which will no longer be supported after this version are:

Windows NT 4.0 SP6/6a (see note!)
Windows 95/98/Me
BSDi (Intel)
IRIX (MIPS)
Solaris-x86 (Intel)

(Note: Windows NT 4.0 is no longer supported as of LISTSERV 14.3)

Sites running these operating systems should start planning now for a migration to a different operating system. Please contact your sales representative for further information.

Sites running the Windows 95 shareware should note that their licenses will not activate the product under Windows XP. Please contact your sales representative for alternatives if you are planning to upgrade to Windows XP (optionally you may migrate to the LISTSERV Lite Free Edition). Sites running the Windows 95 Lite Free Edition can simply upgrade to the Windows NT/2000/XP LISTSERV Lite Free Edition. (Naturally you may also elect to continue running LISTSERV under Windows 95/98/Me, but there will be no further new versions or fixes for that platform.)

It should be noted that L-Soft dropped support for the following operating systems with the original release of LISTSERV 1.8e (14) (in other words, LISTSERV 13 or 1.8d was the last version for these platforms):

Windows NT 3.5, 3.51, 4.0 pre-SP6 (Intel)
Windows NT (Alpha AXP)
SunOS 4.x (SPARC)
Ultrix (MIPS)
OpenVMS (VAX)
VM/SP, VM/HPO

On the plus side, L-Soft now formally supports FreeBSD (Intel) and Linux (S/390) in LISTSERV 1.8e (14).

A comprehensive list of operating systems (and versions) under which LISTSERV is supported can be found at

http://www.lsoft.com/products/default.asp?item=listserv-ossupport

3. Principles of Operation

LISTSERV® is software that allows you to create, manage and control electronic "mailing lists" on a corporate network or on the Internet. Since its inception in 1986 for IBM mainframes on the BITNET academic network, LISTSERV has been continually improved and expanded to become the predominant system in use today. LISTSERV is now available for VM, OpenVMS, unix and the Windows NT "family" (including NT 4.0 SP6 and later, and Windows 2000).

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.

LISTSERV accomplishes its design goals very efficiently and very quickly. This is due primarily to its use of the proprietary DISTRIBUTE algorithm (described in RFC1429, and in the *Developer's Guide for LISTSERV*, available separately) and to the large (and growing) network of LISTSERV servers.

The LISTSERV network of servers helps to enhance LISTSERV's performance by providing a "backbone" through which large quantities of mail can be quickly distributed. The backbone also allows LISTSERV servers to "talk" to each other and exchange information. Among other things, this exchange of information between servers allows your own local server to participate in the global List of Lists service and L-Soft's CataList service on the World Wide Web (just point a web browser at **http://www.lsoft.com/catalist.html** to use the CataList service).

LISTSERV's nature as a distributed network of interconnected servers also makes it possible to identify and eliminate unsolicited advertisements sent to multiple lists (known colloquially as "spams") before they do much harm. While it is virtually impossible for a small isolated server to detect a spam (unless the sender listed the thousands of lists he was targeting in the "To:" field), for the simple reason that it will only ever receive a few copies for its own public lists, the LISTSERV network as a whole receives thousands of copies of the spam. By comparing notes with each other, the servers can quickly determine that a spam is occurring and raise a network-wide "spamming alert", stopping the message before it does much damage at all. Since the introduction of LISTSERV's anti-spam technology in version 1.8b, the growing number of sites that are participating in the anti-spamming warnings have virtually stopped the distribution of such messages in their tracks. L-Soft's developers are constantly upgrading and refining the anti-spam

algorithms, to the effect that LISTSERV version 1.8e has an even better anti-spam filter than before.

In addition to the anti-spamming filter, LISTSERV also incorporates an **anti-spoofing filter**, to keep mischevious (and often malicious) users from subscribing other users to mailing lists in order to "mailbomb" them.

LISTSERV makes it possible for you to offer the same mailing list in four different formats:

- **Individual mail messages** sent out as they are processed
- **Digest mode**, where a compendium of messages processed by the list is sent at specified intervals
- **Indexed mode**, where an index consisting of the message number, sender, and the subject line of each message is sent each day, along with instructions on how to retrieve postings from the server
- Users can read, search, and respond to postings via LISTSERV's **Web Archive Interface**.

These modes are set by sending SET commands to LISTSERV. Unlike some other mailing list management systems, LISTSERV does not require the user to unsubscribe from one version of the list and resubscribe to another just to change delivery modes.

LISTSERV includes **database search capability** for list archive notebooks. A fast **reverse indexing feature** is available for servers running lists with large archives. Users can use a simple search syntax to comb list archives for specific terms of interest. And L-Soft provides a **World Wide Web archive interface** (not currently available on VM for technical reasons unrelated to LISTSERV itself) with which the notebook archives for all public lists can be viewed and searched from a web browser. The new WWW interface differs from (and has advantages over) "hypermail" style web archiving in that new postings are shown as soon as they are received; postings can be organized in a manner that best suits the reader; there is no duplication of effort, as the LISTSERV WWW interface works from the list's notebook archives rather than creating a separate HTML file for each posting; and the list owner can customize the main page for their list by simply modifying their mail template file.

LISTSERV also includes a number of **list and server management functions** in its WWW interface, including the ability to edit list headers and associated mail and WWW templates, and to manage subscribers via the Web. These features have been substantially rewritten in LISTSERV 1.8e. See chapter 11 of this manual for details.

Since Version 1.8d LISTSERV has also contained **DBMS and mail-merge support**. These features are documented in the *Developer's Guide for LISTSERV*, available separately.

New in LISTSERV 1.8e is an **Anti-Virus Scanning feature** for messages passing through the server. This is a value-added enhancement which requires a special LAK and a special version of F-Secure Anti-Virus. At this writing the feature is only available for Windows NT/2000 and Linux servers running LISTSERV Classic or LISTSERV Classic HPO. (Other OS platforms may be supported in the future; there is no intent to make this functionality available in the Lite version of the product.)

Many other enhancements have been introduced in 1.8e. Please see the release notes for complete details at **http://www.lsoft.com/manuals/index.html** .

4. A LISTSERV How-To for Site Managers

This how-to section is not intended to replace the LISTSERV FAQs available from L-Soft's documentation web site (**http://www.lsoft.com/manuals**). It is an attempt to bring together certain basic operations and how to accomplish them in one place. However, note that some of these how-to answers will redirect you to existing external documentation or to other sections of this manual in order to avoid duplication of effort.

4.1. Installation/Startup Questions

How do I install LISTSERV?

Installation guides are available on the web and are also shipped in the version-specific installation kits. You can read the guides on the web at

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

Why do I need a DNS A record and a static IP number for my LISTSERV machine?

The best analogy is to consider why you need to put a return address on a piece of postal mail that you expect someone to respond to (or to be returned to you if the person you are trying to reach no longer lives at the address you have for him). In order for people to be able to send mail to your server, it must have a "street address" so that the "postman" can deliver mail to it, and that "street address" must be known to the "post office" so that mail can be properly routed. The DNS A record tells the world where your LISTSERV machine is located by both its name (eg LISTSERV.EXAMPLE.COM) and its IP address, so that other mail machines on the Internet can correctly route mail to it. If the IP address is not static, in other words if it changes every time you dial up, or whenever you disconnect and reconnect your DSL service, it is not possible to add an A record for it to DNS. This is why both a static IP address and a DNS A record are required in order for LISTSERV to work properly.

Can LISTSERV read mail from POP mailboxes?

No. LISTSERV is designed to work with SMTP mail servers and is not able to read POP mailboxes.

How do I install the web archive/administration interface?

Please see either your version-specific installation guide or chapter 5.4, below.

How do I start LISTSERV?

This is version specific and documented in the version-specific installation guides.

How do I stop LISTSERV?

The supported method is to send e-mail from your POSTMASTER= address to LISTSERV@your server with the command

STOP PW=*createpw*

in the body of the message, where "createpw" is the CREATEPW= value from your site configuration file.

Under Windows NT and later, assuming that LISTSERV is running as a system service (which is the recommended method), you can also stop LISTSERV from the Control Panel/Services applet, or by issuing a NET STOP LISTSERV command from a DOS prompt (both of these assume that you are logged into the machine with administrative privileges).

Under unix, it is possible to stop LISTSERV by issuing a 'kill -TERM' command on the PID found in $LSVSPOOL/listserv.PID . However, this is not 100% guaranteed to kill all of the existing 'lsv' processes which may be running at the time (for instance you may end up with zombie processes left over from web interface queries), so L-Soft recommends that the e-mail method using the STOP command as documented above be used in preference to the 'kill -TERM' method from a shell prompt. It is vital that all 'lsv' processes be stopped before restarting LISTSERV, as the web interface may not properly re-initialize if this is not done.

LISTSERV also stops automatically when the system is rebooted.

4.2. Initial configuration

How do I add, change, and delete LISTSERV Maintainers (aka postmasters)?

LISTSERV Maintainers are defined by their e-mail addresses in the site configuration file, by setting the site configuration variable **POSTMASTER=**. This is normally done by opening the site configuration file in a text editor (never in a word processor or other non-flat-ASCII editor) and changing the value in the variable, then saving the file and stopping and restarting LISTSERV.

Windows sites can alternatively use the SITE.EXE configuration GUI to make these changes, but must also stop and restart LISTSERV after making the change.

Note carefully that the syntax for the POSTMASTER= variable (like all other site configuration variables) differs from one OS platform to another. See Appendix C of this manual for OS-specific syntax examples.

How do I create passwords for postmasters, and what are they used for?

LISTSERV Maintainers use two different passwords, depending on the particular commands they are attempting to authenticate.

When creating a list by the e-mail method, or when using the **PWC** privileged password-management command, a LISTSERV maintainer must use the password set in the **CREATEPW=** site configuration variable.

All other commands (as well as list PUT operations performed on a list after the list is created, for example, to modify the list header) are authenticated by the personal password associated with the LISTSERV maintainer's e-mail address. This password can be created in one of two ways:

- Via the web interface, where a clickable password creation link will appear when you try to log in for the first time; or
- Via mail, by using the **PW ADD** command documented elsewhere in this manual.

Note that some mailing list commands do not always require password authentication, depending on the setting of the **Validate=** list header keyword for the list in question.

See Appendix B for more information on how the various **`Validate=`** settings affect command authentication.

How do I make my first list?

Please see chapter 7.1, below.

How do I delete a list?

There is no LISTSERV command to delete a list. This was a design decision taken very early on in LISTSERV's history, for both security reasons and to avoid accidental deletion of lists. Please see chapter 7.11, below for a procedure for deleting lists.

Does LISTSERV have a GUI interface?

LISTSERV's GUI interface is its web administration/archive interface. Many site-level and most list-level functions can be accessed via the web interface.

5. Configuring your LISTSERV® site

Please note that this manual is not intended to replace the individual installation manuals for LISTSERV on the various platforms supported by L-Soft. This is because the installation procedures vary radically from platform to platform and this manual is intended to assist LISTSERV maintainers on operational LISTSERV sites. The installation guides for all platforms are included in the software distributions, and are also available on L-Soft's World Wide Web and FTP sites.

For the purposes of this chapter, therefore, it is assumed that you have already installed LISTSERV on your host computer and have been able to start it in successfully in interactive mode. If you have not reached this point, this chapter will be of little use to you.

5.1. Site configuration files

These files have different names depending on the platform. They are located in the same directory with the executable binaries.

Platform	Site configuration file
VM	LOCAL SYSVARS
OpenVMS	SITE_CONFIG.DAT
Unix (all)	go.user
Windows NT/2000	SITE.CFG
Windows 95/98/Me	SITE.CFG

These are the *only* configuration files that should be changed on any LISTSERV installation. Software upgrades may overwrite any other configuration files located in LISTSERV's home directory. They will *never* overwrite the files listed above. The intent is to help preserve your system settings from one version to the next so that you do not experience the inconvenience of having to reconfigure LISTSERV after an upgrade.

L-Soft international, Inc., is not responsible for system downtime or misoperation occassioned by the loss of any changes that you make to configuration files other than the ones listed above.

5.2. What can be configured?

Depending on the platform, a large number of control variables are available to "fine tune" the performance and behavior of LISTSERV. The following table indicates the variables, under which platforms they are supported, and briefly what they control. Please see Appendix C of this manual for details before setting any control variable. Some variables shown in the table are VM legacy settings and are not otherwise discussed in this manual.

Table 5.1. LISTSERV site configuration variables

Variable	Short Description	VM	VMS	Unix	Win
ALL_REQUEST_ALLOWED_USERS	Specifies non-POSTMASTER users who are allowed to post to the ALL-REQUEST alias (1.8e)	yes	yes	yes	yes
ANTI_VIRUS	Boolean value determining whether or not LISTSERV's	yes	yes	yes	yes

	anti-virus scanning is enabled (1.8e)				
BITNET_ROUTE	Defines the hostname of a machine that knows how to route mail to BITNET addresses	**yes**	**yes**	**yes**	**yes**
BOUNCES_TO	Tells LISTSERV where to send bounces not related to any particular list	**yes**	**yes**	**yes**	**yes**
CHECKMDISK	List of library minidisks to be checked at startup	**yes**	no	no	no
CLI_*	Three configuration variables under the **CLI_*** rubric are available for use with DBMS/Mail Merge. Please see the *Developer's Guide for LISTSERV* (available separately) for documentation. (1.8e)	no	no	**yes (AIX only)**	no
CMDPIPE_HOSTNAME	Defines the hostname used by the LCMD utility	no	no	no	**yes**
CMSNAME	The name of the CMS system to be used on IPL commands	**yes**	no	no	no
CRASH_MONITOR	Where to send VMS or NT crash reports	no	**yes**	no	**yes**
CREATEPW	The password required to create new lists	**yes**	**yes**	**yes**	**yes**
DATABASE	Indicates whether the (old) VM database functions are enabled or not	**yes**	no	no	no
DBRINDEX	Determines whether or not the new LISTSERV database functions use reverse indexing	**yes**	**yes**	**yes**	**yes**
DEFAULT_CHANGELOG_PERIOD	Sets a default period for rotating change-logs. (1.8e)	no	**yes**	**yes**	**yes**
DEFAULT_LANGUAGE	Sets the default national language template for use by all lists on the server	**yes**	**yes**	**yes**	**yes**
DEFAULT_PROBE	Sets defaults for passive probing	**yes**	**yes**	**yes**	**yes**
DEFAULT_SPLIT	Provides a default value for the "SPLIT=" command line keyword	**yes**	**yes**	**yes**	**yes**
DELAY	The delay between two reader-scan operations	**yes**	no	no	no
DIAGD4	Indicates whether LISTSERV should use diagnose X'D4' to mimic the RSCS origin on files it DISTRIBUTEs	**yes**	no	no	no
DIST_ALLOWED_USERS	In 1.8d and following, space-separated list of non-**POSTMASTER** users who are to be allowed to use DISTRIBUTE	**yes**	**yes**	**yes**	**yes**

DIST_SECURITY	In 1.8d and following, Boolean value which controls security validation feature for the DISTRIBUTE command. **WARNING:** See Appendix C before changing this value.	**yes**	**yes**	**yes**	**yes**
FILEDISK	The filemode of the DEFAULT disk to be used for storing files via a PUT command	**yes**	no	no	no
FILEMAXL	The maximum number of lines for any incoming **non-mail** file to be accepted	**yes**	**yes**	**yes**	**yes**
FILTER_ALLOW	Defines users or classes of users who should be exempt from LISTSERV's standard filter	**yes**	**yes**	**yes**	**yes**
FILTER_ALSO	Defines users or classes of users who should not be allowed to post to any list on the server.	**yes**	**yes**	**yes**	**yes**
FIOC_INUSE_RETRY	Defines the number of seconds LISTSERV will try to open a file locked by an external process	**yes**	**yes**	no	**yes**
FIOC_TARGET	Defines (in kilobytes) the "target size" for LISTSERV's file cache.	**yes**	**yes**	**yes**	**yes**
FIOC_TRIM	Defines (in kilobytes) the threshold at which point LISTSERV should start aggressively trimming the cache.	**yes**	**yes**	**yes**	**yes**
FIOC_WARNING	Defines (in kilobytes) the cache size at which LISTSERV should write a warning to the console log.	**yes**	**yes**	**yes**	**yes**
GETQWAIVE	Internet addresses of persons to be granted an "infinite" GET quota	**yes**	no	no	no
IGNORE	A list of userid@nodes whose messages and files are to be ignored	**yes**	no	no	no
IGNORE_EXTERNAL_PRIME	Boolean value determining whether or not LISTSERV will ignore the PRIME setting on incoming DISTRIBUTE jobs.	**yes**	**yes**	**yes**	**yes**
INDEX_VIA_GETPOST	On VM, determines whether INDEX subscriptions use **GETPOST** or old-style database jobs by default.	**yes**	no	no	no
INSTPW	The optional local "installation password" associated with the INSTALL command	**yes**	no	no	no

JOB_STAT_DEFAULT	Boolean value determining whether or not the "Summary of resource utilization" is generated or suppressed in a CJLI JOB command response.	**yes**	**yes**	**yes**	**yes**
LICENSE_WARNING	Toggles license warnings on and off. **WARNING:** See Appendix C before changing this value.	**yes**	**yes**	**yes**	**yes**
LIST_ADDRESS	Default value for the "List-Address=" keyword	**yes**	**yes**	**yes**	**yes**
LIST_EXITS	Filenames of executable files that can be defined as exits by an "Exit=" list header keyword	**yes**	**yes**	**yes**	**yes**
LMCPUT	a boolean variable indicating how PUT commands for datafiles associated with the LMC FAC are handled	**yes**	no	no	no
LOCAL	a list of nodes to be associated with the hardcoded LCL FAC. Also used as the default for the "Local=" list keyword	**yes**	**yes**	**yes**	**yes**
MAILER	Internet address of the local mailer	**yes**	no	no	no
MAILMAXL	the maximum number of lines for an incoming MAIL file to be accepted	**yes**	**yes**	**yes**	**yes**
MAXBSMTP	Maximum number of 'RCPT TO:' lines per BSMTP file sent to the mailer	**yes**	**yes**	**yes**	**yes**
MAXDISTL	(HPO only) The maximum number of recipients to be listed in the LISTSERV console log as recipients of an SMTP job	**yes**	**yes**	**yes**	**yes (NT)**
MAXDISTN	Maximum number of recipients in forwarded DISTRIBUTE jobs	**yes**	**yes**	**yes**	**yes**
MAXGET	Maximum number of GET requests a user can make per day	**yes**	no	no	no
MAXGETK	Maximum number of kilobytes of data a user may obtain a day via GET requests.	**yes**	no	no	no
MDISK.cuu	Information about library minidisks	**yes**	no	no	no
MSGD	Userid of the virtual machine running a RFC1312/MSP server, if "Internet TELL" support is desired	**yes**	no	no	no
MYDOMAIN	The list of domain names which are equivalent to **NODE**--e.g., MX addresses, CNAMEs, etc.	**yes**	**yes**	**yes**	**yes**

MYORG	Short organization name that appears in the RFC-822 "Sender:" line.	**yes**	**yes**	**yes**	**yes**
NDMAIL	Whether to send mail to the local MAILER in Netdata format	**yes**	no	no	no
NODE	Internet address of this LISTSERV host	**yes**	**yes**	**yes**	**yes**
NO_NJE_JOBS	Directs a VMS™ server running in NJE mode to send all outgoing server-to-server requests via the Internet	no	**yes**	no	No
OCI_*	Three configuration variables under the **OCI_*** rubric are available for use with the DBMS/Mail Merge functionality introduced in LISTSERV 1.8d. Please see the *Developer's Guide for LISTSERV* (available separately) for documentation.	no	**yes**	**yes**	**yes**** (NT, 1.8d only)
ODBC_*	Three configuration variables under the **ODBC_*** rubric are available for use with the DBMS/Mail Merge functionality introduced in LISTSERV 1.8d. Please see the *Developer's Guide for LISTSERV* (available separately) for documentation.	no	no	no	**yes (NT)**
OFFLINETHR	Defines the system and RSCS spool thresholds for automatic offline/online control	**yes**	**yes**	no	no
POSTMASTER	A list of userid@nodes, of which the first one is the "main" postmaster (to receive transferred files).	**yes**	**yes**	**yes**	**yes**
PRIMETIME	Defines the "prime time" for your node	**yes**	**yes**	**yes**	**yes**
QUALIFY_DOMAIN	Defines domain to be appended to all non-qualified addresses	**yes**	**yes**	**yes**	**yes**
RESMODES	Defines a list of filemodes which are to be considered as "reserved" and never available for dynamic ACCESS	**yes**	no	no	no
RSCS	A list of local userids which must be treated as RSCS virtual machines	**yes**	**yes**	no	no
RUNMODE	The mode (NETWORKED, STANDALONE, or TABLELESS) that LISTSERV runs in.	**yes**	**yes**	**yes**	**yes**
SEARCH_DISABLED	Determines whether or not the (new) SEARCH command is enabled.	**yes**	**yes**	**yes**	**yes**
SMTP_FORWARD	The Internet hostname of the	no	**yes**	**yes**	**yes**

	server to which all outgoing SMTP mail should be forwarded for delivery				
SMTP_FORWARD_n	Defines n number of "SMTP workers" used to split up the SMTP forwarding load	no	**yes**	**yes**	**yes**
SMTP_LISTENER_IP	Dotted-decimal IP address which sets the IP address to which the SMTPL.EXE "listener" will bind at boot time.	no	no	no	**yes**
SMTP_LISTENER_PORT	Integer value which sets the port number to which the SMTPL.EXE "listener" will bind at boot time	no	no	no	**yes**
SMTP_RESET_EVERY	Directs LISTSERV to reset open SMTP connections every n minutes	no	**yes**	**yes**	**yes**
SORT_RECIPIENTS	Determines whether or not to sort recipients in the RFC821 mail envelope	**yes**	**yes**	**yes**	**yes**
SPAM_DELAY	Sets the server-wide value (in minutes) for the anti-spam quarantine period	**yes**	**yes**	**yes**	**yes**
SSI	Flag telling LISTSERV that it runs in a SSI system	**yes**	no	no	no
STARTMSG	Recipients of start and stop messages	**yes**	**yes**	**yes**	**yes**
STOREPW	The password to be used by postmasters when executing CP/CMS commands and when storing files in the server by means of the PUTC command	**yes**	**yes***	**yes***	**yes***
SYSTEM_CHANGELOG	Enables a system-level changelog	**yes**	**yes**	**yes**	**yes**
TCPGUI_IPADDR	Defines the IP address used by the TCPGUI interface	no	**yes**	**yes**	**yes**
TCPGUI_PORT	Defines the port number used by the TCPGUI interface	no	**yes**	**yes**	**yes**
TRAPIN	List of userid@node templates from whom LISTSERV should never accept mail	**yes**	**yes**	**yes**	**yes**
TRAPOUT	List of userid@node templates to whom LISTSERV should never send mail	**yes**	**yes**	**yes**	**yes**
VM30091	Indicates whether or not the VM30091 message suppression functions are available	**yes**	no	no	no
WEB_BROWSER_CONFIRM	Indicates whether or not LISTSERV should require "OK" confirmation for commands sent from WWW browsers.	**yes**	**yes**	**yes**	**yes**

WWW_ARCHIVE_CGI	The (preferably) relative URL that leads to the WWW archive CGI script. (This is a URL, not an OS path name.)	no	**yes**	**yes**	**yes**
WWW_ARCHIVE_DIR	The full OS path name to the WWW archive directory	no	**yes**	**yes**	**yes**
WWW_AUTHINFO_DISABLE	Disable/enable the web archive interface's IP address verification function	no	**yes**	**yes**	**yes**
XFERTO	Userid of the virtual machine to which files found in the lists readers should be transferred	**yes**	no	no	no

There are also a number of configuration variables pertaining to the DBMS/Mail Merge functions. These variables are documented in the *Developer's Guide for LISTSERV*, available separately.

Notes:

* For non-VM systems, STOREPW is a secondary password that is functionally identical to CREATEPW. You should use the same value for both passwords, i.e., set **STOREPW= %CREATEPW%** (for Windows NT/2000 and 95/98), etc.

** The native OCI interface was available for Windows servers only in version 1.8d. It was removed in LISTSERV 1.8e. See Appendix C, **OCI_*** for details.

5.3. Files used by LISTSERV

The proper operation of LISTSERV is dependent on LISTSERV's ability to find a number of files that belong to it. The following list of files are **required** to operate the product, and in most cases must be located in the same directory with the LISTSERV executables. (The notable exception is under unix, where all of the data files other than the '**go***' files are required to be placed one directory below the executables, typically in **~listserv/home** .)

(Note that under certain conditions, some required files aren't necessary; these will be noted where applicable. Note also that some files are not shipped with the distribution, but are generated automatically the first time you run LISTSERV.)

Program Executables

Dependent on the platform, the required executables are:

VM:	**LSV EXEC**
OpenVMS:	**LSV.EXE**
unix:	**lsv***
Windows (all):	**LSV.EXE**

For OpenVMS and Windows systems, the executable **SMTPW.EXE** is also required.

For Windows NT/2000 systems not running LSMTP and for all Windows 9x/ME systems, the executable **SMTPL.EXE** is also required.

The executables listed above belong in the following places (depending on the platform):

VM:	on LISTSERV's A disk
OpenVMS:	in LISTSERV's "A" directory, normally **LISTSERV_ROOT:[MAIN]**
unix:	in the LSVROOT directory, i.e., **~listserv/**
Windows (all):	in LISTSERV's "A" directory, normally ***drive*:\LISTSERV\MAIN**

Finally, the Web Archive ('**wa**') interface CGI script is shipped in the A directory of the non-VM servers. This script must be copied into the appropriate script directory for your Web server (per chapter 5.4, below) if you plan to use the Web Archive interface. On OpenVMS and Windows servers, this file is **WA.EXE**, while on unix machines it is **wa*** .

BITNET Network Table files

These files are *not* required when running LISTSERV with **RUNMODE=TABLELESS**, and are not shipped with evaluation copies for Windows 9x/ME or with LISTSERV Lite. Network table files include:

bitearn.nodes	**bitearn.dbindex**	**bitearn.distsum2**	**bitearn.linkdef2**
bitearn.linksum2	**bitearn.nodesum3**		

With the exception of BITEARN NODES, all files are regenerated whenever BITEARN NODES is updated or when an explicit **NODESGEN** command is issued. For pre-1.8c servers (or non-registered 1.8c or later servers), BITEARN NODES must be downloaded on an approximately monthly basis from

ftp://ftp.lsoft.com/listserv-data/bitearn.nodes

or from the European mirror at

ftp://segate.sunet.se/listserv-data/bitearn.nodes

Beginning with 1.8c, BITEARN NODES on registered networked servers is updated using the same mechanism as PEERS NAMES and other LISTSERV tables. Note that this requires that your mail server support incoming files of at least 1.5M. VM sites have not been included as they typically maintain this file using the UPNODES procedure and store it on a public disk, applying change control procedures in the process.

Internet and Peer Networking Table files

aliases.names	**intlinks.file**	**intpeers.names**	**linkswt2.file**
peers.dbindex	**peers.dbnames**	**peers.distsum2**	**peers.names**
peers.namesum	**service.names**		

These files are used by LISTSERV to define its "backbone" and other peer servers, as well as to help determine the best routes for mail sent via the DISTRIBUTE algorithm.

For registered sites they are updated periodically by mail from other servers. The update process is automatic and does not require LISTSERV maintainer intervention unless a problem is noted.

For non-registered sites the files must be updated manually. See **http://www.lsoft.com/table-updates.html** for information on how to accomplish this.

Sites running in TABLELESS or STANDALONE mode do not require these files. This

includes all LISTSERV Lite and LISTSERV Shareware sites.

LISTSERV's external data files

LISTSERV uses these files for a number of purposes. The fact that they are external to the executables makes it easy to update them when needed. These files include:

country.file	**default.mailtpl**	**default.wwwtpl**	**errfac.file**
listkwd.file	**lsvhelp.file**	**lsvinfo.file**	**permvars.file**
signup.filex	**stdcmd.file**	**sysff.file**	**site.catalog**
system.catalog			

PERMVARS.FILE is LISTSERV's main "permanent variables" file; among other things, this is where LISTSERV registers spammers and users that have been served off. **NOTE VERY CAREFULLY** that this file should **NEVER** be modified manually. It is in a binary format and, if corrupted, LISTSERV will not start.

The SIGNUP.FILEx files (initially there are 9 [or 31 if you are licensed for HPO], for example, SIGNUP.FILE1, SIGNUP.FILE2, etc.) are used to register users and their real name fields.

SYSTEM.CATALOG is used by LISTSERV to register system files; it should not be modified, as it is always shipped with new versions and will thus overwrite itself. Instead, SITE.CATALOG should be used to register files and list file archive catalogs (listname.CATALOG) for users to retrieve. (SITE.CATALOG is not shipped with LISTSERV; please see the chapter on *Notebook and File Archives* for details.)

DEFAULT.MAILTPL and DEFAULT.WWWTPL are the files from which LISTSERV gets its default mail templates and default web templates for responses to user input. See Chapter 9, *Creating and Editing LISTSERV's Default Mail Templates*, for details.

User reference material

The following files are LISTSERV's online documentation.

listdb.memo	**listfile.memo**	**listkeyw.memo**	**listmast.memo**
listpres.memo	**listjob.memo**	**listlpun.memo**	**listownr.memo**
listserv.memo	**listall.refcard**		

LISTALL.REFCARD is broken into three parts internally. Part 1 is the response to the INFO REFCARD command; Parts 1 and 2 are the response to a GET LISTOWNR REFCARD command; and the whole document is sent in response to a GET LISTMAST REFCARD command.

Command line utilities (non-VM)

Depending on your platform, the following executables may have been shipped (under unix they must all be complied from the corresponding ***.c** files):

LCMD.EXE (or **lcmd***)

> LCMD is a command-line named-pipes interface to LISTSERV. You can use it to send commands directly to LISTSERV from the console and receive information in return, either on the console itself (Windows and OpenVMS) or via mail (unix). The syntax is:

Windows: **lcmd [*computer*[*serverid*]] *command***
OpenVMS and unix: **lcmd *command***

The user running LCMD must have appropriate permission (e.g., must be a list owner or LISTSERV maintainer) in order to issue the various protected commands.

LISTVIEW.EXE (or **listview***)

LISTVIEW is a utility that allows you to type the binary-format **.LIST** files to standard output so that they can be viewed and/or redirected to text files. The syntax is:

listview [-a] [-e[h]] [-h] [-r *nnnn*] [-s] *file1 file2...*

You can choose only one of the command line options at a time, except that you can specify one of the other options along with the **-r** option if needed. The options are:

-h Show the header only
-s Show the header + the subscribers (without the option string in columns 81-100)
-e Show the list of subscribers only (without the option string)
-eh Similar to -e, but show only the hostnames without userid@
-a Show the entire list file
-r ***nnnn*** Generate two files (***listname*.view1** and ***listname*.view2**) for each list file viewed with this option. The view1 file contains ***nnnn*** subscriber addresses chosen at random from the list, where ***nnnn*** is an integer value between 1 and the number of users on the list. The view2 file contains the rest of the subscriber addresses from the list, randomly sorted. (The view2 file is useful in cases where you wish to pull x names at random from your mailing list, and then pull x more names at random without duplication. Note however that you would have to add the subscribers in the view2 file to a regular LISTSERV list in order to be able to run **listview** against those subscribers.)

While you can specify one other option with **-r** to manipulate the output, the following caveats should be noted:

- **listview -h -r** results in a blank file
- **listview -s -r** does not output the list header
- **listview -eh -r** outputs a list of random hostnames, but they are not unique.
- **listview -a -r** is the same as **listview -r**

Note carefully that running **listview -r** against a mailing list with a value of ***nnnn*** greater than the actual number of subscribers in the list will result in duplicates being written to the view1 file and the generation of a view2 file of length 0.

LISTVIEW executed with no option is the same as '**listview -a**'.

You can redirect the output of LISTVIEW with standard OS-dependent redirection symbols. For instance,

`listview -h mylist > mylist.file`

redirects the output to the ASCII file '**`mylist.file`**'.

`JOBVIEW.EXE` (or **`jobview*`**)

JOBVIEW allows you to read the Base64-encoded spool files created by LISTSERV (see below for the types of files created in the spool directory that may be read with this utility). The syntax is simply

`jobview` ***`file1 file2...`***

GUI site configuration utility (Windows NT and Windows 95 only)

`SITE.EXE and SITE.HLP`

The Site Configuration Utility for LISTSERV allows you to easily configure LISTSERV's operation. While this can also be done by manually editing LISTSERV's SITE.CFG file, the GUI gives you an easier way to take care of this task. Online help for the various configuration variables is provided, and new LAKs can be entered. Basic optimization for various pre-calculated loads can also be performed.

Line-mode site configuration utility (OpenVMS only)

`LISTSERV_CONFIGURE.COM`

A very basic line-mode utility that allows you to modify the OpenVMS version of the site configuration file. Useful for initial configuration. Most OpenVMS sysadmins will probably prefer to edit the SITE_CONFIG.DAT file by hand with a text editor.

Other files that will appear during use

While in use, LISTSERV creates various files for itself. On the A disk or in the MAIN or HOME directory, these are typically:

`.AUTODEL` files	Maintain data for LISTSERV's autodeletion functions; one for each list that has Auto-Delete enabled. If no auto-deletion reports are pending, this file will not exist.
`.CHANGELOG` files	Contain data regarding subscription changes for a given list if that list has the "Change-Log= Yes" list header keyword setting. These files are called ***`listname`*** **`CHANGELG`** on VM.
`.DIGEST` files	These files are the (volatile) digest files for each list that has digests enabled. They are deleted and restarted when the digest is cut. Note that if the location parameter of the Digest= keyword is not set to something that points to the MAIN or HOME directory, .DIGEST files will not appear in the MAIN or HOME directory, but rather in the directory specified.
`.LIST` files	Mailing list files, including the header and subscriber information. Do not attempt to edit these files with a text editor; use the **`GET`** and **`PUT`**

commands instead.

.OKxxxxxx files — Usually found for edited lists, but can also appear for non-edited lists if users are set to REVIEW. These are mail messages that are awaiting "OK" confirmation. If they are not confirmed, they are automatically deleted after about a week.

.OLDLIST files — These files contain the last saved version of the list file. If you PUT a header and find that you've made a fatal mistake (like adding users "on the fly" and deleting everyone else on the list, or editing the list file by hand and corrupting the record structure) you can send the command **GET listname (OLD** to have the listname.OLDLIST file sent to you.

.SUBJECT files — Maintain the list of subjects for the digest. Again, if digests are not enabled for a specific list, this file does not exist for that list. Also, the same note for the location of these files as for .DIGEST files applies. .SUBJECT files are deleted and restarted when the digest is cut.

In the SPOOL directory, the following file types will be found:

.ERROR — LISTSERV generates an .ERROR file in the spool when it encounters an error in a JOB file. These can be viewed with the **jobview** utility and are important for tracing certain errors back.

.JOB — Files that have been received by LISTSERV and are queued for processing. These files are in Base64 format and can be viewed with the **jobview** utility.

.JOBH — Held .JOB files. Such files are either being processed by LISTSERV (and are thus locked) or have generated an error message. These can also be viewed with the **jobview** utility.

.MAIL — Files that have been processed through LISTSERV and are queued for delivery to the outgoing SMTP mail agent. These are plain-text files.

.MAIL-ERR — Files that have been processed through LISTSERV and for which delivery has been attempted, but for which a "permanent" SMTP error has resulted. If you have reason to believe that the error was not actually "permanent", simply rename the file with the **.MAIL** extension and LISTSERV will pick it up for another try.

JOBH files containing the string **$NOJOB$** in the filename are typically waiting to be processed because the list they are going to has an explicit **Prime=** variable set and the non-prime time has not yet arrived.

5.4. Installing and configuring LISTSERV's WWW Archive and Administration Interface

LISTSERV 1.8d and later includes an optional WWW archive and administration interface (not enabled by default). This interface is used to allow users to browse and search notebook archives for lists with the feature explicitly enabled, as well as to allow list owners to manage almost every aspect of their lists and to allow LISTSERV maintainers to perform a number of common site management tasks. The interface is secured by the

use of LISTSERV personal passwords. List owners have administrative access only to their own lists; general users have access only to the archives of public lists or to private lists to which they are subscribed (in other words, there is no difference between the access one receives via the web interface and the access one receives via the mail interface).

LISTSERV 1.8c also included an optional WWW archive interface which did not have any of the administrative functions described in this chapter.

5.4.1. The WWW Archive Interface described

Postings can be organized by date, by topic or by author, and a search function with online help is provided. LISTSERV's WWW interface has the following advantages over "hypermail" style web archiving:

- The information on the web is always up to date. New postings are shown as soon as they are received.

- The postings can be organized in the manner that best suits the reader: by date, by author, by topic, with or without table of contents, with or without showing the author, etc.

- Only one copy of the information is kept, and in particular there is no need to create an individual HTML file for each posting. This design allows the interface to scale up gracefully to lists with hundreds of thousands of archived postings, which would otherwise require hundreds of thousands of individual HTML files, wasting disk space (each file takes up at least one disk block) and stressing the file system past reasonable limits.

- The search forms can be used to create search requests matching (for instance) all postings in the last X days. The resulting URL can then be bookmarked and reloaded on a regular basis.

- List owners can customize the main page for their lists without any intervention by the LISTSERV maintainer, by updating one of the mail template forms for their list (**WWW_INDEX**). The LISTSERV maintainer can customize common pages and header/trailer HTML statements by updating system templates.

To take advantage of this new interface, you must first ensure that the "Notebook=" options for your list are compatible with the WWW interface. In most cases, you will not have to do anything, but certain options are incompatible with the use of the WWW interface and may need to be changed:

- The archive frequency MUST be WEEKLY, MONTHLY or YEARLY. SEPARATE and SINGLE notebooks are not supported. L-Soft generally recommends converting lists with SINGLE notebooks to YEARLY unless there is a compelling reason to have all the messages in exactly one file.

- For optimal performance, the archive frequency may need to be adjusted to produce an "adequate" number of topics and messages in each archival period. The definition of "adequate" depends on your users, the kind of equipment they have, and how they connect to the Internet. As a rule, home users will prefer a larger number of smaller archives whereas office users with large screens and T1 or better connectivity will tolerate a larger table of contents.

- Under the 1.8c version of the interface, the archives must be public as there is no userid/password control in the web archive interface. Under 1.8d this restriction has been removed.

- Under 1.8c, on most systems, the directory in which your list archives are kept must be specified in absolute rather than symbolic form, or the WWW interface will not be able to access it. Symbolic form is when the directory name is a single letter, for instance "**Notebook= Yes,A,Monthly,Public**" ("A" being a logical filemode defined in LISTSERV's site configuration which points to the directory where LISTSERV keeps its internal files). In most cases, your list header will probably read something like "**Notebook=Yes,E:\LISTS\XYZ-L,Monthly,Public**" and you will not have to worry about this.

  Under 1.8d this restriction has been removed (LISTSERV will translate the logical filemode into a full path), but L-Soft still strongly discourages the use of logical filemodes for the "where" parameter of the **Notebook=** keyword, primarily for security reasons but also to keep things orderly. A full path is *always* preferred, and each list should imperatively have its own subdirectory. See 5.8, below, and chapter 8 for details.

The LISTSERV maintainer must then enable the list for the WWW interface. This may require the installation of a web server and of the WWW interface code itself. You can then modify the **WWW_INDEX** mail template form to customize the main archive page for your list. See chapter 9 of this manual for more information on customizing mail templates.

5.4.2. The WWW Administration Interface described

In 1.8d and later, assuming that the WWW interface has been installed per the instructions below, the WWW administration interface is enabled automatically for all lists on the server that are not coded "Validate= Yes,Confirm,NoPW" or "Validate= All,Confirm,NoPW". The basic URL for the list owner section of the interface is

http://*hostname*/*script-directory*/wa[.exe]?LMGT1

where ***hostname*** is the name of the LISTSERV host, and ***script-directory*** is the name of the directory where "wa" is installed. For unix you specify "**wa?LMGT1**" and for Windows and VMS you specify "**wa.exe?LMGT1**". With some non-unix web servers you may have to type "**WA.EXE?LMGT1**" (that is, all in upper case) in order for this to work.

Site managers have a different entry point at

http://*hostname*/*script-directory*/wa[.exe]?ADMIN

which allows them to create lists, customize site-wide WWW templates, and manage DBMS and mail-merge operations. This entry point also has a link to the list owner section, so site managers may wish to bookmark this entry rather than the **LMGT1** entry.

See chapter 11 for detailed information on the web administration interface.

5.4.3. Installing a web server

Please note that L-Soft cannot help you with the installation or configuration of your web server itself. L-Soft does not recommend or endorse specific web servers, nor does L-Soft have development machines with every possible web server installed. You should

ensure that the web server software you choose is installed and operating properly before attempting to install the LISTSERV WWW interface script.

If you do not already have a World Wide Web server installed and operating on your LISTSERV machine, you will need to obtain and install one. There are quite a few free web servers available for downloading on the Internet for most systems; you may want to start your search for server software at the W3 Consortium's web site at **http://www.w3.org/**. Naturally, commercial web servers can also be used.

Please note that for security purposes you should always disable directory browsing if it is not disabled by default by your web server.

5.4.4. Installing the web archive interface script

The CGI script for the web archive interface must be installed in the directory where your web server normally keeps CGI scripts and from which they are authorized to run. If in doubt, please read the manuals that came with your web server and/or contact the web server manufacturer's support group; *L-Soft cannot help you with this.* LISTSERV cannot install the script for you because installation depends on which server you use, which operating system you are running, how the server has been configured, etc.

Please note carefully that the web interface is not designed to be run on a machine separate from the LISTSERV server. It MUST run on the same machine. This means that a web server MUST be installed on the LISTSERV machine or you will not be able to use the web interface.

System specific instructions:

- Windows: Copy **WA.EXE** to the appropriate directory. For Microsoft's Internet Information Server (IIS), this is normally **C:\INETPUB\SCRIPTS** (*not* **C:\INETPUB\WWWROOT\SCRIPTS**).[1]

- Windows NT/2000: **WA.EXE** builds shipped with LISTSERV 1.8d and later communicate with LISTSERV via TCP/IP rather than via named pipes. If your **%SystemRoot%** directory (e.g., **C:\WINNT**) is on an NTFS partition, in order for this to work properly you must grant the "Everyone" user (or at least the user that invokes **WA.EXE**, for example, IUSR_xxx under IIS) **R/X** permissions on the following files in

---

[1] When using IIS version 4.0, you will likely get an Error(5) from 'wa' when trying to do things like search, post to the list, or anything else that requires 'wa' to pipe a command to LISTSERV. We believe this to be due to a bug in IIS 4.0 (the problem does not appear in IIS 3.0 or earlier), specifically in the code that enables automatic password synchronization for the anonymous user stage. The only fix our support group are aware of is to open the Internet Service Manager, find the "wa.exe" executable in the tree, and then do the following:

1. Open the property sheet for "wa.exe" by double-clicking it
2. Click on the "File Security" tab
3. Under "Anonymous Access and Authentication Control", click on the "Edit..." button, which pulls up a dialog box entitled "Authentication Methods"
4. Click the "Edit..." button next to "Account used for Anonymous Access:" which brings up a dialog entitled "Anonymous User Account"
1. UNCHECK the box for "Enable Automatic Password Synchronization"
2. Click "OK"

At this point 'wa' should resume working again. You might need to type the password for the IUSR_xxxx account but our experience was that you needed only to uncheck the box and click "OK".

the **%SystemRoot%\system32** directory:

MSAFD.DLL WS2_32.DLL WS2HELP.DLL WSHTCPIP.DLL WSOCK32.DLL

Under IIS the invoker is normally the **IUSR_xxx** user created when you install IIS. Other web servers are probably different and you may have to check the logs to see what user is invoking **WA.EXE**.

This instruction can be ignored if your **%SystemRoot%** directory is on a FAT or FAT32 partition.

The Windows NT/2000 1.8d installation kits offer to grant world-read permissions to the above files at install time for your convenience if **%SystemRoot%** is on an NTFS partition.

- unix: copy '**wa**' to the appropriate cgi-bin directory, change its owner to '**listserv**' and set the suid bit (typically, '**chmod 4755 wa**'). This authorizes the interface to read archive files. Please note that one of the most common problems with '**wa**' under unix is that the installer has not followed this instruction.

- OpenVMS: you will have to link **WA.OLB** with the CGI library provided with your web server, then copy it to the appropriate directory. Make sure to arrange for the program to have read access to the archive files for the lists you want to serve on the web. This may vary from one web server to another.

While the script can be renamed, a short name will help keep the HTML documents small and speed up the site.

5.4.5. Creating a subdirectory for the archive interface

Create a subdirectory on your web server to contain the various files LISTSERV will be creating for the web archive interface. The suggested name (and the name LISTSERV will expect by default) for the subdirectory you will create in this step is '**archives**'. Under IIS, you would typically make the directory **C:\INETPUB\WWWROOT\ARCHIVES** for this purpose. For a unix server running Apache it might be **/usr/local/etc/httpd/htdocs/archives** . Please note the following IMPORTANT restrictions carefully:

- Do *not* simply use your main HTML documents directory as LISTSERV will create quite a few files. It is much more orderly to keep the web archive interface's files and subdirectories in their own place in any case.

- Do *not* use the directory you keep the list's notebook archives in for this purpose. Notebook archives should always be kept separate from the web interface, preferably in a completely separate directory hierarchy.

- Corollary to the above: Do *not* set the **Notebook=** keyword for any list so that the list's notebook archives are kept in the subdirectory used by the web archive interface for the list.

For specifics on what should be kept in what directories, see section 5.8, below.

System specific steps:

- OpenVMS: define the systemwide logical **LISTSERV_WWW_ARCHIVE_PATH** to point to the directory you just created, and **LISTSERV_WWW_ARCHIVE_URL** with the URL to the directory in question (preferably relative).

- unix: create a world-readable file called **/etc/lsv-wa.config** with the following two statements:

```
PATH xxx
URL yyy
```

Where '**xxx**' is the absolute path to the directory you've just created and '**yyy**' is the URL to this directory (preferably relative). For instance:

```
PATH /usr/local/etc/httpd/htdocs/archives
URL /archives
```

- Windows NT/2000: if necessary (and it shouldn't be), you can update the registry key **HKEY_LOCAL_MACHINE\SOFTWARE\L-Soft\LISTSERV\WWW_ARCHIVE_URL** to override the default URL to the directory you have just created. Again, this is not normally necessary and is only provided for weird web servers, etc. Don't do it unless it didn't work without it.

5.4.6. Configuring LISTSERV to activate the web archive interface

This is done by modifying LISTSERV's site configuration file (see Appendix C) to add two variables:

- **WWW_ARCHIVE_CGI** is the (preferably) relative URL that leads to the CGI script you have just installed. Typically this will be something like '**/cgi-bin/wa**' or '**/scripts/wa.exe**'. This is a URL, not an OS path name.

- **WWW_ARCHIVE_DIR** is the full OS path name to the directory you created in the previous step (**C:\INETPUB\WWWROOT\ARCHIVES** or whatever).

Under unix, you may have to export these variables (you can check the 'go' script to see if they are already exported for you; in early versions of the interface they were not) by adding the lines

```
export WWW_ARCHIVE_CGI
export WWW_ARCHIVE_DIR
```

at the end of **go.user**. Again, if these variables are already exported in the 'go' script, there is no need to do this.

LISTSERV will then create and maintain a file called **http://localhost/archives/index.html** from which you can access all the postings. (This is made from the template **WWW_ARCHIVE_INDEX**--see below.)

Note that proper operation of the interface under LISTSERV 1.8d requires that the 'wa' script be able to talk to LISTSERV via TCP/IP to port 2306 (LISTSERV's TCPGUI port).

5.4.7. Customizing the web pages LISTSERV creates

Under LISTSERV 1.8d, the simplest (and recommended) way to make changes to the templates which contain the information for these pages is to use the tools provided in

the WWW administration interface for changing the "look" of your site.

The LISTSERV maintainer can create a file called **www_archive.mailtpl** in the main LISTSERV directory to override the web archive template forms found in **default.mailtpl**. (See chapter 9 for more information on LISTSERV's mail templates.) There are 3 templates you typically might want to override:

- **$WWW_ARCHIVE_HEADER**: this is added at the top of *every* HTML file generated by the interface. This can be used to set the background color and insert a logo. Usage should be kept to a minimum since it appears on every page and gets in the way of the information people want to see.

- **$WWW_ARCHIVE_TRAILER**: same but added at the bottom of every HTML files. Can contain legal disclaimers, copyright information, and useful links.

- **WWW_ARCHIVE_INDEX**: this is the main page for the web archive interface. The default works fine but is a bit bland.

The list owner can also control the main page for his own lists by creating the usual ***listname*.MAILTPL** file with a **WWW_INDEX** template. (Again, the recommended way of doing this is to use the template editing tools in the web administration interface.)

There are more templates for other parts of the web interface which are found in the file **default.wwwtpl**. These templates can be overridden by placing the edited template forms in a file called **site.wwwtpl** (or by using the template editing tools as already mentioned).

In any case you should not make changes to either **default.mailtpl** or **default.wwwtpl** themselves as any changes you make to these files will be overwritten during an upgrade of the software.

5.4.8. Enabling individual lists

Once the interface is installed, LISTSERV will automatically make any mailing list with public archives available through it, *provided* that a subdirectory has been created for them in the '**archives**' subdirectory created above, and *provided* that LISTSERV has read/write access to the subdirectory.

WARNING:
CONFIDENTIAL= YES DOES NOT LIMIT ACCESS TO ARCHIVES!

Note that public notebooks for any list coded "Confidential= Yes" will be available via the interface if a subdirectory under '**archives**' is created for that list. However, unlike the behavior in 1.8c, the list will *not* appear on the main archive index page if you have coded "Confidential= Yes". In this case there are two avenues for users who know the list exists and want to access the web archives:

- Users can bookmark or manually enter the link to the list's archives, for example,

  http://www.yourhost.com/archives/yourlist.html

- Users can click on the link at the bottom of the main index page that pulls up an "unlisted archive" form, into which they can type the name of the list.

On the other hand, if you code "Confidential= Service", your list *will* show up on the main archive index page for your server but *will not* show up in the CataList or global list of lists.

Under 1.8d and later, "Private" notebooks can be viewed via the WWW interface by following the same instructions as for "Public" notebooks. However, in order to view the notebooks, subscribers must log in with their *subscribed* userid@host and their LISTSERV password (set with the **PW ADD** command or via the WWW interface). Please note carefully that if the user is subscribed as "joe@unix1.host.com" and tries to log in as "joe@host.com", he will be refused access. Also note that unless the list is coded "Confidential= Yes", there will be a link to its archives in the main archive index page.

If you do not want the confidential and/or "Private" list's notebooks available via the WWW interface at all, simply do not make a subdirectory for it under '**archives**'.

Please note that when removing a list from the WWW archive interface, you MUST delete the list's directory under '**archives**'. Otherwise someone with a bookmarked URL may still be able to access some of the archives via the web.

Also note that under unix, if the '**wa**' script does not have the suid bit set, the interface will appear to work normally until you try to read a message. If the suid bit is not set, you will receive a message to the effect that the archives are not available and to try again in 30 seconds.

As an example, let's assume that you have a list called XYZ-L that you want to make available through the Web interface, and that so far you have used the defaults for the installation of the interface.

First, under the 'archives' directory you created above, you must create a directory with the same name as your list. Thus, in order to make the XYZ-L list accessible through the interface, you must create the directory '**archives/xyz-l**'

Next, you would edit the XYZ-L header to indicate how you want the list to appear to the interface. If you want the archives to be wide open, you must code

```
* Confidential= No
* Notebook= Yes,where,interval,Public
```

If you want the archives to be "wide open" but don't want a link on the main archives page, you would code

```
* Confidential= Yes
* Notebook= Yes,where,interval,Public
```

If you want the archives to be accessible only by subscribers (with a password) and to have a link on the main archives page, you would code

```
* Confidential= No
* Notebook= Yes,where,interval,Private
```

And if you want the archives to be accessible only by subscribers (with a password) but you do *not* want a link on the main archives page, you would code

```
* Confidential= Yes
```

*** Notebook= Yes,*where*,*interval*,Private**

Finally, if you want the archives to be available via the interface (either with or without a password), and you want a link on the main archives page, but you do *not* want your list to appear in the CataList or global list of lists, you would need to code

*** Confidential= Service**

and "Notebook=" would be either Public or Private depending on your preference, as above.

Please note carefully that coding the Confidential= keyword has other implications. For instance, if you want your list to show up in the CataList or be available via the Global List Exchange (GLX), you must set "Confidential= No". Thus advertising your list globally is not compatible with having your archives available via the web but not having a link on the server's main archives index page.

Finally, you would simply perform a GET and PUT of the XYZ-L header. When you PUT the header, LISTSERV will create the XYZ-L.HTML file in the '**archives**' directory and build indexes for the list in the '**archives/xyz-l**' directory.

Note: If you do not execute a list **PUT** operation after creating the directory for the list under '**archives**' (for instance, if the list already had public archives and it was not necessary to edit the header), LISTSERV will wait until midnight to create the web archive files for the list rather than creating them immediately. (Naturally, stopping and restarting LISTSERV will also force a rebuild of all of the web interface files but is not recommended as the normal way to accomplish this.)

At this point you will be able to access XYZ-L's archives from the URL **http://www.yourhost.com/archives/xyz-l.html**.

5.4.9. Enabling web-based bulk operations

Bulk operations (part of the list owner administration section of the interface) are *not* enabled by default when the interface is installed. As the site manager, you must create a directory called "upload" under the directory specified in the **WWW_ARCHIVE_DIR=** site configuration variable, and give the userid under which the "wa" CGI program is run write permission in that directory. This is the ***only*** directory in which "wa" needs write authority, and only for this functionality. If you do not want the functionality, do not create the "upload" directory.

Please note carefully that your browser MUST support the RFC1867 file upload extension or you will not be able to use the bulk operations page. Most current browsers do support this extension, including but not limited to Netscape 3.x and later, and Internet Explorer 4.x and later.

(If you get an error 2 when you click on the "Import" button, this means that the "upload" directory has not been created. If you get an error 13 when you click on the "Import" button, this means that the "upload" directory has been created but the CGI program user does not have write permission in that directory.)

5.5. The "spam" detector and anti-subscription-"spoofing" feature

L-Soft acknowledges that these features have been continually upgraded and enhanced throughout the 1.8e development process, but in keeping with previously-announced policy, specifics are proprietary and will not be documented.

5.5.1. Spam quarantine

One of the most arduous problems the spam detector has to face is the accurate detection of the first few copies of the spam. When the first copy reaches the first LISTSERV server worldwide, it is just a posting like any other. It will take repeated occurrences of this same posting for LISTSERV to realize that it is in fact a spam. However, it is desirable to block this very first copy as well, and this can only be accomplished by introducing a delay in the processing of "suspicious" messages. This "quarantine" gives the spam detector some time to gather the necessary evidence to determine if the message is a spam or not. The default value is 10 minutes, and can be changed by adding:

*** Loopcheck= Spam-Delay(*xxx*)**

in the list header (the value is in minutes). The LISTSERV maintainer can also change the system default by adding a **SPAM_DELAY** variable to the LISTSERV configuration with the desired value (also in minutes). A value of zero disables this part of the spam filter.

(It should be carefully noted that setting **SPAM_DELAY=0** *does not* turn off LISTSERV's spam filter. It turns off only the spam quarantine part of the overall filter. There is no setting to disable the spam filter server-wide; it can be turned off only at the list level, with "Loopcheck= NoSPAM" in the list header.)

The default value of 10 minutes is adequate in most cases; it can be lowered on fast, large, active servers and may need to be increased on servers with chronical backlogs. Currently, LISTSERV determines whether a message is suspicious or not based on the sender's posting history. This however may be changed in future versions to further improve the efficiency of the spam detector.

5.5.2. "Anonymous" spam alerts

On occasion, you may receive a spam alert from LISTSERV where the offender's e-mail address is replaced with the word "anonymous". These alerts are generated by new detection algorithms where, for various reasons, it may sometimes be desirable to hide the identity of the potential offender, usually because there is a fair chance that the posting is in fact legitimate for the particular lists to which it was posted (for instance because these lists were configured to tolerate a high degree of cross-posting). In this case, information about the text of the message may be released and ultimately lead to a spamming alert that will block further copies of this same message, while the identity of the poster remains hidden.

5.5.3. Subscription anti-spoofing feature

Not long before the release of LISTSERV 1.8c (January 1997), a number of point and click utilities began to appear on anonymous FTP servers, allowing mischevious users to forge Internet mail on an industrial scale and subscribe an unfortunate victim to thousands of mailing lists. The resulting mail onslaught fills the victim's mailbox in minutes, rendering the account forever unusable. It also brings the mail server on which the account is hosted to its knees, causing, in some cases, tens of thousands of dollars in consequential damages as other users of the same system also lose precious e-mail.

In most cases, the account ends up being closed. Unfortunately, this usually doubles the load on the recipient's mail server, as a delivery error needs to be generated for every message received from the mailing list servers. Thus, it is not uncommon for the service provider to leave the account open and simply reconfigure it in such a way that incoming mail continues to be accepted, but is summarily discarded without generating a costly delivery error notification. While it is difficult to blame the service provider for wanting to minimize impact to their customers, the drawback is that the list owners may never be notified of the fact that the account was closed. On any large LISTSERV system, there are likely to be dozens of these addresses, each being sent hundreds or possibly thousands of messages a day which are simply discarded and waste resources.

Until now, the only defense against this attack was to configure mailing lists to require subscription confirmation:

```
* Subscription= Open,Confirm
```

LISTSERV will then send a confirmation request to the victim, who does not reply and thus is not added to the list. While this line of defense is 100% effective, it may not always be practical or desirable to configure the list to require confirmation.

LISTSERV version 1.8c and later is able to detect these "spoofed" subscription attacks automatically. When more than 50 subscription requests are received from the same account in a short time frame, LISTSERV automatically undoes all the subscription requests and rejects any further subscription attempt for a certain period of time. This applies even to requests that LISTSERV forwarded to other servers; LISTSERV will then send a SIGNOFF request to the remote server for the address in question. Note that, in some cases, the subscription may not be undone, either because of a temporary condition (locked list, etc.) preventing LISTSERV from deleting the user, or because the list was configured with "Subscription= By_Owner" and the owner manually added the victim after the arrival of the undo request.

This mechanism offers a very good degree of protection against the adverse effects that dead "spoofed accounts" can have on the performance of the LISTSERV host system. It does not, unfortunately, mean that people no longer have to fear subscription spoofing, as only LISTSERV lists are monitored and protected by the "spoof detector". Requests to subscribe to lists hosted by other mailing list managers are sent directly to the list managers in question, and LISTSERV can only act on the requests that it does receive.

5.6. Server Registration

5.6.1. Registering LISTSERV Classic Servers

NOTE: This section and 5.6.2, following, do not apply to evaluation kits or to LISTSERV Lite kits. Evaluation copies of LISTSERV should not be registered because they are (presumably) temporary servers running test lists, whose existence should not be broadcast. LISTSERV Lite copies run only in **TABLELESS** mode and therefore cannot be registered in the same manner as LISTSERV Classic, nor may they participate in the LISTSERV backbone. Further, Windows 9x/ME Classic servers may not participate in the backbone as typically they do not have the capacity or stability to qualify for backbone status. For information about how LISTSERV Lite servers are registered, please see 5.6.3.

Also note that only those LISTSERV Classic servers running in **NETWORKED** mode may be registered.

Once the server is ready for production use (that is, once you have installed a permanent License Activation Key, and once you have arranged for LISTSERV to be started automatically when the system boots), you should register it with L-Soft by filling in a server registration form, and returning it to SUPPORT@LSOFT.COM. Registering the server is necessary to broadcast its existence to the other LISTSERV servers. Once you have registered, your server will be sent periodic updates about the lists hosted by other LISTSERV sites and updates to the files whose versions are shown in the output of the **RELEASE** command, among other things, and, similarly, other LISTSERV sites will receive information about the public lists you are hosting.

The LISTSERV site registration request form is found at the URL

http://www.lsoft.com/regform.html

along with the requirements that must be met for registration.

5.6.2. The LISTSERV backbone

The last question on the registration form is whether or not you wish for your site to participate in the LISTSERV backbone.

The LISTSERV backbone is a collection of servers which are operating 24 hours and maintained on a regular basis by their respective operation staffs. This backbone is used to support the DISTRIBUTE command and to host heavy-traffic network-wide peered lists.

LISTSERV servers can fall into one of two categories:

- Local server: A local server has by definition no obligation towards the rest of the network. It can run any release of the code, with or without local modifications. Its operation staff can update it at irregular intervals, place it offline 14 hours a day, or do just anything they might see fit to do. The server will appear in the network routing files but it will be flagged as being a local server.

  The only two restrictions placed on local servers are:

  1. If the server's operation staff encounter a problem with the software and the latest available release of the code has not yet been installed on the server, in general L-Soft support will recommend upgrading to the latest release before trying to diagnose a problem which may have been fixed between releases.
  2. Local servers are not allowed to create peer lists. Note that the term "peer list" should be interpreted as meaning "network-wide public peer list". A closed peer list local to the various nodes of an institution does not fall into that category.

  A local server can create a network-wide list by means of the Mail-Via= DISTRIBUTE feature. Local servers can submit DISTRIBUTE jobs to the backbone, but will not receive any. If a peer sub-backbone is required for the list (e.g. if large archive files are to be made available), the local LISTSERV operations staff should try to find hosts in the backbone or should join the backbone.

- Backbone server: A backbone server can do all of the above, can create peer lists and is supposed to receive DISTRIBUTE jobs. The restrictions placed on the backbone sites are the following:

  1. A backbone server should always be at the latest available level. This means that the operations staff must take whatever steps are necessary to update it in a

timely basis. The average delay should not exceed one week, with the deadline being two weeks.

2. A backbone server cannot be placed offline on a regular basis. It must operate 24 hours/7 days. It can of course be placed offline manually under particular conditions, lists can be held by their respective owners, etc.
3. (VM) A backbone server must be AUTOLOG-ed when the system is IPL-ed, and ought to be automatically restarted at regular intervals in case it logs off due to some hardware failure (e.g. paging error). This applies only if such a restart facility is already available at the site hosting the server. In any case, local operators should be able to restart it if they are also able to restart RSCS and other service machines. That is, the host site should do its best to ensure that the server is restarted on a regular basis in case it crashes.
4. (Non-VM) A backbone server must start automatically whenever the system is rebooted, and should have some facility to restart if it crashes during operation. As with VM servers, the host site should do its best to ensure that the server is restarted on a regular basis in case it crashes.
5. A backbone server should have the latest version of **BITEARN NODES**, or in the worst case, the version from the previous month. This applies only if the NODUPD files are received in due time of course. In 1.8c and following releases, **BITEARN NODES** can be updated automatically--see the 1.8c release notes for details.

Sites which are willing to become part of the LISTSERV backbone should indicate it in the **:backbone** tag of the registration form returned to **support@lsoft.com**. However, please note that unless you have a large number of lists, or a number of very large lists, it is probably not necessary for you to join the backbone. Sites running a few small support or hobby lists, for instance, don't need to be on the backbone; sites running hundreds of lists both large and small *do* need to be on the backbone. Also, sites running one or two huge lists (greater than, say, 50K subscribers each) probably should be on the backbone; such sites should contact L-Soft for more information.

5.6.3. Automatic Registration for LISTSERV Lite Servers

LISTSERV Lite servers are registered automatically when you start the software for the first time. This auto-registration is not optional for Free Edition servers, and can only be disabled for non-Free Edition Lite servers by specifying **STANDALONE** runmode (see "**RUNMODE=**" in Appendix C).

The auto-registration allows you to take part in the global List of Lists and CataList services maintained by L-Soft. Registrations are verified on a regular basis by a central L-Soft server, which sends out several informational commands that return non-privileged information about your server (anyone can issue these commands). Since these registrations are maintained by regular communication with your server, please note that, should you decommission the server, registration verifications will continue to be mailed to your server for several days until the central server decides that your server is actually gone, and not simply unable to receive mail for some reason. Please note carefully that it is not possible for L-Soft to stop these registration queries manually even if you write to us and tell us that the server has been shut down permanently. They *will* stop after several days without a response.

5.7. Inter-server Updates

Because networked LISTSERV servers operate as part of a distributed system, they do a certain amount of inter-server "chatting". This "chatting" takes the form of DISTRIBUTE jobs, X-LUPD jobs, X-SPAM jobs, and so forth. Some of the jobs are requests for

statistics for the LISTSERV network; other jobs are updates to the list of lists; still other jobs are spam alerts. *None of these jobs contain privileged or private information;* L-Soft does *not* query your server for licensing information or any non-LISTSERV-related data, and in fact, all data sent out regarding your server can be retrieved by any user with documented LISTSERV commands.

If you are running LISTSERV Classic, and you do not want to participate in the full-fledged LISTSERV network for whatever reason, you can make a change in your site configuration file to run your server in "standalone" rather than "networked" mode. Simply set the **RUNMODE=** variable to the value "**STANDALONE**" and stop and restart your server (see Appendix C for the syntax applicable to your OS). Note that this will remove your server and all otherwise-public lists running on it from the CataList and the global List of Lists.

LISTSERV Lite (Free Edition) sites are not allowed to change their runmode. If this is a security issue for your site, L-Soft suggests purchasing either the commercial edition of LISTSERV Lite or LISTSERV Classic and running in "standalone" mode.

5.8. Setting up archive and notebook directories for use with LISTSERV

We have found that often people get confused about the difference between the directories where the mailing list's notebook archives are kept and the directories where the mailing list's web archive interface files are kept. Here are a few guidelines:

L-Soft's STRONG RECOMMENDATION is that each list be given a separate directory in which its notebook archives and any files available via LISTSERV's file server are kept. On VM this is not always practical, but the security concerns are different and (to date) the 'wa' interface is not available in any case. For OpenVMS, unix, and Windows systems, our STRONG RECOMMENDATION is that a separate directory tree be established for the purpose of storing list notebook archives and other related files. For instance, you might create

On OpenVMS:	Where LISTSERV's base directory is	**LISTSERV_ROOT:[MAIN]**
	Create the directory	**LISTSERV_ROOT:[LISTS]**
On unix:	Where the LSVROOT directory is	**/home/listserv**
	Create the directory	**/home/listserv/lists**
In Windows:	Where LISTSERV's base directory is	**C:\LISTSERV**
	Create the directory	**C:\LISTSERV\LISTS**

Then, under the main "lists" directory, you would create further subdirectories, one for each list that has archives.

For OpenVMS this would be something like **LISTSERV_ROOT:[LISTS.MYLIST-L]**
for unix something like **/home/listserv/lists/mylist-l**
and for Windows something like **C:\LISTSERV\LISTS\MYLIST-L**

Please note carefully that under unix, the directory path specification for notebook archives MUST IMPERATIVELY be in lower case. LISTSERV will not write notebooks to directory paths specified in upper- or mixed-case.

Where you locate list archives is particularly important with regard to LISTSERV's file server functions. You MUST NOT set up a file server sub-catalog entry in **site.catalog** that points to LISTSERV's A directory! The catalog owner will be given

FULL ACCESS to all the files in the directory you specify in the sub-catalog entry. Therefore in order to maintain security you MUST make a separate directory for each list catalog you define in **site.catalog**. For simplicity's sake, it is generally best to use this directory for the list's notebooks as well as any files the list owners may want to store *except* for the web archive interface files.

Files generated by LISTSERV for the web archive interface MUST NOT be stored in the notebook directories (or vice versa). You MUST make a separate directory tree where the HTML files and index files for the 'wa' interface are kept. Our best recommendation is to place this directory tree under your web server's document root directory, so that the HTML files for the web archive interface are reached by using a URL such as **http://yourserver/archives/** . The location of this directory naturally varies from platform to platform and web server to web server; the guidelines in 5.4.5, above, will give you a start on finding this location.

5.9. DBMS and Mail Merge Functions

For information on installing and using these functions, please refer to the *Developer's Guide for LISTSERV*, available separately.

5.10. Synonymous host name registration via ALIASES NAMES

LISTSERV has supported hostname aliases since BITNET added support for this function in 1990. You could define that BITNET node (say) VTVM1 was the same as VPIVM1 and VPIVM2 (old names) and was also known as VTVM1.CC.VT.EDU. Since this was designed into the first major rewrite of LISTSERV, it is very efficient and there is almost no performance penalty. It also works globally, i.e., once the equivalence is defined, it works for all lists and all users.

However, the Internet did not have a similar mechanism for registering aliases, so this function was only useful to BITNET sites, although the underlying code would also have worked with Internet aliases if there had been a way to register them.

LISTSERV 1.8d and later supports a centralized list (called ALIASES NAMES) of synonymous Internet hostnames. The main purpose is to address problems with ISPs where the "From:" line is rewritten from (say) "joe@isp.net", which is what Joe wanted, to "joe@alpha.isp.net", "joe@beta.isp.net", "joe@gamma.isp.net" and so forth, where "alpha", "beta" and so on are known machine names (with the understanding that they may add machines from time to time) and "joe" is the same in all cases. In this way it is possible for LISTSERV to match joe@alpha.isp.net with his actual subscribed address of joe@isp.net or any of the other cluster hosts in his domain.

This can also handle a situation where an ISP changed name and for instance "joe@oldname.net" is rewritten to "joe@newname.net". However this does not handle "joe@isp.net" -> "J.Doe@isp.net" and the like.

Requests for additions to ALIASES NAMES are handled by a web form:

http://www.lsoft.com/regaliases.html

Note that while it is possible to add entries to your local copy for your local host, you should NOT do this as they will not be propagated through the network and will simply be overwritten by the next update.

5.11. Real-Time Anti-Virus Scanning

This feature is not available in LISTSERV Lite.

LISTSERV Classic or LISTSERV Classic HPO running on Windows NT/2000 or Linux with current maintenance is required. Other OS platforms may be supported in the future.

LISTSERV 1.8e introduces a long-awaited new feature: real-time, on-the-fly anti-virus scanning of all messages sent to LISTSERV mailing lists. All parts of such messages, including inline uuencoded binaries and MIME attachments in those messages, are scanned and bounced back to the sender if viruses are present. Messages sent to the **`*-request`** and **`owner-*`** pseudo-mailbox addresses used by LISTSERV (see 17.3.4, below) are also scanned and discarded if they contain viruses. This includes mail sent to the **`listserv-request`** and **`owner-listserv`** pseudo-mailboxes.

This is a value-added feature which, in addition to a regularly-licensed LISTSERV Classic or LISTSERV Classic-HPO installation, requires the following:

1. A "maintenance" LAK in addition to your regular LAK, meaning that you must purchase maintenance (which includes automatic anti-virus signature updates for the term of the LAK) for LISTSERV in order to use the feature; and

2. A special LISTSERV-specific version of F-Secure Anti-Virus, which is available for download from L-Soft's web site. (The standard version of F-Secure Anti-Virus is not supported.)

The anti-virus scanning feature includes:

- A 45-day warning when maintenance/AV support is about to expire.

- A 5-day warning when virus databases have not been updated.

The above warnings are controlled by the site configuration variable **`LICENSE_WARNING=`** as usual. And, as usual, it is not recommended to disable the license warnings as you may miss an expiration date without any warning and/or your anti-virus signature databases may not be kept up to date without your knowledge.

There is a new site configuration variable, **`ANTI_VIRUS=`** , which defaults to 1 (enabled) if the supported AV system is detected and 0 (disabled) if it is not. Manually enabling the variable is ignored if the supported AV system is not detected.

The virus checking capability is enabled if (1) the supported AV system is present, (2) a maintenance LAK is loaded and not expired, and (3) **`ANTI_VIRUS=0`** is not specified in the site configuration file. List owners may not turn off AV checking (design decision -- security). Messages containing viruses are always returned to the sender (design decision -- the sender ought to be warned) even if filtering is otherwise enabled. However, attachments which have been filtered are not scanned for viruses (they are simply discarded).

6. LISTSERV Commands

(For a quick reference card of LISTSERV commands, see Appendix A of this document.)

This chapter is divided into five parts. The first three correspond to those commands available for use by the general user, list owners and file owners, and the LISTSERV maintainer. The last two describe how to send commands to LISTSERV and how to register LISTSERV passwords. Non-privileged users can send commands by mail or by interactive commands. (Note that interactive commands can only be sent if a two-way NJE or MSGD connection exists.) Privileged users can send commands by mail, interactive commands (subject to the same restriction previously noted) or via the console (VM) or the LCMD utility (non-VM).

Unless otherwise noted, commands are listed in alphabetical order, with the minimum acceptable abbreviation in capital letters. Angle brackets are used to indicate optional parameters. All commands which return a file accept an optional '**F=*fformat***' keyword (without the quotes) that lets you select the format in which you want the file sent; the default format is normally appropriate in all cases. Some esoteric, historical or seldom-used commands and options have been omitted.

Note that some commands are not available on all platforms; these commands are marked appropriately.

Continuation cards (see Chapter 2 in the *Developer's Guide for LISTSERV* regarding LISTSERV's Command Jobs Language) can be used to split long commands (for instance, ADD commands for users with long X.500 addresses) into two or more 80-character cards. In that case you must insert "// " (two slashes followed by a space) before the command text and a comma at the end of each line of the command so that CJLI considers it as a control card and performs the required concatenation. For instance,

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

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

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

Be sure to put a space before the comma at the end of the first line, as LISTSERV will not add the space for you.

6.1. General Commands

6.1.1. List subscription commands (from most to least important)

SUBscribe *listname* [*full_name* | ANONYMOUS] [WITH *options*]

The **SUBscribe** command is LISTSERV's basic command, issued by users to join mailing lists. This command can also be used to change one's "full_name" field in LISTSERV's SIGNUP database (simply reissue the command with the changed name). Note that the ***full_name*** is not required if the user has previously signed up to lists on the same LISTSERV server, or if the user has previously registered in LISTSERV's SIGNUP database by using the **REGISTER** (q.q.v.) command.

LISTSERV 1.8c and later supports the following syntax:

SUBSCRIBE *listname* ANONYMOUS

This indicates that the user wishes to join the list anonymously, that is, without specifying a name. The **CONCEAL** subscription option is automatically set, granting the subscriber the maximal level of protection available.

LISTSERV 1.8d and later supports the following additional syntax:

SUBSCRIBE *listname full_name* WITH *option1 option2 ...*

This syntax allows you to "preset" subscription options at subscribe time. For instance, you might want to subscribe to MYLIST-L in order to be able to search its archives, but don't want to receive postings. You would use the command

SUBSCRIBE MYLIST-L Joe User WITH NOMAIL

Or you might want to receive individual postings with the **SUBJecthdr** option and receive copies of your own postings instead of the standard acknowledgement that your message was distributed to the list:

SUBSCRIBE MYLIST-L Joe User WITH SUBJecthdr REPRO NOACK

LISTSERV 1.8e and later supports the following additional syntax:

QUIET SUBSCRIBE *listname full_name* WITH *option1 option2 ...*

This syntax suppresses the command response normally sent by LISTSERV that looks like this:

```
Confirming:
> SUBSCRIBE MYLIST-L Joe User
You have been added to the MYLIST-L list.
```

JOIN *listname* [*full_name* | ANONYMOUS]

JOIN is a synonym for **SUBscribe**.

SIGNOFF *listname*|*|* [(NETWIDE]

The **SIGNOFF** command allows the user to cancel his or her subscription to lists. **SIGNOFF** requires a qualifying parameter, as follows:

listname	Sign off of the specified list
*****	Sign off of all lists on that server
*** (NETWIDE**	Sign off of all lists in the LISTSERV network

The "*** (NETWIDE**" parameter causes the LISTSERV server to forward a copy of the signoff request to all other registered LISTSERV servers. This is a good option for a user who is changing service providers or otherwise losing a specific address that will not be forwarded. Please note that this parameter will *not* remove the user from non-LISTSERV lists or from LISTSERV lists running on non-registered sites.

LISTSERV will attempt to sign off the address it finds in the RFC822 "From:" line

and will not "fuzzy match" for "similar" addresses.

UNSUBscribe *listname*|*|* [(NETWIDE]

UNSUBscribe is a synonym for **SIGNOFF**.

CHANGE *listname*|* *newaddr*

This form of the **CHANGE** command can be used by any subscriber. It must be sent from the currently-subscribed address and results in an OK confirmation request being sent back to that address. This cookie then MUST be confirmed by the currently-subscribed address, exactly as it was entered, or the command will fail. This is the only case where a 1.8d cookie must be confirmed by a specific address. Note that this assumes that the user still has login access to both addresses, or at least the ability to send mail from the old address.

SET *listname option1* [*option2 ...*]

Allows the user to change his or her subscription options without administrative intervention. The options available to be changed are as follows:

ACK	A mail message acknowledging the receipt and distribution of the user's posting is sent back to the user.
NOACK	No posting acknowledgement is sent. In general, this setting should only be used if the user has also set himself to REPRO, as it is desirable in most cases that some indication of whether or not the posting was received by LISTSERV be sent.
MSGack	An interactive message is sent to acknowledge receipt and distribution. Note that this works only if both the machine running LISTSERV and the user's machine have NJE connectivity (e.g., BITNET). If NJE connectivity is not available on both ends, this option is effectively the same as NOACK.
CONCEAL	Allows the user to be concealed from the REVIEW command. Note that the list owner or LISTSERV maintainer can always get the complete list of subscribers, regardless of this setting.
NOCONCEAL	"Unhides" the user
Files/NOFiles	These options toggle the receipt of non-mail files from the list. Note that this is NJE-specific, and thus obsolete for systems without NJE connectivity, but retained for compatibility.
Mail/NOMail	These options toggle the receipt of mail from the list. Users who will be away from their mail for an extended period of time may prefer to simply turn the mail off rather than to unsubscribe, particularly if subscription to the list is restricted in some way.

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

DIGests/INDex/NODIGests/NOINDex

These options change the format in which list mail is received by the subscriber. **DIGEST** turns on digest mode, in which the subscriber receives a digest of postings at set times dependent on how the "Digest=" keyword of the list is set. **INDEX** turns on index mode, in which the subscriber receives a daily listing of subjects posted to the list, from which he or she may order postings of interest. **NODIGEST** and **NOINDEX** toggle the mode back to individual postings sent as received by LISTSERV. Note that these options are interrelated; setting one will negate another.

REPro/NOREPro — Causes LISTSERV to send you a copy of your own postings as they are distributed. Some users may prefer this behavior to the **ACK** option (see above).

MIME/NOMIME — Toggles MIME options on and off. Currently only digests are available in MIME format. If **DIGEST** mode is set, the user will receive a MIME digest instead of the regular plain-text digest. Note that you must have a mail client that supports MIME digests (Pegasus is one that does) or this setting will do you little good. This option is automatically set at subscribe time for users who send their subscription command using a MIME-compliant agent, unless "Default-Options= NOMIME" is specified for the list.

HTML/NOHTML — Toggle the HTML function for digests and indexes on and off. New in 1.8d.

TOPICS: ALL | [+/-]*topicname*

For lists with topics enabled (see the Topics= list header keyword), subscribe or unsubscribe to topics. For instance, if a list has topics SUPPORT and CHAT, a user could subscribe to CHAT by sending **SET TOPICS +CHAT** . Or the user could unsubscribe to SUPPORT by sending **SET TOPICS -SUPPORT** . Finally, the user can subscribe to all available topics by sending **SET TOPICS ALL** .

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

FULLhdr — "Full" mail headers, (default) containing all of the routing information.

IETFhdr	Internet-style headers.
SHORThdr	Short headers (no routing information).
DUALhdr	Dual headers, useful with PC or Mac mail programs which do not preserve the RFC822 return email address.
SUBJecthdr	"Full" mail headers (like the default) except that setting this option tells LISTSERV to add the list's default subject tag to the subject line of mail coming from the list. (See the listing in Appendix B for "Subject-Tag=" for more information.) Note that if the user is set to SHORThdr (or any other header option other than FULLhdr), LISTSERV will automatically switch the user to FULLhdr, as subject tags require full headers. Under 1.8c subject tags are not generated for messages sent without an RFC822 "Subject:" header; starting with 1.8d a subject tag is generated (for subscribers with the **SUBJecthdr** option set) even if the original message had no "Subject:" header. To turn the subject tagging off, the user simply sends a new **SET** command with any of the other header options (e.g., **SET listname FULLhdr**) and the **SUBJecthdr** option is reset.
FULL822	Essentially the same as "full" mail headers, but with the important difference that the recipient's email address is specified in the "To:" line rather than the address of the list. "**FULL822**" headers should be used with extreme caution, as they cause LISTSERV to create a separate mail envelope with a single RFC821 RCPT TO: for each address so set. *This behavior can significantly affect the performance of both LISTSERV and of your external mail system.*
SHORT822	Essentially the same as "short" mail headers, with the same caveats as noted for **FULL822**.

Note that **FULL822** and **SHORT822** headers should only be used if a specific problem indicates that they might solve the problem. One possible use would be to determine which subscriber from a specific site is causing the site to throw back delivery errors if that site does not specify which RCPT TO: is generating the error. These headers should *never* be used by default.

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

CONFIRM *listname1* [*listname2*]...]]

The **CONFIRM** command should be issued when LISTSERV requests it. A request for **CONFIRM** should not be confused with a "command confirmation request" which requires an "OK" response. The **CONFIRM** command is used in two cases:

- When the list in question requires periodic subscription renewals (controlled by the **Renewal=** keyword). In this case, the amount of time between the request for confirmation and termination of the subscription is controlled by the **Delay()** parameter of the **Renewal=** keyword; the default is seven days.

- When LISTSERV's automatic address probing function fails and you receive a message to that effect. The response time is controlled by the settings of the **Auto-Delete=** keyword for the list in question.

6.1.2. Other list-related commands

INDex [*listname*]

The **INDEX** command sent to LISTSERV without further qualification sends back the contents of the "root" level archive filelist on VM systems (LISTSERV FILELIST) or archive catalog on non-VM systems (SITE.CATALOG plus the contents of SYSTEM.CATALOG).

If the **INDEX** command is sent with the name of a list (e.g., **INDEX MYLIST**) or the name of a special filelist or catalog file (e.g., **INDEX TOOLS** , if TOOLS FILELIST on VM or TOOLS.CATALOG on non-VM exists), LISTSERV sends back the contents of the specified filelist or catalog. Several possibilities exist:

- For mailing lists without an associated filelist or catalog, LISTSERV creates an index "on the fly" containing entries for the accumulated notebook archives for that list. If notebook archives are not enabled for the list, LISTSERV will respond, "This server does not have any file by the name '*listname*.filelist'."

- For mailing lists with an associated filelist or catalog, LISTSERV will append the "on the fly" index of notebook archives to the entries in the associated filelist or catalog. For instance, for a list called MYLIST with associated catalog MYLIST.CATALOG, **INDEX MYLIST** might return:

```
*
* MYLIST FILELIST from LISTSERV@LISTSERV.MYCORP.COM
*
* :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
*
*  The GET/PUT authorization codes shown with each file entry describe
*  who is authorized to GET or PUT the file:
*
*     ALL = Everybody
*     CTL = LISTSERV administrators
*     OWN = List owners
*     PRV = Private, ie list members
*     LMC = LISTSERV master coordinator
*     N/A = Not applicable - file is internally maintained by LISTSERV
*     MSC = Miscellaneous - contact administrator for more information
*
* :::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::::
*
*
* Information files for MYLIST
*
* filename      filetype        GET PUT size (bytes) date       time
* --------      --------        --- --- ------------ ---------- --------
  MYLIST        FAQ             ALL MSC       22,528 1996-02-09 21:30:10
  MYLIST        WELCOME         ALL MSC          279 1998-02-02 09:59:44
  MYLIST        FAREWELL        ALL MSC           92 1998-02-05 11:06:14
```

```
*
* Archive files for the MYLIST list at LISTSERV.MYCORP.COM
* (monthly logs)
*
* filename      filetype        GET PUT size (bytes) date       time
* --------      --------        --- --- ------------ ---------- --------
  MYLIST        LOG9603         LOG OWN        8,668 1998-05-27 15:29:57
  MYLIST        LOG9605         LOG OWN        7,865 1998-06-29 08:43:26
  MYLIST        LOG9606         LOG OWN       17,298 1998-07-23 12:46:20
```

Figure 6.1. Sample output of an **INDEX *listname*** command.

- Lastly, for catalogs or filelists without an associated list, the INDEX command returns only the entries in the catalog or filelist, since there are no associated list archives to be indexed.

Under VM, instead of the size in bytes, three separate VM-specific columns are used. Please note the following definitions for them:

rec -fm	Indicates whether the file is in a fixed or variable record format
lrecl	Logical record length. For a file with fixed record format (F), this is the length of each record. For a file with variable record format (V), this is the maximum record length.
nrecs	Number of records (lines) in the file

Lists [*option*]

Access the global list of lists maintained by LISTSERV. If no options are specified, then LISTSERV returns only local lists, one line per list. The available options are:

Detailed	All local lists, complete with full header information.
Global *xyz*	Only those whose name or title contains 'xyz'
SUMmary [*host*]	Membership summary for all lists on specified host, or the host to which the command is sent if no host is specified
SUMmary ALL	Membership summary for all hosts (long output, send request via mail!)
SUMmary TOTAL	Membership totals only

"Lists Global" without a search string, which returns the entire list of lists, may no longer be issued by general users. If you are asked about this, you should advise users to use the "Lists Global /xyz" format to search the list of lists, or use L-Soft's CataList service at **http://www.lsoft.com/catalist.html**.

"Lists SUMmary", when issued to an unregistered host or to a host running in STANDALONE mode will generate the response "No information available yet - please try again later." because the file required for this function does not exist.

Query *listname*

Query your subscription options for a particular list (use the SET command to change them). Using the "*" wildcard in place of the name of a single list queries subscription options on all lists on the server.

REGister *full_name* | OFF

Register's the user's full name field in LISTSERV's SIGNUP files, or changes the current value of that field. When a user's name is registered, he or she can omit the full name field from subsequent **SUBscribe** requests to that server. Sending "REGISTER OFF" to LISTSERV deletes the user's entry from the SIGNUP file.

REView *listname* [(*options*]

Get information about a list, assuming the list header keyword "Review=" is set appropriately. For general users, **REVIEW** does not return address information about subscribers who are set to **CONCEAL**. The options are:

BY *sort_field*	Sort list in a certain order:
Country	by country of origin (ISO country codes)
Date	by subscription date (newest to oldest)
Name	by user name (last, then first)
NODEid	by hostname/nodeid
Userid	by userid
BY (*sort_field1 sort_field2*)	You can specify more than one sort field if enclosed in parentheses. For instance: **BY (NODE NAME)**
Countries	Synonym of **BY COUNTRY**
Topics	(1.8d and later) Adds a breakdown of subscribers per topic (if **Topics=** is defined in the list header) at the end of the subscriber list. If you just want the breakdown, use **REVIEW *listname* SHORT TOPICS** . This does *not* show topics by individual subscribers (see the **QUERY** command instead). If **Topics=** is not enabled for a given list then this option is ignored.
LOCal	Don't forward request to peers. This is only useful if the list is peered; normally it should not be necessary to issue this option.
Msg	Send reply via interactive messages (BITNET users only)
NOHeader	Don't send list header, just send the subscriber list
Short	Don't list subscribers, just send the header and the membership summary for the list.

Note that you can get a quick read of the number of subscribers on the list by sending the command **REVIEW *listname* SHORT NOHEADER**.

In 1.8d and later list owners and site maintainers may also use the additional option:

ALL	List concealed members (who will show up with "[concealed]" next to their entry) as well as non-concealed members. (NOT available to general users even if **Review= Public**.)

SCAN *listname text*

Scan a list's membership for a name or address. Helpful if a user attempts to send a SET command or an UNSUB command and is informed that their address is not subscribed to the list. At the non-privileged level, this command will show all non-concealed entries that match the search text, assuming that the list is set to "Review= Public".

The following command is available on VM servers only:

Stats *listname* [(*options*]

Get statistics about a list. **NOT AVAILABLE ON NON-VM SERVERS.** This command is VM-specific, and was originally intended to return BITNET data. The single option is:

LOCal	Don't forward to peers

6.1.3. Informational commands

Help

Obtain a list of commonly-used LISTSERV commands. Also explains how to get the comprehensive reference card and tells who the (non-hidden) server manager(s) are.

Info [*topic*|*listname*]

Order a LISTSERV manual, or get a list of available ones (if no topic was specified); or get information about a list. For **Info *listname***, the text in the INFO template form of *listname*.MAILTPL is used; however, if *listname*.MAILTPL does not exist or does not contain an INFO template form, the INFO template form of DEFAULT.MAILTPL is used.

Query File *fn ft* [*filelist*] [(*options*]

(*Available only on VM*) Get date/time of last update of a file, and GET/PUT file access code. The single option is:

Flags	Get additional technical data (useful when reporting problems to experts)

RELEASE

Find out who maintains the server and the version of the software and network data files.

SHOW [*function*]

Display information as follows:

ALIAS *node1* [*node2* [...]]

BITNET nodeid to Internet hostname mapping

DISTribute Statistics about DISTRIBUTE

HARDWare or **HW** Hardware information; what kind of machine is LISTSERV running on?

LICense License/capacity information and software build date

LINKs [*node1* [*node2* [...]]] Network links at the BITNET node(s) in question

NADs [*node1* [*node2* [...]]] Addresses LISTSERV recognizes as node administrators for the specified site(s)

NODEntry [*node1* [*node2* [...]]] BITEARN NODES entry for the specified node(s)

NODEntry *node1 /abc*/xyz* Just the ':xyz.' tag and all tags whose name starts with 'abc'

POINTs [ALL | *list1* [*list2*...]] Graduated (LISTSERV Classic) license point information. This information can help you plan orderly expansion of your site if you are running with a graduated LISTSERV Classic license. Under Lite this command shows Classic point usage.

STATs Usage statistics for the server (this is the default option)

VERSion Same output as **RELEASE** command

If no function is specified, the output is per **SHOW STATS**.

The following options are available for VM servers only:

BITEARN Statistics about the BITEARN NODES file

DPATHs *host1* [*host2* [...]] DISTRIBUTE path from that server to specified host(s)

DPATHs * Full DISTRIBUTE path tree

FIXes List of fixes installed on the server (non-VM see **SHOW LICENSE**)

NETwork Statistics about the NJE network

PATHs *snode node1* [*node2* [...]] BITNET path between 'snode' and the specified node(s)

6.1.4. Commands related to file server and web functions

GET

VM Syntax:
GET *fn ft* [*filelist*] [(*options*] [F=*fformat*] [SPLIT=*integer*]
Non-VM Syntax:
GET *fn ft* [*catalogname*] [([F=*fformat*] [SPLIT=*integer*]]

For non-VM see also below for special TCPGUI parameters.

Order the specified file or package from LISTSERV. The single option is for VM servers only and is:

PROLOGtext xxxx Specify a 'prolog text' to be inserted on top of the file

Examples:

```
GET MYFILE TEXT
GET MYFILE TEXT MYLIST-L
```

Typically the filelist name or catalog name is the same as the name of the list to which the files belong. A password (PW=xxxxx at the end of the command) is required to retrieve any file that does not have a GET FAC of ALL. For more information on FAC (File Access Control) codes, see 8.3.5 of this manual (for VM) or 8.4.1 (for non-VM).

Do not use dots (periods) in the file specification when specifying the filelist or catalog name, as this will result in an error. For instance

```
GET MYFILE.TEXT MYLIST-L
```

will result in an error, whereas

```
GET MYFILE TEXT MYLIST-L
```

will not.

To control the format in which LISTSERV returns the file(s) to you, you can specify the **F=*fformat*** parameter. Supported formats are **Netdata, Card, Disk, Punch, LPunch, UUencode, XXencode, VMSdump, MIME/text, MIME/Appl, Mail**

To split very large files into manageable chunks, you can specify the **SPLIT=*integer*** parameter. The integer value is the size you want the chunks to be generated, in kilobytes. For instance if you were ordering a 2MB notebook log and wanted to break it into 100KB chunks, you would specify **SPLIT=100**. This is handy for people whose mail systems place a limit on the size of an individual mail message that may be received by a given user.

TCPGUI parameters for Change-Logs: Under non-VM LISTSERV 1.8e and later, the following syntax is accepted:

GET *listname* CHANGELOG (MSG [COLUMNS(*colspec1 colfilter1* [*colspec2 colfilter2*[...]])

The design goal was to provide access via the TCPGUI (and thus the web interface) for change-log data. Further information can be found in the *Developer's Guide to LISTSERV*.

GIVE

VM Syntax:	**GIVE *fn ft* [*filelist*] [TO] *userid@host***
Non-VM Syntax:	**GIVE *fn.ft* [TO] *userid@host***
	GIVE *fn ft catalogname* [TO] *userid@host*

(Note: Prior to 1.8d this command is not available on non-VM servers.)

Sends a file stored in a LISTSERV file archive to someone else. For instance, you may want to send LISTSERV REFCARD to a new user. Rather than retrieving LISTSERV REFCARD and then forwarding it to the user, you simply issue a GIVE command to tell LISTSERV to send it directly. Note that the token "TO" is optional. Examples:

For LISTSERV running under VM:

```
GIVE LISTSERV REFCARD joenewuser@hishost.com
GIVE LISTSERV REFCARD TO joenewuser@hishost.com

GIVE README TEXT MYLIST-L joenewuser@hishost.com
GIVE README TEXT MYLIST-L TO joenewuser@hishost.com
```

For LISTSERV running on non-VM hosts there are two syntaxes, depending on whether or not you need to specify a catalog name for the file in question. Note that the only real difference is whether or not you are required to specify a dot between the filename and the extension. Examples are:

```
GIVE LISTSERV.REFCARD joenewuser@hishost.com
GIVE LISTSERV.REFCARD TO joenewuser@hishost.com

GIVE README TXT MYLIST-L joenewuser@hishost.com
GIVE README TXT MYLIST-L TO joenewuser@hishost.com
```

INDex [*filelist*|*catalog*]

Same as **GET xxxx FILELIST**. If no filelist is specified, the default is LISTSERV FILELIST (on non-VM, SITE CATALOG is returned as LISTSERV FILELIST in this case).

PW *function*

Define/change a "personal password" for protecting AFD/FUI subcriptions, authenticating PUT commands, and so on.

ADD *firstpw* Define a password for the first time, or after a **PW RESET**. Requires confirmation via the "OK" confirmation method.

CHange *newpw* [PW=*oldpw*]
Change your existing password. If you do not include your old password for authentication, LISTSERV will require confirmation via the "OK" confirmation method.

REP *password* Starting with 1.8d, this function is a hybrid of "**ADD**" and "**CHange**". If a password does not exist for the user, one will be added. If a password does exist for the user, it will be changed (with confirmation required via the "OK" confirmation method). "**REP**" was added primarily for use by the web archive and

administration interface but can be used in e-mailed **PW** commands as well.

RESET Reset (delete) your password. This function always requires confirmation via the "OK" confirmation method.

SENDme

Same as **GET**

The following commands are available on VM servers only:

AFD

Automatic File Distribution. The functions are as follows:

ADD *fn ft* [*filelist* [*prolog*]]
Add file or generic entry to your AFD list

DELete *fn ft* [*filelist*]
Delete file(s) from your AFD list (wildcards are supported)

List Displays your AFD list

For node administrators:

FOR *user* ADD/DEL/LIST etc
Perform requested function on behalf of a user you have control over (wildcards are supported for DEL and LIST)

FUI

File Update Information: same syntax as **AFD**, except that **FUI ADD** accepts no 'prolog text'

6.1.5. Other (advanced) commands

DISTribute *type source dest* [*options*]

Note: Starting with 1.8d, the ability to send DISTRIBUTE jobs is limited to LISTSERV Maintainers by default, and requires a password. This section is retained for compatibility with 1.8c and earlier, and for 1.8d and later servers which have the DISTRIBUTE security feature turned off.

Distribute a file or a mail message to a list of users (see the *Developer's Guide for LISTSERV* for more details on the syntax). The various parameters are, briefly:

Type:

MAIL Data is a mail message, and recipients are defined by '<dest>'

MAIL-MERGE Data is a mail-merge message. See the *Developer's Guide for LISTSERV* for specifics.

FILE Data is not mail, recipients are defined by '<dest>'

RFC822 Data is mail and recipients are defined by the RFC822 'To:'/'cc:' fields

Source:

DD=*ddname* Name of DD holding the data to distribute (default: '**DD=DATA**')

Dest:

<TO> *user1* <*user2* <...>>
List of recipients

<TO> DD=*ddname*
Use a DD called *ddname* for the destination addresses, one recipient per line

Options for the general user:

ACK=NOne/MAIL/MSG
Acknowledgement level (default: **ACK=NONE**)

CANON=YES 'TO' list in 'canonical' form (**uid1 host1 uid2 host2...**)

DEBUG=YES Do not actually perform the distribution; returns debug path information

INFORM=MAIL Send file delivery message to recipients via mail

TRACE=YES Same as **DEBUG=YES**, but file is actually distributed

AV=YES[,FORCE] (1.8e Classic and later) Check the message for viruses. See the *Developer's Guide for LISTSERV* for specifics.

Options requiring privileges:

FROM=*user* File originator

FROM=DD=*ddname* One line: 'address name'

GETPost *listname post_number* [*post_number* [...]] [NOMIME]

GETPost is used after receiving the output of a **SEARch** command to retrieve the postings you want from the **SEARch** output. For instance, if you want postings numbered 1730, 1731, 1732, and 1840 from the MYLIST list, send the command

```
GETPost MYLIST 1730-1732 1840
```

GETPost is analogous to the VM database command **PRINT**.

In previous versions, the **GETPost** command returned messages that contained MIME attachments in their "raw" form, which could not be extracted automatically by MIME-aware mail clients. Customers who wished to use list notebooks to archive word-processing documents (for instance) found this to be a problem. From LISTSERV 1.8e, attachments returned in messages by way of the **GETPost** command will now display as inline clickable links in the individual messages.

Users of certain email clients (specifically Pine, which handles attachments in a secondary viewing area) may find the new format difficult to use. If preferred, the

pre-1.8e behavior may be reverted to by specifying "NOMIME" as the last parameter of the **GETPost** command.

Search

For lists running on VM servers, see also below at **DATABASE**.

The **Search** command syntax is similar to that of the **SEARCH/SELECT** commands in the "old" database functions. A very basic **Search** command for list MYLIST would look like this:

Search *search_string* IN MYLIST

You can also restrict your search by date, sender, or other criteria, for example,

Search *search_string* IN MYLIST SINCE 96/01/01
Search *search_string* IN MYLIST WHERE SENDER CONTAINS ERIC

etc. The specific syntax is outlined in LISTDB MEMO (available from LISTSERV with the command "INFO DATABASE") and in the *Developer's Guide for LISTSERV*. Note that the new **Search** command does not require a CJLI job framework to operate; simply send the **Search** command in the body of an email message to the appropriate server. LISTSERV will respond with an index of the postings matching your criteria and instructions on how to use the **GETPost** command to retrieve the posts you want.

SERVE *user*

Restore service to a user whose access to LISTSERV has been disabled. This generally occurs when a user has sent 51 incorrect commands in a row to LISTSERV, which LISTSERV interprets as a possible mail loop. (Note also that certain mail packages that send "Read:/Not Read:" notifications back to LISTSERV will trigger this scenario after 51 iterations. The best solution would be for the user to disable receipt notifications.) The user in question cannot restore his or her own service; this command must be issued from another userid. Note that if the user has been manually served out by privileged user (a LISTSERV maintainer), the **SERVE** command must be issued by a similarly-privileged user (who must also be a LISTSERV maintainer, although naturally the same user who issued the **SERVE OFF** command can issue the **SERVE** command). For 1.8d and later please note that the **THANKs** command will *not* reset the serve-off counter (so vacation messages or auto-replies that contain a sentence starting with something like "Thanks for writing" will not defeat the system and users sending them will eventually be served off instead of continuing to loop ad infinitum).

THANKs

You can send this command to check to see if the server is alive. If it is, the server politely responds, "You're welcome!".

The following commands are available only on VM servers:

DATAbase *function*

Access LISTSERV database(s). The functions are explained in detail in the

version of LISTDB MEMO available from VM servers, but the basic syntax is:

Search DD=*ddname* <ECHO=NO>
: Perform database search (see the VM version of LISTDB MEMO for more information on this)

List
: Get a list of databases available from that server

REFRESH *dbname*
: Refresh database index, if suitably privileged

Dbase

Same as **DATABASE**

6.2. List Owner and File Owner Commands

6.2.1. File management commands (for file owners only)

PUT *fn ft* <*filelist* <NODIST>>

Update a file you own. Options are:

<**PW=*password***>
: Supply your password for command authentication

The following options are VM-specific and will not work on the non-VM servers.

The "NODIST" option prevents AFD and FUI distributions when the file is updated. Other available VM only options include:

<**CKDATE=NO**>
: Accept request even if the current version of the file is more recent than the version you sent

<**DATE=*yymmddhhmmss***>
: Set file date/time

<**RECFM=F** <**LRECL=*nnn***>>
: Select fixed-format file (not to be used for text files).

<**REPLY-TO=*userid***>
: Send reply to another user

<**REPLY-TO=NONE**>
: Don't send any reply

<**REPLY-VIA=MSG**>
: Request reply via interactive messages, not mail (Requires NJE connectivity)

<"***parameters***">
: Special parameters passed to FAVE routine, if any

Standard parameters supported for all files:

TITLE=*file title*
: Change file "title" in filelist entry

The following commands are available on VM servers only:

AFD/FUI

Automatic File Distribution privileged commands. In addition to the AFD/FUI functions listed above, a file owner may use the following function:

GET *fn ft* <*filelist*>
: Get a list of people subscribed to a file you own

GET *fn* FILELIST <(*options*>

Special options for filelists:

CTL	Return filelist in a format suitable for editing and storing back
NOLock	Don't lock filelist (use in conjunction with CTL)

REFRESH *filelist* <(*options*>

Refresh a filelist you own. The single option is:

NOFLAG	Don't flag files which have changed since last time as updated (for **AFD/FUI**)

UNLOCK *fn* FILELIST

Unlock filelist after a **GET** with the **CTL** option if you decide not to update it after all

6.2.2. List management functions

Commands that support the QUIET keyword are marked (*)

ADD(*) *listname user* [*full_name*]
ADD(*) *listname* DD=*ddname* [IMPORT [PRELOAD]]

The first syntax is used to add an individual user to one of your lists, or update his name field. Note that you can substitute an asterisk ("*****") for ***full_name*** and LISTSERV will substitute "<No name available>" in the list.

The second syntax is used for bulk ADD operations where a dataset (DD=*ddname*) is used add multiple users, one address/name pair per line. For bulk operations you may also use the **IMPORT** option, which implies a **QUIET ADD** (in other words you do not need to specify **QUIET** if you use **IMPORT**) and otherwise vastly speeds up the **ADD** process by loosening syntax checking and omitting success messages. The **IMPORT PRELOAD** option first appeared in 1.8d and is used to direct LISTSERV to preload the existing e-mail keys in memory before starting the transaction, which speeds the operation up considerably. This option is used primarily with DBMS lists to speed up bulk adds. **PRELOAD** is not necessary for traditional LISTSERV lists and does not normally lead to a significant performance improvement. However, when importing a new list (no existing subscribers), it does reduce CPU usage somewhat.

For a bulk **ADD** operation, the users are defined in a separate dataset beginning on the line following the **ADD** command. For instance,

```
ADD listname DD=ddname IMPORT
//ddname DD *
userid@host.com User Name
userid2@host.com User2 Name
... more address/name pairs, one per line ...
/*
```

Please see chapter 7.17, below, for specific instructions for bulk **ADD** operations.

ADDHere(*)

Same as **ADD**, but means "add the user on this peer, do not forward this request to a (possibly) closer peer". For non-peered lists, is functionally identical to **ADD**.

CHANGE(*) listname|* newaddr
CHANGE(*) listname|* oldaddr|pattern newaddr|*@newhost

The first form can be used by any subscriber and results in a cookie being sent to the new address. This cookie MUST be confirmed by the new address, exactly as it was entered, or the command will fail. This is the only case in 1.8d and later where a cookie must be confirmed by a specific address.

The list owner form does not use cookies but simply applies the standard "Validate=" rules (as for a **DELETE** command). You can specify a wildcard pattern for the old address and *@newhost for the new address to rename certain addresses to a new hostname. The CHANGE1 template is sent unless you specify **QUIET**.

Change log entries are made (**CHANGE *oldaddr newaddr***) and there is a CHG_REQ exit point which allows you to reject the operation.

DELete(*) *listname user* [(*options*]
DELete(*) *listname* DD=*ddname* [BRIEF]

The first syntax is used to remove a single user from one of your lists, or from all local lists if *listname* is '*****' The available options are:

Global	Forward request to all peers
LOCal	Don't try to forward request to closest peer if not found locally
TEST	Do not actually perform any deletion (useful to test wildcard patterns)

The second syntax is used for bulk **DELETE** operations (similar to a bulk **ADD** operation). See chapter 7.17 of this manual for details. The single available option is:

BRIEF	Good for deleting wildcard patterns (such as *****@*****) when you don't want a "userid@host has been deleted from list xxxx" for each user deleted. Returns instead only a count of the users that were deleted.

FREE *listname* <(*options*>

Release a held list. The single option is:

Global	Forward request to all peers

GET *listname* <(*options*>

Get a copy of a list in a form suitable for editing and storing the list and lock it so that other list owners can't modify it until you store it back (or until you or they issue an **UNLOCK** command). The options are:

Global	Forward request to all peers
HEADer	Send just the header; on the way back, only the header will be updated. This is the recommended way to

modify your list header.

NOLock	Do not lock the list
OLD	Recover the "old" copy of the list (before the last **PUT**)

HOLD *listname* <(*options*>

Hold a list, preventing new postings from being processed until a **FREE** command is sent. The single option is:

Global	Forward request to all peers

Lists [*option*]

Additional options available for list owners and moderators:

OWNed	Returns a list of local lists owned by the invoker.
MODerated	Returns a list of local lists that are moderated by the invoker.

MOVE(*) *listname user* <TO> *node*

Move a subscriber to another peer. Do NOT use this command to move users from one list host site to another during migration. It is strictly for moving subscribers from one peer to another peer.

listname* DD=*ddname	Move several subscribers to various peers

PUT *listname* LIST

Update a list from the file returned by a GET command. This is the standard "PUT command" or "list PUT" referred to throughout this document.

Starting with LISTSERV 1.8d, use of the PUT command to store a list header with new subscriber data at the bottom (e.g., an attempt to add subscribers "on the fly") will result in only the header of the list being stored, and in the generation of the following warning:

```
WARNING: new subscriber data was found in the replacement list you sent,
possibly due to the use of a signature file with an unusual separator line.
If you really meant to update the subscriber data, please resend your
request with the word "PUT" replaced with "PUTALL". For now, only the list
header will be updated.
```

PUTALL *listname* LIST

Starting with 1.8d, this command allows you to **PUT** an entire list file, that is, the list header followed by the list of subscribers.

Documented restriction: **PUTALL** does NOT work with DBMS lists; only the header information is replaced. Subscriber information in the DBMS table is not changed. For DBMS lists where the subscriber information needs to be replaced *in toto*, either the DBMS should be manipulated with your regular DBMS tools or you should use **ADD IMPORT** .

Query *listname* <WITH *options*> FOR *user*

Query the subscription options of another user (wildcards are supported).

*** <WITH *options*> FOR *user*** Searches all the lists you own for the specified user(s) with the specified option(s).

SET(*) *listname options* <FOR *user*>

Alter the subscription options for other users (wildcards are supported when setting options for another user or set of users).

Additional options for list owners:

NORENEW/RENEW	Waive subscription confirmation for this user
NOPOST/POST	Prevent user from posting to list
EDITor/NOEDITor	User may post without going through moderator
REView/NOREView	Postings from user go to list owner or moderator even if user is otherwise allowed to post

UNLOCK *listname*

Unlock a list after a GET, if you decide not to update it after all, or unlock a list if it has been locked by another list owner or by the LISTSERV maintainer. Note that if you are not the person who originally locked the list, it is considered good practice to ask the person who originally locked the list whether or not they are done with the list before you unlock it.

The following commands are available only on VM servers:

EXPLODE *listname* <(*options*>

Examine list and suggest better placement of recipients, returning a ready-to-submit **MOVE** job.

BESTpeers *n*	Suggest the N best possible peers to add
Detailed	More detailed analysis
FOR *node*	Perform analysis as though local node were '***node***'
PREFer *node*	Preferred peer in case of tie (equidistant peers)
SERVice	Check to see that service areas are respected
With(*node1 <node2 <...>>>*)	Perform analysis as though specified nodes ran a peer
WITHOut(*node1 <node2 <...>>>*)	Opposite effect

Stats *listname* (RESET

Resets (BITNET) statistics for the list.

6.3. LISTSERV Maintainer Commands

All LISTSERV maintainer commands require a password for validation when issued by email. Commands issued by TELL or SEND from the local host or via the LCMD utility do not require password validation. (Commands issued by LCMDX *do* require password validation. LCMDX, the LISTSERV TCPGUI demonstration program, is not the same as the LCMD utility shipped with LISTSERV.)

FOR *user command*

Execute a command on behalf of another user (LISTSERV maintainers only). Note that this command is provided for debugging purposes only -- it provides a method for a LISTSERV maintainer to send commands "from" the specified user. It is not recommended to use this command syntax in production, for instance to issue SET or SUBSCRIBE or UNSUBSCRIBE commands on a user's behalf. For instance, the LISTSERV maintainer should use, respectively, the "SET l*istname options* FOR *userid@host*", "ADD *listname userid@host*", or "DELete *listname userid@host*" syntaxes in preference to the "FOR *userid@host command*" syntax.

Lists [*option*]

Global All known lists, one line per list, sent as a (large!) file. Only LISTSERV maintainers may request this list, as it has become a favorite pastime of Internet mailbombers to issue **LIST GLOBAL** commands on behalf of users whose mailboxes they wish to bomb. You should direct users who request "the whole list of lists" to L-Soft's CataList service at the WWW URL **http://www.lsoft.com/catalist.html**.

Starting with 1.8e, additional options available for site maintainers are

OWNED BY *internet_address* Returns a list of local lists owned by the userid@host specified. Wildcards are acceptable.

MODerated BY *internet_address* Returns a list of local lists moderated by the userid@host specified. Wildcards are acceptable.

NODESGEN [WTONLY]

Regenerate all LISTSERV network tables, or just compile the links weight file (debugging command). This happens automatically when LISTSERV is rebooted if a new BITEARN NODES file is found. Otherwise you should issue a NODESGEN whenever you update BITEARN NODES.

PUT *listname* LIST

Create a new list. Requires the **CREATEPW** for validation when issued from a remote node. You may specify initial subscribers, one per line, following the list header when creating a list. See also the **PUTALL** command at 6.2.2.

PWC *function*

Password file management:

ADD *user newpw*	Define a password for the specified user
DELete *user*	Delete password for that user
Query *user*	Query the password of the specified user

REGister *name*|OFF FOR *user*

Set or delete a user's SIGNUP FILE entry

SERVE *user* OFF [DROP] | LIST

SERVE *user* OFF permanently suspends access from an abusive user or gateway (restore service with **SERVE *user***).

The following options were added in LISTSERV 14.3:

Adding "DROP" (for example, **SERVE *user* OFF DROP**) to the command is identical to **SERVE *user* OFF** except that the postmaster will not receive any notification messages from LISTSERV when/if the user continues to try to post.

Issuing a **SERVE LIST** command causes LISTSERV to return a list of all users who are currently served off or who are spam-quarantined. For instance,

```
> serve list
JOE@EXAMPLE.COM                    DROP 2003-08-20 15:51:20 by nathan@EXAMPLE.COM

FOOBAR@EXAMPLE.EDU                 HARD 2003-04-07 14:55:29 by NATHAN@EXAMPLE.COM

BLAB@FOO.EXAMPLE.COM               SOFT 2004-09-14 10:53:18

SPAMMER@SPAMDOMAIN.COM             SPAM 2003-08-20 15:50:55

4 matching entries.
```

SHUTDOWN [REBOOT|REIPL]

Stop the server. *Under VM and OpenVMS only,* **REBOOT** or **REIPL** are also allowed as options to the **SHUTDOWN** command. Under unix and Windows, the **REBOOT** or **REIPL** feature is *not* available and these options, if issued, are ignored and the server is simply shut down, requiring a manual restart.

STOP

Same as SHUTDOWN

The following commands are available only on VM servers:

CMS *command_text*

Issue a CMS command and get the last 20 lines of response sent back to you, the rest being available from the console log

CP *command_text*

Issue a CP command and get up to 8k of response data sent to you (the rest is lost)

DATAbase *function*

Control operation of databases:

DISAble	Disable interactive database access, without shutting down existing sessions
ENAble	Re-enable interactive access

SHUTDOWN	Shut down all interactive database sessions, and disable interactive access

INSTALL *function*

Software update procedure (LISTSERV-NJE only):

CLEANUP *shipment*	Remove an installed shipment from the log
CLEANUP BEFORE *dd mmm yy*	Remove all shipments installed before that date
PASSWORD *shipment* PW=*instpw*	Confirm installation of a shipment, when requested by LISTSERV
RELOAD *shipment*	Attempt to reload a shipment which failed due to a disk full condition
STATus	Get a list of installed "shipments"

OFFLINE

Suspend processing of reader files and disable the **GET** command

ONLINE

Cancel **OFFLINE** condition

PUTC *fn ft* <*fm*|*cuu*|*dirid*> <RECFM=F LRECL=*nnn*>

Update a CMS file on one of LISTSERV's R/W minidisks; note that this is similar to SENDFILE + RECEIVE or LINK + COPYFILE and should NOT be used to update file-server files

SENDFile *fn ft* <*fm*|*cuu*|*dirid*>

Request the server to send you a file from one of its disks

SF

Same as **SENDFILE**

SHOW <*function*>

In addition to the standard **SHOW** functions available on other servers, VM servers support the following functions:

BENCHmarks	CPU/disk/paging benchmarks
EXECLoad	Statistics about EXECLOADed REXX files
LSVFILER	Statistics about LSVFILER file cache
PREXX	Statistics about PREXX functions usage
STORage	Information about available disk space and virtual storage

SHUTDOWN <REBOOT|REIPL>

Stop or reboot the server (the two options are synonyms meaning to restart the

server after shutting it down). **REBOOT** or **REIPL** are also allowed as options to the **SHUTDOWN** command under OpenVMS, but are not available under unix or Windows.

Note: some debugging commands and options have been omitted.

6.4. Sending commands to LISTSERV

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

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

6.5. Defining Personal Passwords

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

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

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

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

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).

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

7. Creating and Maintaining Lists

You can create and maintain lists from any userid listed in the POSTMASTER keyword of LISTSERV's site configuration file. Note that a LISTSERV maintainer has the authority to GET and PUT any list, filelist, catalog, or archive file on the server (although for any list not set to "Send= Public", the LISTSERV maintainer must be subscribed to the list in order to post to it, and must additionally be a list Editor if the list is set to "Send= Editor...").

7.1. Basic list creation

At its simplest, creating a list is a matter of setting certain keywords to desired values in a file (called the "list header file") and storing the file in a place where LISTSERV can find it. The format of a typical list header file is relatively free-form, with only a few basic rules:

1. All header lines (including those inserted for "white space") must begin with the character "*" (ASCII 0x2A).
2. Header lines can be up to 100 characters long (including the initial "*" character). However in practice you will probably want to limit them to no more than 80.
3. All words ending with the character "=" (ASCII 0x3D) are evaluated as keywords.
4. The first non-"white space" line of the header file is evaluated as the descriptive name of the mailing list, and will be displayed as such by the **LIST** command.

Additionally, for **PUT** operations, you must add a line of the format

PUT *listname*.LIST PW=*password*

to the top of the file before mailing it. This **PUT** line does *not* begin with an asterisk. (Note that the filename for the list can be either in the format ***listname* LIST** or ***listname*.list** . The "." character is not necessary, but the word **LIST** is *always* necessary.)

Here is a sample of a basic list header with its **PUT** command at the top:

```
PUT SAMPLE LIST PW=CCCCCCCC
*
* Title of sample LISTSERV list
*
* Review= Public      Subscription= Open          Send= Public
* Notify= Yes         Reply-to= List,Respect      Validate= No
* Notebook= Yes,A,Monthly,Public
*
* Owner= someone@somewhere.com
*
```

Figure 7.1. A sample list header.

The preferred method of creating a new list is as follows:

1. Using a text editor, prepare a "list header", for instance using the sample in figure 7.1. You can also get the header of an existing (L-Soft) LISTSERV list and use it as a sample.

2. The first line of the list header MUST be as follows:

PUT LISTNAME.LIST PW=CCCCCCCC

Replace "**LISTNAME**" with the name of your list, for example,

PUT MYLIST-L.LIST PW=CCCCCCCC

Then replace "**CCCCCCCC**" after "**PW=**" with the value of "**CREATEPW=**" in your site configuration file. If your **CREATEPW** is **FIATLUX**, then your complete **PUT** line for a list called MYLIST-L would be as follows:

PUT MYLIST-L.LIST PW=FIATLUX

Note that one of the most common errors made by new LISTSERV users is to leave out the "**.LIST**" part of the **PUT** command. If you leave this part out, LISTSERV will bounce the header back to you with the comment that it does not have any file by the name "**MYLIST-L PW=FIATLUX**".

3. Following the PUT line, you insert as many "list header" lines as you need (see the sample). Each of these lines MUST begin with an asterisk in column 1, for example,

   *** Notebook= Yes,C:\LISTS\PUBLIC,Monthly,Public**

   If your mail software indents paragraphs by default, you must turn off paragraph indentation, or an attempt to store the list will be returned to you with a message that there did not appear to be any list header lines.

   Each "list header" line contains information needed by LISTSERV to operate your list. Most of this information is provided by you in the form of values for standard keywords. You can use the sample header provided above as an example; a complete list of keywords recognized by LISTSERV along with descriptions of their functions can be found in Appendix B of of this manual.

4. Mail the resulting file to the LISTSERV address.

   The "LISTSERV address" is the address formed by "LISTSERV@" + the value you defined in the site configuration file for NODE=. For instance, if you defined NODE=XYZ.COM, the LISTSERV address would be LISTSERV@XYZ.COM.

   This mail must be sent as Internet mail from a username defined as a "postmaster" in the LISTSERV configuration. For instance, from a VMS™ system, you would save your list file (say, in a file called 'newlist.create'), and then do:

   ```
   $ mail
   MAIL> send newlist.create
   To: in%"listserv@xyz.com"
   Subj:

   MAIL>
   ```

   Or, from a unix® system:

   ```
   $ mail listserv@xyz.com < newlist.create
   ```

   On a PC, you would use your POP client or other GUI-based mail program. Make sure to cut+paste the file via the Clipboard and not send it as an "attachment" or use

drag and drop. "Attachment" mechanisms are often proprietary or PC-specific and cannot be guaranteed to work. Sending plain text pasted from the Clipboard always works.

The above is the preferred method for creating and editing list headers. LISTSERV will respond with a report that either the list has been successfully created or that various problems (fatal and/or non-fatal) have been detected. If only non-fatal problems are detected, the list will be stored anyway (non-fatal problems include no list password having been defined). Any fatal problem detected will abort the storage operation.

A less-desirable method of creating lists is to copy the list header file into LISTSERV's main directory and restart LISTSERV. LISTSERV will log a message to the effect that the list is not formatted properly and will then reformat the list. This assumes that the list header has been constructed properly and that there are no errors in the file that will cause LISTSERV to crash or to reject the list file. ***This method is useful only for creating lists; never attempt to edit a production list file in place and restart the server. The GET and PUT operations are the only supported methods for editing list files. Particularly under unix and Windows, LISTSERV will not always accept the edited list file because some editors will insert control characters or CR-LF combinations that LISTSERV cannot parse. Under VM or VMS, it is always possible that hand-editing the list will introduce some sequence that will cause an operational error. L-Soft suggests that this method be used sparingly, if at all, and does not support it. The first method is always preferable to the second.***

BITNET users may also use the LSVPUT utility to store lists on BITNET-connected servers. However, LSVPUT is not documented here as the number of sites with BITNET connectivity is dropping rapidly and fewer and fewer users will be using LSVPUT.

7.2. Architecture-Specific Steps for List Creation

7.2.1. Unix: Creating required Sendmail aliases

This section is for use only by Unix sites running Sendmail.

Please note that the file you need to edit in this step, and the commands you need to issue will require root privileges. Also, while the procedure for manually modifying the sendmail aliases file is described below, you can also enter "**`make list name=`*`listname`***" (where ***`listname`*** is the name of the list) to have the installation program complete this step automatically. The automated procedure assumes that your sendmail stores aliases in the file **`/etc/aliases`**, that the "**`newaliases`**" command will rebuild the aliases database, and finally that "**`kill -HUP `cat /etc/sendmail.pid`**" will cause Sendmail to read in the updated alias list.

LISTSERV accepts and responds to several e-mail addresses. Even before you setup mailing lists, mail sent to listserv and owner-listserv should be handed to LISTSERV (see the installation guide for details). The link between LISTSERV and your mail system is the lsv_amin program. If you are running Sendmail, the best way to route incoming mail to lsv_amin is by adding entries to your "aliases" file. Refer to the manual pages for sendmail on your system if you are not sure where the alias file is stored. On many systems the file will be called **`/etc/aliases`**.

Once you have constructed a list header file, and sent it to your Unix LISTSERV server, you need to instruct your mail system to route mail for that new list to the LISTSERV mail interface. That involves adding entries to your aliases file, much as you did when

installing the server itself. For each new list, you'll need to add eight entries to the aliases file. The format of those lines is as follows,

```
NAME: "|/BBB/lsv_amin /SSS NAME"
owner-NAME: "|/BBB/lsv_amin /SSS owner-NAME"
NAME-request: "|/BBB/lsv_amin /SSS NAME-request"
NAME-search-request: "|/BBB/lsv_amin /SSS NAME-search-request"
NAME-server: "|/BBB/lsv_amin /SSS NAME-server"
NAME-signoff-request: "|/BBB/lsv_amin /SSS NAME-signoff-request"
NAME-subscribe-request: "|/BBB/lsv_amin /SSS NAME-subscribe-request"
NAME-unsubscribe-request: "|/BBB/lsv_amin /SSS NAME-unsubscribe-request"
```

where "**NAME**" is the name of the mailing list, "**/BBB**" in the directory where the mail interface was installed (BINDIR in the Makefile), and "**/SSS**" is the LISTSERV spool directory (LSVSPOOL in the Makefile). Note that "**/SSS**" can be either:

- An explicit directory definition, for example, **/var/spool/listserv** ; or
- The switch **-t** , which is equivalent to the value in LSVSPOOL. (Note that the "**make list**" command makes aliases using **-t**.)

Note: If you use the precompiled copy of **lsv_amin** from the distribution kit rather than compiling your own from the source at install time, you **cannot** use the **-t** switch because the LSVSPOOL value is not compiled into the precompiled program.

For example, assuming the default values were chosen for BINDIR and LSVSPOOL, the aliases for a new list called "mylist" (using the **-t** option) would be,

```
mylist: "|/usr/local/bin/lsv_amin -t mylist"
owner-mylist: "|/usr/local/bin/lsv_amin -t owner-mylist"
mylist-request: "|/usr/local/bin/lsv_amin -t mylist-request"
mylist-search-request: "|/usr/local/bin/lsv_amin -t mylist-search-request"
mylist-server: "|/usr/local/bin/lsv_amin -t mylist-server"
mylist-signoff-request: "|/usr/local/bin/lsv_amin -t mylist-signoff-request"
mylist-subscribe-request: "|/usr/local/bin/lsv_amin -t mylist-subscribe-request"
mylist-unsubscribe-request: "|/usr/local/bin/lsv_amin -t mylist-unsubscribe-
request"
```

(note that the aliases may *not* wrap to the next line in **/etc/aliases**)

If you should decide to use the explicit definition for the LSVSPOOL parameter, the aliases would look like this instead:

```
mylist: "|/usr/local/bin/lsv_amin /var/spool/listserv mylist"
```

and so forth. Once you've added the new aliases to the file, you need to issue the "newaliases" command and (on some systems) send your Sendmail daemon a hangup (HUP) signal before they will take effect.

7.2.2. OpenVMS: Creating required PMDF aliases

This section is for use only by OpenVMS sites running Innosoft International, Inc.'s PMDF® product, version 4.2 or later.

Please note that you will require system level privileges to edit the file in this step .

If PMDF is installed, in addition to the **listserv** and **owner-listserv** aliases which

you've created in **PMDF_ROOT:[TABLE]ALIASES** at install time, you will need to add the following eight aliases for each new mailing list you create, where ***listname*** is the name of the list:

***listname*:**	***listname*@LISTSERV**
owner-*listname*:	**owner-*listname*@LISTSERV**
***listname*-request:**	***listname*-request@LISTSERV**
***listname*-search-request:**	***listname*-search-request@LISTSERV**
***listname*-server:**	***listname*-server@LISTSERV**
***listname*-signoff-request:**	***listname*-signoff-request@LISTSERV**
***listname*-subscribe-request:**	***listname*-subscribe-request@LISTSERV**
***listname*-unsubscribe-request:**	***listname*-unsubscribe-request@LISTSERV**

Note: You can get around this bit of tediousness (and also solve a problem with address probing under VMS with PMDF as documented in 13.5.3, below) simply by creating a dedicated domain for LISTSERV (eg, LISTSERV.XYZ.COM) and adding a rewrite rule to redirect all traffic for that host to the LSV channel. This also simplifies the creation of new lists since it is no longer necessary to make all of the PMDF aliases shown above every time you make a new list.

7.3. A sample checklist for creating lists

1. Check to see that the list name is legal and not duplicated elsewhere. You can use the CataList (**http://www.lsoft.com/catalist.html**) as one resource for the latter.

2. If **Notebook= Yes**, then make the appropriate directory and make sure that LISTSERV has appropriate r/w permissions in it.

3. If **Notebook= No** but **Digest= Yes**, then make the appropriate directory and make sure that LISTSERV has appropriate r/w permissions in it.

4. Optionally, add the list to the quota file (ISP scope licenses only; see chapter 19 for details)

5. VM: Optionally, make a ***listname*.FILELIST** for this list (see chapter 8 for details)
   Non-VM: Optionally, make an entry in **SITE.CATALOG** for a sub-catalog belonging to this list (see chapter 8 for details) and create a dummy ***listname*.CATALOG** in the specified directory.

6. Non-VM: Optionally, assuming that **Notebook= Yes** and you have installed the web archive interface as described in chapter 5, create the ***listname*** directory under the base '**archives**' directory. If you do this now, you won't have to GET/PUT the list header later to initialize things.

7. Create and store the list header with the list owner and you as the only subscribers.

8. Architecture-specific steps:

- Unix, running Sendmail: Create the required Sendmail aliases for the list, either by hand or by using '**make list name=*listname***' . Note that this is a *required* step for unix servers; if you don't make the Sendmail aliases, the list won't work. See section 7.2.1 for details.

- OpenVMS, running PMDF®: Create the required PMDF aliases for the list in

PMDF_ROOT:[TABLE]ALIASES. Note that this is a *required* step for VMS servers running PMDF; if you don't make the aliases, the list won't work. See section 7.2.2 for details (and an alternative workaround).

9. Send a boilerplate "your list has been created" message to the list as the final test that the list works--if it doesn't, go back and find out why, then return here. See Appendix D for a sample boilerplate message for this step, or use your own.

10. Delete yourself from the list (assuming you don't want to be subscribed)

At this point the list should be ready for use.

7.4. Naming Conventions

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

The "-L" convention

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

Reserved names

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

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

For instance, lists cannot be made with names like "owner-loyalty", "linux-server", and "donation-request". While it is physically possible to create a list with a name that matches one of the above wildcards, attempts to send mail to the list (for example, a list called "linux-server") will result in an error, logged as follows in the LISTSERV log:

```
4 Dec 2001 11:47:02 -> Invalid list (LINUX), generating bounce.
```

These "pseudo-mailboxes" have a special meaning to LISTSERV, which has internal rules that govern how mail sent to these addresses is handled. See chapter 17.3 for more information on what happens to mail sent to these special addresses.

Reserved characters

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

! which can be confused for "bang-path" addressing, for example, UUCP
@ which is a reserved character

which can cause problems with some mail software which uses it for addressing

$ which may have a special meaning to the unix shell

% another addressing character that could cause problems

& is sometimes reserved by non-unix systems (specifically on NT it has a special meaning to the shell). However, please note that *use of this character in the name of a list or in a sendmail alias for a list will cause LISTSERV on unix to choke.* Note that it *is* possible under unix to create a list with a "&" character in the name quite easily, and it is also possible to create a sendmail alias with a "&" character in the alias. That does not mean it will work.

***** is, of course, the wildcard character.

() Parenthesis are generally reserved and can't be used in file names.

+ The plus character should be avoided because recent versions of sendmail deliver mail addressed to "user+whatever@somedomain" to "user@somedomain." Whether or not this is an intelligent thing to do on sendmail's part is left as an exercise for the user, but it can affect mail being sent to a list with a "+" character in the listname.

/ The slash character is reserved and can't be used in file names.

. Although on some systems it is physically possible to create lists with a dot character in the name, LISTSERV will not accept this nomenclature. The only place a dot can or should be used is before the word "LIST" in the **PUT** command; for example, **PUT MYLIST-L.LIST** is equivalent to **PUT MYLIST-L LIST**.

" Double-quote characters are not allowed.

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

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

Maximum length of the list name

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

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

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

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

Make it easy on your users

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

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

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

7.5. 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. We will discuss these keywords in detail in subsequent chapters, and a complete alphabetical listing of list header keywords, including default settings and all options available, is provided in Appendix B.

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

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

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

List Maintenance and Moderation Keywords. A fairly large group of keywords having to do with how the list is operated, including definitions for the list owner, list editor, and the list archive notebook; whether or not (and 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.

Security Keywords. These keywords control who can "see" the list (that is, whether or not the list appears in the List of Lists for a given user, based on the user's host site), 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.

Subscription Keywords. These control whether or not the list is open to general subscriptions, whether or not a mailing path confirmation is required, and what user options are set by default upon subscription.

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

7.6. Retrieving and editing the list – some considerations

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

Once the list has been created, 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 (1.8c and earlier): If you **GET** the header with the **(HEADER** switch, do not add new subscribers "on the fly" to the bottom of the header. If you do, your subsequent **PUT** will replace the entire list online with what you have sent, canceling the subscriptions of every user on the list (except for the ones you added to the header).

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

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

```
DELETE XYZ-L *ADMD=ABC*PRMD=DEF*@X400.SOMEHOST.COM
```

You can also enclose the address in double quotes:

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

7.7. Adding a list password (obsolete since 1.8c)

This section is retained for compatibility with those sites still running 1.8b or earlier.

When creating the list, the LISTSERV maintainer should assign a password for the list. However, note that in 1.8c and later, if the LISTSERV maintainer does not assign a password at the time of the list's creation, LISTSERV will generate a random password for the list. This random password can be changed later, but until and unless it is changed, administrators must provide their personal LISTSERV password (created with the "**PW ADD *password***" command) when updating the list.

Compatibility note: When upgrading to LISTSERV 1.8d and later from 1.8b or earlier, lists without passwords will not be altered during the upgrade. However, the first **PUT** operation for such lists after the upgrade will cause LISTSERV to add the random password to the list. List owners should be encouraged prior to the upgrade to create personal passwords for themselves with the "**PW ADD password**" command (if they have not done so already) and plan to use those passwords after the upgrade.

The list owner can change this password when storing the list (with the "PW=" keyword), but the first time the list owner stores the list, the original password or the list owner's personal password must be used. Note that not all LISTSERV maintainers assign list passwords by default; the new random password feature addresses that. However, for pre-1.8c servers it is *highly recommended* that one be assigned by adding a "PW=" header line as follows:

```
* PW=MYPASSWD
```

Replace "MYPASSWD" with the word chosen. Note that there should *not* be a space between "PW=" and the password. The list password is never changed unless specified explicitly in the list header when it is stored on the server. For additional security, the list

password will not appear in the list header on subsequent **GET**s; to all intents and purposes it is invisible once it is assigned.

L-Soft's position on list passwords is that they have become obsolete with version 1.8c (they were actually obsolete as far back as 1987), and that personal passwords should be used instead to validate commands (such as the **PUT** command).

7.8. Storing a modified list on the host machine

(If you are creating a list, see 7.1. These instructions are for storing a list once it already exists on the server, for instance, if changes have been made to the list header after a GET operation.)

When you are ready to store your list 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. 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 that could actually affect the correct operation of the list, the list will not be stored and the old version will be retained. (For instance, creating a list with no list password defined in the header will generate a "soft" error under 1.8b and before, and the list will be stored. On the other hand, setting a list to "Send= Editor" and not defining an editor with "Editor=" is considered a "hard" error, and you will have to fix the error before LISTSERV will accept the list for storage.)

Caution: If you are using a mailer such as Eudora, Pegasus, 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.

If your mail software inserts page formatting (margins) or quoting characters (such as ">") in forwarded mail, you need to either turn these features off or you must cut and paste the header into a new mail message. The **PUT** line MUST be on the first line of the message, and all header lines including the **PUT** MUST start in column 1. Specific problems have been noted with cc:Mail (where top and left margins get inserted) and with certain POP clients including Eudora and Microsoft Exchange (where forwarded mail is quoted with ">" by default).

Also, *be sure to turn off your signature file* (if you use one) before sending a **PUT** command to LISTSERV. If you don't, LISTSERV will attempt to parse the data in your signature file as RFC822 addresses to be added to the list, and you will receive either an error to the effect that the file includes invalid RFC822 addresses and it has therefore not been stored, or a warning that your **PUT** operation contains new subscriber information and only the list header has been stored (see 7.6 for information on the **PUTALL** command).

7.9. 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. (In LISTSERV 1.8d and later you should use the **PUTALL** command for this purpose since you will be storing the entire list, not just the header.)

It is also possible for the LISTSERV maintainer to restore the list by deleting or moving the ***listname*.LIST** file from LISTSERV's A directory and renaming the ***listname*.OLDLIST** file to ***listname*.LIST**. Naturally this method requires that the LISTSERV maintainer in question have appropriate access to LISTSERV's files and directories or be able to log in as the 'listserv' user.

7.10. A sample list header file

A basic list header file for a list to be created might look like this (**CREATEPW** must be replaced with the appropriate password):

```
PUT MYLIST.LIST PW=CREATEPW
* The Descriptive Title of My List
*
* Owner= NATHAN@EXAMPLE.COM (Nathan Brindle)
* Notebook= Yes,E:\LISTS\MYLIST,Monthly,Public
* Errors-To= Owner
* Subscription= Open,Confirm
* Ack= Yes                 Confidential= No
* Validate= No
* Reply-to= List,Respect   Review= Public
* Send= Public
* Default-Options= NoRepro,NoMIME
*
* This list installed on 96/06/02, running under L-Soft's LISTSERV-TCP/IP
* for Windows NT.
*
* Comment lines...
*
```

Figure 7.2. A sample list header file for a list called MYLIST.

A list owner might take the created list and modify it as shown below. Note that the **PUT** command has been modified to include the password you've assigned with the **PW ADD** command.

```
PUT MYLIST.LIST PW=MYPASSWD
* The Descriptive Title of My List
*
* Owner= NATHAN@EXAMPLE.COM (Nathan Brindle)
* Owner= Quiet:
* Owner= nathan@linus.dc.example.com
* Owner= ncbnet@linus.dc.example.com
* Notebook= Yes,E:\LISTS\MYLIST,Monthly,Public
* Auto-Delete= Yes,Full-Auto
* Errors-To= ncbnet@linus.dc.example.com
* Subscription= Open,Confirm
* Ack= Yes                 Confidential= No             Notify= No
* Mail-Via= Distribute     Validate= No                 Send= Public
* Reply-to= List,Respect   Review= Public               X-Tags= Yes
* Default-Options= NoRepro,NoMIME
*
* This list installed on 96/06/02, running under L-Soft's LISTSERV-TCP/IP
* for Windows NT.
*
* Comment lines...
*
```

Figure 7.3. The edited list header file ready to be sent back to the server.

7.11. Deleting a list

For security reasons, LISTSERV does not have an explicit command for deleting lists. The LISTSERV administrator simply deletes the list file from the system command prompt with the appropriate file system command (**CMS ERASE** for VM, **DEL** for VMS, **ERASE** for Windows, **rm** for Unix). A suggested procedure for deleting an established list (one with archives and so forth) follows:

1. Back up any files you wish to keep, such as notebook archives

2. For a digested list, you may want to send a **QUIET SET *listname* NODIGEST FOR *@*** command. This will cause LISTSERV to send out its accumulated digest to those who were set to DIGEST mode. If the list hasn't been active or if it's not digestified, you don't need to take this step.

3. Delete the ***listname*.LIST** file with the appropriate file system command.

4. If the list has web archives, delete the **/archives/*listname*.html** file and the **/archives/*listname*/*listname*.ind*** files. You can also remove the **/archives/*listname*** directory at this time.

5. Although it is not absolutely necessary, stopping and restarting LISTSERV will complete the procedure. If you do not stop and restart LISTSERV, LISTSERV will fairly quickly notice that the list is gone, and will take care of this on its own.

7.12. Adding HTML to a list header for the CataList

L-Soft's CataList service allows users to search the global list of LISTSERV lists via the World Wide Web. Adding an HTML description to a list is easy, and can do a lot to enhance the appearance of a list in the database. All the list owner or LISTSERV maintainer has to do is update the list header and add the text of your choice. Here is an example:

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

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

1. The **<HTML>** and **</HTML>** tags *must* appear on a separate line, as shown in the example above. You cannot have anything else on that line and, in particular, you cannot mix keyword definitions with HTML data.

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

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

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

7.12.1. Update latency

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

7.12.2. Inserting a pointer to another list

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

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

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

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

7.12.3. Restrictions on the placement of equal signs

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

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

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

7.13. How to set up lists for specific purposes

Under LISTSERV 1.8d and later, you can create certain types of lists from standard templates via the web. See chapter 11.9, below, for information on how to access the web-based server administration interface.

7.13.1. Public discussion lists

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

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

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

For more security, you might want to code

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

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

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

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

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

instead.

CAUTION: "REPLY-TO:" HEADERS ARE NOT UNIVERSALLY HONORED!

PLEASE NOTE CAREFULLY: There is one major caveat with regard to the use of the **Reply-To=** list header keyword. Setting this parameter guarantees only one thing -- that LISTSERV will generate an appropriate RFC822 Reply-To: header in the mail it distributes to subscribers. THERE IS UNFORTUNATELY NO GUARANTEE THAT THE MAIL TRANSFER AGENT (MTA) OR MAIL CLIENT ON THE RECEIVING END WILL HONOR THE Reply-To: HEADER. This is because some mail clients, out-of-office robots, and Internet MTAs either simply do not recognize the existence of Reply-To: or do not implement it properly. Specifically RFC2076 "Common Internet Message Headers" reports that the use of Reply-To: is "controversial", that is, "The meaning and usage of this header is controversial, meaning that different implementors have chosen to implement the header in different ways. Because of this, such headers should be handled with caution and understanding of the different possible interpretations." (RFC2076, page 4). While L-Soft recognizes that it is sometimes important to provide an explicit Reply-To: header to indicate a response path, L-Soft cannot be held responsible for problems arising from the inability of a remote server to properly process Reply-To: headers.

7.13.2. Private discussion lists

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

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

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

in 7.13.1).

7.13.3. Edited lists

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

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

To set up a basic edited list, simply add

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

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

Please note that L-Soft recommends setting "Send= Editor,Confirm" so as to add a level of security against malicious users forging mail from an "Editor=" address to get around your moderation settings, or against badly-configured "vacation" programs that simply reflect the message back to the list in a manner that makes it appear that the mail is coming from the editor's address. The "Confirm" option causes LISTSERV to request an "OK" confirmation from an editor when it receives mail claiming to be from that editor.

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

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

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

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

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

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

For security purposes, you can code

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

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

Naturally, you can also code

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

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

7.13.4. Moderated lists

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

A moderated list is similar to an edited list, but for LISTSERV's purposes it refers to a list that uses the Moderator= list header keyword to "load-share" posting approvals among several editors. It is set up similarly to an edited list, as follows:

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

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

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

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

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

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

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

Please note that something like

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

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

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

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

The "Hold" and "Confirm" options for "Send=" can also be used with these examples, if desired. L-Soft recommends that "Confirm" be used by default.

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

Moderation "OK" requests and MIME attachment display: In versions previous to LISTSERV 1.8e, an OK confirmation request for a message coming to a moderated list displayed the message to be approved in its "raw" format; that is, there was no attempt made to display/decode MIME attachments that might be present in the message to be approved. LISTSERV 1.8e addresses the problem by including a copy of the first text/plain part (if one exists in the message) for the purpose of quick screening. The following restrictions apply:

1. This is only done for MIME messages (even simple single-part ones, but they must have MIME headers).

2. The text part in question is sent pretty much 'as is', that is, as an extra text/plain part in the message, with all the options and encoding and what not supplied in the original message. The reason is quite simply that it would be a lot of work and, in some extreme cases (incompatible code page, etc), completely impossible, to embed it into the first text/plain part with the LISTSERV message. The drawback is that some mail agents might conceivably only show the first part until you take some kind of clicking action.

It is important to understand that only the first text/plain part is extracted in this fashion. The goal was to make it easier to approve or reject simple text messages, not to build a factory around a simple problem. The ENTIRE message is available at an extra click.

Where security is a concern, it is important to review the ENTIRE original message and not just the plain text part. There could be an obscene GIF or another text part or a text/html part not matching the contents of the text/plain part or whatever. This is why, again, you are given the ENTIRE original message.

List owners using certain email clients (specifically Pine, which handles attachments in a secondary viewing area) may find the new format difficult to use. If preferred, the pre-1.8e behavior may be reverted to by specifying "NOMIME" in the Send= list header keyword; for instance,

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

7.13.5. Semi-moderated lists

"Semi-moderation" was developed some years ago after a great debate on whether or not an "urgent" message should be allowed to be posted to an edited list without having to go through the approval process. Although this option is still available, it can be misused by anyone who knows about it, and is therefore not generally recommended for use. However, should this feature be deemed necessary, it is activated by setting

```
* Send= Editor,Semi-Moderated
```

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

7.13.6. Self-moderated lists

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

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

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

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

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

There is one caveat to this kind of list. If a user subscribes to the list, and later his mail address changes (for instance, the hostname changes slightly but mail sent to the old address is automatically forwarded to the new address), any postings from him to the list from the new address will be forwarded to the editor because the new address is not subscribed to the list. Thus there is a certain amount of list-owner overhead on this kind of list in keeping track of users whose addresses have changed and modifying the subscriber list to reflect those changes. The "**CHANGE**" command added in 1.8d can be of help in this regard.

7.13.7. Private edited/moderated lists

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

Using the same header you would create for an private discussion list (see 7.13.2, above), simply add the following line to the header:

```
* Default-Options= REVIEW
```

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

Note the following carefully:

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

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

QUIET SET *listname* REVIEW FOR *@*

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

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

7.13.8. Auto-responders

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

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

A simple auto-responder header might look like this:

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

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

POSTACK1 template, like the following:

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

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

7.13.9. Announce-only lists

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

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

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

L-Soft *strongly recommends* that all announce-only lists use the "**Send= Editor,Confirm**" or "**Send=Editor,Hold,Confirm**" setting. The "**,Confirm**" parameter tells LISTSERV to require a confirmation for any posting sent by a user defined as an Editor=. This is important for two reasons:

1. Security. This setting tells LISTSERV to request confirmation from the Editor for all postings it receives that purport to be from that Editor. This prevents hackers from forging mail under an Editor's address, because any forgeries will require that the Editor in question approve them before they go to the list.

2. Loop protection. Certain broken mailers can and will bounce mail back to your list in a "reflected" manner, that is, the bounce will appear to be a legitimate posting from

the Editor to the list instead of looking like an error. This is different from a forgery attempt because (it is assumed) the mailer on the other end is not doing this with malicious intent. Requiring the editor confirmation will stop these potential loop-generating messages from getting through to the list.

To stop a posting from going to the list under this scenario, simply don't OK it and delete the confirmation request message.

7.13.10. Restricted subscription lists with automatically-generated questionnaire

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

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

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

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

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

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

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

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

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

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

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

```
>>> ADDREQ1 &LISTNAME Membership Survey
.RE OWNERS
```

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

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

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

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

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

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

7.13.11. Peered lists

This functionality is not available in LISTSERV Lite.

Occasionally the need to split a very large list may arise. This was more common when LISTSERV ran only on BITNET, whereas the TCP/IP version of LISTSERV is not limited by BITNET constraints. However, because of the fact that subscribers may be scattered all over the world, in rare cases it can make sense to split (or "peer") a list and share the mail load among two or more LISTSERV servers. Peering also makes it possible to have list archives located in more than one place; for example, a list might be peered between a European host and a North American host, making it possible for subscribers on each continent to retrieve archives from the nearer host.

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

Linking two or more LISTSERV mailing lists

Please observe the following points:

1. All lists should have a Peers= keyword setting that includes all of the other peers in the group as its arguments. For example, consider a peer group containing ListA, ListB, and ListC. ListA must have "Peers= ListB@its.host.com,ListC@its.host.com", whereas ListB must have "Peers= ListA@its.host.com,ListC@its.host.com" and finally ListC must have "Peers= ListA@its.host.com,ListB@its.host.com".

For lists running on LISTSERV for VM, setting the Peers= keyword 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.)

2. All lists must have the same list-level password, set with the PW= list header keyword. If this point is ignored, messages approved on one peer will not be accepted by the other peer and an error message will be generated, i.e.,

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

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

This means that under LISTSERV 1.8c and later you must explicitly set the **PW=** list header keyword for each peer and not use the password LISTSERV generates automatically at list creation time. (This is the only case in which it is important to manually set PW= for a list.)

3. Each peer must be subscribed to at least one other peer, and the "real name" field for the subscription MUST be set to "Peer Distribution List".

Moving users from one (peer) server to another:

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

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

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

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

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

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

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

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

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

Special commands for peered lists only

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

The ADDHERE command is strictly identical to ADD, with the exception that the placement of the user is not checked against the list of peer servers; in other words, 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.

7.13.12. "Super-lists" and "sub-lists"

This functionality is not available in LISTSERV Lite.

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

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

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

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

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

2. The DIGEST and INDEX options are ignored and internally converted to MAIL. The first reason is that, since in most cases the user will be on multiple sub-lists (otherwise you don't need a super-list in the first place), the only safe method to set subscription options for super-messages is by subscribing to the super-list so that

there is no ambiguity. The second reason is that, in most cases, super-lists will be used for out of band administrative messages rather than for large volume discussions, so it is actually preferable to have the message sent directly. The third reason is that the super-list and sub-lists may not necessarily offer the same options (DIGEST and INDEX). In particular it is expected that many super-lists will not have archives. If you want a DIGEST or INDEX for the super-messages, you must subscribe to the super-list directly.

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

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

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

Note carefully that a **REVIEW** command sent for the super-list *will not* recurse into the sub-lists pointed to by the super-list. If you have a super-list called SUPER and you send a **REVIEW SUPER** command, LISTSERV will respond with only the people who are subscribed directly to SUPER. The only way to find out what users are covered by the super-list is to send **REVIEW** commands for the super-list and all of its sub-lists.

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

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

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

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

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

If you have many sub-lists, you can specify multiple Notebook= lines, for example,

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

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

7.13.13. "Cloning" lists

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

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

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

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

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

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

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

The **.IK** command tells LISTSERV that whenever it uses this list, it should read the keyword definitions from the file **M101 KEYWORDS** (note carefully that the syntax is "**.IK**

M101", *not* "**.IK M101 KEYWORDS**"). Now, whenever the professor in charge of the class wants to make a change to all of the M101 lists (for instance, he has a new graduate assistant), he simply **GET**s the file **M101 KEYWORDS**, makes the changes, and **PUT**s the file back, instead of having to **GET** separate headers for each list and make the changes to all of them individually.

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

NOTE CAREFULLY THE FOLLOWING: In order to see the complete list header, send a **REVIEW *listname*** command. The response to a **GET** will be only the skeleton header with the **.IK** command. If GET did not work this way, you would not be able to change or remove the **.IK** command line once you set it.

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

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

7.14. Merging existing LISTSERV lists

It is sometimes desirable (or necessary) to merge two or more existing lists. There are a couple of different ways to do this.

7.14.1. Merging list A into list B; list A user options not preserved

This is perhaps the simplest merge operation and requires only that you get the list of subscribers from list A and add them to list B, probably with a bulk operation as explained in section 7.17, below. User options are not preserved across the move and the users from list A will be subscribed to list B with whatever default options are set for list B.

7.14.2. Merging list A into list B; list A user options preserved

In this case you need to **GET** both lists A and B, header and all (so you do not use the **(HEADER** switch in this case). LISTSERV will return copies of the entire list files to you including all of the subscribers along with an encoded option string for each subscriber. Usually this will look something like this:

```
* My test list
* (remaining header lines removed for clarity)
*
xxxxx@APK.NET Pxxxx Axxxxx
2AAARAA4HAAA
xxxxxxxxxx@AOL.COM Rxxxxx Axxxx
2AAARAA2bAAA
xxxxxx@LSOFT.COM Nxxxxx Bxxxxxx
2AAARAA2bAAA
xxxxxxxx@CS.ROSE-HULMAN.EDU Mxxx Dxxxxxx
2AAARAA3nAAA
```

Depending on how your mail client handles long lines, the subscriber lines will be either:

1. Kept as one single long line in which the option string starts in column 81 of the line; or

2. Split into two separate lines, the subscriber address and real name on one line followed by the option string on the next line.

If case 1, you should have no problem with the following operations. If case 2, you must either

- Reformat the lines so that the option strings start in column 81 rather than being on separate lines; or

- If your mail program supports MIME, re-order the file as a MIME/TEXT attachment by adding **F=MIME/TEXT** to your **GET** command (e.g., **GET MYLIST-L F=MIME/TEXT**). This should preserve the long lines without inserting end-of-line characters.

In any case a correctly-formatted subscriber line looks like this:

```
nxxxxx@LSOFT.COM Nxxxxx Bxxxxxx                                    2AAARAA2bAAA
```

Next, assuming that the subscriber lines are correctly formatted, cut and paste list B into a new mail message addressed to LISTSERV. Make sure that your mail client has all formatting options turned off; for instance, make sure line wrap and any automatic "rich text" or HTML mail formatting is turned off. If you do not do this there is no guarantee that the list file will reach LISTSERV properly formatted.

At the bottom of this new message, you can cut and paste the subscribers from list A. Note that you don't want the header of list A, just the subscriber lines. Make sure that there is no blank line between the subscribers you pasted from list B and the subscribers you have just pasted from list A.

Finally, you can now **PUT** your new merged copy of list B.[2]

7.14.3. Merging list A and list B into list C

[2] There is an alternate method of doing this kind of merge which requires access to the machine LISTSERV is running on. You can simply use the 'listview' program to output the list files in text format, and redirect them into editable files (with 'listview -a listname > listname.txt', for instance). Then you can use a text editor to combine the two lists and either mail the resulting merged list to LISTSERV or store the merged file as listname.list and restart LISTSERV to reformat the file (as explained above in 7.1). However, note carefully that the latter method (save and restart LISTSERV) is not recommended, nor is it the supported method for storing lists, as it bypasses a number of checks LISTSERV does when you use PUT.

In this case (where you may be starting a completely new list and want to merge two old lists into it), follow the directions above depending on whether or not you want to preserve user options across the merge or not. The only difference is that you will be combining the subscribers from two lists into another list instead of combining subscribers from one list into a second list. In this case you do need to be careful not to add duplicate addresses, as LISTSERV will not catch them when you **PUT** the new list file. In fact it is probably more sensible to set appropriate defaults to the new list and store the header by itself, then add the users with a bulk operation (not preserving their old options) so that LISTSERV can catch any duplicates you might add.

7.15. Migrating lists from one site to another

In migrating lists to LISTSERV, there are three typical possibilities:

1. You are migrating lists from an existing LISTSERV site (e.g., moving from VM to unix)
2. You are migrating lists from a non-LISTSERV site to LISTSERV
3. You are creating a LISTSERV list from a Sendmail alias or other database of e-mail addresses

7.15.1. Migrating lists from one LISTSERV site to another LISTSERV site

Naturally, this is the simplest migration, but it still requires a few important steps. The preferred method (and the one that generally works the best) is to GET the list from the old server, make any changes necessary to the header (e.g., location of Notebook archives) and PUT the resulting list file on the new server. This method (assuming no corruption or reformatting of the list file by intervening mail systems) is preferred because it involves LISTSERV's internal syntax checking and other error-handling functions, LISTSERV knows exactly where to put the files, and the migration isn't restricted by possible architecture-specific problems.

The drawback to the preferred method is that you have to migrate one list at a time, which may not be acceptable if you need to migrate many lists in a short period of time. In general, you can simply FTP your list files from the old server to the new server, but note the following:

- You can migrate only from VM to non-VM, or from non-VM to non-VM. You *cannot* migrate using the FTP method from non-VM back to VM (unless you are prepared to reconstruct your list files and so forth from scratch). Naturally a GET and PUT *will* work if you need to move from non-VM to VM.

- *If migrating from VM to non-VM:* Be sure to FTP the list files, archives, and so forth in ASCII mode. If you use binary mode, the files will be unreadable on your new system.

- *If migrating from non-VM to non-VM*, you can FTP the list files in binary mode and any other files in ASCII mode. Please note that if moving the list files by binary FTP does not work, you will have to migrate the list using the preferred method outlined above. You could also create a new list header on the new server and add the users with an **ADD IMPORT** job as detailed in the next section.

- Once you have FTP'd the files to the new server, decide where you want things to go. The list file itself should go into LISTSERV's A directory (typically **~listserv/home** on unix systems, **LISTSERV\MAIN** on Windows systems, and

LISTSERV_ROOT:[MAIN] on VMS systems). You may want to make a separate directory on your new server for archives, then subdirectories of that directory for each list (see chapter 5.8, above). If so, make the appropriate directories and move the archive files there. (This is particularly important if you intend to use the ISP options such as the file quota subsystem.) If you are copying non-notebook archives, you should read the chapter on *Notebook and File Archives*, below, in order to set up a filelist or catalog file for these files.

- Next, you should restart LISTSERV. This is particularly important when moving lists from VM to other platforms, as LISTSERV will need to reformat the file into the binary format used on non-VM machines. If this is successful, you will see two messages in the console log:

  ```
  6 May 1996 12:50:14 Invalid record format for list XXXXX-L.
  6 May 1996 12:50:14 -> List reformatted successfully.
  ```

  If this is *not* successful, you will need to open the list file in a text editor and look for anything that might have caused a problem. Note that list header lines have a limit of 100 characters in length.

- Before releasing the list to the general public, be sure to GET the list header and make any changes that need to be made. Typically, changes will need to be made to the location parameters of the Notebook= and/or Digest= keywords, particularly if you are moving from one platform to another.

Note that the first digest sent from the new site will say "First ever".

7.15.2. Migrating lists from non-LISTSERV sites

Non-LISTSERV list files (notably from Majordomo and ListProc, but from other MLM software as well) are not directly compatible with LISTSERV. While it is probably possible to write a script or batch file for the purpose of converting one format to the other, it is outside the scope of this manual to describe this process.

Majordomo users will note that LISTSERV does not require two separate lists for those who want individual messages and those who want digested summaries. LISTSERV handles digesting internally for those who have set the personal option **DIGEST** for the list. Thus those sites migrating to LISTSERV from Majordomo will probably want to merge the digested and non-digested subscribers into one single list and let all subscribers know that they can set themselves to **DIGEST** mode with the **SET listname DIGEST** command. (It would also be possible to send commands to LISTSERV to set all of the old digest subscribers to **DIGEST** before releasing the list to the public.)

Under most conditions, the method recommended by L-Soft for migrating a non-LISTSERV list into LISTSERV format is the following:

- Create a list header for your list as noted above and store it on the LISTSERV server. If you plan to set "Validate=" to any value but "No", set it to "No" until the following steps are completed.

- Create a LISTSERV command language job as follows:

  QUIET ADD *listname* DD=X IMPORT

```
//X DD *
internet-address1
internet-address2
/*
```

where "***listname***" is the name of the new list, and "***internet-address1***", "***internet-address2***" and other users are the internet addresses from the original list that you want to add to the new list. Optionally, you can add the user's "real name" field, for example,

```
QUIET ADD listname DD=X IMPORT
//X DD *
internet-address1 full_name
internet-address2 full_name
/*
```

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

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

and reduce it at least to

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

Otherwise, the ListProc options will become part of the **full_name** field.

- Send the command language job to LISTSERV.
- You will receive in return a confirmation that the job executed and whether or not it was successful:

```
> QUIET ADD XXXXX-L DD=X IMPORT
ADD: no error, 2 recipients added, no entry changed, none
forwarded.
```

List archive notebooks from non-LISTSERV sites can be copied into a file archive area for the list and registered in the ***listname*** **FILELIST** (VM) or ***listname*****.CATALOG** (non-VM), but it is *not* recommended that non-LISTSERV notebooks be renamed with LISTSERV naming conventions, as this may cause problems with LISTSERV's database functions. For instance, if you have ListProc or Majordomo notebook archives that were kept monthly, L-Soft does *not* recommend renaming them with the ***listname*****.log*****yymm*** format.

For information about how to convert non-LISTSERV archives to LISTSERV format, please see 8.10.3, below.

Alternate method of creating the list[3]: You can send the list header and subscriber list to

[3] Another alternate (but unsupported) method of creating the list: Note that you can also create the list, header and subscriber list together, using a text editor. Simply start adding subscribers right after the last header line, save the file with the extension ".list", and restart LISTSERV as noted above in 7.15.1 to reformat the list. Do not, however, attempt to edit the list with a text editor once it has been created--use

LISTSERV in the body of an e-mail (attachments will be ignored, the header and subscriber list MUST be plain text in the body of the mail message). Only one list can be created per e-mail, and the body of the mail must look like this:

```
PUT listname LIST PW=createpw
* Long title of list
* (more list header lines, must begin with asterisks in column 1)
userid1@example.com His Name
userid2@example.net Her Name
```

In the above syntax example, "listname" is the name of the list, and "createpw" is the CREATEPW value from your site configuration file. The text of all lines must begin in column 1. All header lines must begin with an asterisk. There must not be any blank lines anywhere in the text (they would be considered as end-of-file markers).

Subscribers added in this fashion will inherit any Default-Options and Default-Topics from the list header.

7.15.3. Migrating lists from Sendmail alias files, databases, etc.

In general, you will follow the same procedure outlined in 7.15.2 to migrate from these types of lists. You may wish to write an executable script of some sort to pull the addresses and names (if you have them) from your database and surround them with the appropriate CJLI commands, particularly if your database is made from a web site and you need to run a periodic job to add users to your lists.

7.16. Changing the name of an existing list

Changing the name of an existing list on the same server as opposed to migrating a list from another server is somewhat different. Here is a checklist of the basic steps involved in renaming an existing list. For the purpose of this example we will assume that the list is named **MYLIST-L** and we want to rename it to **JOESLIST-L**. Note that operations that call for using OS-level commands are not performed by issuing commands to LISTSERV, but rather by opening a console session and typing the commands at your system's command prompt.

1. Stop LISTSERV.

2. Find the **mylist-l.list** file. LIST files are kept on LISTSERV's A disk (VM) or in its A directory (non-VM). The A directory for non-VM servers is normally **~listserv/home** for unix servers, **LISTSERV_ROOT:[MAIN]** for VMS servers, and **LISTSERV\MAIN** for Windows servers.

3. Copy (using your OS's command for copying files) **mylist-l.list** to **joeslist-l.list**. Note carefully that under unix you must name the file in lower case. Copying the list file will preserve all subscribers and all subscriber options.

4. If necessary, create the directory for **joeslist-l**'s archives. If you had **mylist-l**'s archives in **~listserv/lists/mylist-l**, for instance, you should create the directory

---

only LISTSERV commands sent via mail, the VM console, or the LCMD utility to make changes. And note carefully that subscribers added in this fashion (i.e., without a pre-existing user options string) DO NOT inherit any default options set in the header, and would require that any required options be set manually (that is, with the command `QUIET SET` *`listname options`* `FOR *@*`). You can avoid this problem by not using this method.

~listserv/lists/joeslist-l. Once this directory is created, you can copy the **mylist-l** archive notebooks over to it, then rename any **mylist-l.*** file to **joeslist-l.***. Note that you will want to copy the current notebook over again later, to make sure you get all of the postings up to the time of the switch. Note further that it is not necessary (and probably not desirable in any case) to copy the DBNAMES, DBINDEX, DBRINDEX, or -RAC files as they will be rebuilt automatically by LISTSERV. Also, you don't need to copy the DIGEST or SUBJECTS files as we're going to take care of them later.

5. Again, if necessary, you should also copy over any files referenced by the list's catalog or filelist and make a new catalog or filelist for **joeslist-l**. You will also need to make an entry in **site.catalog** (non-VM) or **listserv.filelist** (VM) for the new joeslist-l catalog or filelist.

6. If the list was available through the web archive interface, make a **joeslist-l** directory for the web archive indexes (see chapter 5 for details).

7. Restart LISTSERV.

8. Issue a **GET JOESLIST-L (HEADER NOLOCK** command to get the header. Make any changes you feel necessary, for instance, in the list's description or in the comments which may or may not contain the old list's name. You will also need to make changes to any keyword that contains a directory reference, for instance the Notebook= and Digest= keywords, so that they point to the right place. **PUT** the list header back on the server. (Note that this **PUT** will cause LISTSERV to build web archive indexes for the list.)

9. Issue a **HOLD JOESLIST-L** command to keep the list from processing any postings from earlybird users :).

At this point you are finished copying the old list to the new list. Now you need to do some housekeeping before notifying the users of the change.

10. Issue a **QUIET SET MYLIST-L NODIGEST NOINDEX FOR *@*** command to LISTSERV. This will force LISTSERV to send out the accumulated **MYLIST-L** digest and index issues to all users who had those options set.

11. Issue a **HOLD MYLIST-L** command to LISTSERV.

12. Copy the final **MYLIST-L** notebook archive file over to the **JOESLIST-L** directory so that you have all of the postings up to the time you issued the **HOLD**.

13. Get the header of the **MYLIST-L** list. You can now add a "New-List=" keyword to the header to let people know that the name of the list has been changed. This requires that you remove all other keywords from the header except "Owner=" and "Confidential=". You can set

```
* New-List= JOESLIST-L@LISTSERV.MYHOST.COM
* Confidential= Yes
```

in the list header so that a) the list no longer appears in the global List of Lists and in the CataList and b) so that all mail and inquiries sent to the old list address will be forwarded on to the new one. When you've made the changes to the header, **PUT** it back on the server.

14. Issue a **FREE JOESLIST-L** command to LISTSERV. (You should not need to issue

a **FREE MYLIST-L** command.)

Congratulations, you've finished renaming the list. At this point you should probably announce the change and let people know where to find the archives, etc.

7.17. Bulk operations (ADD and DELETE)

It is possible to use "bulk" operations to "front-load" or otherwise simplify the job of adding and/or deleting users from lists. This will typically be used on very large announce-type lists but the functionality is naturally available for all lists.

7.17.1. Bulk ADD operations

To front-load or just to add a large number of users to an existing list, you construct a LISTSERV JOB framework as follows and then send it to LISTSERV. The **QUIET** and **IMPORT** command words are optional; omit the square brackets if you use them. The "full name" field is optional as long as you use the **IMPORT** option; otherwise you must either specify "*****" (for an anonymous subscription) or a full name consisting of at least two separate words.

```
[QUIET] ADD listname DD=ddname [IMPORT] PW=yourpassword
//ddname DD *
userid1@host1.com [full name]
userid2@host2.com [full name]
...
useridn@hostn.com [full name]
/*
```

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

It is also possible to do bulk operations through the Web Administration Interface; see chapter 11 for details.

7.17.2. Bulk DELETE operations

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

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

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

```
[QUIET] DELete listname DD=ddname PW=yourpassword
//ddname DD *
userid1@host1.com
```

```
userid2@host2.com
...
useridn@hostn.com
/*
```

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

It is also possible to do bulk operations through the Web Administration Interface; see chapter 11 for details. However, note that very large bulk ADD and DELETE jobs should be sent via e-mail in preference to using the web interface.

7.18. Content filtering

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

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

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

[*prefix*:] *pattern*

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

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

Documented Restriction: You cannot match literal asterisk characters in a string as there is no way to escape them. Any asterisk in a pattern will always be evaluated as a wildcard.

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

```
Subject::
```

There are two significant differences between exact and regular match:

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

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

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

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

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

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

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

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

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

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

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

```
>>> CONTENT_FILTER
Subject: Out of office
Action: REJECT OOO messages are not allowed on this list.
From: JOE@EXAMPLE.COM
```

```
Action: ALLOW
Text: Click here to be removed
Action: REJECT Buzz off, spammer.
```

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

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

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

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

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

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

7.19. DomainKeys Message Signing (14.5)

This feature is not available in LISTSERV Lite.

Starting with LISTSERV 14.5, DomainKeys message signing is available to sites running LISTSERV Classic or LISTSERV Classic HPO. Current LISTSERV maintenance is also required. For more information on how to configure LISTSERV for DomainKeys support, please refer to our document *Using LISTSERV with DomainKeys*.

Assuming that it is available for your use, DomainKeys support for lists is enabled by default. This means that all list postings and administrative messages related to a list will be signed to assert that they actually originated from your LISTSERV server.

If for some reason you wish to disable DomainKeys message signing for a given list, you can do so by adding

```
* Misc-Options: NO_DKIM_SIGNATURE
```

to your list header. Or if you prefer to disable it server-wide by default, you can add NO_DKIM_SIGNATURE to the DEFAULT_MISC_OPTIONS site configuration variable setting.

Incoming DomainKeys or DKIM signatures submitted to a mailing list will be removed unless "`Misc-Options= KEEP_DKIM_SIGNATURE`" is set in the list configuration. This is necessary because these signatures almost never match after the message has been processed. The worst thing that could possibly happen to your deliverability is a DomainKeys signature that does not match and causes the message to be flagged as suspicious.

The KEEP_DKIM_SIGNATURE option is experimental and not meant for general use. As DomainKeys is specified today, signatures DO NOT survive posting to mailing lists (LISTSERV or otherwise), so LISTSERV removes them by default to avoid triggering alerts for subscribers on systems that have implemented the client side of DomainKeys. The DKIM specification may be more robust in this respect, but even DKIM signatures will probably not survive when posted through a mailing list. Use the KEEP_DKIM_SIGNATURE option at your own risk.

8. File and Notebook Archives

Documented restriction: The hierarchical listname.catalog system documented in 8.4, below, is not available under LISTSERV Lite. You may store files on a Lite server for people to retrieve, but the files must be registered in the site.catalog file and must reside in the same directory with the *.list files so that LISTSERV can find them.

There are three file server systems currently in use by various versions of LISTSERV:

- The VM (mainframe) version of LISTSERV continues to support the "traditional" file server system. While it is very powerful, this file server system dates back to 1986 and suffers from a few annoying limitations. In addition, it is written in a non portable language. This will be replaced eventually with the "new" file server system, currently under development.

- The non-VM versions of LISTSERV 1.8d enhanced further the new file server system introduced in non-VM 1.8c, which included most of the functionality of the "traditional" file system. Notably, **GIVE** and file "packages" became available. Most end user commands continue to work as before. However, there is no guarantee that the internal data files manipulated by the file server functions will remain as before. Note that **SITE.CATALOG** files from versions 1.8a through 1.8c *are* still supported and will not need to be changed in order to work with 1.8d and later.

- The non-VM versions of LISTSERV 1.8a and 1.8b supported a "temporary" file server system, to provide an interim solution while the new system was being developed. This temporary system supports only a subset of the functions of the traditional system. This system is no longer supported by L-Soft as it has been superseded by the new non-VM file server referenced above.

In general, the three systems are compatible, with the understanding that the temporary system does not include all the possible options. However, the mechanism for registering files (defining them to the file server system) is different.

Since the first and third systems will eventually be replaced by the second system, rather than providing an exhaustive chapter detailing all filelist aspects from the management side, we have provided only a basic overview of the two systems currently in the field with 1.8e, 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

The LISTSERV maintainer stores "root-level" file definitions in a file called SITE.CATALOG, which should be placed in the same directory with the SYSTEM.CATALOG file.[5] Beginning with 1.8c, the LISTSERV maintainer can also define "sub-catalogs" which in turn can define further files. You should be aware of the differences between VM and workstation file server functions as many people are using and will continue to use the VM file server with different conventions, and may give you incorrect advice. Non-VM sites should skip section 8.3, and use the information below in section 8.4 to maintain their file archives.

8.3. Filelist maintenance (VM systems only)

If you are running LISTSERV under unix, Windows, or VMS, please skip this section as it does not pertain in any way to your implementation of LISTSERV.

Maintaining the filelist for your archive is not difficult. It requires only that you have a working knowledge of VM XEDIT (or your local system's editor) and understand how to send files via e-mail.

8.3.1. VM only: Creating a filelist

Please see FSV GUIDE (available at ftp://ftp.lsoft.com/documents/fsv.guide) for details.

8.3.2. VM only: Adding FAC codes

Please see FSV GUIDE (available at ftp://ftp.lsoft.com/documents/fsv.guide) for details.

8.3.3. VM only: 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:

---

[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@LISTSERV.NET as FSV GUIDE, or at ftp://ftp.lsoft.com/documents/fsv.guide .

[5] Under unix, all files accessed by LISTSERV must be named in lower case (i.e., 'ls' must show site.catalog, not SITE.CATALOG or Site.Catalog). Internally LISTSERV does not care about case since it translates everything to lower case for the purpose of accessing the unix file system, and you may use upper or mixed case within the catalog file itself.

```
*  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.

8.3.4. VM only: 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**), logical 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.

8.3.5. VM only: File Access Codes (FAC) for user access

FACs define which users have access to files in the file archive. The FAC for **GET** indicates who may retrieve the files, and the FAC for **PUT** indicates who may store the files on the server. (Note that some special FACs exist for "superusers" such as the LISTSERV maintainer(s) and the LISTSERV Master Coordinator, who may **GET** and **PUT** any file regardless of its **GET/PUT** permissions.)

The basic FAC codes that are always available for VM servers are:

ALL	universal access.
PRV	only members of the associated mailing list have access.
OWN	only the owners of the associated mailing list have access.

(The FAC codes **PRV** and **OWN** work only on the VM filelist system. They do not work on the non-VM catalog system. See section 8.4 if you are configuring the non-VM systems.)

(Note that this assumes the name of the filelist is identical to the name of the associated mailing list – for instance, MYLIST@FOO.BAR.EDU would have a **MYLIST LIST** file and a **MYLIST FILELIST** file. Ask your LISTSERV maintainer for assistance if this is not the case or if you need special FACs added for special user access to files.)

8.3.6. VM only: 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.

8.3.7. VM only: Storing the filelist

1. Create a mail message to LISTSERV at the appropriate host. (Sending a filelist to LISTSERV@LISTSERV.NET will not work. The filelist must be sent to the host it resides on.)

2. Include the filelist file as plain text in the body of the mail message. Do not attach it

with MIME or another encoding scheme, as LISTSERV does not translate encoded messages.

3. Make sure that your mail client does not automatically add a signature file to the bottom of your mail. If it does, your signature file will be treated as part of the filelist and will be stored along with it.

4. At the top of the filelist, add a single line as follows:

   PUT *filename* FILELIST PW=XXXXXXXX

   where **XXXXXXXX** is your personal password for LISTSERV on that host. Note that this is similar to the PUT command used when storing the list file.

5. Send the filelist to LISTSERV.

Once LISTSERV acknowledges the receipt and storage of the filelist, you can send the files that correspond to the file descriptors in your filelist. See section 8.5, below, for instructions.

8.4. The listname.CATALOG system on non-VM systems

NOTE: If you are running LISTSERV 1.8a or 1.8b, please refer to your Installation Guide or to the List Owner's Manual for LISTSERV 1.8b for information on maintaining your file server.

Documented restriction: The hierarchical listname.catalog system documented below is not available under LISTSERV Lite. You may store files on a Lite server for people to retrieve, but the files must be registered in the site.catalog file and must reside in the same directory with the *.list files so that LISTSERV can find them.

LISTSERV version 1.8c and later uses a file archive registration system similar to (but differing in important respects from) the old VM FILELIST system. This system is available on the VMS, unix, and Windows ports only. VM sites will continue to use the old FILELIST system indefinitely as it still offers more functionality than the new system.

Files to be made generally available to users (e.g., not specific to any one list on your server) should still be registered in the site.catalog file as before.

Prior to 1.8c, entries in site.catalog were written like this:

```
MY.FILE         my.file./home/lists/xyz      ALL JOE@XYZ.COM
```

In 1.8c a new "native" format for these entries was introduced, and the new format is used in all of the examples below. The old format remains supported for compatibility. However, note that you MUST use the old format if any of the directories in the path contains a period.

Documented restriction: All files manipulated by LISTSERV must be accessible through LISTSERV's OS-independent file access methods. This means that files whose name contains spaces or control characters (or, under unix, upper case characters) cannot be accessed. Similarly, files whose name does not contain a period cannot be manipulated by LISTSERV. There is no limit on the length of the file name, only on its contents. Note that these "system filenames" are not visible to the end users, who refer to the files by the names assigned in the catalog.

8.4.1. Adding files to the SITE.CATALOG

This is the most basic way to add files to LISTSERV's file archive system so they can be made available to users via the **GET** command.

To register a new file to the server on workstation systems, the LISTSERV maintainer adds a line to the SITE.CATALOG file. If SITE.CATALOG does not already exist (it is not shipped with the installation kits), simply open a new *ASCII text* file named **site.catalog** in the same directory as **system.catalog** and add entries to it as shown below. (Do not just add entries to **system.catalog** as this file will always be overwritten during a software update.)

Here is what a typical SITE.CATALOG entry looks like under Windows NT:

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

And the same entry under Unix would look like this:

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

(Note that under Unix, LISTSERV does *not* observe case-sensitivity internally. 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**. But note carefully that the physical files you store *must* be named in lower-case; in other words, the output of an '**ls**' command must show **my.file**, not **MY.FILE** or **My.File**. LISTSERV will handle this issue automatically when you **PUT** the files, but be forewarned if you store the files on the server via ftp or the Unix file system.)

Finally, here is an OpenVMS example:

```
MY.FILE     XYZ:[FILES]MY.FILE   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 contain only one period.

The second item, for instance **C:\FILES\XYZ\MY.FILE**, is the name LISTSERV will use for the actual disk file, in native OS format. 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 for non-VM servers (for VM FAC codes, see 8.3.5, above):

ALL	universal access.
CTL	only the LISTSERV maintainers have access.
PRIVATE(xxx)	only members of the xxx list have access.
OWNER(xxx)	only the owners of the xxx list have access.
SERVICE(xxx)	only users in the service area of the xxx list have access.
NOTEBOOK(xxx)	same access as the archives of the xxx list.
user@host	the user in question is granted access.

Except for **ALL**, which must occur on its own, multiple file access code entries can be specified, separated by a comma with no intervening space. For instance:

```
MY.FILE C:\FILES\XYZ\MY.FILE JOE@XYZ.EDU,JACK@XYZ.EDU,PRIVATE(XYZ-L) CTL
```

defines a file that Joe, Jack and the subscribers of the XYZ-L list can order via the **GET** command, but that only the LISTSERV administrator can update.

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.

8.4.2. Delegating file management authority

The sub-catalog enhancement allows the LISTSERV administrator to delegate file management authority in a controlled and secure manner. Multiple list owners can be given the authority to maintain their own sub-catalog in a predefined directory. With the LISTSERV-ISP add on (under development), a quota can be imposed on the directory in question.

The procedure works as follows:

1. The LISTSERV administrator creates the sub-catalog and identifies the directory where the files will be stored, and the person(s) who will be in charge of managing it ("catalog owners").

2. The catalog owners use the **GET** and **PUT** commands to update their catalog and register new files in their directory. Each file has the usual **GET** and **PUT** file access codes, allowing the catalog owners to further delegate the management of individual files to third parties ("file owners").

3. The file owners manage the files in question using the **GET** and **PUT** commands. Authorized users can retrieve the files using the **GET** command.

Note that this functionality is available in the VM version, using a different syntax. See Chapter 8.3, above, for information on managing the VM file archive system.

If you are migrating from VM to one of the non-VM versions of LISTSERV, please note that it is not necessary to create a subcatalog file for WELCOME, FAREWELL and MAILTPL files. If a subcatalog for these files is not created, they do not appear in the output of an INDEX command. However, there are two ways to force them to appear:

1. As the result of an **INDEX** command without qualifier: simply define the file in SITE.CATALOG.

2. As the result of an **INDEX** ***listname*** command: simply define the file in the *listname*.CATALOG.

8.4.3. Creating a sub-catalog

To create a sub-catalog, the LISTSERV administrator edits the file called SITE.CATALOG (or site.catalog under unix) in LISTSERV's main directory (the directory where SYSTEM.CATALOG/system.catalog is located). A sub-catalog is defined as follows:

```
MY.CATALOG      /home/lists/xyz/my.catalog   ALL JOE@XYZ.COM

(1)             (2)              (3)         (4) (5)
```

Notes:

(1) The name must end in '.CATALOG', but otherwise it can be anything. In particular, there does not need to be a list by that name.

(2) The directory specification indicated for the catalog file (e.g., **`/home/lists/xyz`**) is where ALL the files defined in the sub-catalog will be stored. **DO NOT USE LISTSERV'S MAIN DIRECTORY FOR THIS PURPOSE!** The catalog owner will be given FULL ACCESS to all the files in this directory, so make sure to create a new, empty directory. If the sub-catalog is being set up for a list owner, it may be a good idea to put the list archives and the sub-catalog in the same directory.

(3) A file name must be provided for the sub-catalog file itself. This name, however, does not need to match (1).

(4) This file access code controls the authority to INDEX the sub-catalog. This will also be the default GET access code for all the files registered in the sub-catalog.

(5) This file access code defines the catalog owner(s) and default file owner(s) for all the files in the sub-catalog.

Note that there is no need to reboot LISTSERV after updating the SITE.CATALOG file. Also, bear in mind that you are responsible for the OS-level security of the directory you create for the catalog. The file access codes in SITE.CATALOG only affect operations that go through LISTSERV; it is your responsibility to make sure that other users of the computer are given the appropriate access level to any directory you create for LISTSERV's purposes.

8.4.4. Updating the sub-catalog

Once the sub-catalog is created, the catalog owner(s) can register new files using the following procedure (in this example, it will be assumed that the sub-catalog is called MY.CATALOG):

1. Send a **`GET MY.CATALOG`** command to LISTSERV (or, if the catalog is brand new, start from an empty file).

2. Register new file(s) in the catalog (see below).

3. Use the **`PUT MY.CATALOG PW=XXXXX`** command to store the updated catalog.

Alternatively, if the catalog owner has an account on the LISTSERV host system and write access to the directory associated with the sub-catalog, the file can be edited directly. Note however that, in that case, the LISTSERV-ISP quota system will be inoperative as it has no control over disk accesses which do not go through LISTSERV itself.

The format of sub-catalogs is similar to that of SITE.CATALOG:

```
MY.FILE         my.file                      ALL JOE@XYZ.COM
(1)             (2)                          (3) (4)
```

Notes:

(1) This defines the name of the file as seen by LISTSERV users. That is, the command to retrieve the file will be GET MY.FILE.

(2) This defines the name of the actual disk file where the contents of MY.FILE will be stored. Normally, you should specify the same as (1), or just an equal sign (LISTSERV will then substitute the name you provided for (1)). However, in some cases you may want to make a particular file available under multiple names. This can be done by registering multiple files (ie multiple values for (1)), and using the same (2) value every time.

(3) This file access code determines who can order the file through a **GET** command. See section 8.4.1, above, for more information on FAC codes.

(4) This file access code determines who can update the file with the **PUT** command. See section 8.4.1, above, for more information on FAC codes.

Note: (2) defaults to the value of (1), and (3) and (4) default to the **GET** and **PUT** access codes of the sub-catalog itself, respectively. So, in most cases a sub-catalog entry will be as simple as:

```
MY.FILE
```

Additionally, comment lines (starting with an asterisk) or blank lines can be interspersed with file definitions. These comments will be echoed when the sub-catalog is indexed (see below), in sequence with the file definitions. For instance, your catalog could read:

```
*
* Files for the XYZ sub-project
*
XYZ.AGENDA
XYZ.BUDGET
XYZ.PROPOSAL-1
XYZ.PROPOSAL-2
```

8.4.5. Indexing the sub-catalog

If MY.CATALOG is defined as:

```
MY.CATALOG     /home/lists/xyz/my.catalog   xxx JOE@XYZ.COM
```

then any user who matches the 'xxx' file access code is authorized to issue an **INDEX MY** command to get a formatted version of the catalog. For compatibility with older versions of LISTSERV, **GET MY.FILELIST** will produce the same results. If there is a mailing list called MY, a list of the archive files will be appended automatically.

8.5. Storing files on the host machine

Please note that LISTSERV does not currently recognize "attachments" created by many popular mail clients as files to be stored with the **PUT** command. Such files must be part of the body of the message that contains the **PUT** command. This means that binary files

must be stored either in 7-bit format (uuencoded, etc.) or ftp'd to the server and placed in the appropriate directory by the LISTSERV maintainer or other privileged user.

If you store binary-format files on the server, you should be careful to note in the file catalog or filelist that users who want to **GET** the files will need to use an **F=** modifier (e.g., **GET BINARY.FILE F=MIME/APPL**) when ordering them by e-mail.

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 (square brackets indicate optional parameters):

PUT *filename extension* [*filelist*|*catalogname*] PW=XXXXXXXX

(This line will not appear to people who GET the file from LISTSERV.) Replace **XXXXXXXX** with your personal password. If you specify the filelist or catalog name, do not put the square brackets around the name.

There are a couple of issues that need to be noted here:

- If the file you are going to store is registered in the sitewide catalog or filelist, do not specify the name of the catalog or filelist.

- If the file you are going to store is registered in a sub-catalog or filelist other than the sitewide one, you may have to specify the name of the sub-catalog or filelist in order to be able to store the file. This is because it is entirely possible that two lower-level filelists or catalogs may have files registered with the same name (for instance, README TXT). If LISTSERV has two sub-catalogs registered (for instance, MYLIST CATALOG and HISLIST CATALOG) that both have a file called README TXT registered, then a PUT README TXT command will tell LISTSERV to try and store the file in the first catalog it comes to in the hierarchy. If MYLIST CATALOG is registered before HISLIST CATALOG in SITE CATALOG, LISTSERV will try to store the file as if it belonged to MYLIST (which we assume is what you want). However, if HISLIST CATALOG is registered before MYLIST CATALOG (and many sites like to keep things in alphabetical order, so this is a most likely scenario), LISTSERV will try to store the file as if it belonged to HISLIST, and you will get an error stating that you aren't allowed to store the file.

- NOTE CAREFULLY that you MUST turn off your signature file (if one is enabled in your mail client) in order to successfully store files. If you do not, LISTSERV will store your signature file at the end of the file.

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* [*filelist*|*catalogname*] PW=XXXXXXXX

   (Replace **XXXXXXXX** with your personal password.) The same issues noted in 8.5 regarding the filelist/catalog name are operative here.

   NOTE CAREFULLY that you MUST turn off your signature file (if one is enabled in your mail client) in order to successfully delete files. If you do not, LISTSERV will store your signature file in place of the file you are trying to delete instead of deleting the file.

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 tell you that the file has been successfully deleted.

5. For VM Systems ONLY: **GET** the ***listname*** **FILELIST** for your list and delete the line for the file you've just deleted. **PUT** the ***listname*** **FILELIST** back on the server.

6. For Workstation and PC Systems ONLY: Get the ***listname*****.CATALOG** for your list and delete the line for the file you've just deleted. PUT the ***listname*****.CATALOG** back on the server. Note that this is not necessarily required since under non-VM, if the physical file does not exist, LISTSERV will not include it in the output of an INDEX command. This is primarily a housekeeping measure.

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.

If you are running LISTSERV under unix, Windows, or VMS, please skip the rest of this section as it does not pertain in any way to your implementation of LISTSERV.

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 ADD** or **FUI ADD** command, LISTSERV warns the user that it would be a good idea to password protect the subscription.

8.8. File "Packages"

This feature is available for VM (all versions) and non-VM (beginning with 1.8d).

You can define a group of files as a "package" that can be retrieved by users with a single **GET** command. First, ensure that all the files in the package are defined in the appropriate filelist and stored on the server as detailed above.

Next, create a file descriptor in the appropriate filelist or catalog for a file called ***filename*** **$PACKAGE** (or ***filename*****.$PACKAGE** for non-VM), where ***filename*** is the name you have chosen for the group of files. Be sure that the filetype or extension is $PACKAGE, with a leading $ sign, and store your filelist.

Now create the actual ***filename*** **$PACKAGE** file. At the top of the file you can insert comment lines beginning with asterisks, for example:

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

Following these comment lines, you insert lines for each of the files contained in the package. There are two ways to format entries in your $PACKAGE file:

- A "compatibility" mode that works on all platforms, and which is identical to the original method used on VM (and which VM servers still must use). In the compatibility mode the basic format for the entries is

  filename filetype filelist <optional_comments>

  for example,

  ```
  MYLIST   $PACKAGE MYLIST The packing list
  INTEREST FILE     MYLIST Interest groups
  NETIQUET FILE     MYLIST How to behave
  ANOTHER  FILE     MYLIST No comment
  ```

- In the second (new) mode for non-VM servers only, the entries are formatted like this:

  filename.extension <optional_comments>

  for example,

  ```
  MYLIST.$PACKAGE The packing list
  INTEREST.FILE   Interest groups
  NETIQUET.FILE   How to behave
  ANOTHER.FILE    No comment
  ```

Note that anything that is not the name of a file in the package must be commented out

with an asterisk in the leftmost column of the line. It is possible to create a package file without any comment lines at all, but this is not preferable in practice. Often users will get the package file itself just to see what is in it. You should include a reference to the package file itself so that the user will get a copy of the "packing list" to check against the files he receives from LISTSERV.

The final step is to send the package file to LISTSERV like any other file.

Now users can do one of two things:

1. They may get the entire package of files sent to them by sending LISTSERV the command **GET *filename* PACKAGE** (without the $ sign); or

2. They may request that LISTSERV send only the package file itself by sending LISTSERV the command **GET *filename* $PACKAGE** (with the $ sign).

Packages may be subscribed to with the **AFD** and **FUI** commands (VM only).

8.9. Where to find more information on File Archives

Other guides that refer to File Archive setup and maintenance are referenced in Appendix E, *Related Documentation and Support*.

8.10. Notebook Archives

Notebook archives are files in which postings to the list are stored (assuming that notebooks are enabled for the particular list). In general, they are managed automatically by LISTSERV, with certain functions left to the list owner(s). For instance, there is no need to register notebook archives in the ***listname*.FILELIST** or ***listname*.CATALOG**; this is taken care of automatically.

8.10.1. Setting up notebook archives for a list

Setting up notebook archives requires only a few steps:

1. Make sure that you have disk space for the notebook archives and that the directory in which they will reside has been created with appropriate security privileges. LISTSERV needs read and write access to any directory it uses for notebooks. Note that, for security reasons, LISTSERV will not create the directory if it does not exist.

2. Add the **Notebook=** keyword to the list header with appropriate settings. (If you are not the LISTSERV maintainer, you will have to ask the LISTSERV maintainer to do this for you.)

3. Store the list header back on the server.

For instance, let's assume you have a list called **MYLIST** running on a unix server and you wish to store its archives in a directory called **/usr/listserv/home/mylist-archive**. Notebooks are to be kept on a monthly basis and are to be available to anyone. Your Notebook= keyword would look like this:

```
* Notebook= Yes,/usr/listserv/home/mylist-archive,Monthly,Public
```

Note that only the LISTSERV maintainer may change the location of Notebook archives

(or change Notebook= No to Notebook= Yes). Anyone else attempting to PUT the list header after changing these values will result in the following message being sent in response:

```
The following problems have been detected in the list header:

* Notebook= ...
Error: The first two parameters of the "Notebook=" keyword may only be updated
       by the LISTSERV administrator.

Please refer to the list keyword documentation (available via the "INFO
KEYWORDS" command) for more information about keyword syntax.

PUT operation rejected, old list remains unchanged.
```

Figure 8.3. This output will appear either if an attempt is made to change "Notebook= No" to "Notebook= Yes", or if an attempt is made to change the location where notebook archives are stored on the server, by anyone who is not a LISTSERV maintainer.

Similar restrictions also apply to the Digest= keyword. See Appendix B for details.

8.10.2. Migrating old notebook archives to a new site (LISTSERV to LISTSERV)

If migrating old notebook archives from one LISTSERV site to another, you can simply ftp (in TEXT mode) the notebooks from the old host to the new host, put them in the directory reference in the Notebook= keyword settings, and LISTSERV will immediately recognize their presence. You can also migrate the notebooks with **GET** and **PUT**.

8.10.3. Migrating old notebook archives (non-LISTSERV to LISTSERV)

LISTSERV notebooks follow a modified VM MailBook format, which is as follows:

A line of 73 "=" signs (ASCII 0x3D)
RFC822 headers, starting with the Date: header[6]
Blank line (actually part of RFC822 headers)
Message body

For instance:

```
==========================================================================
Date:         Fri, 6 Mar 1998 17:05:01 -0500
Sender:       Test list <TEST@XXXXXX.NET>
From:         Nathan Brindle <nathan@XXXXXX.NET>
Mime-Version: 1.0
Content-Type: text/plain; charset="us-ascii"

This is a test.
==========================================================================
Date:         Thu, 12 Mar 1998 13:23:07 -0500
Sender:       Test list <TEST@XXXXXX.NET>
From:         Nathan Brindle <nathan@XXXXXX.NET>
Subject:      Test

This is another test
==========================================================================
Date:         Thu, 12 Mar 1998 13:24:58 -0500
Sender:       Test list <TEST@XXXXXX.NET>
From:         Nathan Brindle <nathan@XXXXXX.NET>
Subject:      Test 3
Mime-Version: 1.0
Content-Type: text/plain; charset="us-ascii"

Yet another test.
```

The last message in the archive is not followed by a separator line (in other words, the separator line is found at the beginning of each message, not at the end of each message).

If you can reformat your non-LISTSERV archives this way then you can rename them using standard LISTSERV filenames:

For monthly archives: ***listname*.log*yymm***
For weekly archives: ***listname*.log*yymmw***
For yearly archives: ***listname*.log*yy***

(where ***yy*** = 2 digit year, ***mm*** = 2 digit month, ***w*** = letter A-F denoting the week of the month), place them in the directory pointed to by the **Notebook=** keyword for the list, and LISTSERV will index them and make them available via the web archive interface and so on.[7] Note that in order for the web archive interface to notice new notebook files

---

[6] Note that by default, LISTSERV requires that the RFC822 Date: header be the first header in the message. If some other header or headers come before Date:, LISTSERV will not properly delimit the messages in the notebook.

You can work around this restriction for archives migrated from non-LISTSERV systems by using the list header keyword "Notebook-Header= Full"; however you must still remove unix mailbox separator lines like the one shown at the end of this section or LISTSERV will not be able to correctly index the notebook.

It should also be noted that lists with "Notebook-Header= Full" consume more disk space for very little gain, which is why the default is "Notebook-Header= Short", and why we suggest reformatting such migrated archives to the LISTSERV "short" header format.

[7] If your old archives are from Majordomo or ListProc you might want to look at an unsupported Perl script found at **ftp://ftp.lsoft.com/CONTRIB/notebook-conv.pl** which converts sendmail-style

you must either **GET** and **PUT** the list header or restart LISTSERV.

If a list owner is planning to store archive files via the **PUT** method, the LISTSERV maintainer must first make dummy files with the same filenames in the list's notebook directory so that LISTSERV will not say that the file does not exist and reject the **PUT** operation. However please note that you should *not* make entries for the notebooks in ***listname*.catalog** (if one exists). LISTSERV makes its list of notebooks "on the fly" every time an **INDEX** command is issued for the list.

If your old archives have lines at the beginning of each message like this:

```
From userid@host.com Thu Feb 2 15:27:02 1995
```

you should delete them; this is the message separator used by sendmail. LISTSERV does not use it and it may in fact cause problems with indexing if left in.

8.10.4. Deleting old notebook archives

The LISTSERV maintainer may delete old notebook archives that are no longer needed in one of two ways:

- Use standard file system commands from the console prompt to delete the files. On VM, use **CMS ERASE**; on Unix, **rm**; on VMS, **DEL**; on Windows systems, **DEL** or **ERASE**.

- Send a PUT command by itself (in essence, you are storing a zero-length file) via mail to LISTSERV. For instance:

  ```
  PUT MYLIST LOG9607 PW=mypersonalpw
  ```

  by itself would delete the file MYLIST LOG9607.

  Two important issues:

  1. This command MUST be issued by e-mail. It cannot be issued via the "Execute a LISTSERV command" facility of the web management interface.

  2. NOTE CAREFULLY that you MUST turn off your signature file (if one is enabled in your mail client) in order to successfully delete files. If you do not, LISTSERV will store your signature file in place of the file you are trying to delete instead of deleting the file.

8.10.5. Indexing existing notebook archives

LISTSERV creates the notebook archive index "on the fly" as required. If there is an existing ***listname*.FILELIST** or ***listname*.CATALOG**, it appends the index of notebook archives to the end of the index of other files. Otherwise, the index of notebooks is generated and sent by itself. The user simply issues the command **INDEX *listname*** to receive the index of available files and notebooks.

---

notebooks to LISTSERV format. This is really ListProc-specific but it would probably work with Majordomo archives. In any case this user-contributed script is not supported in any way by L-Soft, so please direct any questions about it to its authors.

9. Creating and Editing LISTSERV's Mail and Web Templates

9.1. What LISTSERV uses templates for

Templates are used to generate some of the mail LISTSERV sends to users in response to commands it receives. Among these are the "You are now subscribed . . ." message, the message sent to users when LISTSERV cannot find a subscription for them in a specified list, and others. Note that certain administrative mail (for instance, the response to the STATS and RELEASE commands) is hard-coded into LISTSERV and cannot be changed.

Other templates are used to generate the HTML code used by the web archive and administration interfaces.

A word about nomenclature: When we talk about "templates" we are talking about "files that contain one or more template forms", in other words, files like DEFAULT MAILTPL or DEFAULT WWWTPL. A "template form" is an individual section of a template which begins with a title line (three ">" symbols followed by a space, the name of the template form, and (optionally) a short description of the template, which for some template forms is also used as the subject of the mail LISTSERV constructs with the template form), followed by one or more lines of copy and/or imbedded commands, and ends at the next title line or the end of the file, whichever is reached first. A template may contain one or more template forms.

9.2. The default template files and how to get copies

LISTSERV stores its default mail template information in a file called DEFAULT MAILTPL, which can be requested by list owners and LISTSERV maintainers from LISTSERV with the GET command, just like any other file. The LISTSERV maintainer will find this file in LISTSERV's "A" directory (usually **`~listserv/home/default.mailtpl`** on unix, **`LISTSERV\MAIN\DEFAULT.MAILTPL`** on Windows systems, and **`LISTSERV_ROOT:[MAIN]DEFAULT.MAILTPL`** under VMS). Note that DEFAULT MAILTPL contains some (but not all) of the web interface template forms.

LISTSERV stores the rest of its default web interface template forms in a file called DEFAULT WWWTPL, which can be retrieved in a manner identical to that for DEFAULT MAILTPL.

Note that it is considered unwise (and it is not supported) to modify the contents of DEFAULT MAILTPL or DEFAULT WWWTPL themselves, as these files will be overwritten by upgrades. It is possible to make sitewide changes that will not be overwritten without disturbing either of these files.

Under 1.8d and following, all template forms may be edited using the web administration interface described in chapter 11. Edited template forms are placed in template files that will not be overwritten by software upgrades.

9.3. Mail template format and embedded formatting commands

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

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

Please note carefully the following instructions for the form name and subject line:

- The template form starts with the line containing the form name and subject, and ends with the next line starting with '**>>>**', or at the end of the file.
- The subject line may contain substitutions (such as "**&LISTNAME: &WHOM requested to join**").
- Ensure that there is a blank space (ASCII 0x20) between '**>>>**' and the name of the form, or LISTSERV will not recognize the form.
- Also note that the names of the template forms *must* be typed in **UPPER CASE**.

A template form contains text and, optionally, formatting/editing commands, which start with a period in column 1. All other lines are treated as normal text: sequences starting with an **&** sign are substituted, then lines are joined together to form a paragraph, which is finally formatted like with any non-WYSIWYG text processor. You can suspend formatting with **.FO OFF** and resume it with **.FO ON**; when formatting is suspended, LISTSERV no longer joins lines to form a paragraph, but simply writes one line of text to the message for each line read from the template form. This makes it possible to include tables or a text-mode logo, but can create seriously imbalanced text if substitutions are used. For instance, a typical **&WHOM** substitution can range from a dozen characters to 60 or more, even though it only takes up 5 characters on your screen when you enter it.

The following substitutions are always available:

&DATE	Long-style date (04 Jan 1998)
&TIME	hh:mm:ss
&WEEKDAY	Three-letter day of the week, in English
&MYNAMES	The substitution you will use most of the time when you need to refer to LISTSERV. For Internet-only or BITNET-only servers, this will display LISTSERV's only e-mail address. For servers with both Internet and BITNET connectivity, it will say "LISTSERV@*hostname* (or LISTSERV@*nodeid*.BITNET)".
&MYSELF	LISTSERV's address, in the form LISTSERV@XYZ.EDU or, if no Internet hostname is available, LISTSERV@XYZVM1.BITNET.
&MYNODE	LISTSERV's BITNET nodeid, without the '**.BITNET**', or its Internet hostname if no NJE address is available.
&MYHOST	LISTSERV's Internet hostname or, if none is available, its NJE address (with '**.BITNET**').
&MBX(*addr*)	Looks up the specified address in LISTSERV's signup file and displays "name <addr>" if a name is available, or just the original address otherwise. This is typically used to give the name of the command originator or target, along with his e-mail address: **&MBX(&WHOM)** or **&MBX(&INVOKER)**. Please note however that **&WHOM** and **&INVOKER** are not always available in every template.
&RELEASE	LISTSERV's release number (e.g., "1.8e").
&OSTYPE	The operating system under which LISTSERV is running.
&OSNAME	The full operating system name including the version number, e.g., "VM/ESA 1.2.3", "Windows NT 4.0", "Linux 2.0.27", "SunOS 5.4", etc.

&HARDWARE The type of machine LISTSERV is running on, e.g., "Pentium (512M)".

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

&LISTNAME Either the short or long name of the list based on the value of "List-Address=" and/or its system default. By default the long ("List-ID=") name is used if present.

&TITLE Title of the list, or empty string.

&KWD(*kwd*) Value of the specified keyword for the list. You do not need to specify the name of the list - it is implicit. You need not put quotes around the keyword names either, although quotes will be accepted if present. Optionally, you can specify a second numeric argument to extract just one of the terms of a list header keyword; for instance, if the list header contains "Notebook= Yes,L1,Monthly,Private", **&KWD(NOTEBOOK,4)** has the value "Private". A third argument, also optional, specifies the default value for the keyword in case it was not initialized. It is meant to be used for conditional formatting in the default templates and list owners should not worry about it.

&LITE (1.8c and following) Has the value 1 when running the LISTSERV Lite product, and 0 otherwise. This variable can be used to write generic templates that account for the differences between the two products.

&ISODATE (1.8c and following) Returns today's date in ISO format, i.e., yyyy-mm-dd.

&DAYSEQ(*n*) (1.8c and following) Used to create FAQ templates with rotating topics. May also be used to create bottom banners with rotating text (e.g., for lists with multiple commercial sponsors who get "ad space" in the banner on a rotating basis).

In addition, many template forms have their own specific substitutions, meaningful only in their specific context. For instance, a message informing a user that he was added to a mailing list may have an **&INVOKER** substitution for the address of the person who issued the ADD command. This is not meaningful for a template form intended to inform a user that he must confirm his subscription to a list within 10 days, so it is not generally available. If you attempt to use a substitution which is not available, the template processor writes an error message to the mail message it is generating, but sends it anyway, in the hope that the recipient will be able to figure out the meaning of the message in spite of the error. If you need to include a sentence with an ampersand character, you will have to double it to bypass the substitution process, as in "**XYZ &&co.**"

The mail template processor also supports HTML-like variable closure, in addition to the traditional LISTSERV closure (both methods are supported concurrently; there is no need to select one over the other). For example:

Traditional: **For more information, please send mail to &EMAIL or call &PHONE.**

HTML: **For more information, please send mail to &EMAIL; or call &PHONE;.**

Previously, HTML writers who used HTML closure conventions would not get the expected results. This change makes it easier for webmasters to get the desired results the first time.

Any line starting with a period in column 1 is processed as a formatting command. Note that neither substitutions nor formatting commands are case sensitive. Here is a list of the formatting commands list owners may need to use:

.* Comment: anything on this line is simply ignored. This is useful for recording changes to template files when there are multiple owners. Just add a comment line with the date and your initials every time you make a change, for the benefit of the other owners.

.FO OFF Turns off formatting: one template line = one line in the final message. You can resume formatting with **.FO ON** or **.FO RAGGed**. (**.FO RAGGed** requires LISTSERV 1.8e-2002a or later, that is, build date of 31 October 2002 or later)

.FO RAGGed Changes right-justified text formatting to left justified text formatting. You can resume right-justified formatting with **.FO ON**. (**.FO RAGGed** requires LISTSERV 1.8e-2002a or later, that is, build date of 31 October 2002 or later)

.CE text Centers the text you specify (just the text you typed on the same line as the **.CE** command). This can be useful to highlight the syntax of a command.

.RE OWNERS Adds a 'Reply-To:' field pointing to the list owners in the header of the generated message. Use this command when you think users are likely to want to reply with a question. You can also use **.RE POSTMASTER** to direct replies to the LISTSERV administrator, if this is more appropriate.

.CC OFF Removes all "cc:" message recipients, if any. You can also add message recipients by specifying a series of e-mail addresses after the **.CC** statement, as in **.CC JOE@XYZ.EDU**. PC mail users should note that in this context "cc:" is a RFC822 term that stands for "carbon copy". RFC822 messages may have "cc:" recipients in addition to their "primary" recipients. There is no real technical difference between the two, the "cc:" indicator just denotes a message that is being sent for your information. Some administrative messages sent to list owners are copied to the user for their information, and vice-versa; this behavior can be disabled by adding a **.CC OFF** statement to the template.

.TO Replaces the default recipients of a message with the value specified. For instance, if you use the ADDREQ1 template form to send new subscribers a questionnaire, application form or similar material, you will need to add a '.TO &WHOM' instruction to your modified template form, as by default the user will not receive a copy.

.QQ Cancels the message. LISTSERV stops reading the template form and does not send anything. This is useful if you want to completely remove a particular message; note however that this can be confusing with certain commands, as LISTSERV may say "Notification is being sent to the list owners" when in fact nothing will be sent because of the **.QQ** command in the template form.

.QU (Starting with 1.8e) Ends processing of the current template as if you had reached the end, but *without* cancelling the message. The main purpose is to avoid multi-level nested .BB/.EB conditional blocks (see below) that are hard to keep track of.

A number of more advanced commands are available to list owners with more sophisticated needs and some programming experience. If you encounter one of these commands in a template, you will probably want to leave it alone.

.IM name Imbeds (inserts) another template form at this point in the message. This is used to avoid duplicating large pieces of text which are mostly identical, such as the templates for "you have been added to list X by Y" and "your subscription to list X has been accepted".

As noted below, LISTSERV will not pick up an "imbedded" template form from $SITE$.MAILTPL. If you wish to include an "imbedded" template form (e.g., **$SIGNUP**) in $SITE$.MAILTPL, you must also include the template form that calls it with the **.im** command.

.DD ddname Copies the contents of the specified DD into the message. This is meaningful only if a DD has been set up by LISTSERV for this purpose. As a rule of thumb, you should either leave these statements unchanged or remove them.

.BB cond Begin conditional block. The boolean expression following the keyword is evaluated and, if false, all the text between the **.BB** and **.EB** delimiters is skipped. Conditional blocks nest to an arbitrary depth. The expression evaluator is recursive but not very sophisticated; the restriction you are most likely to encounter is that all sub-expressions have to be enclosed in parentheses if you are using boolean operators. That is, "**.BB &X = 3**" is valid but "**.BB &X = 3 and &Y = 4**" is not. String literals do not require quoting unless they contain blanks, but quotes are accepted if supplied. Comparison operators are **= <> ^= IN** and **NOT IN** (the last two look for a word in a blank-separated list of options, such as a keyword value). These operators are not case-sensitive; **==** and **^==** are available when case must be respected. Boolean operators are **AND** and **OR**. Note that a conditional block must be contained on one physical line and may not wrap, so be careful when sending MAILTPL files back to LISTSERV that you do not accidentally wrap long **.BB** lines.

Starting with LISTSERV 1.8d the operators **=*** and **^=*** are available to perform wildcard matches in conditional blocks. For instance **JOHN_DOE@UNIX.EXAMPLE.COM =* J*DOE@*EXAMPLE.COM** is a true statement. The wildcard specification is on the right-hand side whereas the actual text (or variable) you are evaluating is on the left.

.EB End conditional block (see **.BB**).

.QU Stop (in other words, QUit) processing of the current template as if you had reached the end, but without cancelling the message. The main purpose is to avoid multi-level nested **.BB/.EB** conditional blocks that are hard to keep track of. Available in 1.8e and following.

.SE var text Defines or redefines a substitution variable. This is convenient for

storing temporary (text) expression results which need to be used several times. Even standard variables such as **&LISTNAME** can be redefined - at your own risk. You must enclose the text expression in single quotes if you want leading or trailing blanks.

.CS text Define a (non standard) character set for the template in question, i.e.,

```
.CS ISO-8559-7
```

This setting is ignored if the template does not actually contain special characters (for instance, if the template is written in 7-bit ASCII). Otherwise the appropriate headers are created for the message in question when it is sent out.

.TY text Types one line of text on the LISTSERV console log. This can be useful to the LISTSERV maintainer for debugging, and also to record information in the console log.

.ASIS text Tells LISTSERV to leave the text immediately following the .ASIS directive alone, that is, don't convert "<" and ">" characters into HTML < and > when creating pages. This is specifically for use in HTML templates where it is important not to convert parts of a URL reference. For instance,

```
.ASIS Click <a href="http://some.host.com/some-doc.html">here</a>.
```

As with the **.CE** directive, the text you intend to affect with the **.ASIS** directive must not wrap. The **.ASIS** directive will only work on text it finds on the same physical line into which it is coded.

9.3.1. 8-bit characters in templates

Starting with 1.8d, if you include 8-bit characters (e.g., accented or national language characters) in templates, LISTSERV will automatically encode the templates on-the-fly using MIME quoted-printable encoding. While there is no guarantee that every mail program will be able to properly display 8-bit characters, those mail programs that do understand MIME quoted-printable encoding should have no trouble doing so.

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

Please note that list-level mail templates are not available in LISTSERV Lite.

Make a copy of DEFAULT.MAILTPL on your local machine and name it *listname*.MAILTPL.[8] Keep the original DEFAULT.MAILTPL around in case you make a mistake and need to start over.

At this point, you could theoretically store the *listname*.MAILTPL back on the LISTSERV host. However, without making any changes that would be somewhat pointless. At the very least you should edit the INFO template form before storing the template. Note also

[8] If your local machine is running MS-DOS and/or Windows 3.x, obviously this will not work--you will have to conform to the 8.3 naming convention. Probably the best thing to do in this case is simply name the file *listname*.mai, then rename it when you upload it to a mainframe or network workstation account (or use the web administration interface described in chapter 11, instead).

that you need only store the sections of the template that you have changed. For instance, if you edit the INFO template form but leave the rest of the template untouched, you can delete the rest of the template and store the INFO template form alone as *listname*.MAILTPL. The benefit to this approach is that any administrative changes to the rest of the default template are automatically applicable to your list as soon as they are made, rather than requiring that you edit your mail template individually to reflect such changes. L-Soft recommends that this approach be followed as the default.

Under LISTSERV 1.8d and following it is not necessary to do the GET and PUT; you can edit individual template forms by using the web administration interface (described in chapter 11) instead.

9.4.1. The INFO template form

The first section of DEFAULT.MAILTPL is called the INFO template form, and it is LISTSERV's response to the command **INFO *listname***. By default, it contains the following:

```
 >>> INFO Information about the &LISTNAME list
There is no information file for the &LISTNAME list. Here is a copy of
the list "header", which usually contains a short description of the
purpose of the list, although its main purpose is to define various
list configuration options, also called
"keywords". If you have any question about the &LISTNAME list, write to
the list owners at the generic address:

.ce &LISTNAME-Request@&MYHOST

.dd &LISTHDR
```

Figure 9.1. The default contents of the INFO template form of DEFAULT.MAILTPL.

Note the replaceable parameters **&LISTNAME** and **&MYHOST**. Don't change **&MYHOST**; LISTSERV replaces it with the correct value for the name of the host site. **&LISTNAME** automatically inserts the name of the list. It's probably best to use **&LISTNAME** to refer to the list throughout the document rather than to replace it with something like "MYLIST-L". This ensures that the template form will be consistent with the default and will be simpler to debug should a problem arise. Also, in the event the name of the list changes, it will be unnecessary to edit the template form (although it would have to be renamed to match the new name of the list, of course).

Should it be desirable to replace the default INFO template form with information about the list, it is probably best to remove the **.dd &LISTHDR** line. This line instructs LISTSERV to read in the header of the list and add it to the response in lieu of any other data about the list. Many list owners add descriptive comment lines to their list headers, thus this default.

Here is a minimally-edited sample INFO template form for a list called MONKEYS:[9]

```
 >>> 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:
```

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

```
.ce &LISTNAME-Request@&MYHOST
```

Figure 9.2. Sample edited INFO template form.

9.4.2. Other available template forms

Traditionally, message templates have contained the text of "long" administrative messages, such as messages informing subscribers that they have been removed from a mailing list. These notices were sent unconditionally, as a separate message. Since 1.8b, the template processor has supported "linear" messages, which are sent as a normal command reply and allow the list owner to modify the replies from selected commands, and "optional" messages, which are only sent if a template for this action has been specifically provided by the list owner.

In a linear message, most special instructions are ignored. This is because the contents of the template form are just a few lines out of a larger message that is being prepared by LISTSERV to contain the reply to the user's command(s). For instance, you do not have any control over the "Reply-To:" field of the message, because the message in question is shared with other commands and, in fact, may not be a mail message at all but an interactive message to the user's terminal, a GUI request, etc. Generally speaking, with a linear message you are providing the TEXT of the reply to be shown to the user, but you do not have any control over the methods used for delivering this information.

Here is a list of all of the template forms (other than **INFO**, described above) available in DEFAULT.MAILTPL, in the order in which they appear and with a short description for each. Linear and optional template forms are noted where applicable.

- **MOVE1:** Usually active only for peered lists. This message is sent to the subscriber when the list owner or LISTSERV maintainer changes which peer the subscriber receives his or her mail from.

- **SIGNOFF1**: a notification to the list owner that someone has unsubscribed from the list. Whether or not the list owner receives this notification is controlled by the "Notify=" list header keyword.

- **SIGNOFF2**: this message is sent to any user who attempts to unsubscribe from a list to which he or she is not subscribed under the userid from which the unsubscribe command has been sent. For instance, joe@unix1.somehost.com may be subscribed to list MYLIST-L. If his Pine client is set so that his mail comes from his root domain (e.g., joe@somehost.com), he will get this message if he tries to unsubscribe from MYLIST-L.

- **DELETE1**: the message sent when a list owner or the LISTSERV maintainer deletes a user from a list. You can suppress the sending of this message by prepending "QUIET" to your "DELETE" command.

- **AUTODEL1**: this is the message that is sent to users who are deleted by the delivery error monitor. You can customize it to fit your needs, or suppress it for your list by simply redefining it in the 'listname.MAILTPL' and using the **.QQ** instruction:

  ```
  >>> AUTODEL1 This message is not wanted for our list
  .QQ
  ```

  Note that L-Soft does not generally recommend suppressing this message, as it may indicate a serious problem for the deleted subscriber.

- **ADD1**: the message sent when a list owner or a LISTSERV maintainer manually adds a subscriber to a list.

- **ADD2**: the message sent to the list owner(s) when someone subscribes to their list. As with **SIGNOFF1**, whether or not the list owner(s) receive this message is controlled by the "Notify=" list header keyword.

- **ADDREQ1**: this message is sent to the list owner when a user requests to join a list with "Subscription= By_Owner". Only the list owner is sent a copy of the **ADDREQ1** message. If you use this template form to send new subscribers a questionnaire, application form or similar material, you will need to add a '**.TO &WHOM**' instruction to your modified template form, as by default the user does not receive a copy.

- **SETINFO**: the message sent to the subscriber when the list owner or LISTSERV maintainer changes their personal subscription options. Can be suppressed by the invoker with the use of the "QUIET" command modifier.

- **CHANGE1**: the message sent when a list owner or LISTSERV maintainer uses the CHANGE command to change a subcriber's address.

- **ADDMOD2**: the message sent to the subscriber when the list owner or LISTSERV maintainer changes the subscriber's "real name" field in the SIGNUP database.

- **ADDPW1:** the message sent to the user when a LISTSERV maintainer adds a personal password for that user.

- **ADDPW2:** an informational message sent to the LISTSERV maintainer when a user adds or changes his password, but only if an LSV$PW exit has been enabled to do so. Most installations will never use this template form, but it should not be deleted from DEFAULT.MAILTPL in any case.

- **ADDPW3:** an information message sent to the LISTSERV maintainer when a user tries to add or change his password, but only if an LSV$PW exit has been enabled to do so. Most installations will never use this template form, but it should not be deleted from DEFAULT.MAILTPL in any case.

- **DELPW:** the message sent to the user when a LISTSERV maintainer deletes that user's personal password.

- **RENEW1:** this message is sent to subscribers whose subscriptions are due for renewal (see the Renewal= list header keyword for more information).

- **RENEW2:** this message is sent to subscribers who did not renew their subscriptions within the grace period after being notified that their subscription was due for renewal.

- **SIGNUP1:** the basic "Your subscription request has been accepted" message.

- **$SIGNUP:** a template form included with **SIGNUP1** and **ADD1** (assuming that **SIGNUP1** and **ADD1** template forms include an ".im $SIGNUP" line, which by default they do) which gives the subscriber a basic outline of how to use the list, how various options are set, and where to get more information on using LISTSERV. This template can be used in lieu of a WELCOME file for a list if the list owner doesn't

want two messages to go to the user at subscription time.

- **SUB_CLOSED** (linear): this is the message that is sent to a subscriber attempting to join a list with "Subscription= Closed". The default is "Sorry, the **&LISTNAME** list is closed. Contact the list owner (**&OWNER**) for more information."

- **SUB_NEWNAME** (linear): this message is sent to a current list subscriber who has issued a new **SUB** command in order to change his "real name" field in the SIGNUP files.

- **SUB_OWNER** (linear): this message is sent to a subscriber attempting to join a list with "Subscription= By_Owner". The default is "Your request to join the **&LISTNAME** list has been forwarded to the list owner for approval. If you have any question about the list, you can reach the list owner at **&OWNER**." Because this is a linear template form (see above), it is not the best place to put long questionnaires, application forms, terms and conditions, or other material that the subscriber should be required to review prior to joining the list. See the "Tips" section below.

- **POST_EDITOR** (linear): this is the message LISTSERV sends to people attempting to post to the list, if it is moderated. The default is "Your **&MESSAGE** has been submitted to the moderator of the **&LISTNAME** list: **&MBX(&MODERATOR)**."

- **REQACK1**: this message is sent automatically in reply to any message sent to the xxx-request address. The message acknowledges receipt, explains the difference between the LISTSERV and xxx-request addresses, and contains instructions for joining and leaving the list. To suppress this message for your list, simply redefine it in the '*listname*.MAILTPL' and use the **.QQ** instruction:

  ```
  >>> REQACK1 This message is not wanted for our list
  .QQ
  ```

- **REQNAK1**: (1.8e and following): this message is sent automatically in reply to any message sent to the special ALL-REQUEST address by a user who is not authorized to post to that address.

- **CONFIRM1**: The message sent whenever an "OK" confirmation is required.

- **WWW_INDEX**: this template form is used by sites which have implemented LISTSERV's WWW archive interface. It includes the HTML code for the main archive access screen for the list. List owners should probably should leave this alone unless they know exactly what they are doing.

- **WWW_REBUILD_ALL**: This template is used internally by LISTSERV and should NEVER be edited by end users.

- **PROBE1**: this template form is sent as part of LISTSERV's new bounce processing feature if this feature is activated for your list. The desired response from the user is to discard the message and do nothing. See chapter 4.6.2 of the *List Owner's Manual* or chapter 13.5 of the *Site Manager's Operations Manual* for details on the "Probe" option.

- **PROBE2**: If the mail containing the **PROBE1** message bounces, this template form is sent along with a copy of the bouncing mail. See chapter 4.6.2 of the *List Owner's Manual* or chapter 13.5 of the *Site Manager's Operations Manual* for details on the

"Probe" option. (If you have **Auto-Delete= ...,Delay(0)**, **PROBE2** is *not* sent, rather the bouncing user is deleted immediately.)

Several template forms for the WWW archive interface follow **PROBE1**. For more information on these template forms, see section 9.7, below.

- **HTML_DIGEST**: Preamble for HTML digests (for users who have set the personal subscription option **HTML**)

- **HTML_INDEX**: Preamble for HTML indexes (for users who have set the personal subscription option **HTML**)

- **BAD_CONTENT** (linear, 1.8e): If a posting to a list violates one of the content rules defined in the optional **CONTENT_FILTER** mail template form (see 7.18, above) and is rejected, this template form is sent back to the poster.

- **BAD_ATTACHMENT** (linear, 1.8d 2000a): If a posting to a list contains an attachment of a type not allowed by the "Attachments=" setting for the list, this template form is sent back to the poster. In 1.8e this template form is also sent if virus scanning is enabled and a virus is detected in the posting.

- **DIST_VIRUS** (linear, 1.8e): Assuming that the anti-virus feature introduced in 1.8e is enabled, if LISTSERV detects a virus in a DISTRIBUTE job, the message is returned to sender along with this template text.

- **SIZELIM_EXCEEDED** (linear, 1.8e): This template form is used if a posting to a list exceeds the limit set by the Sizelim= list header keyword.

The following are template forms that can be defined, but which are not present in DEFAULT.MAILTPL. If you want to define them for a particular list, the easiest way to do so is via the web interface.

- **POSTACK1** (optional): when present, this message is sent in reply to any message posted to the list. This is very useful for creating "infobots", or just for returning a standard acknowledgement to contributors. The **&SUBJECT** variable contains the subject of the original message, and naturally the usual substitutions (**&LISTNAME**, **&DATE**, **&TIME**) are available.

- **TOP_BANNER**, **BOTTOM_BANNER** (optional): when these template forms are present, their contents are automatically inserted at the top (respectively bottom) of each and every message posted to the list. Typically, the top banner would be used for a copyright or short legal warning which absolutely has to be seen by each and every reader. The bottom banner could contain instructions for signing off the list, a disclaimer, an acknowledgement of a sponsor's contribution, a "tip of the week", etc.

  Documented Restriction: *The use in banners of substitutions which do not yield a constant result (e.g.,* **&TIME***) will defeat the duplicate mail detection part of LISTSERV's loop-checking heuristics in any case where a subscriber is forwarding all mail back to the list. L-Soft advises that such substitutions never be used in a* **TOP_BANNER** *or* **BOTTOM_BANNER**.

  Prior to 1.8e, for digests, note that the **BOTTOM_BANNER** is printed only once, at the top of the digest, directly following the table of contents. This avoids having the banner repeat after every message in the digest. The default behavior can be

overridden if preferred by adding the "BOTTOM_BANNER" parameter to the **Digest=** list header keyword.

In LISTSERV 1.8e, a major change to the way LISTSERV attaches banner messages to postings changed the digest behavior described above. The change lets LISTSERV correctly insert banners in MIME messages, and solves certain quoted-printable error and message rejection problems observed in the previous versions. However, LISTSERV no longer attempts to remove bottom banners from individual messages in digests. In 1.8d, new banners were not actually inserted into the digest, so they appeared to have been successfully removed. While banner removal did work with single-part, unencoded messages, this was the only case when it worked. In 1.8e banners are inserted in a different way which precludes attempting to remove existing banners when the digest is generated.

Note that the **Digest=** keyword's "BOTTOM_BANNER" override parameter still works in 1.8e, insofar as it will prevent the banner from being printed at the top of the digest.

- **TOP_BANNER_HTML**, **BOTTOM_BANNER_HTML** (optional, 1.8e and following): When these template forms are present, they will be used "as is" for HTML message parts. If absent, the regular banner is used for HTML, probably with less than 100% satisfaction.

  Documented Restriction: *The use in banners of substitutions which do not yield a constant result (e.g.,* **&TIME***) will defeat the duplicate mail detection part of LISTSERV's loop-checking heuristics in any case where a subscriber is forwarding all mail back to the list. L-Soft advises that such substitutions never be used in a* **TOP_BANNER_HTML** *or* **BOTTOM_BANNER_HTML**.

- **CONTENT_FILTER** (optional, 1.8e and following): When present, provides LISTSERV with a ruleset for message content filtering that can be configured at the list level. See chapter 7.18, above, for more information on how to use content filtering.

9.4.3. Tips for using templates

- Many list owners require prospective subscribers to fill in a little questionnaire before being added to the list, or to explicitly state that they have read the list charter and agree to follow all rules or be removed from the list. The most convenient method, for both list owner and subscriber, is to have the SUBSCRIBE command return a copy of the questionnaire (or list charter, etc), and not forward the request to the owner. The user answers the questions and returns them directly to the list owner, who then adds the subscriber manually. Naturally, it is more convenient for the user if this information arrives in a separate message, with a 'Reply-To:' field pointing to the list owner's address. Thus, you should not use the **SUB_OWNER** template form for this purpose, because it is a linear template form and does not give you any control over the 'Reply-To:' field. The **SUB_OWNER** template form could be modified to read "A copy of the list charter is being sent to you, please read it carefully and follow the instructions to confirm your acceptance of our terms and conditions." The list charter would then be sent separately, through the **ADDREQ1** template form. You would use the **.RE OWNERS** command to instruct LISTSERV to point the 'Reply-To:' field to the list owners, and **.TO &WHOM** to change the destination from list owner to subscriber. If you want to receive a copy of the message, you can use **.TO &WHOM cc: xxx@yyy**.

- When writing template forms, it is a good idea to use substitutions (**&XXXX**) for information which may change in the future. In particular, it is not uncommon for lists to have to be moved from one host to another, and this will be a lot easier if the template forms use substitutions for the list address and list host. The **&LISTADDR** substitution translates the full address of the list (XYZ-L@XYZ.COM), whereas **&LISTNAME** is just the name (XYZ-L). For references to the server and host, use **&MYHOST** for the Internet hostname, **&MYSELF** for the server address (normally **LISTSERV@&MYHOST**), and **&OWNER** for the xxx-request mailbox address. These substitutions are "universal" and can be used in all template forms. For instance, if you decide to make a bottom banner with instructions for leaving the list, the text could read: "To leave the list, send a SIGNOFF **&LISTNAME** command to **&MYSELF** or, if you experience difficulties, write to **&OWNER**."

9.5. Storing the <listname>.MAILTPL file on the host machine

The procedure differs slightly on VM systems, but the following will work for unix, VMS and Windows systems:

1. Get a copy of DEFAULT.MAILTPL and edit it.

2. Be sure that you have defined a "personal password" to LISTSERV with the **PW ADD** command before you **PUT** the template file. If you have done this but can't remember the password, send a **PW RESET** command to LISTSERV, then a new **PW ADD** command.

.

3. Send the file to LISTSERV with a **PUT *listname* MAILTPL PW=XXXXXXXX** command at the top of the file, just as if you were storing the list itself. Replace **XXXXXXXX** with your personal password.

The variation for VM systems is that the LISTSERV maintainer will have to create a fileid for the file before it can be **PUT** on the server and seen by LISTSERV.

Note also that the LISTSERV maintainer can create and edit these files in place with any standard text editor. Changes made to template files in this way are available to LISTSERV immediately after they are saved.

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. The DIGEST-H and INDEX-H files are plain text files, like the WELCOME and FAREWELL files, and the instructions for storing them on the server are identical. Note that, as with the WELCOME and FAREWELL files, you cannot use the template formatting commands and replaceable parameters discussed above.

A typical DIGEST-H or INDEX-H file for a list called MYLIST might contain:

```
The MYLIST list is sponsored by ABig Corporation.

See http://www.abig.com for information on ABig Corporation's products.
```

Figure 9.3. Typical contents of a DIGEST-H or INDEX-H file.

The contents of DIGEST-H and INDEX-H are appended to the digest or index,

respectively, immediately following the list of topics. For instance,

```
Date: Tue, 11 Jun 2001 11:52:41 -0500
From: Automatic digest processor <LISTSERV@MYHOST.COM>
Reply-To: My test list <MYLIST@MYHOST.COM>
To: Recipients of MYLIST digests <MYLIST@MYHOST.COM>
Subject: MYLIST Digest - 10 Jun 2001 to 11 Jun 2001

There is one message totalling 10 lines in this issue.

Topics in this issue:

  1. Testing 125...3 sir!

The MYLIST list is sponsored by ABig Corporation.

See http://www.abig.com for information on ABig Corporation's products.
```

Figure 9.4. Sample DIGEST output for a list with a DIGEST-H file. The INDEX-H output would be similar, following the list of postings.

(Note that you can't add a digest or index "footer" because according to the standard anything after the end of the digest text is supposed to be discarded.)

9.7. Templates and template forms for the WWW interface

The following describes the available template files and their respective template forms for the WWW archive and administration interface. ***L-Soft does not advise modifying these templates unless you know exactly what you are doing. If you modify the templates it is strongly recommended that you keep copies of the originals in a safe location for fall-back.***

9.7.1. Forms contained in DEFAULT MAILTPL

Note that, although these template forms are available in DEFAULT MAILTPL (and thus theoretically available for list owners to modify), individual list owners cannot tamper with them. If the LISTSERV maintainer desires to change the "look" of the site, it is preferable to create a file called **www_archive.mailtpl** (per chapter 5.4.5 and below) rather than to edit the forms in DEFAULT MAILTPL.

- **WWW_ARCHIVE_INDEX**: The basic INDEX.HTML page for the WWW archive interface. While this template form is available in DEFAULT MAILTPL, it cannot be changed by list owners.

- **WWW_ARCHIVE_USER_FORMS**: Tells LISTSERV which additional "user" forms to format for the list.

- **WWW_ARCHIVE_TRAILER**: The page trailer file included by the WWW interface's CGI script. When this template is included in a ***listname*.MAILTPL** file it controls ONLY the trailer for the ***listname*.html** main index page. See **WWW_LIST_TRAILER**, below.

- **WWW_ARCHIVE_HEADER**: The page header file included by the WWW interface's CGI script. When this template is included in a ***listname*.MAILTPL** file it controls ONLY the trailer for the ***listname*.html** main index page. See **WWW_LIST_HEADER**, below.

- **$WWW_IMAGES_URLDEF**: Default URLs for standard images, do not change

- **$WWW_IMAGES_URL**: URLs for standard images. If these images are stored in non-standard locations you put the URLs for those locations here. Otherwise LISTSERV uses the defaults in **$WWW_IMAGES_URLDEF** .

- **$WWW_ARCHIVE_HEADER**: Contains the header text for the WWW archive interface, that is, what prints at the top of the page. This template form is included by default in the **WWW_ARCHIVE_HEADER** template form.

- **$WWW_ARCHIVE_TRAILER**: Contains trailer text for the WWW archive interface, that is, what prints at the bottom of the page. This template form is included by default in the **WWW_ARCHIVE_TRAILER** template form.

- **XHTML_LISTSERV_REPLY_TRAILER**: Contains a trailer used for HTML digests (technically a mail template rather than an HTML template)

- **DIRECTORY**: Template directory for **X-GETTPL** (overrides only). You probably should not change this template form unless advised to do so by L-Soft.

- **WWW_ARCHIVE_DIRECTORY**: Template directory for **X-GETTPL** (**WWW_ARCHIVE** only). You probably should not change this template form unless advised to do so by L-Soft.

- **WWW_LIST_HEADER**: A second-level header file that can be defined by the list owner for all pages other than the main ***listname*.html** page (see **WWW_ARCHIVE_HEADER** for the main page). If defined, this header appears after the **WWW_ARCHIVE_HEADER** and before the rest of the page's content on all pages below ***listname*.html**.

- **WWW_LIST_TRAILER**: A second-level trailer file that can be defined by the list owner for all pages other than the main ***listname*.html** page (see **WWW_ARCHIVE_TRAILER** for the main page). If defined, this trailer appears before the **WWW_ARCHIVE_TRAILER** and after the rest of the page's content on all pages below ***listname*.html**.

The following will help clarify the page placement of **WWW_LIST_HEADER** and **WWW_LIST_TRAILER** when they are defined in ***listname*.mailtpl**:

WWW_ARCHIVE_HEADER
WWW_LIST_HEADER
[page content]
WWW_LIST_TRAILER
WWW_ARCHIVE_TRAILER

Except as noted for the main list archive page (***listname*.html**), list owners may not override **WWW_ARCHIVE_HEADER** or **WWW_ARCHIVE_TRAILER** as they are defined on a site-wide basis.

9.7.2. The www_archive.mailtpl file (optional)

Rather than changing DEFAULT MAILTPL to customize your site's "look", it is recommended that you place modified templates from DEFAULT MAILTPL in a file called **`www_archive.mailtpl`** , which must be located in the same directory as DEFAULT MAILTPL and which will not be overwritten by a software update.

9.7.3. The `default.wwwtpl` file

The DEFAULT WWWPTL file contains the default templates for the parts of the WWW archive interface that are not defined in DEFAULT MAILTPL. Unless you have specific issues that need to be resolved (such as a national language preference or a need to point certain links to non-standard locations), you should probably leave this file alone. DEFAULT WWWTPL will be overwritten by a software update so you should be sure to keep a copy of your production file in a safe place.

When editing these templates please note two fundamental differences between them and the templates in DEFAULT MAILTPL:

1. Any substitution variable that you use (for instance, **`&LISTNAME`**) must be escaped with a "+" symbol between the ampersand and the name of the variable, thus: **`&+LISTNAME`** . (Note that, as with the regular mail template forms, not all substitution variables are available in every HTML template form.)

2. Any dot-formatting command you use (for instance, **`.CC`** , **`.BB`** , etc.) must have a "+" symbol rather than the dot, thus: **`+CC +BB`**

The templates currently included in DEFAULT WWWTPL are:

- **`Style-sheet:`** Style sheet for dynamic web templates. You will find this template form imbedded in most other web template forms; it makes it easier to change the overall "look" of the pages.

- **`A1-main`**: Second-level archive page (one month/week)

- **`A1-def`**: Special options for second-level archive page (see A1-MAIN)

- **`A2-main`**: Archive browsing (one message)

- **`S1-main`**: Main search page

- **`S2-main`**: Search results

- **`S2-missing`**: Search function, missing argument

- **`OPEN-error`**: Error message, can't access files (generic, returns error code)

- **`OPEN-bad-index`**: Error message, invalid index file

- **`LOGIN-main`**: Login screen

- **`LOGIN-cookie`**: Login confirmation after password saved in cookie

- **`LOGIN-cookie-reset`**: Confirmation after resetting login cookie

- **`CHPW-main`**: Change password screen

- **NEWPW-main**: Main password registration screen
- **LOGIN-CHECK-COOKIE**: Offer to reset cookie if authentication failed (imbedded template)
- **LOGIN-BROWSE-NOTAUTH**: Error screen, not authorized to browse
- **LOGIN-SEARCH-NOTAUTH**: Error screen, not authorized to search
- **LOGIN-ADMIN-NOTAUTH**: Error screen, not a LISTSERV administrator
- **LOGIN-MANAGE-NOTAUTH**: Error screen, not a list owner
- **LOGIN-MANAGE-NOPW**: Error screen, list cannot be managed via WWW
- **POST-NOTAUTH**: Error screen, not authorized to post
- **MM-DBMS-NOTAUTH**: Error screen, not authorized for DBMS mail-merge jobs
- **MM-LIST-NOTAUTH**: Error screen, not authorized for list mail-merge jobs
- **LITE-NOTSUPP**: Error screen, not supported in Lite.
- **NEWPW-mailed**: Awaiting mailed confirmation of new password screen.
- **ACTMGR-main**: Account management functions, main screen
- **ACTMGR-usersel**: Account management functions, user selection screen
- **ACTMGR-subopt**: Account management functions, view/update subscription options
- **ACTMGR-subopt-msglib**: Account management functions, text for subscription options
- **HDREDIT-main**: Edit list header, main screen
- **TPLMGR-formsel**: Template management, form selection screen
- **TPLMGR-formedit**: Template management, form edit screen
- **LMGT-main**: List management, main page
- **P1-QUOTE**: Reply function, text to prepend when including original message
- **P1-main**: Post/reply function
- **SUBEDIT-main**: Authenticated subscribe/leave, main page
- **BULKOP-main**: Bulk operations, main page
- **LAYOUT-data**: Layout customization, data page

- **LAYOUT-SYSTEM-data**: Layout customization, mandatory data (DO NOT EDIT!)
- **LAYOUT-data-wrapper**: Wrapper for layout data, sets useful variables
- **LAYOUT-main**: Layout customization, main page
- **LCMD-main**: Execute an arbitrary LISTSERV command (invoke "wa" with the parameter "**?LCMD1**")
- **MM1-main**: Mail-merge (DBMS based)
- **MM2-main**: Mail-merge (list based)
- **NEWLIST-main**: List creation, main page
- **LIST-LIBRARY**: Library of list header templates
- **LIST-LIBRARY-INIT**: Initialization sequence for library of list header templates
- **SEARCH-HELP**: Help page for the WWW archive interface's search functions
- **SETTINGS-HELP**: Help page for subscription settings
- **LIST-select**: Form to access archives of confidential (unlisted) list. Reached by invoking "wa" with the parameter "**?LIST**" (or by clicking on the link on the first-level archive page).
- **LOGIN-MSGLIB**: Miscellaneous error messages (for translation purposes)

9.7.4. The site.wwwtpl file (optional)

If desired, you can override the **default.wwwtpl** file by providing a customized **site.wwwtpl** file in the same directory. This will prevent your site-wide definitions being overwritten in an upgrade (i.e., when **default.wwwtpl** will normally be overwritten). The **site.wwwtpl** file takes precedence over **default.wwwtpl** (so you don't have to duplicate every template form in **default.wwwtpl**, just the ones you don't want overwritten by an upgrade) but (for list-level templates only) will itself be overridden by definitions in any ***listname*.wwwtpl** files you have installed.

You can also manage **site.wwwtpl** via the web administration interface (see chapter 11).

9.7.5. National language template files (*idiom*.mailtpl) (optional)

National language templates can be written and used with LISTSERV (L-Soft does not provide them). The use of such templates is governed by two settings:

- Site-wide: The **DEFAULT_LANGUAGE=** site configuration variable allows you to set the site-wide national language template for use by all lists on the server. By default this variable is unset and DEFAULT MAILTPL is used.
- List-level: The **Language=** list header keyword can be used to specify a national

language template to be used for a particular list, for instance a Spanish-language list on an otherwise English-language server.

To create a national language template, you simply copy DEFAULT MAILTPL to *idiom* MAILTPL , where *idiom* is the name of the language, and translate it as desired. You also use *idiom* to specify the value for **DEFAULT_LANGUAGE=** and/or **Language=**. For a given language you can specify anything you want for *idiom*; in other words LISTSERV does not care if you call the file FRANCAIS MAILTPL or FRENCH MAILTPL, but if you do call it FRENCH MAILTPL you must specify FRENCH as the *idiom*, and likewise, if you call it FRANCAIS MAILTPL you must specify FRANCAIS as the *idiom*. LISTSERV has no information about what a given language is called, it simply looks for a MAILTPL file for the *idiom* data supplied.

If you are going to translate the web interface template forms into *idiom* as well, you will need to copy DEFAULT WWWTPL to *idiom* MAILTPL and again, translate as desired. Note carefully however that this requires that you add a special template form to WWW_ARCHIVE MAILTPL, so that the 'wa' CGI script will know what to look for, as follows:

>>> LANGUAGE
idiom

where *idiom* is, of course, the same value we've been talking about above. For instance for a FRANCAIS idiom you'd use

>>> LANGUAGE
FRANCAIS

See also 9.3.1, above, regarding the use of 8-bit characters in template forms.

9.7.6. Template precedence

For template forms found in DEFAULT MAILTPL, the following precedence is used when LISTSERV searches for a given template form:

***listname* MAILTPL**
***idiom* MAILTPL**
WWW_ARCHIVE MAILTPL[10]
DEFAULT MAILTPL

That is to say, if LISTSERV needs a copy of the ADD1 mail template form, it will look first in the ***listname*.mailtpl** file for the list in question. If no such file exists, or if ADD1 is not present in ***listname*.mailtpl**, LISTSERV will look in ***idiom*.MAILTPL** (if **Language=** or **DEFAULT_LANGUAGE=** is set to *idiom*). Again, if the ADD1 form is not present in ***idiom*.mailtpl**, or if ***idiom*.mailtpl** does not exist, LISTSERV will then look in **default.mailtpl** (**www_archive.mailtpl** is skipped because ADD1 is not a web template form) and pull out the default ADD1 template form.

For template forms found in DEFAULT WWWTPL the precedence is:

***listname* WWWTPL**

[10] WWW_ARCHIVE MAILTPL is searched only for web-related template forms and is bypassed for mail template forms. For instance WWW_ARCHIVE MAILTPL will not be searched for ADD1 but will be searched for WWW_INDEX.

***idiom* WWWTPL**
SITE WWWTPL
DEFAULT WWWTPL

The same sequence of events applies as for the MAILTPL files, except that **SITE WWWTPL** is never skipped (all template forms in the WWWTPL files are web forms).

9.8. Using the DAYSEQ(n) function

The **DAYSEQ(n)** function is quite powerful. This function allows the list owner to code template forms (such as the **PROBE1** or **BOTTOM_BANNER** messages) that change or "rotate" automatically.

The **DAYSEQ(n)** function is invoked in a **.BB - .EB** conditional block, and n corresponds to the number of days in the rotation period, i.e., to the number of variations that you want to make to the text of the message. **&DAYSEQ(n)** returns a number from 1 to n which increases by 1 every day, with no special regard for weekends. That is, if the rotation period is to last for a week, you code **DAYSEQ(7)**. If the rotation period is 15 days, you code **DAYSEQ(15)**. Two examples follow:

9.8.1. Rotating bottom banner

To create a rotating bottom banner, follow this example. A list has three commercial sponsors, each of whom are provided with an advertisement every three days. (Note that this doesn't take weekends into account; in this example, if company A is featured in the banner on Monday, it will be featured again on Thursday and then again on Sunday. However, in the following week it will be featured on Wednesday, Saturday, and Tuesday, so it will actually get rather good coverage.) Our **BOTTOM_BANNER** template form would look like this:

```
>>> BOTTOM_BANNER
.BB &DAYSEQ(3) = 1
Today's copy of the &LISTNAME newsletter has been brought to you
by Company A.
.EB
.BB &DAYSEQ(3) = 2
Today's copy of the &LISTNAME newsletter has been brought to you
by Company B.
.EB
.BB &DAYSEQ(3) = 3
Today's copy of the &LISTNAME newsletter has been brought to you
by Company C.
.EB
```

(Naturally you can feel free to be more florid with your prose :)

If a company needs to get a higher percentage of "air" time than another, you can simply assign it more than one of the possible n values of &DAYSEQ(n). For instance, if you have two companies but one should get twice as many days of "air" time, you might code something like this:

```
>>> BOTTOM_BANNER
.BB (&DAYSEQ(3) = 1) OR (&DAYSEQ(3) = 3)
```

```
Today's copy of the &LISTNAME newsletter has been brought to you
by Company A.
.EB
.BB &DAYSEQ(3) = 2
Today's copy of the &LISTNAME newsletter has been brought to you
by Company B.
.EB
```

This would cause Company A's message to appear on days 1 and 3 of the rotation period and Company B's message to appear on day 2 only.

9.8.2. Rotating FAQ via the PROBE1 template and "Renewal= xx-Daily"

Subscription renewal can be coded with daily granularity (however, please note that it is and remains inadvisable to use renewal intervals of less than a week). If you further code subscription probing into the "Renewal=" keyword with the ",Probe" parameter, you open up the possibility of turning the standard **PROBE1** template form into a periodic FAQ. Here's how:

We'll assume to start that you will code "Renewal= 15-Daily,Probe" in your list header. (You can experiment with other numbers, but since we have two messages and will be using **&DAYSEQ(2)**, we need an odd renewal period.) We'll also assume that you want to send two versions of your FAQ each month; the first, a complete FAQ document, and the second, an abbreviated "reminder" version that just contains information about how to sign off, how to post to the list, and so forth. The basic algorithm is therefore:

When **&DAYSEQ(2) = 1**, send the full FAQ.
When **&DAYSEQ(2) = 2**, as it will 15 days later, send the abbreviated FAQ.

Your **PROBE1** template form would thus look like this:

```
>>> PROBE1 Periodic FAQ posting for &LISTNAME
&WEEKDAY, &DATE &TIME
.BB &DAYSEQ(2) = 1
This is the complete FAQ for &LISTNAME.  Please read it and keep a copy
for future reference. A FAQ document for &LISTNAME is distributed every
15 days, the full FAQ alternating with a shorter "reminder" FAQ.

<body of the full FAQ document>
.EB
.BB &DAYSEQ(2) = 2
This is the abbreviated FAQ for &LISTNAME. Please read it and keep a copy
for future reference. A FAQ document for &LISTNAME is distributed every
15 days, the full FAQ alternating with a shorter "reminder" FAQ.

<body of the abbreviated FAQ document>
.EB
```

9.8.3. Calculating the value for DAYSEQ()

When you first start using a rotating banner with the &DAYSEQ variable, the **&DAYSEQ(n)= 1** period begins based on the number of days elapsed since a baseline. On VM (and in REXX generally) you can calculate today's value easily with:

```
/* */
say Date('B') + 1
```

If you do not have access to a REXX interpreter, **`Date('B')`** is described as "the number of complete days (that is, not including the current day) since and including the base date, 1 Jan 0001, in the format '**dddddd**' (no leading zeros or blanks)."[11] It also is equal to the C language expression time(0)/86400 + 719162 or, for OpenVMS users, to the Smithsonian base date plus 678575.

For example, for Friday 22 Oct 2004, the value of `Date('B') + 1` is 731876. This value increases by one every day at midnight.

9.9. Serving up custom web pages for your list

This feature is not available in LISTSERV Lite.

Originally in order to serve up custom or special web pages for a list it was necessary to construct those pages as HTML files and place them either into the /archives directory or link them from somewhere else. This was sometimes impossible for list owners who had no administrative access to the server's web directories or who had no other place from which to serve web pages.

In 1.8d and later it is possible to add ad-hoc web page templates by creating new (non-standard) template forms and enabling their display by setting a special variable value, SHOWTPL_ALLOWED, in the template form. For instance, one could set up a page with special administrative information, the list charter, netiquette information, or the like, and serve it and maintain it directly from the LISTSERV web template interface without need for any other access to the server.

9.9.1. A practical example: ADMIN_POST

The author used to serve up an administrative posting via FTP back in the days when his lists lived on a server that had FTP access to the archive notebooks. When FTP access to the server was cut off due to security concerns, he had to find another way to serve the information via the web. Here is how it was done:

(The following example assumes that you have the 1.8e web administration interface installed. New template forms cannot be created this way in previous versions.)

First, log into the web administration interface. Choose the list for which you will be making a new page and click the "Templates" button to enter the mail and web template editing area. Since the template you will be creating is a web template, click the "Switch to WWW templates" button to change modes.

Next, type the name of the new template form into the box provided, and click "Create". A page entitled "Edit List Template" will come up, with the command response

`The ADMIN_POST form has been successfully stored in the TEST template library.`

In the "Description:" box, type a description of the template, for example, "Administrative information page".

In the large text box provided for the template text, first type the following line:

---

[11] Cowlishaw, Michael: *The REXX Language: A Practical Approach to Programming*, 2nd ed., p.92. Englewood Cliffs, NJ: Prentiss-Hall, Inc., 1990.

+SE SHOWTPL_ALLOWED 1

This line tells LISTSERV that it is allowed to serve the page on the web. If the line is not found, the template will not be available.

Following this line you can start adding your HTML. However note carefully that you cannot override the default headers and footers that have already been defined by other template forms in the library. You can start with a <title></title> block but it will be followed by the pre-defined header and *then* by your HTML.

After adding your HTML, click "Update", and the template form will be stored. The URL for the page you are defining in this example will be

http://*your_server_hostname*/*path_to_wa*?SHOWTPL=ADMIN_POST&L=*listname*

(the parameters for 'wa' are case-sensitive and must be sent in upper case). For instance, the author's version of the ADMIN_POST template form can be viewed at

http://peach.ease.lsoft.com/scripts/wa.exe?SHOWTPL=ADMIN_POST&L=VISBAS-L

Documented Restriction: For LISTSERV 1.8d (or LISTSERV 1.8e running with the 1.8d web interface) there is no simple method to create a new (ie previously non-existent) template form from the web interface. If you are not comfortable with the GET and PUT method of updating list-level template forms, a web-based workaround is to open one of the existing template forms and add the template delimiter line at the bottom, then save the existing template form. For instance, we'll choose the first template form that presents itself, A1-DEF. Open this template by clicking on the "Edit form" button. If this template has not been changed from its default (and it probably will not have been), it will look like this:

```
+* You should now set these options using the layout customization
+* screen, except for the special F and S options, which are much too
+* advanced for the layout screen and must still be set in this template.
+IM LAYOUT-data-wrapper
+SE D &+LYT_DVIEW_D;
+SE FP &+LYT_DVIEW_FP;
+SE H &+LYT_DVIEW_H;
+SE O &+LYT_DVIEW_O;
+SE T &+LYT_DVIEW_T;
```

At the bottom of this form, directly after the **+SE T &+LYT_DVIEW_T;** line, type

```
>>> ADMIN_POST Administrative information page
```

(Be sure that there is a space between ">>>" and "ADMIN_POST". This is required.) Next, click on the "Update template" button and the template will update. You will see the message at the top of the page:

The A1-DEF form has been successfully stored in the LISTNAME template library.

(LISTNAME of course will be the name of the list you are working with.)

Next, back out of the preceding pages and return to the main web template editing page, and reload that page. In the drop-down list box you will see a new line that says

(*) Administrative information page [ADMIN_POST]

You now have one final cleanup action to take before you edit the new template form. Go back into the A1-DEF form editing screen (you'll note that it also has a "(*)" next to it, indicating that it has been modified). You'll note when you go into the editing screen that it says at the top,

This form is defined in the LISTNAME template.

Clear the text out of the edit box, and also clear the "Description:" text box (this is important, otherwise you will leave a blank A1-DEF template form in the list's template library, and that is not good). Update the template again. The interface will again tell you that

The A1-DEF form has been successfully stored in the LISTNAME template library.

In actuality what has happened is you have removed the copy of A1-DEF that was placed into *listname*.WWWTPL when you saved the modifed copy that contained the template form delimiter line for your ADMIN_POST template form. If you back out to the template management page again you will see that the "(*)" indicator is no longer present at the start of the A1-DEF line. Now you can open the ADMIN_POST template form and edit it.

9.10. Modifying the output of LISTSERV's HELP command (non-VM)

LISTSERV's HELP command output is designed primarily to show the basic syntax of certain commonly used commands, and point users to other documents that explain how to use LISTSERV. Some sites may wish to amplify the message, and this can be done by creating a file called 'localcmd.helpfile' in the same directory where LISTSERV keeps permvars.file, default.mailtpl, and so forth.

Note carefully L-Soft *does not* recommend removal or alteration of the LSVHELP.FILE that ships with the software (and which contains the default text for the HELP command response) as LISTSERV expects it to be present and it will be overwritten in an upgrade.

LOCALCMD HELPFILE was originally intended to allow non-VM sites to document local commands constructed by using list exits (see chapter 5 of the *Developer's Guide for LISTSERV* for details regarding list exits and local commands). As such, when LISTSERV sees a LOCALCMD HELPFILE, it appends the contents of LOCALCMD HELPFILE after a line that says "The following local commands are also available:". For instance, if you have a local command called /XYZ, you could have a LOCALCMD HELPFILE containing something like:

```
/XYZ      <options>                (One line description of /XYZ)
```

and the output of HELP would then be

```
> help

LISTSERV(R) version 1.8e - most commonly used commands

Info      <topic|listname>       Order documentation
Lists     <Detail|Short|Global>  Get a description of all lists
SUBscribe listname <full name>   Subscribe to a list
SIGNOFF   listname               Sign off from a list
SIGNOFF   * (NETWIDE             - from all lists on all servers
REView    listname <options>     Review a list
Query     listname               Query your subscription options
SET       listname  options      Update your subscription options
INDex     <filelist_name>        Order a list of LISTSERV files
```

```
GET       filename filetype      Order a file from LISTSERV
REGister  full_name|OFF          Tell LISTSERV about your name

There are more commands (AFD, FUI, PW, etc). Send an INFO REFCARD for a
comprehensive  reference card,  or just  INFO for  a list  of available
documentation files.

The following local commands are available at this installation:

/XYZ      <options>              (One line description of /XYZ)

This server is managed by:
 LSTMAINT@LISTSERV.EXAMPLE.COM
```

For most sites, however, locally-added commands probably won't be available. If you use the LOCALCMD HELPFILE functionality at all, it will likely be to enhance the output in order to make it more understandable for users. So you probably would not want to see the line "The following local commands are available at this installation:" followed by text that doesn't document commands. You can turn off this line by simply adding

```
.NH
```

as the first line of LOCALCMD HELPFILE. Note that **.NH** is the only formatting command available in LOCALCMD HELPFILE; it is otherwise a flat ASCII file that outputs exactly as you type it.

Let's say that you are having a problem where the LISTSERV postmasters are fielding a lot of questions that really ought to be sent to list owners (get me off this list, my address changed, etc.) and a little investigation indicates that these people are getting the postmaster address from the HELP command. It's not reasonable to remove the postmaster's address from the output since it should always be possible to find out who is running the server (in case loops develop, etc.), but you could create a LOCALCMD HELPFILE like the following to indicate to users where various kinds of questions should be sent:

```
.NH
For help with a  specific list, please write to the  list owner(s) at the
generic list owner's address.  This address takes the form

                  listname-REQUEST@LISTSERV.EXAMPLE.COM

For  instance, if  you  are subscribed  to a  list  called DOGLIST-L,  to
contact the list owners you would write to

                  DOGLIST-L-REQUEST@LISTSERV.EXAMPLE.COM

PLEASE  DO NOT  ACTUALLY  WRITE  TO DOGLIST-L-REQUEST.  This  is just  an
example, you  must substitute the name  of the mailing list  in question.
There  is no  DOGLIST-L mailing  list  on this  server and  mail sent  to
DOGLIST-L-REQUEST is discarded unread.

Please  do not  write to  the  server manager  unless  you do  not get  a
response from the list owner.  Thank you!
```

9.11. The $SITE$.MAILTPL file

This feature is not available in LISTSERV Lite.

In some cases, it may be necessary for the LISTSERV maintainer to ensure that all subscribers receive certain information or warnings (typically legal notices required by government regulations) when they leave or join a list. The list owner should not be able to disable these warnings, accidentally or otherwise. The $SITE$.MAILTPL file provides

this functionality. However please note that this functionality should be used only if you have a specific need for it as it will increase the number of administrative messages received by users and may cause confusion.

If a $SITE$.MAILTPL file is present in LISTSERV's main directory (the one with DEFAULT.MAILTPL), LISTSERV will look it up every time it needs to deliver an administrative message. If the message is not found in the site template, LISTSERV will process the request normally, looking up the list template file, then the language template file and finally the system template file. If the message is listed in the site template, LISTSERV will deliver *both* the copy in the site template *and* the copy in the list/language/system template.

Note that L-Soft does not ship a $SITE$.MAILTPL file in the LISTSERV distributions. If needed, you must create this template file yourself. $SITE$.MAILTPL will *not* be overwritten during an upgrade. (Note that under unix, this file must be named in lowercase, that is, $site$.mailtpl .)

Also note that LISTSERV will not pick up an "imbedded" template from $SITE$.MAILTPL. If you wish to include an "imbedded" template (e.g., **$SIGNUP**) in $SITE$.MAILTPL, you must also include the template(s) that calls it with the **.im** command. For instance, if you include **$SIGNUP** in $SITE$.MAILTPL, by default you would need to include the **SIGNUP1** and **ADD1** templates in $SITE$.MAILTPL as well if you expected **$SIGNUP** to be sent from $SITE$.MAILTPL.

Web templates from DEFAULT MAILTPL must be placed in WWW_ARCHIVE MAILTPL if you wish to override the defaults. The web interface ignores $SITE$ MAILTPL .

10. Interpreting and Managing LISTSERV's log files

10.1. Logs kept by LISTSERV

LISTSERV keeps logs of all of its activity. On VM systems, this information is kept in the LISTSERV console log. On unix systems, it is kept in **$LSVROOT/listserv.log** by default. Note that unix systems create the log by redirection of standard output to a file; see the 'go' script if you are interested in this process.

On VMS and Windows systems, there may be several different logs depending on the configuration of the system. For instance, in addition to the LISTSERV log itself, there will be logs for the SMTP "workers" if this feature is enabled. On Windows systems, there will also be a log for the SMTP "listener" if it is in service. (Windows systems running L-Soft's LSMTP product will not use the SMTP "listener" service, and thus will not keep logs for it.) By default, logs under VMS are kept in **LISTSERV_ROOT:[LOG]** and logs under NT are kept in **\LISTSERV\LOG** .

10.2. Managing the logs

Making daily logs

While LISTSERV for VM (via WAKEPARM FILE) and Windows "turns" the log at midnight each day, automatically creating daily logs, LISTSERV for VMS and unix does not.

LISTSERV for VMS can "turn" the log via the revision control system if you simply stop and restart LISTSERV once a day (note that you must stop LISTSERV completely and restart it in a new process). This will create files like **LSV_*machinename*.LOG;1** and **LSV_*machinename*.LOG;2** in **LISTSERV_ROOT:[LOG]** (by default).

LISTSERV for unix creates a single file in the **$LSVROOT** directory called **listserv.log**. As this file can get quite long, it will probably be most productive to create a shell script called by a cron job to stop LISTSERV, rename the existing listserv.log and move it to another location (e.g., **$LSVROOT/oldlogs**), and then restart LISTSERV. L-Soft does not provide shell scripts for this purpose.

"Cleaning" your log files periodically

On all systems, you will probably want to clean out your old logs on a regular basis. To do this, you will want to write a script that executes automatically (e.g., via WAKEPARM on VM, as a cron job on unix, or as an AT job on Windows). A sample REXXette for use with Windows NT or 95 that keeps the last 5 days' worth of compressed logs and deletes anything older than that follows:

```
/**/
 logdir = 'E:\LISTSERV\LOG'
 tempfile = logdir'\CLEANLOG.TMP'
 keep = 5

/* First zip all the logs, except today's */
'DIR/B' logdir'\*.LOG >' tempfile
 If rc ^== 0 Then Exit

 today = Date('S')
 Do forever
```

```
    line = Translate(Linein(tempfile))
    If line == '' Then Leave
    Parse var line '-'date'.'
    If date = today Then Iterate
    Parse var line fn'.'
    Say 'ZIPping' line'...'
   'ZIP -j -m -q' logdir'\'fn logdir'\'line
 End
 Call Lineout tempfile

/* Now delete ZIP files older than 'keep' days */
'DIR/B/O-N' logdir'\*.ZIP >' tempfile
 If rc ^== 0 Then Exit

 n. = 0
 Do forever
    line = Translate(Linein(tempfile))
    If line == '' Then Leave
    Parse upper var line pfx'-'date'.'
    n.pfx = n.pfx + 1
    If n.pfx <= keep Then Iterate
    Say 'Deleting' line'...'
   'DEL' logdir'\'line
 End
 Call Lineout tempfile
'DEL' tempfile
```

Figure 10.1. Sample CLEANLOG.REXX script for managing LISTSERV's log files. This particular script runs under Regina REXX on Windows NT or 95.

Please note that while it is of course possible to simply delete the log file on a daily basis, for the purpose of debugging potential problems this is not recommended.

10.3. Interpreting the LISTSERV log

This file, LISTSERV's main logging file, has different names under different ports of the product.

VM:	LISTSERV logs events to the LISTSERV user's console log.
OpenVMS:	**LISTSERV_ROOT:[LOG]LSV_*machine*.LOG;*x***
Unix:	**~$LSVROOT/listserv.log**
Windows:	**LISTSERV\LOG\LISTSERV-*yyyymmdd*.LOG**

Under VM, the console log can be "turned" by placing the appropriate command in the WAKEPARM file. Under Windows NT and 95, LISTSERV automatically "turns" the log at midnight. For OpenVMS, if you reboot daily as explained in 10.2, the logs are numbered using the revision tracking system, e,g, **LSV_PEAR.LOG;1**, **LSV_PEAR.LOG;2**, etc. (if you don't do the daily reboot, LISTSERV just keeps a single log file). For Windows, "***yyyymmdd***" is the year, month and day of the log, for example, **LISTSERV-19980104.LOG** is the log for 4 January 1998. Unix logs are simply the output of the program written to standard output and shell scripts (not provided by L-Soft) can be written to "turn" the logs daily via **cron**.

10.3.1. Expiring cookies

```
25 May 1996 00:00:00 Expiring cookie from xxxxx@ICOM.CA:
> SIGNOFF TECHLINK
25 May 1996 00:00:00 Sent information mail to xxxxx@ICOM.CA
25 May 1996 00:00:00 Expiring cookie from xxxxxxxx@LIGHTSIDE.COM:
> SUBSCRIBE WINNT-L Sxxxx Bxxxxx
25 May 1996 00:00:00 Sent information mail to xxxxxxxx@LIGHTSIDE.COM
```

These entries refer to expiring "OK" confirmation cookies.

10.3.2. Releasing and reallocating a disk slot

```
25 May 1996 00:00:00 Virtual disk slot E(E:\LISTSERV\ARCHIVES\NISTEP-L) released
.
25 May 1996 00:00:00 Directory E:\EASE\PEACH\LISTS\IATN accessed as virtual disk
 slot E.
```

LISTSERV uses "disk slots" in rotation to minimize the overhead involved in opening a file, performing an operation, closing the file, and then possibly having to reopen the file immediately to perform another operation. A given "disk slot" stays open until it is needed for another file.

10.3.3. Reindexing a list

```
25 May 1996 00:00:01 Reindexing MYLIST-L LOG9605D...
```

LISTSERV rebuilds the index for the current notebook archive file of a given list immediately prior to sending the DIGEST and INDEX versions of the list.

10.3.4. Distributing a digest

Here is a typical log sequence for the distribution of a daily digest:

```
25 May 1996 00:00:35 Virtual disk slot E(E:\LISTSERV\ARCHIVES\HONYAKU) released.
25 May 1996 00:00:35 Directory E:\LISTSERV\ARCHIVES\SPAM-L accessed as virtual d
isk slot E.
25 May 1996 00:00:37 Reindexing SPAM-L LOG9605...
25 May 1996 00:00:37 SPAM-L digest is being distributed...
25 May 1996 00:00:37 Distributing mail ("SPAM-L") from owner-SPAM-L@PEACH.EASE.L
SOFT.COM...
25 May 1996 00:00:38 Mail forwarded to H-NET.MSU.EDU     for   2 recipients.
25 May 1996 00:00:38 Mail forwarded to UAFSYSB.UARK.EDU  for   2 recipients.
25 May 1996 00:00:38 Mail forwarded to LISTMAIL.SUNET.SE for   2 recipients.
```

The preceding 3 jobs were forwarded to LISTSERV servers on the DISTRIBUTE backbone. The following mail is posted to 45 users who are not served by the DISTRIBUTE backbone in a single BSMTP "envelope".

```
25 May 1996 00:00:38 Mail posted via SMTP to xxx@AMERICAN.EDU.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxxxx@AOL.COM.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxxx@AOL.COM.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxx@BCVMS.BC.EDU.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxx@BMACADM.BITNET.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxx@CLEMSON.EDU.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxxxxxxx@COMPUSERVE.COM.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxxxx@EMAIL.GC.CUNY.EDU.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxxx@ERE.UMONTREAL.CA.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxx@ETERNA.COM.AU.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxx@GOL.COM.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxxxx@GPU.SRV.UALBERTA.CA.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxxx@HAWAII.EDU.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxx@IBM.NET.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxxxx@IDIR.NET.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxxx@INET.UNI-C.DK.
```

```
25 May 1996 00:00:38 Mail posted via SMTP to xxx@KSUVM.KSU.EDU.
25 May 1996 00:00:38 Mail posted via SMTP to xxxx@LOC.GOV.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxx@LOONY-TOONS.TAMU.EDU.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxx@LTRR.ARIZONA.EDU.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxxx@MAILBOX.SYR.EDU.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxxxx@MO.NET.
25 May 1996 00:00:38 Mail posted via SMTP to xxxx@MUSICA.MCGILL.CA.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxxx@NANDO.NET.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxxx@NETCOM.COM.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxxx@NYIQ.NET.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxx@PAMELA.INT.MED.UNIPI.IT.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxx@PIONEER.STATE.ND.US.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxx@PSC.LSA.UMICH.EDU.
25 May 1996 00:00:38 Mail posted via SMTP to xxxx@PSUVM.PSU.EDU.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxx@PUCC.PRINCETON.EDU.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxx@SCS.UNR.EDU.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxx@SILVERPLATTER.COM.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxx@SJUVM.STJOHNS.EDU.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxx@SPCVXA.SPC.EDU.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxx@TELERAMA.LM.COM.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxx@UA1VM.UA.EDU.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxxx@UABDPO.DPO.UAB.EDU.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxxxx@UCONNVM.UCONN.EDU.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxxx@UTARLVM1.UTA.EDU.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxxx@UVVM.UVIC.CA.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxxx@VM1.HQADMIN.DOE.GOV.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxx@VM1.MCGILL.CA.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxx@VM3090.EGE.EDU.TR.
25 May 1996 00:00:38 Mail posted via SMTP to xxxxxx@WATSON.IBM.COM.
```

LISTSERV always summarizes and tells you what it's done:

```
25 May 1996 00:00:38 Done - 3 jobs (6 rcpts), 1 outbound file (45 rcpts).
```

10.3.5. Daily error monitoring reports

```
25 May 1996 00:00:43 Generating daily nondelivery monitoring reports...
```

Here LISTSERV deletes a subscriber who has gone over the Auto-Delete= limits:

```
25 May 1996 00:00:44 -> Deleted xxxxxxxx@CARIARI.UCR.AC.CR from ACCESS-L.
```

LISTSERV now sends out the daily error monitoring reports to the appropriate people (defined in "Errors-To=" for each list).

```
25 May 1996 00:00:44 Sent information mail to xxxxxxxx@CARIARI.UCR.AC.CR
25 May 1996 00:00:45 Sent information mail to xxxxxx@LINUS.DC.LSOFT.COM
25 May 1996 00:00:46 Sent information mail to xxxxx@AF.PENTAGON.MIL
25 May 1996 00:00:46 Sent information mail to xxxxx@ESA.MHS.COMPUSERVE.COM
```

As LISTSERV continues to run and process errors for the various lists, it will update the listname.AUTODEL file whenever it receives an error that it understands:

```
25 May 1996 00:05:43 Automatic nondelivery report processing for WIN95-L:
25 May 1996 00:05:43 -> All errors temporary, no action taken.
25 May 1996 00:07:34 Automatic nondelivery report processing for EXCEL-G:
25 May 1996 00:07:34 -> 1 monitoring entry updated.
```

10.3.6. Processing mail for local lists

```
25 May 1996 00:39:23 Processing file 8209289 from MAILER@PEACH.EASE.LSOFT.COM
25 May 1996 00:39:25 Processing mail from xxxxxx@PRIMENET.COM for ACCESS-L
25 May 1996 00:39:25 Virtual disk slot E(E:\EASE\PEACH\FTP\EXCEL-L) released.
```

```
25 May 1996 00:39:25 Directory E:\EASE\PEACH\FTP\ACCESS-L accessed as virtual di
sk slot E.
25 May 1996 00:39:26 Distributing mail ("ACCESS-L") from owner-access-l@PEACH.EA
SE.LSOFT.COM...
25 May 1996 00:39:26 Mail posted via SMTP to xxxxxxxxxxxxxx@NR-COMMS.RADIO.BBC.C
O.UK.
25 May 1996 00:39:26 Mail posted via SMTP to xxxxxxxx@OHS.ORG.
25 May 1996 00:39:26 Mail posted via SMTP to xxxxx@POBOX.COM.
25 May 1996 00:39:26 Done - 1 outbound file (3 rcpts).
25 May 1996 00:39:26 Distributing mail ("ACCESS-L") from owner-access-l@PEACH.EA
SE.LSOFT.COM...
25 May 1996 00:39:26 Mail forwarded to LISTSERV@HEARN    for   4 recipients.
25 May 1996 00:39:27 Mail forwarded to LISTMAIL.SUNET.SE for  11 recipients.
25 May 1996 00:39:27 Mail forwarded to LISTSERV@ICINECA  for   4 recipients.
25 May 1996 00:39:27 Mail forwarded to LISTSERV.GMD.DE   for   3 recipients.
25 May 1996 00:39:27 Mail forwarded to LISTSERV@AEARN    for   2 recipients.
25 May 1996 00:39:27 Processed     192 recipients...
25 May 1996 00:39:27 Done - 5 jobs (24 rcpts), 1 outbound file (192 rcpts).
25 May 1996 00:39:27 Distributing mail ("ACCESS-L") from owner-access-l@PEACH.EA
SE.LSOFT.COM...
25 May 1996 00:39:27 Mail posted via SMTP to xxxxx@ADC.COM.
25 May 1996 00:39:27 Mail posted via SMTP to xxxxxxx@MSMEL.PRAXA.COM.AU.
25 May 1996 00:39:27 Mail posted via SMTP to xxxxxxx@SPRYNET.COM.
25 May 1996 00:39:27 Done - 1 outbound file (3 rcpts).
25 May 1996 00:39:27 Message DISTRIBUTEd to 222 recipients.
25 May 1996 00:39:28 Sent information mail to xxxxxx@PRIMENET.COM
```

10.3.7. Administrative mail (X-ADMMAIL)

```
25 May 1996 00:01:16 From MAILER@PEACH.EASE.LSOFT.COM: X-ADMMAIL OWNER-AFNS
25 May 1996 00:01:16 Processing file 8206153 from MAILER@PEACH.EASE.LSOFT.COM
6 Jun 2000 09:12:42 Automatic nondelivery report processing for AFNS:
6 Jun 2000 09:12:42 -> All errors temporary, no action taken.
```

This particular type of mail, when sent to the OWNER-listname address, is a delivery error being returned to the RFC 821 MAIL FROM: address. The last two lines of this log excerpt entry simply indicate that auto-deletion is enabled for the list and that in this case the error received was not a permanent one.

```
26 Jun 2000 09:12:38 From MAILER@PEACH.EASE.LSOFT.COM: X-ADMMAIL TEST-REQUEST JO
E.USER@EXAMPLE.COM
26 Jun 2000 09:12:39 Sent information mail to JOE.USER@EXAMPLE.COM
```

When sent to the listname-REQUEST address, the mail is forwarded to all non-quiet list owners and the REQACK1 template form (see chapter 9) is sent back to the user who wrote to the listname-REQUEST address.

10.3.8. DISTRIBUTE jobs from remote hosts

Just as our LISTSERV sends out DISTRIBUTE jobs to the backbone, it also receives them for distribution from remote backbone hosts.

```
25 May 1996 00:01:16 Processing file 8206155 from MAILER@PEACH.EASE.LSOFT.COM
25 May 1996 00:01:16 From LISTSERV@PSUVM.PSU.EDU: X-B64 ID=PAN-L.MAILDIST CLASS=
A
25 May 1996 00:01:16 Rescheduled as: 8206156
25 May 1996 00:01:16 Processing file 8206156 from LISTSERV@PEACH.EASE.LSOFT.COM
25 May 1996 00:01:16 From LISTSERV@PSUVM: DIST2 MAIL I=Y FROM=owner-pan-l@BINGVM
B.CC.BINGHAMTON.EDU FORW(CRC) HOST(626 626 (...)
25 May 1996 00:01:17 Distributing mail ("PAN-L") from owner-pan-l@BINGVMB.CC.BIN
GHAMTON.EDU...
25 May 1996 00:01:17 Mail posted via SMTP to xxxxxxxx@ACS.RYERSON.CA.
25 May 1996 00:01:17 Mail posted via SMTP to xxxxx@AMAIL.AMDAHL.COM.
25 May 1996 00:01:17 Mail posted via SMTP to xxxxxxx@AOL.COM.
```

and so forth.

10.3.9. Requesting "OK" confirmation for commands

```
25 May 1996 00:01:17 Processing file 8206160 from MAILER@PEACH.EASE.LSOFT.COM
25 May 1996 00:01:17 From xxxxxxx@AUSCONSULT.COM.AU: UNSUB EXCEL-L
25 May 1996 00:01:17 Requesting confirmation, cookie=3EB4F2
25 May 1996 00:01:17 Sent information mail to xxxxxxx@AUSCONSULT.COM.AU
```

10.3.10. Subscription summary updates (SUPD jobs)

X-SUPD jobs update the file used by "LIST SUMMARY". LISTSERV receives the file from another LISTSERV host and distributes it down the line:

```
25 May 1996 00:04:54 Processing file 8206401 from MAILER@PEACH.EASE.LSOFT.COM
25 May 1996 00:04:54 From LISTSERV@INTERNET.COM: X-B64 ID=X-SUPD.JOB ASCII CLASS
=J
25 May 1996 00:04:54 Rescheduled as: 8206402
25 May 1996 00:04:54 Processing file 8206402 from LISTSERV@PEACH.EASE.LSOFT.COM
25 May 1996 00:04:54 From LISTSERV@INTERNET.COM: DIST2 I=Y FROM=LISTSERV@INTERNE
T.COM FORW(CRC) HOST(626 626 626 626 626 691 686  (...)
25 May 1996 00:04:54 Distributing file "X-SUPD JOB" from LISTSERV@INTERNET.COM..
.
25 May 1996 00:04:54 File forwarded to LIME.EASE.LSOFT.COM for   1 recipient.
25 May 1996 00:04:54 File forwarded to WIN95.DC.LSOFT.COM for   1 recipient.
25 May 1996 00:04:54 File forwarded to SPIDER.EASE.LSOFT.COM for   1 recipient.
25 May 1996 00:04:54 File forwarded to HOME.EASE.LSOFT.COM for   1 recipient.
25 May 1996 00:04:54 File forwarded to LISTSERV.GEORGETOWN.EDU for   1 recipient
.
25 May 1996 00:04:54 File forwarded to CC1.KULEUVEN.AC.BE for   1 recipient.
25 May 1996 00:04:54 File forwarded to LISTSERV.USHMM.ORG for   1 recipient.
25 May 1996 00:04:54 File forwarded to LISTSERV.CLARK.NET for   1 recipient.
25 May 1996 00:04:54 File "X-SUPD JOB" distributed to LISTSERV@PEACH.EASE.LSOFT.
COM.
25 May 1996 00:04:54 Done - 8 jobs (8 rcpts), 1 outbound file (1 rcpt).
25 May 1996 00:04:54 Processing file 8206411 from LISTSERV@PEACH.EASE.LSOFT.COM
25 May 1996 00:04:54 From LISTSERV@INTERNET.COM: X-SUPD FWD=NO DATE=1996052500:0
0:04 DATA=5 56 PHOTOPRO 465 PHOTOTECH 268 PHOTOAS (...)
```

10.3.11. Global list of lists updates (LUPD jobs)

X-LUPD jobs update the list of lists. LISTSERV receives the file from another LISTSERV host and distributes it down the line:

```
25 May 1996 00:04:08 Processing file 8206361 from MAILER@PEACH.EASE.LSOFT.COM
25 May 1996 00:04:08 From LISTSERV@LISTSERV.UIC.EDU: X-B64 ID=X-LUPD.JOB ASCII C
LASS=J
25 May 1996 00:04:08 Rescheduled as: 8206362
25 May 1996 00:04:08 Processing file 8206362 from LISTSERV@PEACH.EASE.LSOFT.COM
25 May 1996 00:04:08 From LISTSERV@LISTSERV.UIC.EDU: DIST2 I=Y FROM=LISTSERV@LIS
TSERV.UIC.EDU FORW(CRC) HOST(626 626 626 626 626 691  (...)
25 May 1996 00:04:08 Distributing file "X-LUPD JOB" from LISTSERV@LISTSERV.UIC.E
DU...
25 May 1996 00:04:08 File forwarded to LIME.EASE.LSOFT.COM for   1 recipient.
25 May 1996 00:04:08 File forwarded to WIN95.DC.LSOFT.COM for   1 recipient.
25 May 1996 00:04:08 File forwarded to SPIDER.EASE.LSOFT.COM for   1 recipient.
25 May 1996 00:04:08 File forwarded to HOME.EASE.LSOFT.COM for   1 recipient.
25 May 1996 00:04:08 File forwarded to LISTSERV.GEORGETOWN.EDU for   1 recipient
.
25 May 1996 00:04:08 File forwarded to CC1.KULEUVEN.AC.BE for   1 recipient.
25 May 1996 00:04:08 File forwarded to LISTSERV.USHMM.ORG for   1 recipient.
25 May 1996 00:04:08 File forwarded to LISTSERV.CLARK.NET for   1 recipient.
```

LISTSERV also sends itself a copy:

```
25 May 1996 00:04:08 File "X-LUPD JOB" distributed to LISTSERV@PEACH.EASE.LSOFT.
COM.
```

```
25 May 1996 00:04:08 Done - 8 jobs (8 rcpts), 1 outbound file (1 rcpt).
25 May 1996 00:04:08 Processing file 8206371 from LISTSERV@PEACH.EASE.LSOFT.COM
25 May 1996 00:04:08 From LISTSERV@LISTSERV.UIC.EDU: X-LUPD FWD=NO DATE=19960524
23:00:02 HDR=YES
```

The following entry tells LISTSERV to replace the information it has for NEWIO-L in its global list of lists:

```
> REP NEWIO-L NEWIO-L /Info about replacing ILLINET Online hardware and software
/2616677089
> CKS 3881006631
25 May 1996 00:04:10 GLOBLIST FILE has been successfully updated.
```

Note that entries for deleted lists also are updated (deleted from GLOBLIST FILE):

```
25 May 1996 00:24:11 Processing file 8207550 from LISTSERV@PEACH.EASE.LSOFT.COM
25 May 1996 00:24:11 From LISTSERV@VM1.NODAK.EDU: X-LUPD FWD=NO DATE=1996052423:
00:04 HDR=YES
> DEL WLREHAB
> DEL WDAMAGE
> DEL POWER-L
> DEL QUEST
> DEL RS1-L
> DEL SANGEET
> DEL SPRINT-L
> DEL STAFFGOV
> DEL TAG-L
> DEL TELUGU
> DEL THEORY-A
> DEL THEORY-B
> DEL THEORY-C
> DEL THEORYNT
> DEL TOW
> DEL TWSGIS-L
> DEL UND-SEMI
> CKS 3794276653
25 May 1996 00:24:13 GLOBLIST FILE has been successfully updated.
```

Note that if the "CKS" checksum does not check out, the job is discarded without being processed.

10.3.12. Valid "OK" confirmation received

Here is a set of typical log entries for the receipt of a valid "OK" confirmation. Notice that LISTSERV accepts the OK and then issues itself the command that required confirmation. Other than the "OK", this behavior (at least in the log) is identical to how LISTSERV handles commands that do not require confirmation.

```
225 May 1996 00:04:58 Processing file 8206418 from MAILER@PEACH.EASE.LSOFT.COM
25 May 1996 00:04:58 From xxxxxxxxx@SOL.KISS.DE: OK
25 May 1996 00:04:58 From xxxxxxxxx@SOL.KISS.DE: SIGNOFF AFWEEKLY
25 May 1996 00:04:58 To   xxxxxxxxx@SOL.KISS.DE: You have been removed from the
AFWEEKLY list.
25 May 1996 00:04:58 Sending FAREWELL message to xxxxxxxxx@SOL.KISS.DE
25 May 1996 00:04:58 Sent information mail to xxxxxxxxx@SOL.KISS.DE
25 May 1996 00:04:58 Sent information mail to:
                     > xxxxxxx@AFSYNC.HQ.AF.MIL xxxxxxx@AFSYNC.HQ.AF.MIL
                     > xxxxx@AFNEWS.PA.AF.MIL xxxxxx@MASTER.PA.AF.MIL
25 May 1996 00:04:58 Sent information mail to xxxxxxxxx@SOL.KISS.DE
```

Note that the "OK" may contain the return cookie:

```
25 May 1996 00:08:50 Processing file 8206772 from MAILER@PEACH.EASE.LSOFT.COM
25 May 1996 00:08:50 From xxxxxxxxxx@GENIE.COM: OK 71365E
25 May 1996 00:08:50 From xxxxxxxxxx@GENIE.COM: SUBSCRIBE AFNS Mxxxx Exxx Kxxxxx
er
```

```
25 May 1996 00:08:51 To   xxxxxxxxxx@GENIE.COM: You have been added to the AFNS
list.
25 May 1996 00:08:51 Sent information mail to xxxxxxxxxx@GENIE.COM
25 May 1996 00:08:51 Sending WELCOME message to xxxxxxxxxx@GENIE.COM
25 May 1996 00:08:51 Sent information mail to xxxxxxxxxx@GENIE.COM
25 May 1996 00:08:51 Sent information mail to:
                     > xxxxxxx@AFSYNC.HQ.AF.MIL xxxxxxx@AFSYNC.HQ.AF.MIL
                     > xxxxx@AFNEWS.PA.AF.MIL xxxxxx@MASTER.PA.AF.MIL
25 May 1996 00:08:51 Sent information mail to xxxxxxxxxx@GENIE.COM
```

10.3.13. Invalid "OK" confirmation received

"OK" confirmation codes relate to specific userids. For instance, if you try to execute a command as "someuser@someplace.com" and reply to the "OK" from "someuser@unix1.someplace.com", LISTSERV will not perform so-called "fuzzy matching" or do a DNS lookup to determine whether or not "unix1.someplace.com" maps to "someplace.com". Therefore, since the code and the userid don't match, LISTSERV will respond that the confirmation code does not match any pending job.

This message is also sent if the "OK" comes after the "cookie" expires, since of course there is no longer any pending job matching it.

```
25 May 1996 01:16:28 Processing file 8211043 from MAILER@PEACH.EASE.LSOFT.COM
25 May 1996 01:16:29 From xxxxxx@MSN.COM: ok
25 May 1996 01:16:29 To   xxxxxx@MSN.COM: The confirmation code you gave (78B484)
does not correspond to any pending (...)
```

10.3.14. User is already subscribed to a given list

Note that the user may be trying to change his "real name" field in the list. In any case, if the "real name" field matches the one in the SUBSCRIBE command, the following is logged:

```
25 May 1996 00:11:42 Processing file 8206935 from MAILER@PEACH.EASE.LSOFT.COM
25 May 1996 00:11:42 From xxxxxxxx@NS.NET: SUBSCRIBE IN-TOUCH Txxxxx Hxxxxxx
25 May 1996 00:11:42 To   xxxxxxxx@NS.NET: You are already subscribed to the IN-
TOUCH list as "Txxxxx Hxxxxxx.
25 May 1996 00:11:42 Sent information mail to therrera@NS.NET
```

If the "real name" field is different, the following is logged:

```
25 May 1996 00:16:18 Processing file 8207278 from MAILER@PEACH.EASE.LSOFT.COM
25 May 1996 00:16:18 From xxxxxxx@EROLS.COM: SUBSCRIBE IN-TOUCH Rxxxxxx Sxxx
25 May 1996 00:16:18 To   xxxxxxx@EROLS.COM: The name associated with your xxxxx
xx@EROLS.COM subscription has been (...)
25 May 1996 00:16:18 Sent information mail to xxxxxxx@EROLS.COM
```

10.3.15. User has included non-command text (e.g., a .sig file) in his mail to LISTSERV

```
25 May 1996 00:32:07 Processing file 8207703 from MAILER@PEACH.EASE.LSOFT.COM
25 May 1996 00:32:07 From MAILER@PEACH.EASE.LSOFT.COM: X-ADMMAIL OWNER-EXCEL-L M
ailer-Daemon@inf.com
25 May 1996 00:32:15 Processing file 8207705 from MAILER@PEACH.EASE.LSOFT.COM
25 May 1996 00:32:15 From xxxxxxxx@IX.NETCOM.COM: ok
25 May 1996 00:32:15 From xxxxxxxx@IX.NETCOM.COM: sub WINNT-L Txxxx D. Sxxxx
25 May 1996 00:32:15 To   xxxxxxxx@IX.NETCOM.COM: You have been added to the WIN
NT-L list.
25 May 1996 00:32:15 Sent information mail to xxxxxxxx@IX.NETCOM.COM
25 May 1996 00:32:15 From xxxxxxxx@IX.NETCOM.COM: Txxxx Sxxxx
```

```
25 May 1996 00:32:15 To   xxxxxxxx@IX.NETCOM.COM: Unknown command - "TXXXX". Try
 HELP.
25 May 1996 00:32:15 From xxxxxxxx@IX.NETCOM.COM: xxxxxxxx@ix.netcom.com
25 May 1996 00:32:15 To   xxxxxxxx@IX.NETCOM.COM: Unknown command - "XXXXXXXX@IX
.NETCOM.COM". Try HELP.
25 May 1996 00:32:15 From xxxxxxxx@IX.NETCOM.COM: http://www.netcom.com/~xxxxxxx
x/
25 May 1996 00:32:15 To   xxxxxxxx@IX.NETCOM.COM: Unknown command - "HTTP:". Try
 HELP.
25 May 1996 00:32:15 Sent information mail to xxxxxxxx@IX.NETCOM.COM
```

10.3.16. Response to list owner or LISTSERV maintainer commands

```
25 May 1996 00:36:08 From xxxxxxx@WEBCOM.COM: quiet delete iatn xxxxxx@po.pacifi
c.net.sq pw=xxxxx
25 May 1996 00:36:08 To   xxxxxxx@WEBCOM.COM: xxxxxx@PO.PACIFIC.NET.SQ is not su
bscribed to the IATN list.
25 May 1996 00:36:08 Sent information mail to xxxxxx@WEBCOM.COM
```

10.3.17. Response to a user who tries to post to a held list (or one for which PRIMETIME is in effect)

```
25 May 1996 00:36:42 Processing file 8208564 from MAILER@PEACH.EASE.LSOFT.COM
25 May 1996 00:36:43 To   xxxxx@IQUEST.NET: The distribution of your message da
ted  Fri, 24 May 1996 23:32:59 -0500 with (...)
25 May 1996 00:36:43 Sent information mail to xxxxx@IQUEST.NET
```

10.3.18. Command forwarded via GLX from another host

```
25 May 1996 00:48:56 Processing file 8210093 from MAILER@PEACH.EASE.LSOFT.COM
25 May 1996 00:48:57 From LISTSERV@SEARN.SUNET.SE: X-B64 ID=X-FOR.JOB CLASS=M
25 May 1996 00:48:57 Rescheduled as: 8210094
25 May 1996 00:48:57 Processing file 8210094 from LISTSERV@PEACH.EASE.LSOFT.COM
25 May 1996 00:48:57 From LISTSERV@SEARN: X-FOR FWDED=2 xxxxxxxx@CS.SFU.CA SUBSC
RIBE WINNT-L Pxxxxxxxxx Pxxxxxxxx
25 May 1996 00:48:57 Requesting confirmation, cookie=4E79B8
25 May 1996 00:48:57 Sent information mail to xxxxxxxx@CS.SFU.CA
25 May 1996 00:48:57 Sent information mail to xxxxxxxx@CS.SFU.CA
```

10.3.19. Netwide DELETE (X-DEL jobs)

```
25 May 1996 01:13:25 Processing file 8210957 from MAILER@PEACH.EASE.LSOFT.COM
25 May 1996 01:13:25 From LISTSERV@PSUVM.PSU.EDU: X-B64 ID=X-DEL.JOB CLASS=A
25 May 1996 01:13:25 Rescheduled as: 8210958
25 May 1996 01:13:25 Processing file 8210958 from LISTSERV@PEACH.EASE.LSOFT.COM
25 May 1996 01:13:25 From LISTSERV@PSUVM: DIST2 I=Y FROM=LISTSERV@VM1.NODAK.EDU
FORW(CRC) HOST(626 626 626 626 626 626 691 (...)
25 May 1996 01:13:25 Distributing file "X-DEL JOB" from LISTSERV@VM1.NODAK.EDU..
.
25 May 1996 01:13:25 File forwarded to HOME.EASE.LSOFT.COM for   1 recipient.
25 May 1996 01:13:25 File forwarded to SPIDER.EASE.LSOFT.COM for   1 recipient.
25 May 1996 01:13:25 File forwarded to WIN95.DC.LSOFT.COM for   1 recipient.
25 May 1996 01:13:25 File forwarded to INDIAN.DC.LSOFT.COM for   1 recipient.
25 May 1996 01:13:25 File forwarded to LIME.EASE.LSOFT.COM for   1 recipient.
25 May 1996 01:13:25 File forwarded to LISTSERV.GEORGETOWN.EDU for   1 recipient
.
25 May 1996 01:13:25 File forwarded to CC1.KULEUVEN.AC.BE for   1 recipient.
25 May 1996 01:13:26 File forwarded to LISTSERV.USHMM.ORG for   1 recipient.
25 May 1996 01:13:26 File forwarded to LISTSERV.CLARK.NET for   1 recipient.
25 May 1996 01:13:26 File "X-DEL JOB" distributed to LISTSERV@PEACH.EASE.LSOFT.C
OM.
25 May 1996 01:13:26 Done - 9 jobs (9 rcpts), 1 outbound file (1 rcpt).
25 May 1996 01:13:26 Processing file 8210968 from LISTSERV@PEACH.EASE.LSOFT.COM
25 May 1996 01:13:26 From LISTSERV@VM1.NODAK.EDU: X-FOR xxxx@SDSUMUS.SDSTATE.EDU
 SIGNOFF * FWDED=2 (NETWIDE
```

10.3.20. FIOC cache notifications

LISTSERV caches files that it uses for efficiency. Occasionally, you may see a warning that the FIO cache has reached a preset limit (FIOC_WARNING in the site configuration file). See the FIOC_CACHE, FIOC_TRIM, and FIOC_WARNING site configuration variables in Appendix C for more information. If you get a lot of these warnings, you may want to consider adjusting the cache values.

```
25 May 1996 01:24:06 Virtual disk slot E(E:\EASE\PEACH\FTP\VBDATA-L) released.
25 May 1996 01:24:06 Directory E:\LISTSERV\ARCHIVES\SPAM-L accessed as virtual
disk slot E.

*** FIO file cache now totals 20659k. A list of cached files follows. ***

File size  Usage  Flags  File name
---------  -----  -----  ---------
    5071k      1   U     E:\LISTSERV\ARCHIVES\SPAM-L\SPAM-L.LOG9605
       2k      1         E:\listserv\TMP\LISTSERV.CMSUT1
     242k      1         E:\listserv\main\PERMVARS.FILE
       4k      1         E:\listserv\main\VBDATA-L.DIGEST
     378k      1         E:\EASE\PEACH\FTP\VBDATA-L\VBDATA-L.LOG9605D
     121k      1         E:\listserv\main\POSTNOTE.LIST
     282k      1         E:\listserv\main\FUTURESUPERSTOCK.LIST
     115k      1         E:\listserv\main\AUTOTECHNET.LIST
       6k      1         E:\LISTSERV\ARCHIVES\HONYAKU\HONYAKU.DIGEST
       1k      3         E:\listserv\main\DIGESTS.FILE
      11k      3         E:\listserv\TMP\TEMP.FILELIST
```

(Many more lines were deleted)

10.3.21. Web archive/administration interface logging (starting with 1.8d)

When LISTSERV receives a request from the 'wa' interface, it logs the activity as below. Note that LISTSERV receives an X-LOGIN command from 'wa' and issues a validation code if the password and the user's e-mail address match. Note also that LISTSERV logs the IP address of the machine making the request via 'wa'.

```
5 Aug 1997 10:51:08 From IUSR_XXX@PEACH.EASE.LSOFT.COM: X-LOGIN xxxxxx@lsoft.com
206.241.13.58 PW=YYYYYYYY
5 Aug 1997 10:51:08 To   IUSR_XXX@PEACH.EASE.LSOFT.COM: ***OK*** 6093C01B81CF68D42B
5 Aug 1997 10:51:09 From IUSR_XXX@PEACH.EASE.LSOFT.COM: X-LOGCK 6093C01B81CF68D42B
AUTHINFO(206.241.13.58) WM: LIST OWNED
5 Aug 1997 10:51:21 From IUSR_XXX@PEACH.EASE.LSOFT.COM: X-LOGCK 6093C01B81CF68D42B
AUTHINFO(206.241.13.58) OWNER(TEST) WM: GET TEST (HDR NOL (...)
```

The following indicates a timeout after 60 seconds of inactivity:

```
4 Feb 1998 10:26:53 From IUSR_XXX@PEACH.EASE.LSOFT.COM: X-LOGCK 37BA2700C7AA3AE9EE
AUTHINFO(208.141.38.1) TIMEKILL(60)
```

This indicates a web archive search and the response:

```
4 Feb 1998 10:26:53 From IUSR_XXX@PEACH.EASE.LSOFT.COM: X-LOGCK 3EA2D501187014BB04
AUTHINFO(204.149.110.125) NOTEBOOK(spam-l) DBS: SEARC (...)
4 Feb 1998 10:26:56 Reindexing SPAM-L LOG9802A...
4 Feb 1998 10:27:08 To   IUSR_PEACH@LSOFT.COM: -> 23 matches.
```

This indicates a subscrption via the web interface:

```
4 Feb 1998 10:27:08 From IUSR_PEACH@LSOFT.COM: X-LOGCK 37BA2700C7AA3AE9EE
AUTHINFO(208.141.38.1) WM: SUBSCRIBE WINNT-L Kevin Mc (...)
4 Feb 1998 10:27:08 Requesting confirmation, cookie=1723C284
```

10.3.22. X-SPAM jobs

From time to time a server receives X-SPAM jobs from other LISTSERV hosts. These jobs are "spam alerts" which tell the server that spam has been detected at another site and a user has been put under 48 hour quarantine.

```
14 Jun 1999 06:48:52 Processing file 41256617 from MAILER@PEACH.EASE.LSOFT.COM
14 Jun 1999 06:48:52 From LISTSERV@PLUM.EASE.LSOFT.COM: X-B64 ID=X-SPAM.JOB ASCII
CLASS=J
14 Jun 1999 06:48:52 Rescheduled as: 41256618
14 Jun 1999 06:48:52 Processing file 41256618 from LISTSERV@PEACH.EASE.LSOFT.COM
14 Jun 1999 06:48:52 From LISTSERV@PLUM.EASE.LSOFT.COM: DIST2 I=Y
FROM=LISTSERV@LISTSERV.AOL.COM FORW(CRC) HOST(626)
14 Jun 1999 06:48:52 Distributing file "X-SPAM JOB" from
LISTSERV@LISTSERV.AOL.COM...
14 Jun 1999 06:48:52 File "X-SPAM JOB" distributed to
LISTSERV@PEACH.EASE.LSOFT.COM.
14 Jun 1999 06:48:52 Done - 1 outbound file (1 rcpt).
14 Jun 1999 06:48:52 Processing file 41256619 from LISTSERV@PEACH.EASE.LSOFT.COM
14 Jun 1999 06:48:52 From LISTSERV@LISTSERV.AOL.COM: X-SPAM
exxxx_xxxxxxxxx@YAHOO.COM 4214CE9E
14 Jun 1999 06:48:52 -> Registered.
```

At the end of this, the address `exxxx_xxxxxxxxx@YAHOO.COM` has been registered as a spammer and will be quarantined for 48 hours.

10.3.23. X-TBREG jobs

X-TBREG jobs are sent out by all LISTSERV Lite Free Edition servers and any other servers that are set to TABLELESS runmode (see "RUNMODE" in Appendix C). These jobs register your server with a central L-Soft server, and are important for two reasons: first, so that your server and lists can show up in the L-Soft-maintained CataList service; and second, so that your server can participate correctly in the LISTSERV distributed server model without needing to update LISTSERV's networking tables on a regular basis.

Non-Free Edition servers set to NETWORKED or STANDALONE runmode do not generate X-TBREG jobs, although they may receive them from remote hosts to be cached for DISTRIBUTE purposes.

For more information about server registration, see chapter 5.6. Note that LISTSERV Lite Free Edition servers cannot change their runmode and therefore will always self-register this way.

10.3.24. Responses to LVMON@VM.SE.LSOFT.COM

If you are running in Networked or Tableless mode you may see these from time to time:

```
13 Mar 2000 16:45:13 From LVMON@VM.SE.LSOFT.COM: RELEASE
13 Mar 2000 16:45:13 To   LVMON@VM.SE.LSOFT.COM: LISTSERV(R) High Performance fo
r Windows NT version 1.8d, managed by:
13 Mar 2000 16:45:13 From LVMON@VM.SE.LSOFT.COM: SHOW
13 Mar 2000 16:45:13 From LVMON@VM.SE.LSOFT.COM: SHOW WWW_ARCHIVE_URL
13 Mar 2000 16:45:13 To   LVMON@VM.SE.LSOFT.COM: WWW_ARCHIVE_URL = http://peach.
ease.lsoft.com/scripts/wa.exe
13 Mar 2000 16:45:13 From LVMON@VM.SE.LSOFT.COM: SHOW CTR 200003
13 Mar 2000 16:45:13 From LVMON@VM.SE.LSOFT.COM: SHOW CTR 200002
13 Mar 2000 16:45:13 Sent information mail to LVMON@VM.SE.LSOFT.COM
```

VM.SE.LSOFT.COM is a central L-Soft server that collects publicly-available statistics and other information for the CataList service and for L-Soft's use in developing usage

metrics. All of the commands sent by LVMON are documented and can be issued by any user. See also 5.7 of this manual for more information on inter-server information sharing.

10.3.25. MIME parser messages (1.8e)

In 1.8e, LISTSERV's MIME parser was completely rewritten to address a number of issues, not the least of which were to add uuencoded binary filtering support and anti-virus support. When a MIME message is received by LISTSERV, information about the makeup of the message is logged; for example, here is a message with an RTF file attached which has been sent to a list:

```
9 Nov 2001 10:26:52 Processing file 0118 from MAILER@LISTSERV.EXAMPLE.COM
Part 1 [1-243]: MULTIPART/MIXED; parent=0; boundary="--====================_480
189806==_"
Part 2 [1-6]: TEXT/PLAIN; parent=1; boundary=""
Part 3 [7-235]: APPLICATION/RTF; parent=1; boundary=""
Parts: 3 skipped: 0
```

And here is a similar message posted to LISTSERV itself with a command in the plain text part and an RTF attachment:

```
9 Nov 2001 10:31:44 Processing file 0121 from MAILER@LISTSERV.EXAMPLE.COM
Part 1 [1-235]: MULTIPART/MIXED; parent=0; boundary="--====================_480
480784==_"
Part 2 [1-5]: TEXT/PLAIN; parent=1; boundary=""
Part 3 [6-234]: APPLICATION/RTF; parent=1; boundary=""
Parts: 3 skipped: 0
Rewriting part 2
9 Nov 2001 10:31:44 -> Decoding MIME message parts...
9 Nov 2001 10:31:44 Rescheduled as: 0122
9 Nov 2001 10:31:44 Processing file 0122 from LISTSERV@LISTSERV.EXAMPLE.COM
Part 1 [1-1]: TEXT/PLAIN; parent=0; boundary=""
Parts: 1 skipped: 0
9 Nov 2001 10:31:44 From ncb@EXAMPLE.COM: show lic
9 Nov 2001 10:31:44 Sent information mail to ncb@EXAMPLE.COM
```

In this latter message, LISTSERV sees the three MIME parts of the message and skips them all, passing the message to the MIME parser. LISTSERV then rewrites the second MIME part (the TEXT/PLAIN attachment that actually contains the command(s) to be executed) and places a new job containing only the TEXT/PLAIN attachment in the spool, which is then executed by the command processor. The APPLICATION/RTF attachment is simply discarded.

Finally, here is an example of a subscription request received from a site running an InterScan VirusWall gateway, which wraps the original message in a special MIME wrapper. The actual "raw" message as received by LISTSERV looks like this:

```
X-MimeOLE: Produced By Microsoft Exchange V6.0.5762.3
content-class: urn:content-classes:message
MIME-Version: 1.0
Content-Type: multipart/mixed;
       boundary="------------InterScan_NT_MIME_Boundary"
Subject:
Date: Mon, 26 Nov 2001 14:51:18 -0800
Message-ID: <66C896FC5FA3EB4E9BC85569962F378403BCC430@example.com>
X-MS-Has-Attach:
X-MS-TNEF-Correlator:
Thread-Index: AcF2zNx1xlGA3wa1RES/vvt+eFWNGw==
From: "Joe User" <joe@example.com>
To: <LISTSERV@LISTSERV.EXAMPLE.COM>
Return-Path: joe@example.com
```

```
X-OriginalArrivalTime: 26 Nov 2001 22:51:19.0067 (UTC)
FILETIME=[DCAAC6B0:01C176CC]

This is a multi-part message in MIME format.

--------------InterScan_NT_MIME_Boundary
Content-Type: multipart/alternative;
       boundary="----_=_NextPart_001_01C176CC.DC8A9762"

------_=_NextPart_001_01C176CC.DC8A9762
Content-Type: text/plain;
       charset="us-ascii"
Content-Transfer-Encoding: quoted-printable

SUBSCRIBE TEST Joe User


------_=_NextPart_001_01C176CC.DC8A9762
Content-Type: text/html;
       charset="us-ascii"
Content-Transfer-Encoding: quoted-printable

SUBSCRIBE TEST Joe = User
=00
------_=_NextPart_001_01C176CC.DC8A9762--

--------------InterScan_NT_MIME_Boundary--
```

Versions of LISTSERV previous to 1.8e were unable to parse any text out of this type of message, but from version 1.8e forward the MIME parser is able to find the plain text attachment and execute it:

```
27 Nov 2001 10:13:01 Processing file 0083 from MAILER@LISTSERV.EXAMPLE.COM
Part 1 [1-24]: MULTIPART/MIXED; parent=0; boundary="--------------InterScan_NT_M
IME_Boundary"
Part 2 [3-0]: MULTIPART/ALTERNATIVE; parent=1; boundary="------_=_NextPart_001_0
1C176CC.DC8A9762"
Part 3 [7-14]: TEXT/PLAIN; parent=2; boundary=""
Part 4 [15-21]: TEXT/HTML; parent=2; boundary=""
Parts: 4 skipped: 0
Rewriting part 3
27 Nov 2001 10:13:01 -> Decoding MIME message parts...
27 Nov 2001 10:13:01 Rescheduled as: 0084
XMV.finalrc: 3
27 Nov 2001 10:13:01 Processing file 0084 from LISTSERV@LISTSERV.EXAMPLE.COM
Part 1 [1-1]: TEXT/PLAIN; parent=0; boundary=""
Parts: 1 skipped: 0
27 Nov 2001 10:13:01 From joe@EXAMPLE.COM: SUBSCRIBE TEST Joe User
27 Nov 2001 10:13:01 Sent information mail to joe@EXAMPLE.COM
27 Nov 2001 10:13:01 Sending WELCOME message to joe@EXAMPLE.COM
27 Nov 2001 10:13:01 Sent information mail to joe@EXAMPLE.COM
```

10.3.26. Content filter rejection message (1.8e)

```
5 Dec 2001 17:31:07 -> Rejected:
* Your posting to the TEST list has been rejected by the content filter. OOO
* messages are not allowed on this list.
5 Dec 2001 17:31:07 Sent information mail to ncb@EXAMPLE.COM
```

10.4. Interpreting the SMTP logs (Windows servers only)

Note that if you are running L-Soft's LSMTP™ product and do not have the SMTPL.EXE "listener" running, these logs will not be generated.

These logs are for the SMTPL.EXE "listener" service, and are called **SMTP-*yyyymmdd*.LOG**. They are found in the **/LISTSERV/LOG** directory with the other

LISTSERV system logs. A typical "listener" log looks like this:

```
13 Mar 1998 14:13:31 LISTSERV SMTP listener, version 1.0d
13 Mar 1998 14:13:31 Copyright L-Soft international, 1994-98
13 Mar 1998 14:13:31 Initialization complete.
13 Mar 1998 14:15:59 New connection (1) from 128.118.56.2
13 Mar 1998 14:18:50 New connection (2) from 199.3.65.5
13 Mar 1998 14:18:59 >>>(1) Connection closed by remote host.
13 Mar 1998 14:19:05 Closing connection (2) from 199.3.65.5.
13 Mar 1998 14:19:08 New connection (3) from 205.186.43.7
13 Mar 1998 14:20:08 Closing connection (3) from 205.186.43.7.
```

Figure 10.2. Typical SMTP log for the SMTPL.EXE "listener"

This log simply keeps track of incoming SMTP connections to your server. It adds an entry for each new connection as it opens and closes. Additionally, if a connection is closed abnormally (e.g., by the remote host before any data is sent), the condition is logged. The SMTP log is generally useful only for debugging the SMTP listener, although it may also be of use in debugging problems with specific remote hosts.

10.5. Interpreting the SMTP "worker" log entries (non-VM only)

If you have SMTP "workers" activated on a Windows or OpenVMS server, each "worker" creates a separate log for itself. These logs are found in:

OpenVMS: **LISTSERV_ROOT:[LOG]SMTPS*n*-*yyyymmdd*.LOG**
Windows: **LISTSERV\LOG\SMTPS*n*-*yyyymmdd*.LOG**

If you are using SMTP "workers" under unix, no **smtps*.log** files are generated. Rather, the **lsv** "worker" sub-processes log information to the main **listserv.log** file.

(LISTSERV automatically "turns" the OpenVMS and Windows SMTPS logs at midnight. "***yyyymmdd***" is the year, month and day of the log, for example, **LISTSERV-19980104.LOG** is the log for 4 January 1998. "***n***" refers to the worker that is generating the log. Each worker generates a log, so if you have 10 workers running, you will have logs for **SMTPS1** through **SMTPS10**.)

A typical SMTP "worker" log looks like this:

```
03 Jun 1996 13:49:21 *** LSMTP extensions activated ***

03 Jun 1996 13:49:03 500 error reading data line

03 Jun 1996 13:49:04 Renaming 8891738.MAIL to 8891738.MAIL-ERR1.

03 Jun 1996 14:06:54 Error opening '8894361.MAIL': Permission denied
03 Jun 1996 14:09:55 Error opening '8894695.MAIL': Permission denied
03 Jun 1996 14:40:45 Error opening '8897761.MAIL': Permission denied
```

Figure 10.3. Typical SMTPS log for the SMTPW.EXE SMTP "workers"

The SMTP worker logs keep track of events that occur while the SMTP workers are delivering mail to the external mail host(s) defined in the **SMTP_FORWARD_*n*** variables in the site configuration file. In general, the events logged will be errors of one kind or another. For instance, the first error in the example above indicates an SMTP error, and the second indicates that a file that caused an error has been renamed so that it can be examined for debugging.

The last three errors are normal and can generally be ignored. They refer to the fact that two workers have noticed a .MAIL file that exists in the queue, and that one of them

grabbed it before the other one did. The first worker locks the file and the second worker is denied permission to open it. Sometimes the first worker may process the mail so quickly that the error will read "File not found" rather than "Permission denied"; either way, this should not be considered alarming unless mail is actually not being delivered.

The first line in the example simply indicates that special extensions for use with LSMTP have been activated. This message will appear only if you are licensed for LISTSERV-HPO.

10.6. Change logs

This feature and keyword are not available in LISTSERV Lite.

In version 1.8d and following, this feature is available to track certain operations on lists with "Change-Log= Yes" coded into their headers. As noted elsewhere in this manual, setting the keyword to "Yes" causes LISTSERV to write a file called ***listname*.CHANGELOG** (or ***listname* CHANGELG** for VM) into LISTSERV's A directory or A disk. CHANGELOG files are automatically available for list owners and site maintainers to **GET** and **PUT** (**PUT** normally being used to delete them) like any other file. It is not necessary to make catalog entries for CHANGELOG files.

Non-VM version 1.8e added the ability to define change-logs for regular mailing lists that rotate on a regular basis, either DAILY, WEEKLY, MONTHLY, or YEARLY. (The option SINGLE is provided for backward compatibility and is still the default, that is, LISTSERV does not ever rotate the change-log). The rotation periods are specified as a second parameter to the Change-Log= list header keyword, for example:

```
Change-Log= Yes,Monthly
```

or

```
Change-Log = Yes,Single
```

(the latter being equivalent to "Change-Log= Yes" as noted above). Rotated change-logs are renamed with the format ***listname*.CHANGELOG-*yyyy*[*mm*[*dd*|*w*]]** (depending on the rotation period selected) and may be retrieved with the GET command as usual.

Documented Restriction: NOLIST-* change-logs are always SINGLE.

Documented Restriction: Change-Log rotation is not available under VM.

The operations monitored are **ADD**, **AUTODEL**, **BOUNCE**, **CHANGE**, **DELETE**, **POST**, **READD**, **RESUBSCRIBE**, **SET**, **SIGNOFF**, and **SUBSCRIBE**. All abbreviations and synonyms are translated to their "official" forms, i.e., **SUB**, **JOIN**, and **SIGNON** are all translated to **SUBSCRIBE** for the purposes of the changelog. This makes it easy to write scripts to come up with statistics for a given list--you don't have to take variations of the commands into account.

Sample changelog entries are:

```
20011025100330 ADD xxxxxxx@JPS.NET Lxxxx Pxxxxxxxx
20011025120049 AUTODEL xxxxxxxx@EISA.NET.AU
20011025131221 BOUNCE xxxxxxxx@NONEXIST.COM
20011025214433 CHANGE xxxx@MCS.COM txxx@MCS.NET
20011025214434 DELETE xxxxx@SINGNET.COM.SG
```

```
20011025060052 EXPIRE joe@EXAMPLE.COM
20011025232441 POST xxxxxxxx@M4.SPRYNET.COM Printer Drivers
20011025000400 RESUBSCRIBE xxxx@MAIL.MECHWART.MUMSZKI.HU Mxxxxx Zxxxxx
20011025113947 SET xxxxxx.xxxxxxx@WDC.COM REPRO
20011025082712 SIGNOFF xxxxxxxx@STARNET.NET.AR
20011025085642 SUBSCRIBE xxxxxxxx@LYCOSMAIL.COM Kxx Wxxxxxx
```

As you see, the **SET** entry tells you what options were set, and the **POST** entry tells you what the subject of the posting was. **RESUBSCRIBE** is a **SUBSCRIBE** operation that takes place when the user is already subscribed to the list, for example, to change the real name field in the user's subscription. **BOUNCE** is a special operation that takes place when using the "no-list" bounce-processing mechanism (described in the *Developer's Guide to LISTSERV*). **EXPIRE** indicates that the renewal "grace period" for the subscriber in question has expired without the user sending a **CONFIRM** command and the user has been deleted from the list. Otherwise these entries are fairly self-explanatory.

In 1.8e you may also see

```
20011025165954 IMPORT 2 0 0
```

This type of record shows the basic results of an **ADD IMPORT** job. The numbers following the **IMPORT** recordtype stand for (1) recipients added, (2) entries changed, and (3) recipients forwarded to another host, in other words, the same results that are sent back to the **ADD IMPORT** invoker on completion of the job.

Additionally, in 1.8e the following entry is always written at the top of a new change-log file:

```
20011025120731 SUBCOUNT 111
```

This tells you how many subscribers existed on the list before this particular change-log was started. It is handy to know if you are trying to track historical subscription count trends.

One caveat: Even with the new rotation feature in 1.8e and following, changelog files can get fairly large. It may be necessary to monitor the size of the changelogs on your server and delete them as disk space fills up. If a list owner wants the changelog information for his list, he should be instructed that it is his responsibility to **GET** old changelog files regularly and delete them himself each time. There is no facility in LISTSERV to delete the rotated changelog files automatically.

10.7. Using LISTSERV logs and SHOW CTR to extract server statistics

While LISTSERV does not provide a native method to display statistics, it is entirely possible to use scripts to post-process the LISTSERV console logs and changelogs to provide a wide range of statistics. Additionally, the native SHOW CTR command provides a breakdown of current and past LISTSERV traffic.

10.7.1. Sample log-processing scripts

There are two unsupported REXX scripts available from L-Soft which can be used to extract various statistics from the LISTSERV console log and from changelogs. See

ftp://ftp.lsoft.com/listserv/windows/contrib/cntpost.rexx
ftp://ftp.lsoft.com/listserv/windows/contrib/stats.rexx

Both of these scripts were written for Regina REXX and are (in their current incarnations) Windows-specific, but they could probably be ported to the unix version of Regina with a little work.

Please note that these scripts are completely unsupported. Use at your own risk. Neither the script author or the L-Soft support department are able to field support requests for unsupported scripts.

Be sure to read the internal documentation before attempting to use either of these scripts.

The first script is used to compile posting data from all lists on the server and issues a weekly report that looks like this:

```
LISTSERV posting statistics for LISTSERV.EXAMPLE.COM since 8 Feb 1999
-------------------------------------------------------------------------------
                                Indiv.    Digests    Indexes    Total
List Name                       Postings  Generated  Generated  Recipients
=========                       ========  =========  =========  ==========
LIST1                                223          7          0       14574
LIST2                                  0          0          0           0
LIST3                                 12          7          7         825
-------------------------------------------------------------------------------
Totals:                              235         14          7       15399

DISTRIBUTE and MAIL-MERGE jobs sent by individuals:

  Start Date/Time      End Time Rcpts    Job Name Invoker
= =================== ======== ======== ======== =======
D  8 Feb 1999 14:07:08 14:07:09       29 DISTJOB1 USER@EXAMPLE.COM
M 12 Feb 1999 10:31:23 10:31:25    10334 MMDISTJO USER@EXAMPLE.COM
   >>> Full job name:  MMDISTJOB25
   * FROM=mmdistjob25-nolist@listserv.example.com

This report was generated by cntpost.rexx version 1.8-fix11c 1999/02/19
```

The second script is designed to be run against a ***listname*.CHANGELOG** file and produces a report similar to the old VM STATS command output (addresses have been changed to protect the innocent). There are two options--TOP (which produces posting information for only the top x number of posters--by default this is 20, the value can be raised or lowered by changing the variable setting in the code), and NOP (which suppresses the individual posting data altogether). If neither the TOP or NOP options are specified, individual posting data are produced for each and every address that posted to the list during the period represented by the changelog, which has the potential to produce very long reports. It is probably best to run this script with redirection to a file.

Sample output from the STATS.REXX script is shown below:

```
C:\rexx>rexx stats.rexx access-l top
Statistics for ACCESS-L from 25 Jun 1998 through 10 Mar 2000 (625 days)
Total lines processed: 27495
Total subscribers on list at start of period: 1465
Total subscribers on list at end of period:   2623
Increase: +1158
Number of expired subscriptions since start of period: 289
Last posting in this changelog was on 10 Mar 2000.

                        Total units       Units/day
                        ===========       =========
ADD operations:                   0          0.0000
AUTODEL operations:            1147          1.8352
BOUNCE operations:                0          0.0000
```

```
CHANGE operations:              153          0.2448
DELETE operations:              182          0.2912
IMPORT operations:                2          0.0013
  Recipients added:               0          0.0000
  Entries changed:                2          0.0013
  Forwarded:                      0          0.0000
POST operations:              16758         26.8128
READD operations:                 0          0.0000
RESUBSCRIBE operations:          45          0.0720
SET operations (total):        2733          4.3728
  ACK:                           29          0.0464
  NOACK:                        240          0.3840
  MSGACK:                         0          0.0000
  CONCEAL:                      108          0.1728
  NOCONCEAL:                      4          0.0064
  FILES:                          2          0.0032
  NOFILES:                        1          0.0016
  MAIL:                         344          0.5504
  NOMAIL:                       688          1.1008
  DIGESTS:                     1162          1.8592
  NODIGESTS:                    160          0.2560
  INDEX:                        114          0.1824
  NOINDEX:                        5          0.0080
  REPRO:                        207          0.3312
  NOREPRO:                       97          0.1552
  MIME:                          78          0.1248
  NOMIME:                       274          0.4384
  HTML:                         213          0.3408
  NOHTML:                        79          0.1264
  TOPICS:                         0          0.0000
  FULLHDR:                       13          0.0208
  SHORTHDR:                      31          0.0496
  DUALHDR:                        5          0.0080
  IETFHDR:                        4          0.0064
  SUBJECTHDR:                    69          0.1104
  FULL822:                        0          0.0000
  SHORT822:                       0          0.0000
SIGNOFF operations:           2568          4.1088
SUBSCRIBE operations:         3908          6.2528

Ratio of SIGNOFF operations to SUBSCRIBE operations:  0.6571:1

                                                 Total
Top 20 posters                                   Units Units/day Units/mon(*)
============================================== ===== ========= =========
Gxxxxx.Wxxxxx@xxxxxxxxxxx.xx.xx                     19    3.1667    0.6333
dxxxxxx@xxx.xxx.xx                                  18    3.0000    0.6000
dxx.exxxxxx@xx.xxxxxxx.xx.xx                        11    1.8333    0.3667
fxxxxxxx@xx.xxx                                     11    1.8333    0.3667
mxxxxx@xxxxxxxxx.xxx                                 8    1.3333    0.2667
cxxxxxxx@xxxxxx.xxx                                  8    1.3333    0.2667
Hxxxxxxx@xxxxxx.xxx                                  7    1.1667    0.2333
mxx@xxxxxxxxxxxxxx.xxx                               7    1.1667    0.2333
dxxxxx.xxxxxxx@xxx-xxx.xx                            7    1.1667    0.2333
dxxxxxx@xxxxxxxxxxxx.xxx                             7    1.1667    0.2333
jxxxx@xxxxxxxxx.xxx                                  7    1.1667    0.2333
kxxxxxxxx@xxxxx.xxx                                  6    1.0000    0.2000
nxxxx_x@xxx.xxx                                      6    1.0000    0.2000
Pxxxxxx@xxxx.xxx                                     5    0.8333    0.1667
G.F.G.Wxxxxxxxx@xx.xxxxxxxx.xx                       4    0.6667    0.1333
Nxxxxxxx.Vxx@xxx.xxxxx.xx.xx                         4    0.6667    0.1333
axxxxxxx@xxxxxxxx.xxx.xxx                            4    0.6667    0.1333
kxxxxxx@xxxxxxx.xx.xx                                4    0.6667    0.1333
txxxx.x.x@xx.xxx                                     4    0.6667    0.1333
xxxxxx@xxxxxxx.xxx.xx                                3    0.5000    0.1000

----------
(*) Units per month is total units / 30 days.

Top 20 posters                                   Last posted
============================================== ===========
Gxxxxx.Wxxxxx@xxxxxxxxxxx.xx.xx                  13-Jan-1999
```

```
dxxxxxx@xxx.xxx.xx                              12-Jan-1999
dxx.exxxxxx@xx.xxxxxxx.xx.xx                    12-Jan-1999
fxxxxxxx@xx.xxx                                 13-Jan-1999
mxxxxx@xxxxxxxxx.xxx                            09-Jan-1999
cxxxxxxx@xxxxxx.xxx                             13-Jan-1999
Hxxxxxxx@xxxxxx.xxx                             13-Jan-1999
mxx@xxxxxxxxxxxxx.xxx                           12-Jan-1999
dxxxxx.xxxxxxx@xxx-xxx.xx                       13-Jan-1999
dxxxxxx@xxxxxxxxxxx.xxx                         13-Jan-1999
jxxxx@xxxxxxxxx.xxx                             13-Jan-1999
kxxxxxxxx@xxxxx.xxx                             12-Jan-1999
nxxxx_x@xxx.xxx                                 09-Jan-1999
Pxxxxxx@xxxx.xxx                                11-Jan-1999
G.F.G.Wxxxxxxxx@xx.xxxxxxxx.xx                  13-Jan-1999
Nxxxxxxx.Vxx@xxx.xxxxx.xx.xx                    13-Jan-1999
axxxxxxx@xxxxxxxx.xxx.xxx                       10-Jan-1999
kxxxxxx@xxxxxxx.xx.xx                           13-Jan-1999
txxxx.x.x@xx.xxx                                13-Jan-1999
xxxxxx@xxxxxxx.xxx.xx                           11-Jan-1999

This report was prepared by STATS.REXX 0.9 2002/01/14
```

Please note carefully that these scripts are not supported in any way by L-Soft, and your use of them is strictly at your own risk.

10.7.2. Interpreting the output of SHOW CTR

LISTSERV provides certain raw statistics in response to the command **SHOW CTR *yyyymm*** (where "yyyymm" is the year and month for which you are requesting statistics). For instance, **SHOW CTR 200110** sent to one of L-Soft's hosting servers produces the following:

```
>SHOW CTR 200110
200110 1 0 102276 14829079 9414 351624 1686 5525 218474 189356 125315 29161
245922 59850 11234881 0 1654503 21893 40050348 302198 112853 63348 4373
291703 134 73552 END EOD
```

Unfortunately this is fairly obscure (it was originally intended to be used only by LISTSERV to compile network-wide statistics) and requires a certain amount of interpretation. The fields signify, in order:

Month of report
Version number (of this report format)
Missing days
Postings to mailing lists
Number of recipients
Digests issued
Number of digest recipients
Indexes issued
Number of index recipients
DISTRIBUTE jobs processed
DISTRIBUTE jobs internally generated
Outbound DISTRIBUTE jobs
Outbound NJE files
Outbound files to MAILER
Outbound files to MAILER in non-BSMTP format
Number of recipients in the outbound files to MAILER
GLX requests
Bandwidth usage register #1
Bandwidth usage register #2

CPU usage in microseconds
Number of bounces received
Number of bounces received in non-standard format
Number of bounces detected by heuristics
Probes
Number of virus scan operations (1.8e on)
Number of viruses found (1.8e on)
Number of copies of infected messages stopped (low bit) (1.8e on)[12]
Number of copies of infected messages stopped (high bit) (14.3 on)[13]
Future use
Future use
Future use
Future use
END (tells LISTSERV that this is the end of the regular data)
EOD (tells LISTSERV that this is the end of the report)

The two bandwidth registers are used as follows:

Bandwidth used in MB = (first_register/10) + (second_register/10000000)

The two "infected messages stopped" registers are used as follows:[14]

Infected messages stopped = (low bit/10000000) + (high bit/10)

The "future use" counters are undocumented, but may be observed to increment if the SPAM_EXIT feature is enabled.

If you send a SHOW CTR command for the current month, LISTSERV inserts the following between the END and EOD markers:

XPOL
integer_value
integer_value

For instance, at the time of writing (200508), the output on 10 August at approximately 10:15 -0500 on one of L-Soft's hosting servers was

```
>SHOW CTR 200508
200508 1 0 6151 6704302 1300 223483 704 9003 34593 17229 23136 17628 74524
64977 348776 0 266059 60477 26257717 302407 68659 24196 97223 754088 2614
39464990 0 0 0 0 0 END XPOL 744 227 EOD
```

The XPOL numbers are used by L-Soft to extrapolate data for the current month and can generally be ignored.

[12] For each virus detected, this counter increases by the number of outbound copies of the virus that LISTSERV was about to send when the virus was stopped.

[13] A high-order bit was added to convert this counter from 32- to 64-bit in October 2003.

[14] While normally the high bit of a counter comes first, this case is an exception. For backward compatibility with older statistics-gathering scripts, the new high bit had to come last.

10.8. Using the system changelog to track distributions

This feature is not available in LISTSERV Lite.

This feature is available only in the LISTSERV 1.8d 2000b level set release or later running with a Classic or Classic HPO LAK. If your LISTSERV server does not have a build date of 16 July 2000 or later you do not have this feature (issue a SHOW LICENSE command to LISTSERV to check the build date).

If enabled (see Appendix C, **SYSTEM_CHANGELOG**) a file called system.changelog (SYSTEM CHANGELG under VM) is generated in LISTSERV's A directory (A disk under VM), containing records of the following sort:

```
20000605014426 MAIL-MERGE 1 0 1,MM1,owner-XYZ@GUAVA.EASE.LSOFT.COM,Re: Whatever
20000803010016 MAIL 22544 8304 21,MYLIST-L,LISTSERV@GUAVA.EASE.LSOFT.COM,owner-
mylist-l@GUAVA.EASE.LSOFT.COM,How now brown cow?
20000804134805 DIST-NJE 1 0 1,X-SPAM.JOB,LISTSERV@PLUM.EASE.LSOFT.COM,LISTSERV@
LISTSERV.EXAMPLE.COM
20000804190101 DIST-NJE 1 0 1,Netwide_SIGNOFF,LISTSERV@PLUM.EASE.LSOFT.COM,LIST
SERV@LISTSERV.EXAMPLE.COM
20000804230002 DIST-NJE 0 119 1,SUPD,LISTSERV@GUAVA.EASE.LSOFT.COM,LISTSERV@GUA
VA.EASE.LSOFT.COM
```

The records consist of four comma-separated tokens:

1. Information about the job
2. Name of the job (or list to which the mail was sent)
3. Bounce address
4. Subject line

The first token contains multiple space-separated parameters. As of this writing the parameters are:

A. Date and time the job was processed (**yyyymmddhhmmss**)
B. Type of job - MAIL, MAIL-MERGE, or DIST-NJE. Typically you will see DIST-NJE only for interserver update jobs, for example, SUPD, X-LUPD, etc.
C. Number of recipients processed locally
D. Number of recipients forwarded to another DISTRIBUTE server
E. Size of the message rounded up to the nearest kilobyte

(More parameters may be added to the first token in future versions. If you write a local application to parse the changelog records, be sure to take this into account.)

In LISTSERV 1.8e, a VIRUS job type was introduced to track intercepted viruses. Assuming that 1) the system.changelog is enabled and 2) LISTSERV's anti-virus scanning facility is enabled, LISTSERV will write records like the following to the system.changelog file for each virus encountered:

```
20020110152955 VIRUS TEST 6 EICAR-Test-File
20020128144949 VIRUS *LSWAVDD* 1 EICAR-Test-File
```

The record consists of

A. Date and time the job was processed
B. The VIRUS job type

C. The list to which the message containing the virus was posted (if sent to a -request or -owner mailbox, this part of the record contains ***LSWAVDD***, as in the second example above)
D. LISTSERV's best guess of the number of outbound copies that have been suppressed. This is not always 100% correct, but very close. When LISTSERV does not know, the value 1 is assumed. A value of 0 is possible, for instance if the list had no recipients.
E. The name or description of the virus as provided by F-Secure

10.9. Logging changelog information to a DBMS

Starting with LISTSERV 1.8e, a copy of changelog information may be stored in a DBMS. This requires a pre-existing DBMS connection configured in the usual way (see the Developer's Guide for LISTSERV, chapter 4, if you need guidance), and is controlled by three new site configuration parameters, and a table that must be created manually. The parameters, which are defined in LISTSERV's site configuration file, are **CHANGELOG_DBMS**, **CHANGELOG_DBMS_TABLE**, and **CHANGELOG_DBMS_CONNECTION** . The first two are mandatory while the third is optional.

Note: It is not possible for LISTSERV to write the changelog in multiple different tables based on various combinations of parameters. This can be accomplished on the DBMS side with a stored procedure if required.

Important: The DBMS copy is just that, a copy. The disk files (.changelog) are still generated. Naturally if they are not needed they may periodically be erased with a script.*

CHANGELOG_DBMS (mandatory)

This controls which entries are to be copied to the DBMS. Note that only entries that are actually generated can be copied. If a given change-log is disabled, it will not go to the DBMS even if you request it in this variable.

The value can be ALL or a space-separated combination of SYSTEM, NOLIST (matches any NOLIST-xxx), LISTS (matches any list) or the names of individual lists.

CHANGELOG_DBMS_TABLE (mandatory)

This contains five space-separated names:

1. The name of the table in which to store changelog entries.
2. The name of a time-stamp column in which to write the current date and time. This must be a DATE (Oracle), TIMESTAMP (DB2), DATETIME (SQL Server), or whatever else can store both date and time. This cannot be a character string.
3. The name of a VARCHAR or equivalent column storing the name of the list. The maximum size depends on the list names you choose, but it should be at least 40.
4. The name of a VARCHAR column storing the record type (BOUNCE, etc). This should probably be around 40.
5. The name of a VARCHAR column storing the parameters. This ought to be 256 or so.

If any of these parameters is missing, the setting is ignored.

CHANGELOG_DBMS_CONNECTION (optional)

This contains two optional space separated words:

1. The type of driver to be used (CLI, OCI, ODBC). This defaults to your system default driver type.
2. The server to connect to (similar to SERVER=). This defaults to the empty string, that is, the default server.

As an example, let us say that you have created a table called CHANGELOG in the database to which LISTSERV is connected. The CHANGELOG table has four columns, which are called TIMESTAMP, LISTNAME, RECORDTYPE, and PARAMETERS. So you would open your system configuration file in a text editor and add the following entries:

Windows: (site.cfg)

```
CHANGELOG_DBMS=ALL
CHANGELOG_DBMS_TABLE=CHANGELOG TIMESTAMP LISTNAME RECORDTYPE PARAMETERS
```

Unix: (go.user)

```
CHANGELOG_DBMS="ALL"
CHANGELOG_DBMS_TABLE="CHANGELOG TIMESTAMP LISTNAME RECORDTYPE PARAMETERS"
export CHANGELOG_DBMS CHANGELOG_DBMS_TABLE
```

OpenVMS: (site_config.dat)

```
CHANGELOG_DBMS "ALL"
CHANGELOG_DBMS_TABLE "CHANGELOG TIMESTAMP LISTNAME RECORDTYPE PARAMETERS"
```

If you wanted only to log the information from the system and NOLIST changelogs, you would change

Windows: **CHANGELOG_DBMS=ALL**
Unix: **CHANGELOG_DBMS="ALL"**
OpenVMS: **CHANGELOG_DBMS "ALL"**

to

Windows: **CHANGELOG_DBMS=SYSTEM NOLIST**
Unix: **CHANGELOG_DBMS="SYSTEM NOLIST"**
OpenVMS: **CHANGELOG_DBMS "SYSTEM NOLIST"**

and so forth.

11. Using the Web Adminstration Interface

[Documentation on the new 1.8e web interface was not available in time for the 1.8e product release. A supplemental guide to the new interface is forthcoming.]

LISTSERV 1.8d introduces a powerful web-based interface for the management of existing mailing lists. Virtually all list management operations can be accomplished via this interface, which is tied into LISTSERV's own password manager for security. Site managers have a special interface that can be used (among other things) to create lists--see 11.12 for details.

Please note carefully that this interface *cannot* be used to manage lists that are coded **`Validate= Yes,Confirm,NoPW`** or **`Validate= All,Confirm,NoPW`** , because passwords are not accepted for validation in those cases.

11.1. Default LISTSERV Home Page

Starting with LISTSERV 1.8d the interface includes a default home page for LISTSERV. Typically this is reached by using the URL:

On unix: **`http://yourhost.domain/cgi-bin/wa`**
On VMS: **`http://yourhost.domain/htbin/wa`**
On Windows: **`http://yourhost.domain/scripts/wa.exe`**
or **`http://yourhost.domain/cgi-bin/wa.exe`**

Of course this is not standardized; the location of the 'wa' script is determined by the value of WWW_ARCHIVE_CGI in LISTSERV's site configuration file. In any case, invoking 'wa' without any parameters returns the default home page, which looks like this:

Welcome to LISTSERV!

From this page, you can access the following services:

- Online mailing list archives.
- CataList, the official catalog of public LISTSERV® lists.
- Online documentation in HTML format:
  - LISTSERV user's guide.
  - LISTSERV list owner's quick start.
  - LISTSERV list owner's guide.
  - LISTSERV site manager's guide.
- Mailing list management interface (list owners only).
- Server management interface (LISTSERV administrator only).

If desired, the contents of this home page may be modified in the server management interface.

11.2. Logging in

You can log into the list administration interface from any list's main web archive index page (assuming that this link has not been removed by the list owner; it exists in the WWW_INDEX mail template by default). The interface may also be reached by a link from the default LISTSERV home page mentioned in 11.1, above.

To access the list administration interface without a link, you point your web browser to the "wa" script (just as you do to access the web archive interface) with a value of "LMGT1". This value must be in CAPS. Typically the interface is accessed as follows:

On unix: **`http://yourhost.domain/cgi-bin/wa?LMGT1`**
On VMS: **`http://yourhost.domain/htbin/wa?LMGT1`**
On Windows: **`http://yourhost.domain/scripts/wa.exe?LMGT1`**
or **`http://yourhost.domain/cgi-bin/wa.exe?LMGT1`**

but this is not always standardized. Note that the **`LMGT1`** parameter passed to "wa" MUST be in CAPS. Additionally, under Windows NT and Windows 95, certain web servers may insist that "wa.exe" be in CAPS as well.

In any case, once you link to this URL, you will be asked to log in:

Login required

The function you have requested requires authentication. Please enter your e-mail address and your LISTSERV password (***not*** the password you use to login to your computer or read your mail), and click on the "Login" button. If this is the first time you see this dialog, or if you have forgotten your password, you will need to get a new LISTSERV password first.

E-mail address: **[_____]**
Password: **[_____]**

[Login] [Change password]
[Login and save my password in a cookie]

If you already have a personal LISTSERV password, simply log in with your userid and password. Please note that the userid you use here must be associated with the personal password you have from LISTSERV. If you have registered a password as joe@unix.host.com and try to log in here as joe@host.com with that password, LISTSERV will reject your login attempt.

If you login with the "save my password in a cookie" method, LISTSERV will issue you a cookie that allows you to bypass this login screen (and incidentally to stay logged into the interface for longer than 15 minutes without having to log in again when your session expires). This option is, however, only recommended for people who have physically secure machines (for instance, on your machine at home) or who are able to otherwise keep unauthorized users from logging in, since LISTSERV cannot tell who is using the cookie. Specifically if your browser does not support separate configurations or bookmark files for different users, you should not use the cookie method in a workplace environment.

Please note that there is a known bug in Netscape prior to version 4.0 which allows you to see the userid and password typed into the text boxes if you back up to the login page using the "Back" arrow.

11.3. Setting a LISTSERV password

If you do not already have a personal LISTSERV password (set with the **PW ADD** command or via the web interface) or cannot remember your password, you need to define one now. If you choose to do this via the web interface, simply click the hyperlink and you will get the following page:

Registering your LISTSERV password

Please enter your e-mail address and the desired password, then click on the "Register password" button. If you already had a LISTSERV password, but cannot remember what it was, this procedure will automatically replace your existing password with the new one you will be entering below.

E-mail address: **[_____]**
Password: **[_____]**
Password (again): **[_____]** *(verification)*

[Register password]

When you hit the "Register password" button, you will be transferred to this page:

Confirmation e-mailed

Your password registration request has been accepted. For your protection, the password will not be activated just yet (anyone could have completed this form using your e-mail address).

To activate your password, simply follow the instructions which have been e-mailed to you at *joe@unix.host.com*. Please wait until you receive an e-mail message from LISTSERV saying *"Your new password was registered successfully"* before trying to use it with the WWW interface.

Then you simply need to "OK" the password confirmation message that was mailed to you by following the instructions in that message, and your password will be registered.

11.4. The List Management main page

Once you get logged in, you will see the main List Management page.

List management – main page

This screen allows you to manage your mailing list using LISTSERV's WWW interface. First, select the list you would like to work with:

List name: [_____]

[Subscribers][Configuration][Layout][Templates][Bulk op.][Command][Mail merge]

[Manage subscribers]
These menus allow you to add or delete subscribers, change a subscriber's e-mail address or subscription options, see whether someone is still subscribed to the list, etc. If you have a lot of subscribers to add or delete, see bulk operations.

[Edit list configuration]

This is the place to go if you want to change the configuration options for your list (also known as the *list header*).

[Customize layout]
The layout editor allows you to customize the WWW interface using a simple graphical interface. You can switch between text and graphical (icon-based) layout for the archive pages, disable functions which are not useful or not wanted for your particular list, or even translate the archive pages.

[Edit mail and web templates]
The template editor allows you to customize the administrative messages sent by LISTSERV in response to most commands (known as *mail templates*). You can also use it to exercise finer control on the layout of the WWW interface than is possible through the graphical layout editor. Note that the banners at the very top and bottom of WWW archive pages are under the LISTSERV administrator's control. You can, however, add your own top and bottom banners in addition to the site-wide ones imposed by the administrator.

[Bulk operations]
This screen allows you to add or delete large numbers of subscribers from a text file that will be downloaded through your browser.

[Command]
This screen allows you to execute an arbitrary LISTSERV command and see the results immediately, in your browser window.

[Mail merge]
If enabled by the administrator, this screen allows you to send customized mail-merge messages to your subscribers. You can choose which subscribers should receive the message (for instance, all AOL subscribers who are set to NOMAIL), and you can include customized substitutions or conditional blocks in the message (this is particularly useful when coupled to a DBMS back-end).

Enter the server administration area (LISTSERV administrator only).

List owners who have only one or just a few lists running on the server will be presented with a drop-down list box from which they can choose the list they want to work on (only their own lists will be displayed). Site maintainers or list owners who own many lists on the server will either see the drop-down list box (on servers with up to 51 lists) or a plain text box (on servers with 52 or more lists). Either type or choose the name of the list you want to work on, and click the button for the operation you want to perform.

For the sake of argument, let's say that we have a list called CAT-FANCY that we want to administer. The following sections explain what is back of the function buttons.

11.5. Maintaining subcriptions via the web

Clicking on the "Manage subscribers" button will bring up the following screen:

Account management – CAT-FANCY

Examine or delete a subscription

Name or address: **[_____]**
henry@somewhere.com
Henry Brown
s*lvia

[Search in CAT-FANCY] **[Clear]**

Add a new user to the list

Name & address: **[_____]**
henry@somewhere.com Henry Brown
Henry Brown <henry@somewhere.com>
[•] Send welcome message
[o] Do not notify the user in any way

[Add to CAT-FANCY] **[Clear]**

Back to the list management page

11.5.1. Examine or delete a subscription

This works very much like the "SCAN" command. Simply enter your criteria in the text box and click the "Search in ..." button. If there is no match for your entry, you will get back the same page as above, but with a

Scan: No match.

message at the top. If on the other hand your search *is* successful, one of two things will happen. If there are multiple matches for your criteria, the following screen will be displayed, with a scrollable list box containing all of the matches:

Account management – CAT-FANCY

Select a subscriber

["Joe User" <joe@host.com> **]**
["Sarah Brown" <sarahb@example.org>**]**

When deleting someone from the list:
[•] Notify the user by e-mail
[o] Do not notify the user in any way

[Examine] **[Delete]** **[New search]** **[Delete from all lists]**

Back to the list management page.

You now simply choose the user you want to examine or delete and click on the appropriate button. If there was only a single match to your query, the preceding screen will be bypassed and you will go directly to the next screen. If you didn't find what you were looking for, you can press the "New search" button to get a new search screen.

Account management – CAT-FANCY

View or set subscription options

joe@host.com

Notification options: [•] Notify the user by e-mail
[o] Do not notify the user in any way
[Update] [Delete] [New search] [Delete from all lists]

Name: **[**Joe User **]**
Address: **[**joe@host.com **]**
Subscribed on 7 Jan 1998

Subscription type:

[•] Regular	[NODIGEST]
[o] Digest (traditional)	[NOMIME DIGEST]
[o] Digest (MIME format)	[NOHTML MIME DIGEST]
[o] Digest (HTML format)	[HTML DIGEST]
[o] Index (traditional)	[NOHTML INDEX]
[o] Index (HTML format)	[HTML INDEX]

Mail header style:

[o] Normal LISTSERV-style header	[FULLHDR]
[•] LISTSERV-style, with list name in subject	[SUBJECTHDR]
[o] LISTSERV-style, short	[SHORTHDR]
[o] "Dual" (second header in mail body)	[DUALHDR]
[o] sendmail-style	[IETFHDR]

Acknowledgements:

[o] No acknowledgements	[NOACK NOREPRO]
[•] Short message confirming receipt	[ACK NOREPRO]
[o] Receive copy of own postings	[NOACK REPRO]

Miscellaneous:

[o] Mail delivery disabled temporarily	[NOMAIL]
[•] Address concealed from REVIEW listing	[CONCEAL]
[o] User is exempt from renewal/probing	[NORENEW]
[o] User may bypass moderation	[EDITOR]
[o] All postings sent to list owner for review	[REVIEW]
[o] User may not post to list	[NOPOST]

[Update] [Delete] [New search]

Back to the list management page.

If you are deleting someone or changing/updating their options, the two radio-button options allow you to choose whether or not the operation will be non-"quiet" (where notification is sent), or "quiet" (where no notification is sent). The two options when used with the "Delete" button are therefore strictly equivalent to "**DELETE *listname userid@host***" and "**QUIET DELETE *listname userid@host***", respectively, and the other equivalent commands are formatted identically. "Notify the user by e-mail" is the default.

You will note that this page allows you to set virtually every user option available, excepting only a couple of obsolete header style settings that are still retained in the command set for backwards compatibility. (Should a user be set to one of these obsolete

settings, it will show up as "Special or obsolete header style". Specifically this will happen if a user is set to the **FULL822** or **SHORT822** header options.)

Note also that the subscription date is displayed for the user. If there is no subscription date, then this indicates a user who has been on the list since before your server was upgraded to version 1.8c (and thus prior to the time when the subscription date was tracked by LISTSERV).

"Delete from all lists" is strictly equivalent to the command "**DELETE * *userid@host***" and is used to delete the user from all lists on the local server (for site managers) or from all lists on the local server which are owned by the invoker (for list owners).

If you are making changes to the user's name field, address, or user options, use the "Update" button to commit the changes. If you make changes to both the options and the identification fields, user option settings are updated first, and then changes are made to the name and address fields.

Following both a "Delete" and an "Update" operation, the main Account Management screen is displayed along with a message indicating the success or failure of your operations.

11.5.2. Add a new user to the list

To add a new user to the list, simply type the user's address and full name into the bottom section of the page shown at the beginning of 11.5. (The full name is optional; if omitted the user will be added anonymously to the list.) Then choose whether or not to notify the user that he has been added and click on the "Add to *listname*" button.

11.6. Maintaining the list header via the web

From the List Management Main Page (shown in 11.4, above), type in the name of the list for which you want to display and/or modify the header and click on the "Edit list configuration" button.

You then get a screen as shown on the next page. The list header appears in a multi-line text box which can be scrolled both up and down and left and right. At first glance you will appear to have a standard header, but note carefully one apparently missing element: There are no asterisks in column 1 of the header lines.

For the purpose of this interface, the asterisks denoting that header lines are, in fact, header lines, are not required. You simply type in the changes or added lines just as if you were using a regular text editor. When you are finished, you click the "Update" button to submit the changes. If you make a mistake in the editing or simply want to start over, you can click the "Reload" button to reload the header information from the server.

Edit list header – CAT-FANCY

```
+-------------------------------------------------------------------------+
|The Cat Fancier's List                                                   |
|                                                                         |
| Review= Owners                                                          |
| Send= Public                                                            |
| Notify= No                                                              |
| Reply-to= List,Respect                                                  |
| Validate= No                                                            |
| Notebook= Yes,E:\LISTS\CAT-FANCY,Weekly,Public                          |
| Subscription= Open,Confirm                                              |
| Confidential= No                                                        |
```

```
| Ack= Yes                                                                |
| Renewal= 2/1,5/1,8/1,11/1,Probe                                         |
| Digest= Yes,A,Daily,23:00,Size(1000)                                    |
| Mail-Via= Distribute                                                    |
+-------------------------------------------------------------------------+
```

[Update] [Reload]

Back to the list management page.

When you submit your changes, you will get the same kind of feedback from LISTSERV as you would if you sent a **PUT** operation by mail. The next screen will either say that the header of the list has been successfully updated, or it will indicate that it has found errors and that the header has not been stored. The feedback page also has a text box containing the header information you've just stored (or tried to store) so if you need to make further emendations to the header, you don't have to back up and start over.

11.7. Customizing how a list's pages look

By clicking the "Customize layout" button on the main list management page, you can pull up a page that allows you to customize how a list's pages look by simply answering a few questions. This is much simpler than editing the raw templates and can help avoid formatting or syntax problems that might be difficult to fix.

The page is fully self-documented.

11.8. Maintaining mail and WWW templates via the web

From the List Management Main Page (shown in 11.4, above), type in the name of the list for which you want to display and/or modify the header and click on the third button ("Edit mail and WWW templates").

You then get the following screen:

Template management – CAT-FANCY

Select a form to view or edit

[Welcome message **[V]**

[Edit form] [Switch to WWW templates]

Back to the list management page.

From this page you can either:

- edit the forms from your ***listname*.MAILTPL** file as well as your ***listname*.WELCOME** and ***listname*.FAREWELL** files (or add them if they do not already exist); or
- edit the special WWW templates.

In order to edit templates or your WELCOME or FAREWELL file, you simply choose the form you want to work with from the drop-down list box and click on the "Edit form" button. To switch to WWW templates, click on the "Switch to WWW templates" button; if

working on WWW templates and you wish to return to mail templates, click on the corresponding "Switch to mail templates" button on that page.

(Note: To customize the "look" of a list's web pages, you can alternately use the questionnaire-driven functionality described in 11.7, above.)

If you do not have a WELCOME or FAREWELL file, the response will be a page with the message "The TEST list has no WELCOME message" or "The TEST list has no FAREWELL message". You can then create the message you want by typing it into the multi-line text box (this is very similar to the list header editing interface). There is also a separate box to type the subject line into.

If you choose to edit one of the standard forms found in DEFAULT MAILTPL for your list, simply choose the template form from the drop-down list box and click "Edit form". You will get a page with the message "This form is defined in the DEFAULT template" along with the appropriate subject line and template text which you can then edit and submit. If your listname.MAILTPL already contains the template form you want to edit, the message will say "This form is defined in the *listname* template" and there will be an asterisk on the entry for the form in the drop-down list, indicating that it has been altered from the standard text.

11.9. Bulk operations via the web

Note: Bulk operations are *not* enabled by default. As the site manager, you must create a directory called "upload" under the directory specified in the **`WWW_ARCHIVE_DIR=`** site configuration variable, and give the userid under which the "wa" CGI program is run write permission in that directory. This is the ***only*** directory in which "wa" needs write authority, and only for this functionality. If you do not want the functionality, do not create the "upload" directory.

Please note carefully that your browser MUST support the RFC1867 file upload extension or you will not be able to use the bulk operations page. Most modern browsers do support this extension, including but not limited to Netscape 3.x and later (although Netscape versions for the Macintosh through at least 4.5 do *not* appear to implement RFC1867 properly), and Internet Explorer 4.x and later.

From the List Management Main Page (shown in 11.4, above), type in the name of the list for which you want to display and/or modify the header and click on the last button ("Bulk Operations"). You will get the following page:

Bulk operations – CAT-FANCY

Caution: some of the functions offered through this page will ***remove all subscribers*** from CAT-FANCY. Double-check your selection before submitting!

Input file: [_____] **[Browse...]**

Function:
- [•] **Add** the imported addresses to CAT-FANCY; do not remove any subscribers
- [o] **Remove all subscribers** from CAT-FANCY, then **add** the imported addresses (to remove all subscribers, select this option and omit the input file)
- [o] **Remove** the imported addresses from CAT-FANCY; do not add any subscribers
- [o] **Remove** the imported addresses from **all lists**

[Import]

Note:

- The input file must be a *plain text* file (not a word processor document or spreadsheet) and must contain one address per line, optionally followed with a space (or TAB) and the subscriber's name.
- The subscribers being added or deleted will not be notified.
- These functions require a browser supporting the "file upload" extension (RFC1867). Current versions of Netscape and Internet Explorer both support this operation, but you should try other browsers carefully on a test list.

Back to the list management page.

(If you get an error 2 when you click on the "Import" button, this means that the "upload" directory has not been created. If you get an error 13 when you click on the "Import" button, this means that the "upload" directory has been created but the CGI program user does not have write permission in that directory.)

The input file is created on your own machine with an ASCII text editor (as noted in the instructions on the page). If you have the "upload" directory correctly configured, the next page you will see after clicking the "Import" button will have a command response like the following:

If the first radio button is set (Add but do not delete any existing subscribers):

ADD: no error, 202 recipients added, no entry changed, no duplicate, none forwarded.

If the second radio button is set (Delete *@*, then add from the file if specified):

DELETE: 14 subscribers removed.

ADD: no error, 38 recipients added, no entry changed, no duplicate, none forwarded.

(If the second button is set and no input file is specified, you will only get the DELETE: message.)

If the third radio button is set (Delete the users in the uploaded file):

DELETE: 93 subscribers removed.

If the fourth radio button is set (Delete the users in the uploaded file from all local lists to which they are subscribed):

DELETE: 243 subscribers removed.

DELETE: 109 subscribers removed.

Global deletion process complete, 352 entries removed.

If you do not supply an upload file where required, or if your browser does not support the RFC1867 file upload extension, you get the following message:

Your browser did not upload any file during the transfer. Assuming you did fill in the file input box, the most likely cause is that your browser does not support the file upload extension (RFC1867).

11.10. Sending interactive commands via the web

Clicking on the "Command" button brings up a simple, single-line interface where LISTSERV commands may be typed in for interactive execution (similar to the **LCMD** command line utility).

11.11. Mail merge

Starting with LISTSERV 1.8d, advanced mail-merge features are available and can be accessed either by sending specially-formatted DISTRIBUTE jobs to LISTSERV or by using the web administration interface. The web interface is not a "wizard" but simply an interface that allows you to "cut and paste" a mail merge message and select different standardized groups of list subscribers to whom the message is to be sent.

Documented Restriction: Note that LISTSERV's mail merge functionality REQUIRES the use of LSMTP Classic as the outgoing MTA. Mail merge does not work with sendmail, qmail, Post.Office, Netscape Mail Server, Microsoft Exchange, PMDF, MX, or any other MTA except L-Soft's LSMTP Classic mailer.

Under unixes not supported by LSMTP Classic this may require that you set **SMTP_FORWARD=** accordingly in **go.user**, to point to a separate machine running LSMTP (for instance, a dedicated Windows NT LSMTP machine). Under OpenVMS or Windows NT can run LSMTP Classic either on the same machine (the preferred method), or on a separate machine if desired. The main point is that the outgoing mail-merge postings MUST be handled by LSMTP Classic. (LSMTP Lite does not support mail-merge.)

Mail merge functions are documented fully in the *Developer's Guide for LISTSERV*, available separately.

11.12. Server administration interface

LISTSERV 1.8d also includes a maintainer-level server administration interface. From this interface it is possible to

- Create lists
- Enter the list management (list owner) area
- Customize the default LISTSERV home page
- Customize the default web interface layout
- Customize the site-wide dynamic web templates
- Customize the site-wide static web pages and banners

- Send a mail-merge message using the DBMS back-end (requires a DBMS interface; see the section on DBMS features in the *Developer's Guide for LISTSERV* for more information)
- Execute an arbitrary LISTSERV command (as with LCMD). Note that certain commands will be answered by mail rather than through the interface.

To access the server administation interface, use the appropriate link from the LISTSERV home page described in 11.1. Alternately, the direct link requires that you construct a URL as shown in 11.2, but substitute "**`?ADMIN`**" for "**`?LMGT1`**".

Each feature of the administration interface is fully self-documented.

Note one caveat: When creating lists via the web interface under unix and VMS (with PMDF), it is still necessary to make the mail aliases required in 7.2.1 (for unix) or 7.2.2 (for VMS), above. The web interface will not make these aliases for you.

12. Distribution Features and Functions

For more information on these features, see also the *List Header Keyword Reference* in Appendix B of this manual.

12.1. Controlling the default level of acknowledgement to user postings

You can control the default level of acknowledgement sent back to users when they post to the list with the "Ack=" list header keyword (see Appendix B for details). This is particularly important because it also controls the acknowledgement level for users who are not subscribed to the list and cannot, therefore, set personal options. While the value set for "Ack=" can be overridden for subscribers both by the setting of the "Default-Options=" keyword (which sets the default at subscribe time) and by the "SET" command, this option will always be in effect when distributing mail from people who are not subscribed to the distribution list.

12.2. Controlling the maximum number of postings per day

12.2.1. Controlling total postings to the list per day

You can control the maximum number of postings per day on a list-by-list basis by setting the "Daily-Threshold=" list header keyword. The default value is 50 posts per day.

Note the following:

- If the Daily-Threshold value is not reached by midnight, the number of postings is reset to zero and starts over for the next day.

- If the Daily-Threshold is reached before midnight, and the list is not freed before midnight, postings released after midnight count toward the next day's quota. Thus, if the list is held on Tuesday evening and 15 postings accumulate before you free the list on Wednesday morning, the 15 postings count toward Wednesday's quota.

12.2.2. Controlling the number of postings per day from individual users

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

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

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

12.3. Controlling "prime" time

This feature reserves certain times and days of the week when you don't want LISTSERV to process postings. Although this is not usually necessary today, there can be certain applications for the use of the "PRIMETIME=" site configuration variable and the "Prime=" list header keyword.

"PRIMETIME=" controls the server-wide prime time setting. By default it is set to **MON-**

SUN: -. There should be no need to change this setting under normal circumstances. Please see the entry for PRIMETIME in Appendix C for further details.

The list header keyword "Prime=" controls prime time on a list by list basis. By default it is set to "Prime= Yes", meaning that it does not observe the PRIMETIME= variable. If explicitly set to "Prime= No", the value in the PRIMETIME= variable will be observed. It can be set to an explicit time definition if necessary. For instance, you might have a very large announce-only list (e.g., a newsletter) that should not be posted until after midnight (when network traffic is low and more machine resources are generally available). You might wish to set this list with a "Prime=" setting of

```
* Prime= "MON-SUN: 06:00-23:59"
```

or, if you want the list only to be processed between midnight and 6 AM on weekends, you might code

```
* Prime= "MON-FRI: 00:00-23:59; SAT-SUN: 06:00-23:59"
```

Note that the specification for "Prime=" must be enclosed in *double* quotes. Note also that the minutes specification is cosmetic only. LISTSERV checks on jobs held awaiting non-prime time only once each hour, on the hour. Thus if you have

```
* Prime= "MON-SUN: 06:00-21:00"
```

then jobs awaiting non-prime time will be executed at 22:00, not 21:00 as you might otherwise expect. On the other hand, if you code

```
* Prime= "MON-SUN: 06:00-20:xx"
```

where "***xx***" is any two-digit integer between 01 and 59, then jobs awaiting non-prime time will be executed when LISTSERV runs its hourly check of PRIME jobs at 21:00.

If you need to open only one short window during one or more days, you can do this by coding something like:

```
* Prime= "MON-FRI: 00:00-02:59 04:00-23:59; SAT-SUN: -"
```

This example allows LISTSERV to process mail for the list only between 2 AM and 4 AM Monday through Friday, and at any time on Saturday and Sunday. Note that there is no punctuation--just a space--between the time settings for a given day or day sequence.

Mail sent to lists during prime time is automatically held until non-prime time and then distributed normally, without requiring further intervention by anyone. This means that the newsletter editor of the example list can post their next issue on Friday afternoon and know that it won't be distributed until Saturday at midnight or shortly thereafter.

One of the most common misconceptions regarding the prime time settings is that prime time is the time during which LISTSERV will process postings for your list (or globally for the server if you change "PRIMETIME="). Please remember that when you set a "prime time" either for a list or globally for the entire server, you are setting the time during which LISTSERV *does not* process postings. It is "prime time" for the machine when it should be doing other things, for example, number crunching, daily backups, or any other function during which LISTSERV should not be using cycles.

Note also that when you are coding a prime time specification that LISTSERV's week starts on Monday and runs through Sunday. Thus something like the following examples:

```
Prime= "MON-TUE: 00:00-23:59; WED: -; THU-SUN: 00:00-23:59"

Prime= "TUE: 01:00-4:59; THU-SUN: 00:00-23:59"
```

are correct syntax, whereas

```
Prime= "WED: -; THU-SUN: 00:00-23:59; MON-TUE: 00:00-23:59"
```

is not. Furthermore note carefully the weekdays must be specified in their correct order, that is,

```
Prime= "THU-FRI: 00:00-23:59; SAT-MON: 21:00-23:59"
```

is not correct because it starts on Thursday and ends on Monday The correct specification in this case would be

```
Prime= "MON: 21:00-23:59; THU-FRI: 00:00-23:59; SAT-SUN: 21:00-23:59"
```

12.4. "Holding" and "freeing" a list

12.4.1. Automatic list holds

On occasion, LISTSERV will automatically "hold" a list, i.e., postpone processing new mail for the list until either the list owner or the LISTSERV maintainer manually intervenes. There are two circumstances under which a list will be automatically held:

- When the daily threshold of postings has been reached (see above and in Appendix B under "Daily-Threshold="). Note that a mailing loop may cause this to happen.

- When an error occurs resulting in LISTSERV being unable to process new postings for the list. LISTSERV always sends a traceback of the error to the list owner along with the notification that the list has been held.

In both cases, the list can be freed by either the list owner or a LISTSERV maintainer sending the command

FREE *listname* PW=*yourpassword*

to LISTSERV. However, note that for any hold, it may be wisest to wait until the LISTSERV maintainer has had a chance to check the server for anything untoward that might be causing the error (e.g., a mailing loop) before freeing the list.

Note that an error indicating that the current notebook archive LOG file is open and locked by another process may actually indicate that the archive file is in a directory accessible by anonymous FTP and that someone is in the process of FTPing the current notebook archive, making it impossible for LISTSERV to append the latest posting to the current notebook. This error generally occurs only on systems with FTPable notebook archives; by the time you receive the error, it is usually safe to issue the FREE command to release the list. If, on the other hand, this error persists, you may want to check the server for "zombie" processes that may have the file locked, or set the location parameter of the "Notebook=" keyword to a non-FTPable directory.

12.4.2. Manual list holds

If you need to stop LISTSERV from processing new mail for a list for any reason, simply issue the command

HOLD *listname* PW=*yourpassword*

Then, to free the list, issue the FREE command as noted above.

12.5. Controlling the list digest feature

List "digests" are provided for those users who prefer to receive (typically) one large, comprehensive posting per digest period that includes all of the list traffic from that period, rather than receiving each post individually as processed by the server.

If "Notebook= Yes...", the digest feature defaults to daily digests cut at midnight with the "work" files kept in the same directory as the list's notebook archives. If "Notebook= No", digests are not enabled by default. However, note that lists without notebook archives *can* have digests; it is simply necessary to enable digests manually for such lists by using the "Digest=" list header keyword and specifying a valid path for the location of the digest's work files.

Digests can be set up to cut on a Daily, Weekly, or Monthly basis, and can be further configured to cut after a certain number of lines of data have been stored regardless of the digest period setting. This ability to generate "special issues" when the digest reaches a certain size may be necessary if people complain that your 10,000 line daily digest is getting truncated by their mail host to 1500 (or even 1000) lines.

For example, if a high volume list is set for Daily digests, and the "Size(nnn)" parameter of the "Digest=" keyword for the list is set to "Size(1000)", a "special issue" of the digest will be cut and mailed to digest subscribers whenever the ***listname*.DIGEST** file reaches 1000 lines of text. Note that LISTSERV will not cut the digest at exactly 1000 lines, thereby truncating the last message; LISTSERV will cut the digest after the end of the message that causes the digest file to go over its limit. Thus, if the digest file is 950 lines long and a 200 line message is received, the "special issue" digest will be 1150 lines long.

12.6. Setting up 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 and 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 Chapter 6 of the *List Owner's Manual* and Appendix B for more information on setting up and using list topics.

12.7. Allowing/Blocking MIME Attachments

LISTSERV 1.8d builds starting in July 2000 (the 1.8e 2000b "level set") introduced a new MIME attachment-filtering feature which is configured by setting an **Attachments=** list header keyword. The keyword as first introduced allowed three distinct modes:

- Allow all MIME attachments, no filtering or blocking
- Reject MIME attachments with notice to the poster
- Filter MIME attachments out of messages transparently

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

LISTSERV 1.8e further enhances the **Attachments=** keyword by adding a filter for non-MIME, inline uuencoded files such as are sent by mail clients like Microsoft Outlook. The uuencode filter is strictly on/off; no attempt is made to determine the file type of such inline "attachments".

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

13. Error Handling Features and Functions

13.1. Defining list-level error handling addresses

Every LISTSERV mailing list requires that an e-mail address be defined to which all mail delivery errors are sent for disposition. The error handling address is defined by using the "Errors-To=" list header keyword.

The value for "Errors-To=" can be one of two things:

- An appropriate *access-level*, such as **`OWNER`**
- A specific *internet-address*, such as **`someuser@somehost.com`**

It is strongly recommended that "Errors-To=" point to a real person's mailbox, rather than to an alias that simply dumps errors into something like **`/dev/null`**. Mail delivery errors generally indicate that a problem exists, and it's always possible to filter out the ones that don't via **`procmail`** or by using the filtering features available in most POP mail clients. After a long enough period of time, unhandled errors can grow to a significant percentage of your server's traffic and seriously impact your production.

In LISTSERV 1.8e and following, the internet address of the list is explicitly disallowed as an error-receiving address, and attempting to set Errors-To= to the internet address of the list will raise an error. The list should never be configuired to receive its own errors as this is guaranteed to cause looping.

If not defined in the list header, "Errors-To=" defaults to "Errors-To= Owners".

It should be carefully noted that there is no way to automatically discard errors without sending them to some address. "Errors-To= No" and "Errors-To= None" are both invalid settings and will cause LISTSERV to revert to the default.

13.2. The auto-deletion feature

LISTSERV includes a powerful auto-deletion filter that can be configured in several modes, depending on the level of error handling desired by the list owner. It should be noted, however, that at the present time there is no Internet standard for delivery error messages (although RFC1893 is gaining acceptance as just such a standard). Currently, LISTSERV understands and can take action on errors generated by LMail, LSMTP, Sendmail (version 8.7.x or higher), Innosoft's PMDF (version 4.2 or higher), and MadGoat's MX (version 3.2 or higher), as well as other mailers using the "Notary" format described in RFC1893. As more and more mailers conform to RFC1893, LISTSERV naturally will be capable of handling more and more errors intelligently.

To set up auto-deletion defaults for a list, use the syntax described in Appendix B for the "Auto-Delete=" list header keyword.

A sample error monitoring report, generated daily and sent to the list's "Errors-To=" address if "Auto-Delete=" is activated, is shown below:

```
Date:         Thu, 30 Oct 1998 00:00:48 -0400
From: "L-Soft list server at PEACH.EASE.LSOFT.COM (1.8d)"
              <LISTSERV@PEACH.EASE.LSOFT.COM>
Subject:      EXCEL-G: Daily error monitoring report
To: EXCGERR@LINUS.DC.LSOFT.COM
```

```
The following 3 subscribers were deleted from the EXCEL-G list today:

  sluggo@OMNI.VOICENET.COM
  Last error was: Mailer quasar.voicenet.com said: "550
                  <sluggo@OMNI.VOICENET.COM>... User unknown"

  crenaud@BOSS1.BOSSNT.COM
  Last error was: Mailer BOSS1.BOSSNT.COM said: "550
                  <crenaud@BOSS1.BOSSNT.COM>... User unknown"

  Roger Giellis <rogerg@SUPSUN4.DEN.MMC.COM>
  Last error was: Domain "SUPSUN4.DEN.MMC.COM" doesn't exist.

The following 5 subscribers are currently being monitored:

Err First Last  Address
--- ----- ----- -------
  2 05/28 05/29 "Ronald D. Stepp" <MORITURI@INTELLISYS.NET>
                Last error: Mailer INTELLISYS.NET said: "550
                            <morituri@INTELLISYS.NET>... User unknown"

  1 05/29 05/29 TEST@TEST.POWERNET.CO.UK
                Last error: Domain "TEST.POWERNET.CO.UK" doesn't exist.

  1 05/29 05/29 SIMONC@VOL.NET
                Last error: Mailer h02.VOL.NET said: "550 <simonc@VOL.NET>...
                            User unknown"

  1 05/29 05/29 REG@NROBBO.CO.UK
                Last error: Domain "NROBBO.CO.UK" doesn't exist.

  2 05/29 05/29 jmanring@INETGW.LEGGMASON.COM
                Last error: Unavailable; notary status was 5.1.2

Err=   Number of delivery errors received thus far
First= Date first delivery error was received (mm/dd)
Last=  Date of most current delivery error (mm/dd)

Subscribers will be automatically deleted from the list when delivery errors
have been  reported for  a period  of 2 days  or more,  or when  10 delivery
errors have  been received,  whichever occurs  first. Monitoring  will cease
after 3 days without any reported error.

Note: manually deleted subscribers may remain on the monitoring report under
an alias address. Such entries will expire eventually; you do not need to do
anything about them.
```

Figure 13.1. A typical daily error monitoring report.

13.3. LISTSERV's loop detection feature

LISTSERV has an extremely advanced loop detection heuristic that practically eliminates the chances of a mailing loop being propagated through one of its mailing lists. In general, L-Soft does not recommend that any loop-checking element be disabled, as any one missing element might let a loop through, but under certain controlled circumstances it might be needful to do this. See Appendix B under "Loopcheck=" for specific keyword options to turn off individual parts of the loop-checking feature.

13.3.1. The anti-spamming filter

LISTSERV's anti-spamming filter is built into the loop-checking heuristic. Depending on your local circumstances, it may be desirable to disable the anti-spamming filter for certain lists, particularly if the lists are confidential (and thus not visible in the global list of lists, making it extremely unlikely that a spammer would target them), if the lists are set to reject or transfer to the list moderator postings from non-subscribers (e.g., "Send= Private" or "Send= Editor"), or if your LISTSERV server is not accessible from the

Internet (e.g., you're running it on an internal LAN without Internet connectivity).

If you need to turn the anti-spamming filter off for a particular list, code:

```
* Loopcheck= NoSpam
```

One (tunable) aspect of the anti-spamming filter involves holding mail from non-subscribers for a pre-determined length of time (the default is 10 minutes) to see if further mail arrives from the same user for other lists that may trigger the anti-spamming filter. If you want anti-spamming protection, but want mail from non-subscribers to go directly to a list without being held up for this check, you can code

```
* Loopcheck= Spam-Delay(0)
```

If you want the check to be performed, but think 10 minutes is too long to hold the messages, you can change the value for "Spam-Delay()" to the preferred number of minutes. For instance, to hold non-subscriber messages for five minutes, code

```
* Loopcheck= Spam-Delay(5)
```

Note also that you can configure the server-wide default for this "spam quarantine" feature by adding the new **SPAM_DELAY** variable to your site configuration file and specifying the number of minutes. For instance, setting the variable to a value of 15 would set the server-wide spam quarantine default to 15 minutes, while setting it to zero disables the feature.

You can avoid the problem of a legitimate user being identified as a spammer when cross-posting to multiple lists on your server by setting up a super-list that has as its sub-lists the lists to which the user needs to cross-post. This is the only supported method for cross-posting to multiple lists if the anti-spam feature is not disabled for all the lists in question.

Please note carefully that there is no command or method to lift the 48-hour anti-spam quarantine for addresses that are identified as spammers and blocked from posting.

The rest of the anti-spamming filter algorithm is proprietary and non-configurable.

13.4. RFC822 mail header parsing

LISTSERV is designed to be 100% compatible with the Internet RFCs that govern how mail headers may be formatted (RFC822 et seq.), and includes a powerful RFC822 parser for this purpose. However, not all individual mail clients are compliant with RFC822, either because of poor design or because of mis-configuration. Because of this, LISTSERV postmasters may from time to time see bounces such as the example below:

```
Date: Tue, 5 Aug 1997 15:43:57 -0400
From: "L-Soft list server at Apple (1.8c)" <LISTSERV@MAIL.EWORLD.COM>
To: Nathan Brindle <nathan@OSF1.DC.EXAMPLE.COM>, ERIC@VM.SE.EXAMPLE.COM
Subject: Problem processing mail file from MAILER@MAIL.EWORLD.COM

An error occurred while  processing file 1449650 from MAILER@MAIL.EWORLD.COM:
"Mail message sent to the LISTSERV address contains invalid RFC822 fields and
could not be parsed successfully".

RFC822 parser messages follow:

<W> Incorrect or incomplete address field found and ignored.
```

```
<W> Incorrect or incomplete address field found and ignored.

<W> Incorrect or incomplete address field found and ignored.

<E> Mail origin cannot be determined.

<E> Original tag data was -> "myuserid" <>

------------------- Message causing the problem (29 lines) --------------------
Received: from dfw-ix16.ix.netcom.com by home.ease.lsoft.com (LSMTP for Windows
NT v1.1a) with SMTP id <0.628849B0@home.ease.lsoft.com>; Tue, 5 Aug 1997
15:43:56 -0400
Received: (from smap@localhost)
          by dfw-ix16.ix.netcom.com (8.8.4/8.8.4)
          id OAA06085 for <listserv@mail.eworld.com>; Tue, 5 Aug 1997 14:45:04
-0500 (CDT)
Message-Id: <199708051945.OAA06085@dfw-ix16.ix.netcom.com>
Received: from por-or10-22.ix.netcom.com(204.31.113.150) by
dfw-ix16.ix.netcom.com via smap (V1.3)
        id smaa06017; Tue Aug  5 14:44:47 1997
From: "myuserid" <>
To: <listserv@mail.eworld.com>
Date: Tue, 5 Aug 1997 12:47:20 -0600
...
```

Figure 13.2. Sample RFC822 parser error.

The lines starting with "<W>" are warnings from the parser that indicate a non-fatal problem with one or more of the message's RFC822 headers. In this case the warnings apply to the second "Received:" header of the message, which is misformatted (it includes end-of-line characters and thus the parser treats the header as three separate RFC822 headers and attempts to parse them individually). If these were the only warnings, the message would still be accepted by LISTSERV.

However, there is a second problem in the message, and this one is fatal. The RFC822 "From:" header has a null value for the user's "userid@host" address. It is most likely that this user has decided to remove his address from his POP client's configuration in order to avoid being placed on spammers' mailing lists. However, this is not legal per RFC822, and when LISTSERV tries to determine the origin of the mail message, it can't be done. This is because the value for "From:" is invalid and there is no other header (such as "Sender:") that might be able to indicate where the message is coming from. Therefore LISTSERV writes two "<E>" (error) lines, one that says the mail origin can't be determined and a second to specify what the data was in the one origin header it could find, and bounces the message to the LISTSERV maintainer(s) for further disposition.

Another error you might see is

```
<W> MESSAGE-ID field duplicated. Last occurrence was retained.
```

13.5. Address Probing

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

13.5.1. Active address probing

This functionality is not available in LISTSERV Lite.

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

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

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

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

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

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

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

13.5.2. Passive address probing

This functionality is not available in LISTSERV Lite.

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

probe addresses that will not be used a second time.

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

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

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

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

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

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

If a given list only has activity once in a while (e.g., a large weekly newsletter), passive probing works like this: If you have **Probe(*p*)** set in your **Auto-Delete=** keyword (where *p* is some integer value), and you have *n* subscribers, about (*n* / *p*) will receive a probe during the mailing. Normally you would want to probe 2-10% of your subscribers in this kind of scenario, so *p* would range from 10 to 50.

Please note carefully that LISTSERV ignores Probe(0) in list-based mail-merge jobs. Mail-merge messages are always sent as probes, and in list-based mail-merge there is no attempt made to parse the header of the list that is being used as a datastore for the mail-merge job.

13.5.3. OS-specific issues with probing

Probing is supported automatically by the VM and Windows versions of LISTSERV without need for any special configuration other than noted above.

On unix systems, while LISTSERV itself does support probes, probes are not supported natively by most (if not all) unix MTAs, including sendmail, etc. In order to use probing on such systems the MTA must be patched to divert incoming probe bounces to the lsv_amin mailer for delivery to LISTSERV. Otherwise incoming probes simply bounce since there is no way for the MTA to determine what to do with them.

User-contributed patches to support probing under sendmail are available on L-Soft's ftp site but are not supported by L-Soft--use at your own risk!

On VMS systems, probing is supported natively if you are using LSMTP 1.1a or later, or any version of MX that includes support for the LISTSERV interface. PMDF® users should create a dedicated domain for LISTSERV (eg, LISTSERV.XYZ.COM) and add a rewrite rule to redirect all traffic for that host to the LSV channel. This also simplifies the creation of new lists as with this setup, it is no longer necessary to define PMDF aliases for the lists.

13.6. Defining server-level error handling addresses

13.6.1. BOUNCES_TO=

Starting with LISTSERV 1.8d it is possible to divert some of the server-level error traffic (that is to say, error traffic not specific to a given list, or errors that should never have been sent to LISTSERV to begin with) to a specified place other than the default (the non-quiet LISTSERV maintainers). This is done by adding the **BOUNCES_TO=** variable to your site configuration file and restarting LISTSERV.

The following specific types of traffic are routed to the **BOUNCES_TO=** address: spam alerts, spoofing alerts, quota errors (for sites running with the SCOPE=ISP option), and DISTRIBUTE error messages.

BOUNCES_TO= can point to one or more users (use the same syntax as for **POSTMASTER=**) or to a mailing list set up for the purpose.

13.6.2. Crash reports and CRASH_MONITOR=

Following a severe system crash, LISTSERV 1.8c and later under VMS and Windows NT will generate a "crash report" containing:

- System-specific information about the immediate cause of the crash (access violation, division by zero, etc).
- A traceback showing what LISTSERV was doing at the time of the crash.
- The last 100 lines in the LISTSERV log.

By default, this report is mailed to the LISTSERV maintainer. You can change the destination of these reports by adding a **CRASH_MONITOR** variable to your configuration file (**SITE.CFG** for NT, **SITE_CONFIG.DAT** for VMS) with the e-mail addresses to which the report should be mailed. Note that **CRASH_MONITOR** replaces the entire recipient list, so make sure that all the necessary addresses are listed. This configuration variable follows the same syntax rules as **POSTMASTER**. Please do not add L-Soft mailboxes to **CRASH_MONITOR** without checking with our support group first. While we will be happy to receive these reports, we want to make sure that they are sent to the addresses where we can process them most efficiently. In particular, these reports should never be mailed to a support engineer's personal mailbox. Instead, we use special addresses where these reports are logged for future reference.

IMPORTANT: Crash reports may contain company confidential information! Before forwarding a crash report to L-Soft, make sure that it does not contain any confidential information. L-Soft will not sign non-disclosure agreements related to crash reports. If you include an L-Soft address in your **CRASH_MONITOR** configuration variable, you are implicitly stating that none of the activity taking place on your server is confidential.

When reporting a crash to L-Soft, please forward a copy of the crash report with any confidential information removed or XXX-ed out. Note that the crash report is not actually mailed until LISTSERV is restarted. Crashes are usually caused by conditions which prevent LISTSERV from operating normally; furthermore, image termination may be necessary to cause the operating system to generate the traceback included in the report.

(NT) IMPORTANT: In order for the crash report to be useful, the files **LSV.EXE** and **LSV.SYM** must be updated at the same time. This is done automatically if you install/update LISTSERV using the graphical installation procedure, however if you install patches manually you must ensure that both **LSV.EXE** and **LSV.SYM** are updated.

This feature was not provided for VM because all the information that is available is already gathered at the bottom of the console log, which is normally spooled to a maintenance userid and not accessible to LISTSERV.

Under unix, there is no portable way for an exception handler to obtain a call traceback; a system-specific debugger (which must be installed separately and often requires a separate license) must be run on the core file, which is not available until the process has aborted. LISTSERV does flush buffered log output to ensure that the **listserv.log** file contains all the relevant log information following a core dump.

14. List Maintenance and Moderation Features and Functions

14.1. Setting up edited/moderated mailing lists

As noted above in Chapter 7.13, you need only add the following lines to the list header file:

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

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

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

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

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

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

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

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

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

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

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

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

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

"compare/consolidate" feature may be removed in a future upgrade.

For more information on setting up edited lists, see "Send=", "Editor=" and "Moderator=" in Appendix B, as well as Chapter 7.13.2 of this manual where setting up edited lists is discussed further.

14.2. Restricting the size of messages posted to the list

Using the "Sizelim=" list header keyword, you can restrict the size (in lines) of messages posted to a given list. This may be particularly desirable for lists discussing programming topics where the posting of uuencoded binaries to the list is discouraged, or simply to encourage economy in posting. In any case, if this feature is desired, simply add the keyword

```
* Sizelim= nnn
```

to the list header, where "nnn" is the maximum number of lines (including Internet delivery and addressing headers) to be accepted. Note that unlike the "Size()" parameter of the "Digest=" keyword, LISTSERV will not allow a post to go over the "Sizelim=" setting, but will reject it if it is even a single line over the allowable threshold. When a posting is rejected for size, the original poster receives a notification that his post was too large.

The Sizelim= list header keyword has been enhanced in LISTSERV 1.8e to allow list owners to specify a maximum message size in either kilobytes or megabytes, rather than in lines, if preferred. For instance:

`Sizelim= 100K`	Reject messages over 100Kbytes
`Sizelim= 1M`	Reject messages over 1Mbyte

As before, the limit operates against the entire message file, including all Internet header lines.

Note that some misconfigured mail hosts will try to bounce delivery errors, complete with the text of the message that bounced, back to the list address rather than to the RFC821 MAIL FROM: address. Setting "Sizelim=" to a reasonable level (say, 400 lines, or 25-30 kilobytes) will usually prevent a mail host from bouncing a whole digest back to the list.

14.3. Restricting the number of posts per user per day

You can restrict the number of posts to the list per user per day. This is done with a new second parameter for the "Daily-Threshold=" list header keyword. For instance, setting "Daily-Threshold= 100,5" would tell LISTSERV to hold the list after 100 postings (as in earlier versions), and additionally to stop accepting new postings from any individual subscriber after that subscriber had posted 5 messages during the 24h period from midnight to midnight (server time). After reaching the user threshold, the subscriber simply receives a message to the effect that he has reached the daily limit and that he should try to repost the message later (i.e., after midnight). Please see the entry for "Daily-Threshold=" in Appendix B for further information.

14.4. Moving a list to a new location: the New-List= keyword

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. BITNET sites running 1.8b and later which are moving from VM to workstation versions of LISTSERV should probably use "New-List=" during their migration period.

Note that this only works for a move from one L-Soft LISTSERV server to another, not for a move from a LISTSERV server to a server running another mailing list manager.

See "New-List=" in Appendix B for more information about this keyword and how to use it.

15. Security Features and Functions

LISTSERV's security options are wide ranging, from almost no protection (easiest to administer a list, but also most open to hacker attacks) to total protection requiring validation of each and every command sent to LISTSERV for the list. It is also possible to limit access to various aspects of the list, such as who can subscribe, who can review the list of subscribers, and who can access the list archives. The list can be hidden 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.

Please note carefully that LISTSERV does not set any file system permissions for any files. LISTSERV's security features are meant to allow and deny access to LISTSERV's various files depending on how various keywords are set, but in order to make your system completely safe, you must ensure that you have also set file and directory permissions appropriately for your operating system. For instance, LISTSERV may deny GET access to certain list archive files to non-subscribers, but if you have not set the appropriate file system permissions at the operating system level, you may have left open a window for someone to reach these files via anonymous ftp.

15.1. First line of defense: The VALIDATE= keyword

The **VALIDATE=** keyword controls the level of command validation desired for the list. The default, **VALIDATE= NO**, requires password validation only for storing the list on the server. This is often sufficient for general needs. However, when a list is set this way, LISTSERV only compares the RFC822 "Sender:"/"From:" headers against the **Owner=** keyword(s) in the list header to determine whether or not the person ostensibly sending the commands has authority to do so. Otherwise at this level LISTSERV does not validate commands it receives for the list, under the assumption that the mail it receives is genuinely coming from a list owner. This level of validation does not protect the list from commands issued by hackers who have forged mail in the name of the list owner. If you run a list on a controversial topic or just don't feel comfortable without at least some security, **VALIDATE= NO** is probably not for you.

The next level is **VALIDATE= YES**. At this level, LISTSERV requires a password for all of its "protected" commands. This password is the sender's personal LISTSERV password as defined by the **PW ADD** command. The commands protected by this level are those that affect subscriptions or the operation of the list, for example, **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** and other strictly-informational commands) 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.

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

See Appendix B for complete information on the **VALIDATE=** keyword.

15.2. Controlling subscription requests

Subscription requests are controlled 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. Subscription requests can be refused completely by setting **SUBSCRIPTION= CLOSED**.

To code a list for open subscriptions without list owner intervention, set **SUBSCRIPTION= OPEN**. If it is desired 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.

15.3. Controlling the service area of the list

The Service= keyword is not available in LISTSERV Lite.

It may be desirable to restrict access to a 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 **LISTS GLOBAL** command.

Please note that in LISTSERV 1.8c and following, the local list of (public) lists can be retrieved only by those users who are considered local, per the setting of the server-wide **LOCAL=** variable in LISTSERV's site configuration file. All other users will be told that none of the lists on the server are visible via the LISTS command, and will be referred to the use of the **LISTS GLOBAL *search-text*** command or to the CataList. This is regardless of the setting of **Confidential=** as outlined below.

To simply hide a list from a **LISTS** 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 **LISTS** command. Note also that all other non-subscribers, including users on the local machine, will *not* be able to determine that the list exists via a **LISTS** command.

To hide a 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=**. The LISTSERV maintainer should define hosts and nodes that are considered local with the server-wide **LOCAL=** variable in the site configuration file. If the global definition is not suitable, it can be overriden by defining the **Local=** list header keyword:

```
* LOCAL= bitnode1,bitnode2,some.host.edu,another.host.com
* SERVICE= LOCAL
* CONFIDENTIAL= SERVICE
```

If there are many subdomains within your primary domain, it may be preferable to use the wildcard when defining the **LOCAL** or **SERVICE** list header keywords. For instance:

```
* SERVICE= *.HOST.COM
```

defines the service area for a specific list as "all subdomains ending in .HOST.COM".

Note that defining a service area for a list controls only from which domains subscription requests may be accepted. *It does not control who may post to the list.* Depending on local circumstances, it may be desirable to set lists with controlled service areas to Confidential= Service.

15.4. Controlling who may review the list of subscribers

For whatever reason, it may be desirable 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 was the default prior to LISTSERV 1.8c).

To restrict reviews of the list to subscribers only, set **REVIEW= PRIVATE**. This is the default from LISTSERV 1.8c on.

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

Reviews can also be restricted to users within the list's service area by setting **REVIEW= SERVICE** , and defining the **SERVICE=** keyword appropriately (see the preceding section).

Please note that unless the list is set to "Confidential= Yes" or "Confidential= Service", a request to REVIEW a list by someone who is not allowed to do so will result in the header of the list being sent to the user along with a note to the effect that the REVIEW command is restricted for this list. See the section below regarding hiding header lines if you want to hide parts of the header but do not want to use the "Confidential=" keyword.

15.5. 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.)

Starting with LISTSERV 1.8d it is possible to define "Service=" in terms of IP address blocks in order to limit access to list archive notebooks. See "Service=" in Appendix B for details.

If enabled, notebook archives are private by default.

15.6. 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**. (This is the default.)

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**:

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

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

```
processing large numbers of messages.

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

Figure 15.1. 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 ***(or 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.

See also the **Default-Options=** list header keyword description.

15.7. 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 (an eight-digit hexidecimal number) is registered to the "From:" user id of the original command.

A typical command confirmation request looks like this:

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

Your command:

                            PW REP XXXXXXXX

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

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

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

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

Figure 15.2. A typical command confirmation request.

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

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

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

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

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

- **SEND** a new message to LISTSERV with the text "ok xxxxxxxx" (where xxxxxxxx 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. (See also the "bracketed OK" syntax mentioned below.)

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

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

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

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

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

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

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

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

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

  OK BEGIN
  command1
  command2
  ...
  command3
  OK END

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

Documented restriction*:* In a "bracketed OK" the aggregate length of the data stream (that is, the total number of characters in the command lines falling between **OK BEGIN** and **OK END**) MUST be less than 32K characters. In practice you should use bracketed OKs for limited numbers of commands only, say no more than 10-12 at a time. In particular, if you have many ADD or DELETE commands to send, it is far more efficient (and strongly preferred) to use the bulk ADD and bulk DELETE syntaxes described in chapter 7.17, above.

15.7.1. Explicitly cancelling "OK" cookies (1.8e)

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

```
OK CANCEL xxxxxxxx
```

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

15.8. Denying Service to Problem Users

In addition to methods listed above for restricting service areas and the like, LISTSERV has a varied set of methods whereby "problem users" may be denied service on several levels.

15.8.1. The "Filter=" list header keyword

List owners or LISTSERV maintainers may filter specific userids (or users matching a wildcard specification) on a list-by-list basis by using the **Filter=** list header keyword. For more information, see Appendix B.

15.8.2. The "FILTER_ALSO" configuration file variable

LISTSERV maintainers may add specific or wildcarded userids to LISTSERV's built-in filter by using the **FILTER_ALSO=** configuration file variable. Users matching this specification will be denied service to all LISTSERV services on your server.

15.8.3 The "SERVE" command

The LISTSERV maintainer may selectively "serve out" specific userids with the **SERVE** command. This use of the **SERVE** command creates a "hard" serve-out which only a LISTSERV maintainer may override, as opposed to the "soft" serve-out that occurs when 51 consecutive bad commands (raised from 21 in 1.8b) are sent to LISTSERV which any user (other than the served-out user) can override. This function is particularly useful for temporary suspensions of service to certain userids, as it does not require modifying the site configuration file, but naturally it may be used to serve users out permanently. The **SERVE** command does not accept wildcards, so if you need to suspend service to a class of users rather than to a specific user, you should use the **FILTER_ALSO** configuration file variable as described in 17.8.2.

LISTSERV writes an entry for each served-out user to its **PERMVARS FILE**. (*Note:* **PERMVARS FILE** *is not a plain-text file; do not edit it by hand.*) To serve a user out manually, the LISTSERV maintainer sends the command:

SERVE *internet-address* OFF PW=*password*

The password is required. Acceptable passwords are the list creation password (**CREATEPW**), the system file modification password (**STOREPW**), or your personal password set with the **PW ADD** command.

Note that prepending **QUIET** to the **SERVE** command suppresses notification to the user in question. If you are serving out an ID that is causing a mailing loop, you will probably want to use **QUIET** to avoid sending a fresh message to the looping ID.

To restore service to a user previously served out, the LISTSERV maintainer sends the command:

SERVE *internet-address* PW=*password*

Again, the password is required, and the **QUIET** modifier may be used.

When a user is served out by the LISTSERV maintainer, assuming the **QUIET** modifier is not prepended to the **SERVE** command, a message similar to the following is sent:

```
Date:         Thu, 6 Dec 2001 09:45:40 -0500
From: "L-Soft list server at LISTSERV@LISTSERV.EXAMPLE.COM (1.8e)"
              <LISTSERV@LISTSERV.EXAMPLE.COM>
Subject:      Message ("Your access to LISTSERV has just been suspended...")
To:           baduser@somehost.com

Your access to LISTSERV has just been suspended by nathan@example.com.
```

```
Commands and postings from you will be ignored from now on.
```

15.8.4. The POST_FILTER list exit point

See the *Developer's Guide for LISTSERV* for information on programming list exits and the POST_FILTER list exit point.

15.9. Hiding selected header lines

Starting with LISTSERV 1.8d, it is possible to hide part or all of a list header (except for the list title) from users who send the REVIEW command or who try to view the list's configuration via the CataList. The following syntax is used:

```
* My very own list
*
* blah blah blah
*.HH ON
* This line is hidden
* This line is also hidden
*.HH OFF
* This line is not hidden
```

The sequence can be repeated as many times as required. **GET** will return the unedited header with the **.HH** sequences, **REVIEW** will replace hidden lines with a note saying that lines were hidden. You can't hide the fact that some lines were hidden because it would lead to people spending hours trying to figure out problems which only appear to be problems because some of the keywords are not visible. L-Soft will not field support inquiries with hidden headers; you must send the entire raw header (including the **.HH** lines) when requesting support.

15.10. Tracking subscription changes with the Change-Log keyword

This feature and keyword are not available in LISTSERV Lite.

Starting with version 1.8d, LISTSERV includes a "Change-Log=" list header keyword which, when set to "Yes", causes the server to log information about changes to individual subscriptions for a given list to a ***listname*.CHANGELOG** file (***listname*** **CHANGELG** on VM). Changelogs also log postings to the list. Please see chapter 10.6, above, and Appendix B for more information on the "Change-Log=" keyword.

16. Subscription Features and Functions

16.1. Setting up 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.

16.2. Defining default options for subscribers at subscription time

The LISTSERV maintainer or the list owner may specify subscribe-time 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.

If setting "Default-Options=" for an existing list, you will probably want to issue QUIET SET commands for existing subscribers, particularly if you are defaulting an option such as REPRO. Setting "Default-Options=" does *not* affect current subscribers.

Starting with LISTSERV 1.8d any Default-Options that you set are applied to non-subscribers. This is particularly useful if you want to run a public list but disallow non-subscribers from posting, or at least force their mail to go through a moderator first. For the former you could code "Default-Options= NOPOST", and for the latter, "Default-Options= REVIEW". These settings would also apply to new subscribers, and the latter particularly can help you ensure that the people who join your list are legitimate and

haven't just joined the list so they can spam it.

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

16.3. Setting up subscription renewal

(See also chapter 13.5, *Address Probing*, which is related to 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, add the following keyword to the header of the 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. (Remove the brackets around "QUIET" if you want to use the QUIET option. The brackets indicate that QUIET is an optional part of the command.) *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.*

The LISTSERV maintainer or the list owner can also issue the **CONFIRM** command for a

subscriber:

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

LISTSERV creates a daily report on its subscription renewal activities, telling you what users were requested to confirm subscriptions, and what users were deleted for failing to confirm the renewal request. This report is sent to the "Errors-To=" address for the list. A typical subscription renewal monitoring report is reproduced below:

```
Date:         Thu, 30 Oct 1998 06:00:01 -0400
From: "L-Soft list server at PEACH.EASE.LSOFT.COM (1.8d)"
              <LISTSERV@PEACH.EASE.LSOFT.COM>
Subject:      ACCESS-L: Subscription renewal monitoring report
To: ACCERR@LINUS.DC.LSOFT.COM

The following 3 subscribers were deleted from the ACCESS-L list today:

 AHUIE@FIN-AID.CSUHAYWARD.EDU
 byetter@ECLAT.UCCS.EDU
 mgill1@IX.NETCOM.COM


The following 5 subscribers were requested to confirm their subscription to
the ACCESS-L list:

 Ernest_HILL%KN=NYC=ZE@MCIMAIL.COM
 dave@MAIL.DERBY.K12.KS.US
 dianna@INTEX.NET
 koa@KOAMAR.COM
 leider@STR.DAIMLER-BENZ.COM
```

Figure 16.1. Typical daily subscription renewal monitoring report.

17. Other Features and Functions

17.1. Setting up national language mail templates

While LISTSERV is not shipped with non-English language mail templates, it is possible to create such "national language" templates and make them available on a list-by-list basis by using the "Language=" list header keyword. The procedure to use national-language templates is as follows:

1. Translate DEFAULT.MAILTPL (or a desired subset of the templates in DEFAULT.MAILTPL) into the desired language.

2. Call the translated template file *idiom*.MAILTPL, where "*idiom*" is replaced by the name of the language, for example, FRANCAIS.MAILTPL, ESPANOL.MAILTPL, etc.

3. Store the file on the server with a PUT.

4. For lists that will use the national language template, code "Language= idiom" in the list header. For instance, if the national language template is called FRANCAIS.MAILTPL, code "Language= Francais". If you've called it FRENCH.MAILTPL instead, then code "Language= French".

Note that LISTSERV's hard-coded messages will continue to be issued in English, and any editable template not present in your national language template will be pulled from DEFAULT.MAILTPL (or the list's own MAILTPL file, if it exists) when needed.

See chapter 9 for more information on creating and editing LISTSERV's mail templates. L-Soft does not provide national language mail templates.

17.2. Translating control characters included in list mail

LISTSERV removes control characters from mail messages by default and expands tabs in the process. Control characters usually have undesirable and unexpected results, as they seldom survive the double ASCII->EBCDIC->ASCII translation and cause the recipient's terminal to execute sequences different from the ones meant by the message sender - not to mention the case of ASCII control characters on EBCDIC terminals. Furthermore, certain combinations of control characters were found to create problems with IBM's SMTP (due to unexpected CRLF sequences appearing in the middle of a record after translation). Lists which absolutely require control characters must have a "Translate= No" keyword added to their header.

17.3. Communicating with list owners

The LISTSERV maintainer may have occasion to communicate with some or all of the list owners who run lists on his server. This does not require that the LISTSERV maintainer keep a "super-list" of list owners, but only that he or she use certain addresses when communicating with list owners.

17.3.1. The *listname*-REQUEST alias

If you need to communicate with all of the list owners of a single list, simply address your mail to ***listname*-REQUEST**. This special address will be expanded by LISTSERV to include all non-quiet list owners of the specified list. For instance, mail to the list owners of the PEKINESE list on LISTSERV.SIRIUS.NET would be addressed to **PEKINESE-**

REQUEST@LISTSERV.SIRIUS.NET.

17.3.2. The ALL-REQUEST alias

If you need to communicate with all of the non-quiet list owners on your server, simply write to the special **ALL-REQUEST** alias. **ALL-REQUEST** is expanded by LISTSERV to include all non-quiet list owners of all the lists on your server. Note that **ALL-REQUEST** should only be used in emergencies, or when all non-quiet list owners need to be informed that something is happening, such as a migration from one server to another.

For general news it is probably better to create an administrative mailing list for your list owners rather than to use **ALL-REQUEST**.

In LISTSERV 1.8e and following, the ability to post to the ALL-REQUEST alias, which expands to all non-quiet list owners, has been restricted as follows:

1. The site configuration variable, ALL_REQUEST_ALLOWED_USERS, can be used to specify who can mail to ALL-REQUEST. This variable uses the same syntax as POSTMASTER=, in other words, a space-separated list of userid@host specifications.

2. Postmasters are always allowed to mail to ALL-REQUEST, even when not listed in ALL_REQUEST_ALLOWED_USERS.

3. The determination is made conclusively on the RFC821 MAIL FROM because:

    a. This avoids dealing with header errors. Header error = don't know who sent this = discard silently = unhappy admin who had to send an urgent notice to all his owners.

    b. The main point of this change is to block spammers, and spammers typically have non-working or null MAIL FROM: addresses. Needless to say, null doesn't pass the test.

4. When the MAIL FROM is not null, the REQNAK1 mail template form is sent when the message is rejected.

Although a further request was considered for a way to disable ALL-REQUEST entirely, it was decided not to implement it, as the main concern was to block spammers and general users from posting to the alias at random.

17.3.3. Configuration required for unix servers and VMS servers running PMDF

Note that VM servers, VMS servers running MX, and Windows servers do not require that any special aliasing be added for these aliases. This functionality is built-in for those installations.

You should already have coded "***listname*-request**" aliases into your **/etc/aliases** file (unix) or your PMDF aliases file (VMS running PMDF) for each list on your server. See chapter 7 for more information on coding these aliases.

If you are running a unix server, or a VMS server running PMDF, you will probably have to code an alias for "**all-request**" into your aliases file, as this is not normally done at install time. See your installation guide for information on how to code the base-level "**listserv**" and "**listserv-request**" aliases and follow those instructions to add an

alias for "**all-request**".

17.3.4. Other aliases used by LISTSERV

The following aliases need to be added for lists running on VMS (with PMDF) and unix. All other configurations handle them automatically.

In addition to the aliases for "*listname*" and "*listname*-request", LISTSERV also uses the following aliases for specific purposes:

- **owner-*listname***: This alias is placed by LISTSERV in the RFC821 MAIL FROM: header. The expectation is that, per RFC821 et seq., remote hosts will send any delivery errors back to the RFC821 MAIL FROM: address. LISTSERV considers anything sent to the "owner-*listname*" address as a delivery error (which can cause a problem with some older mail clients, such as Microsoft Mail, which use the RFC821 MAIL FROM: rather than the RFC822 From: header as the originator of the mail). This alias maps to the value in the "Errors-To=" list header keyword. In general this address should not be used to communicate with the list owner (*listname*-request is the preferred alias for contacting list owners) as mail sent to this address is always handled as an error.

- ***listname*-server**: This alias is an artefact from the days when it was not always clear what the name of the account running the mailing list manager was. In general, if you had forgotten if the list was running on LISTSERV or Majordomo or whatever, you could write to *listname*-server@*host* and your commands would reach the server. For LISTSERV's purposes, this alias maps to LISTSERV@host . While this alias is not absolutely required, we recommend that it be made available because old documentation may still indicate that this is a legitimate way to write to the mailing list manager.

- ***listname*-search-request**: This alias handles search requests coming from INDEX mode subscriptions and must be present for each list for INDEX to work properly.

- ***listname*-signoff-request** and ***listname*-unsubscribe-request**: (1.8d and later) Mail sent to these aliases will cause a signoff event for the userid in the From: line (no command is required and the body of the message is discarded).

- ***listname*-subscribe-request**: (1.8d and later) Mail sent to this alias will cause a subscribe event for the userid in the From: line (no command is required and the body of the message is discarded).

18. Special Functionality for ISP's

These functions require that your site is appropriately licensed for them. Specifically, your LAK must contain the ISP "Scope" option. Contact the L-Soft sales department for details.

Currently the ISP functionality is under development and does not include a complete suite of tools that an ISP might find useful. If you have suggestions for useful tools (other than accounting tools which *are* in development), please feel free to write to SUPPORT@LSOFT.COM with your comments, which will be turned over to the developers.

18.1. Directory quotas for individual lists

Currently there is no warning message when a list hits a preset percentage of its storage quota, so this function should be used with care.

18.1.1. The QUOTA.FILE

LISTSERV uses a file called **quota.file** to store quota information for individual lists. The **quota.file** must be installed in LISTSERV's "A" directory (the same directory where LISTSERV keeps list files and its other standard data files). The file is a flat text file with the information for each list kept on one line, as follow:

```
/HOME/LISTS/MYLIST-L     1024     Owner

(1)                      (2)      (3)
```

Notes:

1. The directory where the notebook archives and any other user-maintained files belonging to the list are kept. This specification should be the same as the specification in the "Notebook=" keyword (for lists with notebook archives) or the same as the specification for the file archives directory for the list in **site.catalog** (for lists without notebook archives).

2. The size (in kilobytes) of the list's quota. Note that 1024 kilobytes = 1 megabyte, so multiply the desired number of megabytes by 1024 to set this value.

3. The person who should be notified when the list goes over quota; in this case, the *access-level* "Owner" means that the list owner is notified. This value can also be a regular *internet-address*.

18.1.2. Displaying quota information

To display current quota information, issue the command

SHOW QUOTA

LISTSERV will respond with a listing of the lists for which quotas are set, along with the quota setting and percentage of quota used. A typical **SHOW QUOTA** report is reproduced below, for lists called ALIST, BLIST, and so forth:

```
Date:         Mon, 17 Jun 1996 17:09:38 -0400
```

```
From:         "L-Soft list server at HOME.EASE.LSOFT.COM (1.8d)"
              <LISTSERV@HOME.EASE.LSOFT.COM>
Subject:      Output of your job "NATHAN"
To:           Nathan Brindle <NATHAN@EXAMPLE.COM>

> show quota
Directory                       Quota  Usage
---------                       -----  -----
E:\FTP\LISTS\ALIST                 4M     0k  ( 0.0%)
E:\FTP\LISTS\BLIST                 4M     4k  ( 0.0%)
E:\FTP\LISTS\CLIST                 4M   323k  ( 7.8%)
E:\FTP\LISTS\DLIST                 1M    11k  ( 1.0%)
E:\LISTS\ELIST                     4M    11k  ( 0.2%)
E:\LISTS\FLIST                     4M   940k  (22.9%)
E:\LISTS\GLIST                    50M   9.8M  (19.5%)
E:\LISTS\HLIST                     4M    66k  ( 1.6%)
```

Figure 18.1. Typical output of a SHOW QUOTA command issued by privileged user

Your list owners (or the person(s) indicated by the third parameter in **quota.file** for the list) can also issue the **SHOW QUOTA** command, but they will receive quota information only for the list(s) for which they appear in that third parameter.

18.1.3. Reloading quota information after making changes

Whenever you change values or add or delete lists from **quota.file**, you must issue the command

SHOW QUOTA RELOAD

to reload the quota file. (Rebooting LISTSERV also reloads the information.)

18.2. Limiting the number of subscribers to a list

Using the special **Limits=** keyword, you can limit a list to a specified number of subscribers. Only a LISTSERV maintainer may raise, lower, or disable this limit. An attempt by a list owner to change or disable the limit will result in an error message being returned to the invoker and no change being made to the list header.

To enable the subscriber limit in the list header, code

*** Limits= Sub(nnn)**

where **nnn** is the number of subscribers you want to limit the list to. For instance, a list coded **Limits= Sub(200)** would be limited to 200 subscribers.

19. Contacting L-Soft

19.1. Support

L-Soft international recognizes that the information in this manual and the FAQ questions on our web site (**http://www.lsoft.com/lsv-faq.html**) are not going to solve every problem you may face. We are always willing to help diagnose and correct problems you may be having with your licensed LISTSERV server. To that end, please note the following when you write to L-Soft with a problem report:

1. Make the subject line of your report indicative of the problem. L-Soft receives a great deal of mail with the subject "Help!", which is not very helpful when we receive them.

2. Include any appropriate log entries. LISTSERV keeps logs of everything it does, and without the log traceback, it is often impossible to determine what caused a given error.

3. If you're running a Unix server and LISTSERV dumps core, please run the debugger on the core file, produce a traceback, and include the results.

4. Always send a copy of your site configuration file (with the passwords x'ed out).

5. Send along anything else that you think might be helpful in diagnosing the problem.

If you are running an evaluation version of our software, please send your trouble reports to the evaluation users' list, LSTSRV-E@PEACH.EASE.LSOFT.COM.

If you are running the Windows 95 shareware server, please send your trouble reports to the LISTSERV for Windows 95 mailing list, LISTSERV95-L@PEACH.EASE.LSOFT.COM. This includes registered users of the software.

If you are running LISTSERV Lite, please send your trouble reports to the LISTSERV Lite support mailing list, LISTSERV-LITE@PEACH.EASE.LSOFT.COM. This includes users of the paid version of the software.

If your LISTSERV Classic server for VM, VMS, unix, or Windows NT has paid-up maintenance, you may send problems to SUPPORT@LSOFT.COM for a quick reply.

19.2. Sales

To reach our worldwide sales group, simply write to SALES@LSOFT.COM. You may also call 1-800-399-5449 (in the US and Canada) or +1 301-731-0440 (outside the US and Canada) to speak to our sales representatives.

19.3. Manuals

To comment on this or other L-Soft manuals, please feel free to write to MANUALS@LSOFT.COM, and be sure to mention which manual you are commenting on. (However, please do not send support questions to this address. See above for appropriate support addresses.)

Appendix A: Command Reference Card for LISTSERV® version 14.5

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)
---------------------------------------------------------
Commands that support the QUIET keyword are marked (*)

SUBscribe(*) listname <full_name>             Subscribe to a list, or change
                                              your name if already subscribed
                       ANONYMOUS              -> Subscribe anonymously

             Following either <full_name> or ANONYMOUS you may specify
             individual user options:

                       <WITH opt1 opt2...>    -> with specified user options

SIGNOFF                                       Remove yourself:
             listname                         - From the specified list
             *                                - From all lists on that server
             * (NETWIDE                       - From all lists in the network

CHANGE                                        Change your subscribed address
                                              to "newaddr":
             listname newaddr                 -> on the specified list
             * newaddr                        -> on all lists on the server

SET          listname options                 Alter your subscription options:
             ACK/NOACK/MSGack                 -> Acknowledgements for postings
             CONCEAL/NOCONCEAL                -> Hide yourself from REVIEW
             HTML/NOHTML                      -> Prefer/avoid HTML format
                                                 (especially HTML digests)
             Mail/NOMail                      -> Toggle receipt of mail
             MIME/NOMIME                      -> Prefer/avoid MIME format
                                                 (especially MIME digests)
             DIGests/INDex/NODIGests/NOINDex  -> Ask for digests or message
                                                 indexes rather than getting
                                                 messages as they are posted
             REPro/NOREPro                    -> Copy of your own postings?
             TOPICS: ALL                      -> Select topics you are
                     <+/->topicname              subscribed to (add/remove
                                                 one or replace entire list)

Options for mail headers of incoming postings (choose one):
             FULLhdr or FULL822               -> "Full" (normal) mail headers
             IETFhdr                          -> Internet-style headers
             SHORThdr or SHORT822             -> Short headers
             DUALhdr                          -> Dual headers, useful with PC
                                                 or Mac mail programs
             SUBJecthdr                       -> Normal header with list name
                                                 in subject line

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

Other list-related commands
---------------------------
GETPOST      listname ref1 <ref2 <...>> <opt> Order individual messages from
                                              list archives
```

```
                There is a single option:

                NOMIME                          Retrieve messages in "raw" form,
                                                ie, do not re-encode MIME
                                                attachment links (pre-1.8e
                                                    behavior)

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 /xyz                         -> All known lists whose name or
                                                       title contains 'xyz'
                SUMmary <host>                      -> Membership summary for all
                                                       lists on specified host
                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
                   Date                                by subscription date
                   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
                Topics                              -> Include breakdown of
                                                       subscribers per topic
                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
                ALL                                 -> List both concealed and non-
                                                       concealed subscribers (list
                                                       owners/site maintainers only)

SCAN            listname text                       Scan a list's membership for a
                                                    name or address

SEArch          listname word1 <word2 <...>>        Search list archives
        or:     word1 <word2 <...>> IN listname
                FROM date1                          -> From this date
                     TODAY                          -> From today
                     TODAY-7                        -> In the last 7 days
                TO   date2                          -> To this date
                WHERE
                  SUBJECT CONTAINS xxxx             -> Only this subject
                AND/OR
                  SENDER  CONTAINS xxxx             -> Only this author
                                                    Complex boolean operations are
                                                    supported, see database guide

STats           listname <(options>                 Get statistics about a list (VM)
```

```
            LOCal                               -> Don't forward to peers


Informational commands
----------------------
Help                                            Obtain a list of commands

INFO        <topic>                             Order a LISTSERV manual, or get
            <listname>                          a list of available ones (if no
                                                topic was specified); or get
                                                information about a list

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 NJE
                                                   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)
            POINTs <ALL | list1 list2...>       -> Graduated license point
                                                   information for planning
            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

Other (advanced) commands
-------------------------
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>'
             MAIL-MERGE                   -> Data is a mail-merge message.
                                             See the Developer's Guide to
                                             LISTSERV for specifics.
             POST                         -> (non-VM only) Same as MAIL
                                             except that the message is pre-
                                             approved. See the Developer's
                                             Guide to LISTSERV for specifics.
             FILE                         -> Data is not mail, recipients
                                             are defined by '<dest>'
             RFC822                       -> Data is mail and recipients
                                             are defined by the RFC822
                                             'To:'/'cc:' fields
             Source:
             DD=ddname                    -> Name of 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'
             PRE-APPROVED=YES                   -> Pre-approve message (with
                                                   DISTRIBUTE POST only)

FOR          user command                       Execute a command on behalf of
                                                another user (for node
                                                administrators)

SERVE        user                               Restore service to a disabled
                                                user

THANKs                                          Check 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
             <"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
             listname DD=ddname IMPORT <PRELOAD>
                                                -> Bulk add multiple users, one
```

```
                                                address/name pair per line
                                                PRELOAD option loads addresses
                                                into memory before adding to
                                                speed up operation
ADDHere(*)                                      Same as ADD, but never forwards
                                                the request to a possibly closer
                                                peer

CHANGE(*)    listname|* oldaddr|pattern newaddr|*@newhost
                                                Change a subscriber's address
                                                (List owner's version)

DELete(*)    listname user <(options>           Remove a user from one of your
                                                lists, or from all local lists
             listname DD=ddname <BRIEF>         Bulk delete multiple users, one
                                                address per line.  BRIEF option
                                                omits verbose response of who was
                                                deleted.
             Options:
             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

LISTs        OWNed                              Send back a list of local lists
                                                owned by the invoker
             MODerated                          Send back a list of local lists
                                                moderated by the invoker

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 header from the
                                                file returned by a GET command
```

```
PUTALL       listname LIST                    Similar to PUT but lets you
                                              store the entire list, header
                                              and subscribers together

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

Site management functions
-------------------------

CMS          command_text                     Issue a CMS command and get the
                                              last 20 lines of response sent
                                              back to you, the rest being
                                              available from the console log

CP           command_text                     Issue a CP command and get up to
                                              8k of response data sent to you
                                              (the rest is lost)

DATAbase     function                         Control operation of databases:
             DISAble                          -> Disable interactive database
                                                 access, without shutting down
                                                 existing sessions
             ENAble                           -> Re-enable interactive access
             SHUTDOWN                         -> Shut down all interactive
                                                 database sessions, and
                                                 disable interactive access

INSTALL      function                         Software update procedure:
             CLEANUP shipment                 -> Remove an installed shipment
                                                 from the log
             CLEANUP BEFORE dd mmm yy         -> Remove all shipments
                                                 installed before that date
             PASSWORD shipment PW=instpw      -> Confirm installation of a
                                                 shipment, when requested by
                                                 LISTSERV
             RELOAD shipment                  -> Attempt to reload a shipment
                                                 which failed due to a disk
                                                 full condition
             STATus                           -> Get a list of installed
                                                 "shipments"


LISTs        OWNed <BY> userid@host           Send back a list of local lists
                                              owned by the address supplied
                                              (wildcards acceptable)
             MODerated <BY> userid@host       Send back a list of local lists
                                              moderated by the address supplied
                                              (wildcards acceptable)

NODESGEN     <WTONLY>                         Regenerate all LISTSERV network
                                              tables, or just compile the
```

```
                                          links weight file (debugging
                                          command)

OFFLINE                                   Suspend processing of reader
                                          files and disable the GET
                                          command

ONLINE                                    Cancel OFFLINE condition

PUT          listname LIST                Create a new list

PUTC         fn ft <fm|cuu|dirid>         Update a CMS file on one of
             <RECFM=F LRECL=nnn>          LISTSERV's R/W minidisks; note
                                          that this is similar to SENDFILE
                                          + RECEIVE or LINK + COPYFILE and
                                          should NOT be used to update
                                          file-server files

PWC          function                     Password file management:
             ADD    user newpw            -> Define a password for the
                                             specified user
             DELete user                  -> Delete password for that user
             Query  user                  -> Query the password of the
                                             specified user

REGister     name|OFF FOR user            Set a user's SIGNUP FILE entry

SENDFile     fn ft <fm|cuu|dirid>         Request the server to send you a
                                          file from one of its disks

SERVE        user OFF                     Permanently suspend access from
                                          an abusive user or gateway
                                          (restore with 'SERVE user')
             user OFF DROP                Permanently suspend access from
                                          an abusive user or gateway and drop
                                          all further inbound mail from this
                                          sender on the floor
                                          (restore with 'SERVE user')
             LIST                         Return a list of all addresses that
                                          are currently served off or which
                                          are spam-quarantined

SF                                        Same as SENDFILE

SHOW         BENCHmarks                   -> CPU/disk/paging benchmarks
             EXECLoad                     -> Statistics about EXECLOADed
                                             REXX files
             LSVFILER                     -> Statistics about LSVFILER
                                             file cache
             PREXX                        -> Statistics about PREXX
                                             functions usage
             STORage                      -> Information about available
                                             disk space and virtual
                                             storage

SHUTDOWN     <REBOOT|REIPL>               Stop or reboot the server (the
                                          two options are synonyms)

STOP                                      Same as SHUTDOWN

Note: some debugging commands and options have been omitted.

Syntax of parameters
--------------------
filelist  = 1 to 8 characters from the following set: A-Z 0-9 $#@+-_:
fformat   = Netdata, Card, Disk, Punch, LPunch, UUencode, XXencode, VMSdump,
            MIME/text, MIME/Appl, Mail
fn        = same syntax as 'filelist'
ft        = same syntax as 'filelist'
full_name = firstname <middle_initial> surname (*not* your e-mail address)
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        = A password with characters from the set: A-Z 0-9 $#@_-?!|%
user      = Any valid Internet address not longer than 80 characters; if
            omitted, the 'hostname' part defaults to that of the command
            originator
```

Appendix B: List Keyword Reference for LISTSERV® version 14.5

This document (reference number 0506-UD-01) is available separately. It can be retrieved in plain text from any server running L-Soft's LISTSERV® with the command INFO KEYwords, or may be downloaded from the URL

ftp://ftp.lsoft.com/documents/listkeyw.memo

Last updated 23 Feb 2006

The List Header

The list header contains configuration information for the list. To edit it, use the **GET *listname* (HEADER** command, edit the header, and send it back to LISTSERV with the **PUT listname PW=XXXXXXXX** command. For more details on this procedure, consult the *List Owner's Manual for LISTSERV*. If you have the web archive and administration interface installed, you can also do this via the web (see chapter 11 of this manual).

Each line of the header must begin with an asterisk ("*"). The first line of the header must contain the list title, which must fit on a single line and not exceed 40-50 characters. Succeeding lines hold list control keywords and their values. Any words in the list header followed by the "=" character are assumed to be keywords. Following the list of keywords, you may add a few lines containing a brief description of the purpose of the list. These lines must also begin with an asterisk ("*").

This document is a description of the list control keywords that appear in the header of each list. Whenever default values are supplied for the keywords, they are listed first in the description. Words in *italics* are "generic parameters" which define a set of possible values for a keyword operand, as described below:

Format of List Header Keyword Settings

List header keyword settings can be defined in any of the following formats (we are using three parameters for our examples; note that some keyword settings have more then three parameters and adjust accordingly):

```
* Keyword= parameter1,parameter2,parameter3
```

```
* Keyword=parameter1,parameter2,parameter3

* Keyword= parameter1, parameter2, parameter3

* Keyword = parameter1,parameter2,parameter3

* Keyword= parameter1
* Keyword= parameter2
* Keyword= parameter3
```

The last example above is useful when defining a keyword setting (for instance, Notebook=) which may contain a long directory path. When using this format, note carefully that the last parameter on the line MUST NOT be followed by a comma. LISTSERV will properly concatentate the parameters internally as if they had commas. For instance,

```
* Keyword= parameter1,parameter2
* Keyword= parameter3
```

and variations are also supported; for instance,

```
* Notebook= Yes
* Notebook= /home/listserv/archives/english101-fall2002
* Notebook= Weekly,Private
```

is perfectly legal.

LISTSERV reads list headers one line at a time, assuming that any word followed by an equal sign is a keyword, and then, starting at the equal sign, reads keyword parameter settings either until it encounters a space that is not preceded by a comma (the first parameter excepted), or until it reaches the next word in the line which it evaluates as being a keyword. Thus LISTSERV will completely ignore a keyword setting coded as follows:

```
* Keyword parameter1, parameter2, parameter3
```

because there is no equal sign following the keyword. (This is one way to comment out a keyword setting if you do not want to completely remove it from the list header.) Likewise, while LISTSERV will recognize the following as a keyword setting:

```
* Keyword= parameter1, parameter2 parameter3
```

it will read only to the second parameter because the second parameter is not followed by a comma.

It is also possible to define multiple keywords on a single line, so long as the line does not wrap and does not exceed 100 characters. For instance,

```
* Send= Private     Confidential= Yes     Review= Owners
```

is a legal LISTSERV header line containing settings for the Send=, Confidential=, and Review= list header keywords.

Keyword settings are always evaluated in a case-insensitive manner. Under unix, where case sensitivity is an important issue, file and directory paths defined in the list header are always converted to lower-case for LISTSERV's use. Thus all data files written to or

read from by LISTSERV under unix must be named in lower case, regardless of the case used in the list header keyword setting.

Hiding header lines

Starting with LISTSERV 1.8d, it is possible to hide part or all of a list header (except for the list title) from users who send the REVIEW command or who try to view the list's configuration via the CataList. The following syntax is used:

```
* My very own list
*
* blah blah blah
*.HH ON
* This line is hidden
* This line is also hidden
*.HH OFF
* This line is not hidden
```

The sequence can be repeated as many times as required. **GET** will return the unedited header with the **.HH** sequences, **REVIEW** will replace hidden lines with a note saying that lines were hidden. You can't hide the fact that some lines were hidden because it would lead to people spending hours trying to figure out problems which only appear to be problems because some of the keywords are not visible. Please note that L-Soft will not field support inquiries with hidden headers; you must send the entire raw header (including the **.HH** lines) when requesting support.

In LISTSERV 14.2 (1.8e-2003a) and later,

- .HH commands can be nested.
- The .HH ON and .HH OFF dot commands are respected in KEYWORDS files called from list headers with the .IK dot command. Previous versions ignored .HH commands in KEYWORDS files.

The following should be noted:

- In a KEYWORDS file, .HH OFF found in excess of .HH ON will be ignored. This ensures that a KEYWORDS file called from inside of an .HH ON block will not expose the remainder of that block upon return from the call.
- Similarly, LISTSERV will internally generate as many .HH OFF tags as necessary before exiting the KEYWORDS file and returning to the list, if more .HH OFF tags than .HH ON tags exist in the KEYWORDS file.

Both of these precautions ensure that .HH coding errors in a KEYWORDS file will not result in exposure of keyword settings that it is desired to keep hidden.

Generic parameters

Note: Special parameters used by only one or two keywords are defined along with the keyword and do not appear in this listing.

net-address Describes an Internet address, such as JACK@XYZ.COM.

access-level Controls which category of users has access to the information or service to which this parameter applies. *access-level* can be either:

Public Everybody has access to the information.
Postmaster Only the postmaster (i.e. LISTSERV operations staff) has access to the information.
A1,A2,... with Ai being either:

Private Only users subscribed to the list have access to the information.
(*listname*) Only the subscribers of the named list have access to the information.
Owner Only the list owner can access the information.
Owner(*list*) Only the owner of the named list can access the information.
Service Only people in the service area of the list can see the information.
Service(*list*) Only subscribers of the named list's service area can see the information.

destination Indicates the destination of a piece of mail, message or reply.

List The reply message is sent to the list.
Sender The reply message is sent to the sender of the original piece of mail.
Both The reply message is sent both to the list and to the original sender.
None No reply message is sent at all.
"*address*" The reply message is sent to the specified network address if enclosed in double quotes

interval Is a time interval that indicates how frequently an operation is to be renewed. Note that depending on the operation being performed, some of the options may not be available. For example, "Notebook= Yes,A,Daily" is not available.

Yearly }
Monthly }
Weekly } Self-explanatory
Daily }
Hourly }
Single The operation is to be done only a single time.

peer Is the node-id or network address of a peer list. If the name of the peer list is the same as the name of the local list (which will usually be the case), only the node name needs be given. If the list names are different, the full list network address must be given, for example "REXX-L@UIUCVMD".

area Is a means whereby a node or list of nodes can be identified. An area can be either:

- The name of a network, for example EARN, BITNET
- The name of a country, for example Germany, Canada
- 'Local', in which case it is equated to the value of the "Local="

keyword (q.q.v.).

- A node name, for example SEARN
- A simple wildcard nodename pattern such as FR*, *11, *ESA*, D*ESA*, etc.

mon-address Is a means whereby 'list monitors' can be identified (the term 'list monitor' refers to a human person who monitors the activity of a list). A 'mon-address' can be:

- A single network address, for example INFO@TCSVM
- 'Postmaster', which indicates the "main" postmaster
- 'Postmasters', which indicates ALL the postmasters, main and alternate
- 'Owner', which indicates the "main" list owner (the first to be listed in the "Owner=" keyword)
- 'Owners', which indicates ALL list owners

Some keywords can take more than one parameter. Where multiple parameters are accepted, they will be separated by a logical OR sign (|). Unless specified otherwise, commas have "higher priority" than OR signs, that is to say, "Public|Private, Open|Closed" means "(Public|Private), (Open|Closed)", not "Public|(Private,Open)|Closed".

Optional parameters are indicated by enclosure in square brackets (**[ ]**); for instance, in the case of

Send= Editor[,Hold][,Confirm[,Non-Member | All]][,Semi-Moderated][,NoMIME]

the only required parameter is the first one ("Editor"), and each of the following five parameters is optional. In this case, note carefully the nested brackets signifying that one parameter ("Non-Member | All") is available only if another parameter ("Confirm") is also specified in the keyword setting. Further, because the logical OR symbol is used, the two options "Non-Member" and "All" are mutually exclusive, thus only one of them may be specified.

Do not type the brackets when using the optional parameters! Where the use of square brackets and logical OR signs together could be confusing, we have shown each of the alternate configurations on separate lines.

Keyword Classifications

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

Access Control Keywords

Attachments= No[,Filter]
Attachments= Yes[,*allowed_content_types*[,Filter]]
Attachments= All[,*allowed_content_types*[,Filter]]

LISTSERV 1.8d 2000b "level set" and LISTSERV 1.8e and later include a list-owner-configurable message attachment filter. This feature allows you to control the posting of various types of MIME attachments (images, audio, etc.) to your lists. LISTSERV 1.8e adds the ability to control the posting of inline uuencoded files to your lists on an on/off basis (off being the default if attachment control is enabled).

NOTE: The ability of LISTSERV to filter or reject messages that contain MIME attachments is completely dependent on the ability of the poster's mail client to properly identify the MIME attachment when the mail is originally sent. Filtering/rejection is done based on the Content-Type headers found in the message--NOT by evaluation of the actual contents of the attachment. If for instance an executable binary (normally Content-Type: application/octet-stream) is sent by the client with a Content-Type of "text/plain", it will not be filtered or rejected by LISTSERV since (as noted below) text attachments are not covered by this keyword setting.

A registry of allowable MIME types for attachments, maintained by the Internet Assigned Numbers Authority (IANA) per RFC2048, can be found at

http://www.isi.edu/in-notes/iana/assignments/media-types/media-types

The options are:

`Attachments= Yes` All types of attachments are allowed to be posted to the list (the default). Note however that other configuration options may still disallow the posting of certain attachments, and that "Attachments= Yes" does not override them. For instance, if you have "Language= NoHTML", setting "Attachments= Yes" does not override the Language= setting. Or if you have "Sizelim=" set to a value that precludes a file of x number of lines from being posted to the list, setting "Attachments= Yes" will not override the Sizelim= setting if the message with its attachment exceeds the number of lines specified by Sizelim=.

`Attachments= No` All types of attachments are disallowed, other than plain text (always allowed) and HTML text (which is controlled exculsively by the "Language= NoHTML" keyword setting). With "Attachments= No", LISTSERV rejects messages containing attachments and bounces them back to the poster.

`Attachments= No,Filter` Same as "Attachments= No", except that LISTSERV simply removes the unwanted material from the message and processes it instead of rejecting it out of hand. The removal of material is a silent operation and the poster is not notified that the attachment was discarded.

In LISTSERV 1.8e and later, note that in all three of the above cases, when a message containing one or more uuencoded files is posted to the list, the encoded file(s) is/are stripped from the body of the message and the remainder

of the message is passed through to the list. LISTSERV 1.8d was unable to recognize or strip inline uuencoded files.

Attachments= All (Requires 1.8e or later) This setting tells LISTSERV to *allow* inline, uuencoded files, such as are generated by Microsoft Outlook, overriding the default.

One important restriction: UUencode filtering is strictly on/off. There is no attempt on the part of LISTSERV to guess file types when filtering is enabled (the default). This would be hazardous to begin with as support for these attachments is usually provided on a legacy basis in mail clients; that is, client A and client B could have a very different opinion on the type of the attachment.

It is also possible to allow certain MIME types to be passed through to the list while rejecting or filtering all others. For instance,

```
Attachments= Yes,image,application/*msword
```

allows only the specified attachment types and rejects everything else. If you don't want to reject messages that contain other types of attachments, but just want to remove all other types of attachments, you add the ",Filter" parameter at the end of the line--ie,

```
Attachments= Yes,image,application/*msword,Filter
```

This means, "Allow all image and application/*msword attachments, and strip all other attachments". Again, note that plain text ("Content-Type: text/plain") is always allowed and does not need to be included in the list of allowed attachment types. Likewise, HTML text is controlled exclusively by the "Language= NoHTML" keyword setting. Other text subtypes are, however, controlled by "Attachments=", so they need to be listed if you intend to allow them.

Additionally, should it be desired to allow all inline uuencoded files but restrict the list to certain MIME types, you can specify, similar to the above, something like

```
Attachments= All,image,application/*msword
```

or

```
Attachments= All,image,application/*msword,Filter
```

(In the preceding examples note carefully that "image" by itself is equivalent to "image/*"--in other words, when you code "Attachments= image", you are saying that all MIME image sub-types, for example, "image/jpeg", "image/gif", and so forth, are to be accepted. If only certain sub-types are acceptable, for instance if you want to accept only JPEG graphics and ensure that others don't go through, you must specify the types explicitly--eg "Attachments= image/jpeg".)

Note carefully that simply coding something like "Attachments= image" will not necessarily allow all image files through. This is highly dependent on the client being used by the poster. For instance, if your client attaches all binary files as "Content-Type= application/octet-stream", regardless of whether a given binary is (for instance) an executable image, a Word file, or a compressed archive, and you send a JPEG to a list with "Attachments= image" set in the header, it will be

rejected since the image does not have a "Content-Type: image" tag. Specifically this appears to be the case with Eudora 3.x but may not be limited to that particular client.

In the 1.8d version of the attachment filter, attachments sent by Microsoft Outlook cannot be blocked by LISTSERV as they do not follow MIME standards (at least not up to and including Outlook 97; this writer has not installed Outlook 2000). Outlook sends attachments as imbedded uuencoded files and does not use the MIME Content-Type: header. LISTSERV 1.8e is able to recognize and filter inline uuencoded "attachments" as noted above.

The rejection message sent by LISTSERV when ",Filter" is not specified is found in the **BAD_ATTACHMENT** mail template form (see chapter 9 for information on LISTSERV's mail templates). Note that the **BAD_ATTACHMENT** template form is a linear template and as such does not allow text formatting commands to be used.

The reason HTML text is not subject to "Attachments=" filtering is to allow you to reject (bounce) messages with attachments, while silently suppressing HTML text in multi-part messages which also contain a plain-text alternative. Some mail programs send both HTML and plain-text versions of messages, and, even if you do not want HTML text on your list, there is little point in keeping out people who use it (who are often new to the Internet and aren't aware that their mail programs are sending HTML text) when you can simply remove the HTML part. At the same time, you may want to reject postings containing images out of hand, rather than removing the images and continuing. The same applies to Exchange attachments, which are filtered by default (see "Language= Exchange").

Files=Yes | No

This keyword is not available in LISTSERV Lite.

(NJE only; obsolete in other versions) Indicates whether NJE files can be sent to the list or not. The default value is "No". "**Files= No**" may prevent some non-RFC822 mailer users from posting to lists.

Files= has absolutely no effect under the non-NJE versions of LISTSERV. Specifically it will not prevent users from sending "attached" (MIME-encoded) files to lists. It is provided under all versions for backwards compatibility only (i.e., for lists being migrated from NJE servers). See the **Attachments=** keyword for attachment blocking.

Filter= Only,net-address1,net-address2,...[Allow],net-addressn,... |
Filter= Also,net-address1,net-address2,...[Allow],net-addressn,... |
Filter= Safe,net-address1,net-address2,...[Allow],net-addressn,... |

"**Filter=**" is checked when a user attempts to post or subscribe to a list (but not when the list owner issues an **ADD** command). The first parameter of this keyword is either "Only", "Also" or "Safe", and it is followed by a list of patterns such as 'X400MAIL@*' or '*@*.XYZ.EDU' (without the quotes).

- If "Filter= Also..." is specified, your filter is used in addition to the standard LISTSERV filter; this is useful to register additional looping mailers, to prevent users behind broken gateways from subscribing until the problem is addressed, or to ban anonymous posters.

- If "Filter= Only..." is specified, the addresses you specify are the only ones which LISTSERV prevents from posting to the list. CAUTION: You should not use this option unless you also code "Safe= Yes", and even then you will want to ask your LISTSERV maintainer for permission. This option has been added mostly for LISTSERV maintainers with very specific problems to solve. The minimal filter is very small and you should never need to override it.

- If "Filter= Safe" is specified, LISTSERV uses the "more safe" of its two built-in filters. These built-in filters are (1) a "minimal" one, which is used for safe lists; and (2) a "safe" one which is used for lists running with "Safe= No". That is, the unsafe lists need a safe filter to avoid mailing loops; safe lists only need the minimal filter, but can be made even safer by selecting "Filter= Safe". This, however, prevents usernames such as 'root' from posting to the list, because they are included in the safe filter.

  Messages sent to the LISTSERV userid for execution are always checked with the minimal filter, as people with userids such as 'root' would otherwise not be allowed to subscribe to lists which were set up to allow them.

In all cases, LISTSERV extracts as many e-mail addresses as it can from the userid being checked and runs them all through the filter. For instance if your list receives mail from 'searn.sunet.se!mailer@xyz.edu', LISTSERV will check 'searn.sunet.se!mailer@xyz.edu', 'mailer@searn.sunet.se' and 'mailer@searn' (via the '**:internet.**' tag in **BITEARN NODES**).

Version 1.8d adds the ability to "exempt" certain addresses (or wildcards) from the default filters. You can combine "**Filter= ...,Allow,...**" with the forms documented above as long as you put the "allow" information last. That is, the first word must be **ONLY**, **SAFE** or **ALSO** as before, and you can then have **ALLOW** anywhere (including as the second word) followed by a list of addresses that should be allowed even if present in the filter. The default for **ALLOW** is the empty string. Examples of **ALLOW** usage follow:

```
Filter= Also,userid@host1.com,*@host2.edu
Filter= Allow,niceguy@host2.edu
```

The first example stops everyone from host2.edu from posting to the list, but since we've determined that niceguy@host2.edu is a considerate human being and should be allowed to post, we've defined him as an exception to the general rule by including him in the "Allow" part of the filter.

```
Filter= Safe,Allow,root@host1.edu
```

The second example would invoke the "safe" filter mentioned above, but would allow root@host1.edu to subscribe to and post to the list, instead of bouncing his mail because it matches one of the entries in the "safe" filter. All other "root" users' mail will be caught by the "safe" filter and transferred to the list owner as an error.

See also **Safe=** and **Loopcheck=**.

Review= ***access-level***

This keyword defines the categories of users who are allowed to review the (non-

concealed) Internet addresses and names of the persons subscribed to a list. Beginning with version 1.8c, the default value is "**`Review= Private`**".

Send= Public[,Confirm][,Non-Member]
Send= Private[,Confirm]
Send= Editor[,Hold][,Confirm[,Non-Member | All]][,Semi-Moderated][,NoMIME]
Send= *other-access-level*[,Confirm]

(Note: The "Non-Member" option is available only in LISTSERV 14.3 and later, and only with "Send= Public,Confirm" or "Send= Editor,Confirm...". Similarly, the "All" option is available only in LISTSERV 14.3 and later, and only with "Send= Editor,Confirm...". Please see below for full details.)

Defines the categories of users who can mail or send files to the list. Possibly puts the list under control of an editor. The default value is "Public". Other *access-levels* for use with Send= would include "Private", "Editor", "Owner", etc. (see the beginning of this document for the definition of an *access-level*). A literal Internet e-mail address may also be used in place of the *access-level*, for example, `Send=joe@foo.bar.com`. Using a literal address is one way to ensure that only an authorized person can post to the list, for instance, if the list is an "announce-only" list rather than a discussion list.

Requiring confirmation: When the "Confirm" option is enabled, the sender is required to confirm the posting. When LISTSERV receives mail for the list, it sends an e-mail to the sender requesting a confirmation. The confirmation can be accomplished by replying to the confirmation with another e-mail containing the text "OK" (the subject of the confirmation e-mail contains a validation "cookie") or by clicking on the confirmation URL provided in the e-mail.

List owners can require that non-subscribers actively confirm their messages to the list, while allowing subscribers to post without confirmation. This can dramatically cut down on spam for lists accepting postings from non-subscribers. When "Send=Public,Confirm" is used, the "Non-Member" setting can be specified to limit the confirmation requirements to non-members. When "Send=Editor,Confirm" is used, the "Non-Member" setting can be specified to limit the confirmations to non-members and to require editors to confirm their own posts (see below).

Moderated/Edited lists: 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 "Hold" option is enabled (`Send= Editor,Hold`), the moderator(s) may approve postings using the "OK" mechanism rather than forwarding the posts back to the list. "Hold" is valid only with "Editor".

When the "Confirm" option is enabled on a moderated list (for instance, `Send=Editor,Confirm`), mail sent by any editor or moderator requires confirmation by that editor or moderator. This is to prevent a user from forging mail to the list under an editor's or moderator's return address. The confirmation request is validated with the "OK" mechanism. Please note that this does not mean that a double validation is required when an editor approves other peoples' postings, but only that the editor's own postings to the list and any reposts he

does on others' behalf require a confirmation from him that he actually submitted them. In other words if the editor simply "OK"s a posting, he doesn't have to "OK" his own "OK".

It is also possible to set a list to

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

This allows you to "OK" both subscriber submissions and editor/moderator approvals, as described above.

"Non-Member" and "All" options with "Confirm" (14.3 and later): List owners can require that non-subscribers actively confirm their messages to the list, while allowing subscribers to post without confirmation. This can dramatically cut down on spam for lists that accept postings from non-subscribers. To activate this feature, you would set:

```
* Send= Public,Confirm,Non-Member
```

or either

```
* Send= Editor,Confirm,Non-Member
```

or

```
* Send= Editor,Hold,Confirm,Non-Member
```

in the list header.

It should be noted that, although LISTSERV will accept “Send= Private,Confirm,Non-Member” when the header is stored, that setting is equivalent to just “Send= Private,Confirm” since by definition only subscribers are allowed to post to a list coded "Send= Private".

The intent is to help list owners cut down spam on public discussion lists, without inconveniencing normal list subscribers. For public lists, it works like "Send= Public,Confirm" if you are not a member of the list, otherwise it works as "Send= Public" (no confirmation required from subscribed users). List owners and editors are considered to be members of the list even if they are not subscribed to it, and are thus not subjected to the confirmation requirement.

For edited lists, the behavior is similar -- non-members must confirm their own postings before they are submitted to the editor for approval, whereas members' postings go directly to the editor for approval without the intermediary step. It should be noted that ",Confirm" still activates the anti-spoofing feature that already existed, which requires that the editor must approve his own postings.

Please note carefully: On an edited list, if it is desired for all posters to confirm their own postings regardless of their subscription status, substitute "All" for "Non-Member", that is

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

forces all posters to validate their own postings before they are submitted to the editor for final approval.

IMPORTANT: "Non-Member" or "All" must always be specified in conjunction with "Confirm". For instance, setting "Send= Public,Non-Member" or "Send= Editor,All" will not activate the feature.

Semi-Moderated option for Send= Editor: 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.

"`Send= Private,Semi-Moderated`" was supported in Revised LISTSERV prior to approximately 1994, and, while previously documented in this space, was never actually available in L-Soft LISTSERV (that is, from version 1.8a and following).

A usage example:

```
* Send= Editor,Semi-Moderated
* Editor=NATHAN@EXAMPLE.COM,ERIC@EXAMPLE.COM
```

In this example, a message sent to the list would be:

- Processed, if the sender used the "Urgent:" subject
- Forwarded to the moderator if the sender didn't use the "Urgent:" subject.

Note that in the above example, messages don't get discarded if the sender isn't subscribed.

Moderation "OK" requests and MIME attachment display: In versions previous to LISTSERV 1.8e, an OK confirmation request for a message coming to a moderated list displayed the message to be approved in its "raw" format, i.e., there was no attempt made to display/decode MIME attachments that might be present in the message to be approved. LISTSERV 1.8e addresses the problem by including a copy of the first text/plain part (if one exists in the message) for the purpose of quick screening. The following restrictions apply:

1. This is only done for MIME messages (even simple single-part ones, but they must have MIME headers).

2. The text part in question is sent pretty much 'as is', that is, as an extra text/plain part in the message, with all the options and encoding and what not supplied in the original message. The reason is quite simply that it would be a lot of work and, in some extreme cases (incompatible code page, etc), completely impossible, to embed it into the first text/plain part with the LISTSERV message.

The drawback is that some mail agents might conceivably only show the first part until you take some kind of clicking action.

It is important to understand that only the first text/plain part is extracted in this fashion. The goal was to make it easier to approve or reject simple text messages, not to build a factory around a simple problem. The ENTIRE message is available at an extra click.

Where security is a concern, it is important to review the ENTIRE original message and not just the plain text part. There could be an obscene GIF or another text part or a text/html part not matching the contents of the text/plain part or whatever. This is why, again, you are given the ENTIRE original message.

List owners using certain email clients (specifically Pine, which handles attachments in a secondary viewing area) may find the new format difficult to use. If preferred, the pre-1.8e behavior may be reverted to by specifying "NOMIME" in the Send= list header keyword; for instance,

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

"Hold" now forced for multi-part moderated messages: LISTSERV 14.3 and following will force ",Hold" if it is not specified with "Send= Editor" for multi-part messages, in order to generate an approval request that can display the multi-part message correctly.

Stats= Normal | None,*access-level*

This keyword is not available in LISTSERV Lite.

This keyword is obsolete and has absolutely no effect on all ports of the software except for VM. On non-VM servers it is provided for backwards compatibility only (i.e., for lists being migrated from VM) in order that any existing "Stats=" keyword setting in a migrated list header does not trigger a command parser error.

UNDER VM ONLY, indicates whether or not statistics are to be maintained for the list and if yes, which level of statistics is desired and who is able to retrieve the statistics reports. The default value is "Normal,Private".

Normal statistics include number of mailings for each user on the list, and similar information for file distribution.

Distribution Keywords

Ack= Yes | Msg | No | None

Defines the default value of the "ACK/NOACK" distribution option for the corresponding list, that is, the value assigned to new users when they subscribe to the list. This value can be altered by subscribers ("**SET**" command), but not by users who are not signed on to the list. This means that this option will always be in effect when distributing mail from people who are not on the distribution list. The default is "**`Ack= No`**".

Yes	A short acknowledgment with statistical information on the mailing will be sent back to you.
Msg	Messages will be sent when your mail file is being processed. Statistical information will be sent via messages, but no acknowledgment mail will be sent. [BITNET only]
No	For Internet users, no acknowledgement will be sent. For BITNET users, a single interactive message will be sent as the message is processed. This is the default value.
None	No messages of any kind are sent when your mail file is processed. [same as No for non-BITNET]

Daily-Threshold= *nnn1[,nnn2]*

This keyword limits the number of postings that may be processed by the list in a calendar day (midnight to midnight, server time), and, with the addition of a (new) optional second parameter, limits the number of postings that may be accepted from any individual user per calendar day (midnight to midnight in the server's local time zone).

The default is **`Daily-Threshold= 50`**. When the value of the first parameter is reached, the list is automatically placed on hold, and the list owner or LISTSERV maintainer must issue the **FREE** *listname* command. Note that it may or may not be advisable to increase this parameter for higher-volume lists – individual list owners should study the issue carefully before increasing the daily threshold of their high-volume lists.

When the value of the optional second parameter is reached by an individual user, the user is told that their posting will not be processed and that they should resend it later if they still want it to be posted. The list itself is not held in this situation. The default is to have no such limit, in which case the second parameter is not defined. Note that list owners and list editors are exempt from the individual daily limit. There is no command to reset the limit for an individual user, although the list owner may update the header to increase the value.

Digest= No
Digest= Yes,*where* | Same[,*frequency*][,*when*][,Size(*maxsize*)][,BOTTOM_BANNER]

This keyword controls the automatic digestification function allowing subscribers who do not have the time to read large numbers of messages as they arrive to subscribe to a digestified or indexed version of the list. The list owner decides whether digests are available or not, the frequency at which they are issued and the day of week or time of day when the digest should be distributed.[15]

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

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

over a certain period of time. Unlike real-world digests, LISTSERV digests are not edited; what you see is exactly what was posted to the list. The only difference is that you get all the messages for a given day, week or month in a single batch. This is mostly useful if you are just "listening in" to the list and prefer to read the postings at your leisure. Digests are kept separately from list archives and can be made available for mailing lists which do not archive postings (i.e. which run with "**Notebook= No**").

Indexes, on the other hand, only provide a few lines of information for each posting: date and time, number of lines, name and address of poster, subject. The actual text is not included. You select just the messages you are interested in, and order them from the server. This is useful for mailing lists where most messages really don't interest you at all, or as an alternative to **SET NOMAIL**: when you come back from vacations, you can quickly order the messages you are most interested in. Note that, since indexes are not useful without the ability to order a copy of the messages you do want to read, they are not made available unless the list is archived and digests are enabled.

Users sign up for digestified rather than immediate delivery with '**SET *listname* DIGests**', while indexes are selected with '**SET *listname* INDex**'. These two options are alternatives to **MAIL** and **NOMAIL**. When switching around between these delivery options, users will observe the following behavior (digests will be assumed to be daily for the sake of clarity):

- When switching to **NOMAIL**: delivery stops immediately. The day's digest is not sent, as the user is assumed to desire immediate termination of traffic from the list. (Note that any mail already "in the pipeline" to the subscriber will still be delivered.)

- When switching from any option to **DIGEST** or **INDEX**: mail delivery stops immediately, and the first index or digest may contain some items the user has already seen (if switching from **MAIL** to **DIGEST/INDEX**). This is because the digests and indexes are global to the list - they are the same for everyone, just like regular issues of newspapers.

- When switching from **DIGEST** or **INDEX** to **NODIGEST** or **NOINDEX**, the current, unfinished digest or index is immediately mailed to the user. New messages are delivered normally, as they arrive. Thus, a "trick" to get a copy of the current digest is to switch to **NODIGEST** and then back to **DIGEST**. You can send both commands in the same mail message to make sure they are executed together.

The list owner controls the availability and frequency of digests through the "**Digest=**" list header keyword, which defaults to "**Digest= No**" for lists without an archive and "**Digest= Yes,Same,Daily**" for archived lists. Again, it is not necessary for the list to be archived to keep a digest; LISTSERV just attempts to avoid having to store large amounts of digest data on its private area for lists which, lacking a "**Notebook= Yes,xxx**" keyword, do not specify any suitable directory for the digest data. Conversely, having daily as the default frequency keeps the additional cost in disk space to a minimum.

The syntax of the keyword is "**Digest= Yes,*where*,*frequency*,*when*,*maxsize***" when digests are enabled, or then "**Digest= No**". The second parameter is a disk or directory specification, just as with the "**Notebook=**" keyword, or "**Same**", which means that the digest must be

stored on the same disk as the list archives.

The third parameter is either "**Daily**" (the default), "**Weekly**" or "**Monthly**".

The fourth parameter is optional and specifies when the digest is to be actually distributed. For daily digests, specify 'hh:mm' or just 'hh' in the usual 00-23 scale (24 is also accepted for midnight). Note that daily digests are cut on the hour, regardless of whether or not you have provided ":mm" in your setting. For weekly digests, specify a weekday such as "Tuesday". For monthly digests, you may specify a number from 1 to 31 corresponding to the day of the month when the digest will be distributed, although this is not recommended. The purpose here is to make it possible for digests to be issued at mid-month rather than on the first of the month - if you code a number larger than 28, you may not get a digest every month.

The fifth parameter is also optional. It takes the form "**Size(*number*)**" and specifies the maximum number of lines the digest is allowed to reach before a "special issue" is cut. (Note that your digests may run over the limit set in "**Size(*number*)**". This is because LISTSERV will never truncate a message in order to meet the digest size limit. Thus, if you've reached 950 lines of your 1000 line setting and the next message is 100 lines long, your digest will cut at 1050 lines.) Bear in mind that most unix systems do not accept messages larger than 100 kilobytes, so values larger than 1500 should be avoided. The default is to have virtually no limit - 10,000 lines.

In LISTSERV 14.3 and following, you may code a Size() parameter in kilobytes or megabytes rather than in lines. For instance,

* Digest= Yes,Same,Daily,Size(100K)	Cut the digest at 100Kbytes
* Digest= Yes,Same,Daily,Size(1M)	Cut the digest at 1Mbyte

As before, the digest will be cut whenever the pending digest grows over the size limit setting.

The list owner must take special care when disabling digests for a list, as LISTSERV does not presently have any facility which would allow it to alter subscription options automatically on the basis of changes to the list header. Subscribers who had opted for digests would continue not to receive mail as it arrives, but would not get the digests either. The best way to solve this problem is to announce the change long enough in advance, so that people can switch back before digests are suspended. The reason nothing has been done to remove this limitation is that it is not expected to be a frequent condition. Daily digests take up very little disk space and there is no reason to disable them for a typical list.

(1.8c and 1.8d only) The default behavior of a list with a **BOTTOM_BANNER** template defined in ***listname*.MAILTPL** is to suppress the banner throughout the digest and print it only once at the beginning, between the list of topics and the first message in the digest. This behavior can be disabled so that the banner is printed in its normal position at the end of each message in the digest by adding the **BOTTOM_BANNER** parameter to the **Digest=** keyword. Evaluators should note that this behavior is also standard on evaluation copies, with the difference that the evaluation kit banner cannot be turned off. L-Soft does not expect that this parameter will be much used, but it is included for the sake of completeness.

(1.8e and following) Bottom banners are no longer suppressed in individual messages when **BOTTOM_BANNER** is specified. The only use of the **BOTTOM_BANNER** parameter in 1.8e and following is to force a copy of the banner to be printed at the top of the digest as in previous versions.

Note that **TOP_BANNER**s are always included at the top of each message in the digest. Generally, **TOP_BANNER** contains copyright or other important information that should be included with each message, and therefore it is not suppressed.

The second parameter of the "**Digest=**" keyword ("***where***") may only be changed by the LISTSERV maintainer. A list owner is allowed to change "**Digest= No**" to "**Digest= Yes,Same...**", but any other specification for the digest file location will cause an error. A list owner is also allowed to change "**Digest= Yes...**" to "**Digest= No**" without the intervention of the LISTSERV maintainer. Note that if the list is not archived ("**Notebook= No**"), changing "**Digest= No**" to "**Digest= Yes,Same**" will cause the digest files to be written to LISTSERV's A disk (or equivalent specification on the workstation systems). Since the overhead for a typical digest is small, it is not expected that this will cause any problem for the LISTSERV maintainer.

Internet-Via= *internet-hostname*

This keyword is not available in LISTSERV Lite.

There is no default value. This parameter determines whether or not mail bound for Internet addresses is routed through a specific Internet gateway. In principle this keyword should never need to be set on non-BITNET hosts.

Mail-Via= DISTRIBUTE | DIST2 | Direct

This keyword is not available in LISTSERV Lite.

The default value is **Mail-Via= DISTRIBUTE**. DIST2 is functionally equivalent to DISTRIBUTE, and is included for historical reasons.

The Mail-Via keyword should generally not be set, except by the site administrator for troubleshooting mail delivery problems.

On "Networked" and "Tableless" LISTSERV sites, **Mail-Via= DISTRIBUTE** causes mail to go out over the DISTRIBUTE backbone to spread the delivery load over the LISTSERV Network. **Mail-Via= Direct** causes LISTSERV to send all mail through the local SMTP server.

On "Standalone" LISTSERV sites, it has no practical use.

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

This keyword is not available in LISTSERV Lite.

This keyword defines the RFC822 "Newsgroups:" header for a list. This field may be required by certain news gatewaying software and should only be defined if the list is gatewayed to usenet and if the gatewaying software does require it. The default is **Newsgroups= None**.

A typical setting for this keyword might be:

```
* Newsgroups= bit.listserv.lstown-l
```

NJE-Via= ***internet-hostname***

This keyword is not available in LISTSERV Lite.

There is no default value. This parameter determines whether or not mail bound for NJE addresses is routed through a specific gateway. In principle this keyword should never need to be set on non-BITNET hosts.

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

This keyword is not available in LISTSERV Lite.

Determines whether or not mail for the list is processed during "prime time", a value that is determined by the LISTSERV maintainer and is kept in the system configuration file. The default is "**Prime= Yes**", which means that LISTSERV will allow postings to be processed during prime time. You can also explicitly code "**Prime= No**", which means that LISTSERV will use the value in its **PRIMETIME** site configuration variable to determine "prime time" for the list.

This keyword can be most useful in controlling the load on the machine running LISTSERV. "Prime=" may also be set to an explicit time specification, for example,

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

or (for a once-weekly newsletter issued on Wednesday, perhaps)

```
Prime= "MON-TUE: 00:00-23:59; WED: -; THU-SUN: 00:00-23:59"
```

Note that the time specification for Prime= must *always* be surrounded by double quotes (**""**). Otherwise LISTSERV will stop reading the specification at the first space (ASCII 32) it encounters. For example, the above example coded without quotes would be interpreted as **Prime= MON-FRI:** with the balance of the string ignored. LISTSERV will not issue an error if you omit the quotes.

Note also that when you are coding a prime time specification that LISTSERV's week starts on Monday and runs through Sunday. Thus something like the following examples:

```
Prime= "MON-TUE: 00:00-23:59; WED: -; THU-SUN: 00:00-23:59"

Prime= "TUE: 01:00-4:59; THU-SUN: 00:00-23:59"
```

are correct syntax, whereas

```
Prime= "WED: -; THU-SUN: 00:00-23:59; MON-TUE: 00:00-23:59"
```

is not. Furthermore note carefully the weekdays must be specified in their correct order, that is,

```
Prime= "THU-FRI: 00:00-23:59; SAT-MON: 21:00-23:59"
```

is not correct because it starts on Thursday and ends on Monday The correct specification in this case would be

```
Prime= "MON: 21:00-23:59; THU-FRI: 00:00-23:59; SAT-SUN: 21:00-23:59"
```

IMPORTANT: When you set a "prime time" either for a list or globally for the entire server ("**PRIMETIME=**" in the site configuration file), you are setting the time during which LISTSERV does **not** process postings. It is "prime time" for the machine when it should be doing other things, for example, number crunching, daily backups, or any other function during which LISTSERV should not be using cycles.

Note also that the minutes specification is cosmetic only. LISTSERV checks on jobs held awaiting non-prime time only once each hour, on the hour. Thus if you have

```
* Prime= "MON-SUN: 06:00-21:00"
```

then jobs awaiting non-prime time will be executed at 22:00, not 21:00 as you might otherwise expect. On the other hand, if you code

```
* Prime= "MON-SUN: 06:00-20:xx"
```

where "***xx***" is any two-digit integer between 01 and 59, then jobs awaiting non-prime time will be executed when LISTSERV runs its hourly check of PRIME jobs at 21:00.

If you need to open only one short window during one or more days, you can do this by coding something like:

```
* Prime= "MON-FRI: 00:00-02:59 04:00-23:59; SAT-SUN: -"
```

This example allows LISTSERV to process mail for the list only between 2 AM and 4 AM Monday through Friday, and at any time on Saturday and Sunday. Note that there is no punctuation--just a space--between the time settings for a given day or day sequence.

Reply-To= List|Sender|Both|None|*net-address*,[Respect|Ignore]

Indicates whether the "Reply-to:" tag supplied by the sender of the mail file is to be preserved or discarded (if present), and, if discarded or omitted, what should be placed in the new "Reply-to:" generated by the server. The default value is "**`Reply-To= List,Respect`**".

Note that some mailing systems are unable to process a "Reply-To:" field with multiple addresses correctly and may therefore disregard the **`Reply-to= Both`** option and treat it as **`Reply-to= List`**.

PLEASE NOTE CAREFULLY: Setting this parameter guarantees only one thing -- that LISTSERV will generate an appropriate RFC822 Reply-To: header in the mail it distributes to subscribers. THERE IS UNFORTUNATELY NO GUARANTEE THAT THE MAIL TRANSFER AGENT (MTA) OR MAIL CLIENT ON THE RECEIVING END WILL HONOR THE Reply-To: HEADER. This is because some mail clients, out-of-office robots, and Internet MTAs either simply do not recognize the existence of Reply-To: or do not implement it properly. Specifically RFC2076 "Common Internet Message Headers" reports that the use of Reply-To: is "controversial", that is, "The meaning and usage of this header is

controversial, meaning that different implementors have chosen to implement the header in different ways. Because of this, such headers should be handled with caution and understanding of the different possible interpretations." (RFC2076, page 4). While L-Soft recognizes that it is sometimes important to provide an explicit Reply-To: header to indicate a response path, L-Soft cannot be held responsible for problems arising from the inability of a remote server to properly process Reply-To: headers.

The two parameters are specified as follows:

First parameter:

List	Replies are directed to the list address.
Sender	Replies are directed to the original sender.
Both	Reply to both the original sender *and* to the list (see note regarding this above)
None	No Reply-to: header is generated unless ",Respect" is specified and a Reply-to: header is present in the original posting, in which case replies are directed to wherever the Reply-To: tag points.
net-address	Replies are directed to the specified internet address

Second parameter:

Respect	The original "Reply-to:" tag on the posting, if any, is kept. If no valid Reply-To: tag exists in the posting, the value defined in the first parameter of this keyword is used.
Ignore	The original "Reply-to:" tag on the posting, if any, is ignored and discarded, and the value defined in the first parameter of this keyword is used instead. If **`Reply-To= None,Ignore`**, then a Reply-to: tag will never be generated by LISTSERV.

Sender= List | None
Sender= "list title <*net-address*>",*ietf-address*

This keyword is not available in LISTSERV Lite.

NOTE CAREFULLY: Setting this value DOES NOT change the RFC822 "From:" header. Per standard, LISTSERV is not allowed to change the From: header, but must pass it through unchanged.

Used to define the value LISTSERV will place in the RFC822 "Sender:" field. The second parameter is optional, and is included to allow the specification of a second mailbox for use with IETF headers. The first value is used for non-IETF headers and is expected to contain the name and address of the list, or the keywords LIST or NONE. The second mailbox is used for IETF headers; if it is omitted, the generic "owner-listname" mailbox is substituted. Example:

```
* Sender= "Test List <TEST@LISTSERV.X.EDU>",owner-test@listserv.x.edu
```

Note that the first address must be contained in quotes.

Sub-lists= *sublist1,sublist2...sublistn*

This feature and keyword are not available in LISTSERV Lite.

This keyword first appeared in 1.8c and makes it possible to define a "super-list"

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

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

or (if you want to put each sublist on a separate line):

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

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

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

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

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

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

been removed in LISTSERV 1.8e.

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

Topics= ***topic1*****,*****topic2*****,...*****topic23***

This feature and keyword are not available in LISTSERV Lite.

List topics provide a way to run a mailing list (preferably moderated) where several sub-topics are being discussed in parallel but some subscribers are only interested in a subset of the topics. For instance, a working group might have general discussions, decisions, and messages related to meetings. People who cannot attend the meetings can then opt out of last calls for hotel reservations and discussions about seafood restaurants, whereas people who have no time to follow the discussions can elect to get just the decisions. At any rate, such a compartmented list requires a certain discipline in order to be successful, as the posters must label their messages to indicate which topic(s) they belong to.

Through the **Topics=** keyword, the list owner can define up to 23 topics for the list (note that 1.8c and earlier are limited to 11 topics). For instance, the list owner could code:

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

If necessary, you may use multiple **Topics=** lines in your header in order to fit all of your topics in.

WARNING - YOU MUST NEVER REORDER THE TOPICS= KEYWORD(S)

To save disk space, LISTSERV remembers which topics users have selected through their ordering in the "Topics=" keyword. That is, "News" is "topic number 1" for LISTSERV, "Benchmarks" is "topic number 2", and so on. This means you can change the name of a topic without requiring users to alter their subscriptions (for instance, you could decide that "Tests" is a better name than "Beta-tests" and just make the change). However, you must never change the order of the topics in the "Topics=" keyword. If you want to remove a topic, replace it with a comma. For instance, to remove the "Meetings" topic, you would change the keyword to:

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

This restriction might be removed in a future release.

Topic names can contain any character except space, colon and comma; the use of double quotes or equal signs is discouraged, as they require special attention when coding list header keywords. In addition, topic names may not start with a plus or minus sign, and the words ALL, NONE, RE, OTHER and OTHERS are reserved.

Posters label their messages through the subject field. LISTSERV first skips any

possible sequence of 'Re:' keywords, and takes anything to the left of a colon as a list of topics, separated by commas. The posting is considered to belong to all the topics listed before the colon. If none of these topics is valid for the list, it is classified in a special topic, "Other". If some of the topics are valid but others are undefined, the invalid ones are ignored. At any rate the subject field is left unchanged. Here is an example:

Subject: Benchmarks,News: Benchmarks for XYZ now available!

Messages which should be read by everyone can be posted to the special topic "All". Topic names can be shortened to any unambiguous abbreviation. In our example, "Be" is ambiguous because it could be either "Beta-tests" or "Benchmarks", but "Bench" is acceptable.

Subscribers select the topics they wish to receive with the SET command. The syntax is 'SET listname TOPICS: xxx' where 'xxx' can be:

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

- Updates to the list of topics the user currently receives. A plus sign indicates a topic that should be added, a minus sign requests the removal of a topic. For instance, 'SET XYZ-L TOPICS: +NEWS -BENCH' adds news and removes benchmarks. If a topic name is given without a + or - sign, + is assumed: 'SET XYZ-L TOPICS: +NEWS BENCH' adds news and benchmarks. The first topic name must have the plus sign to show that this is an addition, and not a replacement.

- A combination of the above, mostly useful to enable all but a few topics: 'SET XYZ-L TOPICS: ALL -MEETINGS'.

The colon after the keyword TOPICS: is optional, and TOPICS= is also accepted. Do not forget to include the special OTHER topic if you want to receive general discussions which were not labeled properly. On the other hand, if you only want to receive properly labeled messages you should not include it. ALL does include OTHER.

Finally, it is important to note that topics are active only when your subscription is set to MAIL. Digests are indexes always contain all the postings that were made, because the same digest is prepared and sent to all the subscribers.

(See also **Default-Topics**.)

Error Handling Keywords

Auto-Delete= No
Yes,Semi-Auto[,Delay(*number*)][,Max(*number*)][,Probe(*number*)]
Yes,Full-Auto[,Delay(*number*)][,Max(*number*)][,Probe(*number*)]
Yes,Manual[,Delay(*number*)][,Max(*number*)][,Probe(*number*)]

This keyword is available in LISTSERV Lite, but is not full-featured. The behavior in LISTSERV Lite with Auto-Delete= Yes is Auto-Delete= Yes,Semi-Auto,Delay(0),Max(1). Any other settings are ignored. Specifically the passive probing option available in Classic is disabled in Lite.

LISTSERV includes support for automatic deletion of users whose account has expired or whose system has permanently disconnected. When the delivery error is generated by LMail (any version), MX V3.2 or higher, PMDF V4.2 or higher, or LSMTP(TM) , which all implement the same delivery error format, LISTSERV may be able to automatically process the delivery error and take action based on the value of the "Auto-Delete=" list header keyword. The unix versions of LISTSERV also support sendmail's delivery error format. The auto-deletion code is also fully compatible with RFC1893 "Notary" format error codes

If the list has been coded "Auto-Delete= No", or if the delivery error is not in LMail format and LISTSERV cannot understand it, LISTSERV simply passes it to the list owner. Otherwise LISTSERV processes the message automatically. The algorithm may be refined in a future version, but at present the following steps are taken whenever the auto-deletion feature is enabled:

When auto-deletion is set to "Full-Auto" or "Semi-Auto":

- LISTSERV looks for "permanent" errors (no such user, no such host, and so on). If the failing recipients are subscribed to the list, LISTSERV removes them and notifies you. No action is required from the list owner.

- If there are permanent errors for users LISTSERV could not find on the list (for instance because the account subscribed to the list is a totally different one which forwards mail to a dead account), or if there are only "temporary" errors (host unreachable for 3 days, system error, disk quota exceeded, and so on), LISTSERV passes the actual error message to the list owner for further disposition if running in semi-auto mode. If running in full-auto mode, the error messages themselves are discarded and the errors only show up as entries in the daily auto-deletion monitoring report.

- When running in full-auto mode, LISTSERV never passes back a delivery error unless it took action on it. This means that certain errors may remain unsolved, as LISTSERV presently ignores temporary errors and some of them are virtually permanent (if the owner of the account has left but for some reason his account was not closed, his disk quota is bound to remain exceeded until someone takes action). Full-auto mode should be used only when the list owner positively does not have the time to handle the delivery errors LISTSERV sends every day.

When auto-deletion is set to "Manual":

- When running in manual mode, the auto-delete monitor informs the list owner of the error(s) and takes no further action on delivery errors.

Some considerations for configuring the auto-delete monitor parameters:

- Setting the Delay(number) option. The default is 4. This is the number of days that a subscriber's mail needs to bounce before he's automatically deleted. If "Delay(0)" is coded, LISTSERV won't wait, but will delete on the first bounce.

  Most delivery errors occur on weekends when systems are taken down for maintenance, system administrators are not around to reboot after crashes, and the like. Because of this, most delivery errors only last for 2-3 days and may not be "permanent" even if they seem to be at first.

  The nature of delivery errors is such that LISTSERV has no way to establish that a problem has been fixed because it receives only negative acknowledgements when a message bounces. This taken together with the transient, "weekend" nature of most delivery errors indicates that it is not a good idea to set Delay() to a value close to 7. For instance, if Delay(7) and a subscriber's mail regularly bounces on the weekend, LISTSERV will wait until the next weekend to decide whether or not to delete him, at which point the subscriber will bounce mail again and start the process all over. The bottom line is that LISTSERV might as well have gone ahead and deleted the subscriber as soon as the first bounce occurred.

- Setting the Max(number) option. To prevent auto-deletion monitoring from getting out of hand, subscribers are deleted after a specified number of errors regardless of how many days it has been going on. The default is Max(100). This is so LISTSERV won't spend its life monitoring 50 bogus users x 100 messages = 5000 a day. Note that if Delay(0), the setting for Max() is ignored (in effect it is set to Max(1)).

- Setting the Probe(number) option. This parameter tunes the "passive probing" option (beginning with 1.8d). Passive probing operates by turning a certain percentage of your regular list messages into transparent probes that look like a normal message but also double as a probe, rather than sending out the explicit **PROBE1** template as in active probing. You enable (or tune) passive probing by adding a "**,Probe(xx)**" parameter to the **Auto-Delete=** keyword setting. For instance,

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

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

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

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

  If you have users who for whatever reason should not be probed, you can deactivate passive probing (and any other renewal you have set for the list) with the **SET *userid@host* NORENEW** command. The default for this parameter is Probe(30) for lists up to ~2K subscribers, and Probe(0) for

larger lists (because by its nature, probing can be a non-inconsiderable performance hit).

For more information on passive probes, see chapter 13.5.2 of the *Site Manager's Operations Manual*.

- When you take a vacation, note that it is best to switch auto-delete to MANUAL. Then do not restore to auto on the day you come back, because you will have a number of subscribers on file ready to be deleted. Wait DELAY+n days before changing back to Full-Auto or Semi-Auto, where n is an adjustment to account for the fact that people don't fix all problems right away at 09.00 on the day your vacation ends. n=2 is a reasonable choice.

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

Note that if you have coded "Delay(0)" and/or "Max(0)", LISTSERV simply deletes any error-generating subscriber it can (generally 95-98%), discards any further errors it does not understand, and does not generate daily monitoring reports. If you want the daily monitoring reports you must code at least "Delay(1),Max(1)".

Errors-To= *mon-address1,mon-address2,...*

Defines the person or list of persons that are to receive rejection mail for the list. The default value is 'Owners'.

In LISTSERV 1.8e and following, the internet address of the list is explicitly disallowed as an error-receiving address, and attempting to set Errors-To= to the internet address of the list will raise an error. The list should never be configuired to receive its own errors as this is guaranteed to cause looping.

It should be carefully noted that there is no way to automatically discard errors without sending them to some address. "Errors-To= No" and "Errors-To= None" are both invalid settings and will cause LISTSERV to revert to the default.

Loopcheck= Normal
Loopcheck= Full
Loopcheck= None[,Allow-Bounces]
Loopcheck= *option1*[,*option2*][,*optionn*]

This keyword is not available in LISTSERV Lite.

Determines the type of loop checking performed by LISTSERV to avoid perpetuating mail loops. The default is "**`Loopcheck= Full`**" prior to LISTSERV 14.5, and "**`Loopcheck= Normal`**" starting with LISTSERV 14.5. Loop checking is configured on a list by list basis only.

ALWAYS USE THIS KEYWORD WITH CAUTION!
Misuse of this keyword can and will allow mailing loops onto your list!

The various **Loopcheck=** parameters are defined as follows. "Normal", "Full" and "None" must be specified alone (except in the special "None,Allow-Bounces" case, and except for "Spam-Delay(n)"), whereas the other options may be specified together as required.

Normal	LISTSERV uses its "normal" suite of loop checking heuristics. Tests considered to be obsolete are not conducted. This is the default for Loopcheck= beginning with LISTSERV 14.5. See the LISTSERV 14.5 release notes for more details.
Full	LISTSERV uses its full suite of loop checking heuristics to check incoming mail for loops. This includes tests considered to be obsolete.
None	All LISTSERV loop checking and "command to list" checking is disabled for this list. WARNING: "None" tells LISTSERV that, by definition, anything that reaches its reader is NOT a delivery error. It is *never* a good idea to use this parameter except in special cases where a bug is suspected in the loop checking heuristics. Generally this parameter should not be used without checking with L-Soft first, and only for the diagnosis of an existing problem.
None,Allow-Bounces	From LISTSERV 14.2 (1.8e-2003a), allows the list to accept error messages that would normally be bounced to the list owner. This makes it possible to direct the Errors-To= keyword setting to a LISTSERV list rather than to a specific person. "Allow-Bounces" MUST be specified in conjunction with "None". Specifying just "Loopcheck= Allow-Bounces" will result in a syntax error when the list header is stored.
Noorigin	Allows the list owner to disable the check for "known mailer origins" such as MAILER, POSTMASTER, ROOT, UUCP, et al. Mail whose 'From:' field is the address of the local mailer is still trapped, but wildcard checks on the mail origin are disabled.
Nobody	Allows the list owner to disable the check for identical text in the body of incoming mail only. LISTSERV relies only on the Subject: field of the mail message to determine whether or not mail is looping. This is a very dangerous option: it means that any mailer not using one of the "standard" subjects known to LISTSERV will cause a loop.
NoCRC	Allows the list owner to disable CRC (cyclical redundancy check) check of incoming mail. CRC loop checking calculates a "checksum" based on the contents of the mail message and compares it to other incoming mail to spot duplicates.
NoSpam	Allows the list owner to disable the anti-spamming filters to incoming mail.
Spam-Delay(n)	Allows the list owner to modify the number of minutes LISTSERV holds mail from non-subscribers before releasing it to the list. The assumption is that, within n minutes, a spam alert may or may not arrive regarding non-subscriber mail. The list owner can disable this function for his list by coding "**`Loopcheck= Spam-Delay(0)`**", or can tune it to his preference by simply specifying the number of minutes for LISTSERV to hold the mail. The default is 10 minutes, or "**`Spam-Delay(10)`**".

Please note carefully that L-Soft does not recommend changing **Loopcheck=** from the default value unless you are prepared to accept the very likely possibility of a mail loop occuring on your list; a situation for which L-Soft would not and could not be held responsible for. The only exception would be the "**Loopcheck= NoSpam**" (which might be necessary to keep adminstrative mail to multiple lists on a single host from triggering the anti-spamming filter) or "**Loopcheck= Spam-Delay(*n*)**" options, neither of which stops canonical mail loops *per se*.

See also **Filter=** and **Safe=**.

Safe= Yes | No

The list header keyword, "**Safe=**", controls the e-mail address LISTSERV places in the SMTP MAIL FROM: field, which is where well-behaved mailers will return delivery errors. With "**Safe= No**", these errors are sent to the list address as before, hopefully to be intercepted by the loop detector and passed on to the list owner. With "**Safe= Yes**", the error address is set to 'owner-listname', and delivery errors sent to that address are passed on to the list owner without the risk of creating a mailing loop. The default is "**Safe= Yes**".

IMPORTANT: The use of "**Safe= Yes**" does not guarantee that all errors will go to the 'owner-listname' mailbox. Unfortunately, there are many non-compliant mailers which will continue to send the error back to the list (usually because it is listed in the 'Reply-To:' or 'Sender:' field). Using "**Safe= Yes**" significantly decreases the potential for mailing loops, but not enough to actually code "**Loopcheck= No**", unless you are sure that all your subscribers have compliant mailers.

See also **Filter=** and **Loopcheck=**.

List Maintenance and Moderation Keywords

Configuration-Owner= *conf_owner[, conf_owner2][,...]*

This feature and keyword are not available in LISTSERV Lite, and can only be set by the LISTSERV maintainer.

This new (optional) keyword defines which of the list owners defined in the Owner= keyword setting are authorized to change the list configuration. It is a LISTSERV ACL, much like "Review=", or "Send=" (if you disregard the special options that are specific to "Send="). The default value is "Configuration-Owner= Owner", which allows all addresses and access-levels listed in Owner= to modify the list header.

Note carefully that list owners who are not Configuration-Owners retain the right to change templates, add and delete users, change user options, GET and PUT archive files, and so forth. The setting of Configuration-Owner for a given list restricts only the ability of non-Configuration-Owners to change the list header.

The definition of a *conf_owner* is a sub-set of *access_level*. Valid *conf_owners* are

Owner
Postmaster
net-address (for instance, user@host.com)
Owner(*listname*)
(*listname*)

Examples:

```
Configuration-Owner= Postmaster,(configownerlist)
Configuration-Owner= Owner,Owner(otherlist),joe@example.com
Configuration-Owner= (configownerlist)
Configuration-Owner= sarah@example.com,joe@example.com
```

The purpose of this keyword is to make it easier for LISTSERV administrators to manage non-technical list owners. It allows the LISTSERV maintainer to create a list where the owner can manage subscriptions normally, but cannot make changes to the list configuration.

It is also possible with this keyword to create a separate mailing list of LISTSERV users who do not have full LISTSERV maintainer privileges, but can manage lists other than those they might own explicitly on the same server.

Please note carefully that any addresses specified in Configuration-Owner= must also be specified (either explicitly or as part of an access-level definition) in the Owner= setting for the list.

Editor= *net-address1,net-address2|access-level1,...*

Defines the list editor(s). When used in conjunction with the "**`Send=Editor`**" option, it causes all mail sent to the list to be automatically forwarded to the first person listed in the "**`Editor=`**" keyword, who will then send it back to the list at his discretion. The editors are the only persons (with the list owners) who are allowed to mail directly to the list. Note that ANY editor can send mail to the list while only the FIRST one will receive copies of mail sent to the list (but see also **`Moderator=`**).

The file will be forwarded to the editor 'as is', without being included in a mail envelope. This method makes sure that the original "Resent-" tags (if any) and "To:" keyword are preserved.

Note that the first editor *MUST* be a network address (e.g., **someuser@foo.bar.com**) and not an *access-level*. Subsequent editors may be *access-levels*. For instance, you can code

```
* Editor= joe@baz.net,(MYLIST-L)
```

which allows all subscribers from the MYLIST-L list to post without going through the editor, and diverts all non-subscriber mail to joe@baz.net for approval.

IMPORTANT NOTE: The first editor SHOULD 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.

ALSO PLEASE NOTE: In many support cases it has been noted that lists are often coded with "Editor= Owner" or "Editor= Owners". Prior to LISTSERV 14.3, this would have caused LISTSERV to generate approval request messages to an address such as OWNER@.BITNET, which of course would not be deliverable. LISTSERV 14.3 and later will now avoid this problem and when encountering either of these cases will default to sending the approval request to the first listed non-quiet list owner. L-Soft continues to recommend NOT using either "Owner" or "Owners" as the primary editor of the list, as it is not always the case that the list owner should be the primary (or only) editor.

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

Editor-Header= Yes | No

This keyword is not available in LISTSERV Lite.

If an editor is defined (see **Editor=**), this keyword determines whether or not special header information is prepended to list messages forwarded to the editor. The default (for lists configured with an Editor) is "**Editor-Header= Yes**".

Note that **Editor-Header= No** is ignored if you have **Send= Editor,Hold** or **Send= Editor,Hold,Confirm** . In these cases the editor-header information is required so as to provide the confirmation code for the OK command.

List-Address= *name_info*[@*host_info*]

This keyword is not available in LISTSERV Lite.

This keyword determines how LISTSERV announces its list address in the

header of messages delivered to the list: NJE vs. Internet address, short vs. long list name, etc. The default options (when neither "**List-Address=**" or **LIST_ADDRESS** are defined) are long list name and Internet address. A corresponding **LIST_ADDRESS** configuration option must be added to the LISTSERV site configuration file.

It is important to note that the only effect of the "**List-Address=**" keyword is to change the way the list identifies itself in list postings, command replies, etc. It does not instruct the mail system to create forwarding entries to support the new name, nor does it establish the specified name as an alias for the list (use "**List-ID=**" for this purpose). In general list owners should not use this keyword without first consulting with the LISTSERV maintainer.

In 1.8b and earlier versions, the first token (*name_info*) can be either **LISTNAME** or **LIST-ID**. Do not attempt to specify the actual list name. Use **LISTNAME** if you want LISTSERV to use the "short" list name (always available), and **LIST-ID** if you would rather see the "long" list name ("**List-ID=**" keyword). If there is no "long" name, the short name is substituted.

Version 1.8c introduced the ability to specify the name of the list in the first token (i.e., you may now specify something like "**List-Address= XYZ-L@XYZ.EDU**").

The second token (*host_info*) can be either **NJE**, **FQDN**, or the fully qualified domain name of your choice. That is, you may type the actual hostname that you want LISTSERV to use, which may be useful if the machine on which LISTSERV is running is known under several hostnames.

If you only want to override one of the two parts of the list address, you do not need to specify the other. For instance, if you only want to change the hostname, you can enter "**List-Address= XYZ.EDU**" in the list header and let the left-hand part default from the value of the system default in the LISTSERV 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.
- Starting with version 1.8c, "**List-Address= XYZ-L@XYZ.EDU**" is also valid. You no longer are restricted to specifying **LISTNAME** or **LIST-ID** for the left-hand (username) part.

List-ID= ***longlistname***

This keyword is not available in LISTSERV Lite, and is technically obsolete on all ports of the software except for VM.

On VM systems, this keyword allows the list owner to specify a long list ID in addition to the normal 8-character list name. This is particularly useful for peered or gatewayed lists that have names longer than 8 characters. On non-VM systems, if the normal list name is longer than 8 characters and the list is being migrated from a VM system, it may be a good idea to specify the first 8 characters of the list name in this keyword, at least temporarily. This way subscribers who were used to the old 8-character name can continue to use it on the new system.

Non-VM systems may use this keyword for aliasing. However, today there is really no good reason to use this keyword on non-VM systems, as it is possible to define lists on such systems with native file system names longer than 8 characters. L-Soft's recommendation is that this keyword be used only if you are migrating a list from VM that was known by both its "short" name and its "long" **List-ID=** name. (On unix you can avoid this by simply specifying an extra set of aliases in **/etc/aliases** for the "long" name that point to the same places as do the ones for the "short" name.)

In any case a list owner should not set a value for **List-ID=** without first consulting with the LISTSERV maintainer, since it will be necessary to add appropriate system mailer aliases before the name specified in **List-ID=** will work.

List-ID= will not work properly on NT systems running with the SMTPL "listener" because the "listener" has no way to know that the list ID specified in this parameter is a valid local address.

List-ID= *will* work on NT and VMS systems running LSMTP, but you must first configure a route in LSMTP for the **List-ID=** name so that LSMTP will know to deliver mail addressed to the **List-ID=** address to LISTSERV (as opposed to POP or SMTP, etc.).

Under VMS and unix, it is necessary to add the appropriate aliases to the mailer's aliases file in order for **List-ID=** to work, since mailers such as sendmail and PMDF otherwise have no way to know that the name specified in **List-ID=** is a valid address. This means that lists that have the **List-ID=** keyword specified need two complete sets of aliases defined (unless **List-ID=** is identical to the list name, in which case it should not be implemented to begin with).

Starting with LISTSERV 1.8d, if you do use **List-ID=** to specify a "long" name for a list with web archives, LISTSERV will create an HTML page for both the long and short names. Here again, however, on non-VM systems L-Soft does not recommend the use of **List-ID=** .

Moderator= [All,]*netaddress1*[,*netaddress2*]...

This keyword is not available in LISTSERV Lite.

This keyword defines which editors of a moderated list receive postings for forwarding to the list. The default is the first editor as defined by the "**Editor=**" keyword. If multiple moderators are defined, the load is spread across them.

Note that all editors may still post directly to the list, but only those editors defined by "**Moderator=**" will have messages from non-editors forwarded to

them.

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

```
* Moderator= All,joe@somehost.com,jill@someplace.net
```

Note that this could also be coded as:

```
* Moderator= All,joe@somehost.com
* Moderator= jill@someplace.net
```

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

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

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

New-List= *net-address*

This keyword is not available in LISTSERV Lite.

When a list is moved to a different LISTSERV host, this keyword can be added to the list header left on the original host. This facilitates forwarding of administrative commands and postings from the original host to the new host. Users posting to the old address will also receive a short note in return listing the new address.

Note that success in setting the "**New-List=**" keyword is dependent on whether or not you have deleted practically every other keyword other than "**Owner=**" from the list header. Since the use of "**New-List=**" is intended to be an automatic pointer to the new host and/or new list name, no other keywords should be defined. Keywords that would cause a problem will therefore generate fatal errors on a list **PUT** operation and the list header will not be updated.

Further note that the old list's subscriber list cannot be modified once the "**New-List=**" parameter is defined. The appropriate sequence of operations is:

1. Create the new list

2. Move the subscribers to it

3. **DELETE *oldlistname* *@***

4. Modify the header of the old list by deleting unneeded keywords and adding the "**New-List=**" keyword with its pointer to the new list.

Should this sequence not be followed, note that by removing the "**New-List=**" keyword, the old list will be unlocked and the subscriber list can then be deleted if desired.

Notebook= No
Notebook= Yes,*where,interval*|Separate,*access-level*[,*access-level,...*]

Indicates whether or not an automatic log of every piece of mail sent to the list is to be kept, and defines at which interval of time its file name must be changed and who is allowed to retrieve it from the server. The default values are "**Notebook= No,A,Single,Private**".

where is the filemode of the minidisk (VM) or the disk and directory (non-VM) on which the notebook is to be kept. The default value of "**A**" is equivalent to LISTSERV's main working directory. On VM servers, this is LISTSERV's A disk; on VMS and Windows servers, this is LISTSERV's **MAIN** directory, and on Unix servers it is **~listserv/home** (or whatever value has been used in the Makefile for **$LSVROOT/home**). Naturally, you may change this value to any directory you wish, provided that a) the directory exists (for security reasons, LISTSERV will not make it for you) and b) LISTSERV has read-write access to that directory. Rather than use the "A" directory, L-Soft strongly recommends that you create a separate directory structure with subdirectories for each list and use a full path spec for this parameter. This is important for security purposes related to the file server functions (see chapter 8 for details).

Note that under unix this parameter MUST IMPERATIVELY point to a directory specification that is all lower-case. LISTSERV for unix cannot write archives to directories named in upper- or mixed-case.

If your server is running the Web Archive Interface, L-Soft *does not* recommend that this parameter be pointed to the web archive index directory.

interval Defines the filetype or extension of the "notebook" file for the list, as indicated below (the filename will always be the same as the list name):

Single: A single file with the extension "NOTEBOOK" is created.
Yearly: A new file is started each yearly, extension is "LOGyy"
Monthly: The extension is "LOGyymm"
Weekly: The extension is "LOGyymmw" (w in "A"-"E")
Separate: A separate file is kept for each mailing (e.g. announcements, newsletters). The extension is "yy-

nnnnn" (sequential counter).

While you may change the notebook interval at any time, LISTSERV will not convert existing notebooks into the new interval format. For instance, if you convert from Monthly to Weekly notebooks, LISTSERV will continue to maintain your original notebooks in their monthly format, while writing any new postings into weekly notebooks. This is perfectly normal and does not affect the proper operation of your list (in particular it does not cause any breakage to the archive search feature).

For 1.8c servers with the WWW archive interface interface installed, please note that in order for archives to appear in the interface, the following requirements must be met:

1. Notebooks must be "Public"
2. The notebook interval must be "Monthly", "Weekly", or "Yearly" ("Yearly" is not recommended).
3. The LISTSERV maintainer *must* create an index directory for your list per the instructions in the *Site Manager's Operations Manual*.

Note further that lists that meet the above three requirements will show up in the WWW archive interface *even if the list is set "Confidential= Yes"*. See chapter 5.4.6 of the *Site Manager's Operations Manual* for details.

For 1.8d and later servers with the WWW archive interface interface installed, please note that in order for archives to appear in the interface, the following requirements must be met:

1. Notebooks can be "Public" or "Private" (or any other *access-level*)
2. The notebook interval *must* be "Monthly", "Weekly", or "Yearly" ("Yearly" is not recommended).
3. The "Confidential=" keyword *must* be set either to "No" or "Service"
4. The LISTSERV maintainer *must* create an index directory for your list per the instructions in the *Site Manager's Operations Manual*.

See chapter 5.4.6 of the *Site Manager's Operations Manual* for further details.

Note: Notebooks may be retrieved by means of the **GET** command. On VM only, a list of all available notebooks can be obtained with a **GET NOTEBOOK FILELIST** command.

The first two parameters of the "**Notebook=**" keyword may only be changed by the LISTSERV postmaster.

If necessary, you may break the "Notebook=" keyword into multiple lines in order to avoid running up against the 100-character header line limit. For instance

```
* Notebook= Yes,/home/listserv/lists/mylist-l,Monthly,Private
```

is strictly equivalent to

```
* Notebook= Yes
* Notebook= /home/listserv/lists/mylist-l
* Notebook= Monthly,Private
```

This can be particularly important if it is necessary to specify multiple *access-levels* for the notebooks (for instance if you have many sub-lists and want the subscribers to the sub-lists to be able to access the super-list's notebooks), for example,

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

Notebook-Header= Short | Full

Determines whether or not individual message in notebook archives are stored with full Internet header information or with "short" headers. The default is "**`Notebook-Header= Short`**".

Notify= Yes | No | *mon-address*

Defines whether the list owner (or the person indicated by "**`Notify= mon-address`**") is to receive notification of new subscriptions and deletions, etc. The default is "**`Notify= Yes`**", meaning that non-quiet list owners will be notified.

Owner= *net-address1* | *mon-address1*,[Quiet:,]*net-address2* | *mon-address2*,...

Defines the person or list of persons who "own" the list. They are responsible for controlling access to the list and defining the list control keywords which are best suited to the purpose of the list. The default value for this keyword which should ALWAYS appear in the list header is the list of the userids of the postmasters. Any combination of explicit network addresses and complex access-levels is acceptable, for example: **`Owner= BIG@BLUE,(STAFF-L),Owner(MAIN-L)`**

An interesting application is to create a STAFF-L list containing the userids of all the local LISTSERV staff members and set the "**`Owner=`**" keyword of all local lists to "**`Owner= (STAFF-L)`**". This way when there is a change in the local LISTSERV management it is not necessary to modify the headers of all the lists – just modify the STAFF-L list.

The use of the "**`Quiet:`**" parameter causes all subsequently-defined list owners to be excluded from receiving any delivery error messages or other administrative mail from LISTSERV.

List owners may be defined on a single line or on multiple lines. See Chapter 2.7 of the *List Owner's Manual* for details.

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

This keyword is not available in LISTSERV Lite.

Defines the (global) list of all the servers in the world that are peer-linked to the list, either directly or via one or more other peer servers. This information is used by the various list management commands to determine the "nearest" peer list to a given user. For example, when a **`SUBSCRIBE`** command is received from a user and it is determined that there is a better (nearer) peer list for him, the subscription request is automatically forwarded to the appropriate LISTSERV.

Be sure to read the appropriate sections of the LISTSERV *List Owner's Manual* before peering any list. Note that peers must have the same **`PW=`** keyword setting.

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

This keyword is not available in LISTSERV Lite.

This keyword controls whether or not subscribers are required to renew their subscriptions on a regular basis, and what the subscription period is. Multiple intervals can be set, each interval being one of several things:

- Monthly, Yearly, Weekly, or a numeric variation such as 3-Monthly (meaning, quarterly). Note also that 1.8c introduces the ability to code "**Renewal= *xx*-Daily**", for instance, "**Renewal= 15-Daily**". While the use of intervals of less than a week is and remains inadvisable, FAQ templates with rotating topics may require the selection of a very precise renewal interval (for congruence purposes), which was not possible with "***xx*-Weekly**" granularity. Please refer to chapter 9.9 of either the list owner's manual or the site manager's manual for a discussion of rotating FAQ support.
- An absolute date in the format ***yyyy/mm/dd*** (once on this specific day), or the format ***mm/dd*** (once yearly on this month/day).
- The confirmation delay, in the format **Delay(*n*)**, where **(*n*)**=the number of days between the time the subscriber is asked to confirm the subscription and the day the user is removed from the list. This default is Delay(7), or seven days.

A typical **Renewal=** configuration might be:

```
* Renewal= 6-Monthly,Delay(14)
```

Conceivably **Renewal=** could also be set to something like:

```
* Renewal= 6-Monthly,1998/07/04,12/25,Delay(14)
```

which would cause LISTSERV to send renewal requests once every six months on the anniversary date of the user's original subscription, a specific request on 4 July 1998, and a request every year on Christmas Day. Note that this is provided ONLY as an example. L-Soft does not recommend using a renewal scheme of this sort.

Note: When setting up **Renewal=** for the first time on an older, established list, you may find that a substantial number of subscribers are prompted for confirmation immediately even though you may have set **Renewal=** to a value that might not be expected to cause such behavior. This is because LISTSERV uses the last activity date (which may or may not be the same as the subscription anniversary date) for the purpose of subscription renewal. The last activity date may be one of the following: The subscription anniversary date; the last date the subscriber posted to the list; or the last date the subscriber changed personal options.

Note also that if you code a specific date without specifying a year field (e.g., **Renewal= 6/1**), LISTSERV will immediately request a renewal from any subscriber whose last activity date is prior to that date in the *current* year.

The "**Probe**" parameter, introduced in Version 1.8c (but disabled in LISTSERV Lite) activates LISTSERV's "active probing" bounce processing feature, whereby the users are "probed" regularly using the **PROBE1** mail template. The desired response from the user is to discard the message and do nothing. If the probe bounces, LISTSERV first sends the **PROBE2** template with a copy of the bounce

(assuming that the address actually works regardless of the bounce), and then schedules a new probe for the next day or deletes the user immediately, depending on the list's "**Auto-Delete=**" policy. For more information see chapter 4.6 of the list owner's manual.

Subscription renewal is disabled by default. To turn it off for a specific list, simply remove the "**Renewal=**" keyword from the list header.

See also **Auto-Delete=**.

Sizelim= *number* | *number*K | *number*M

This keyword is not available in LISTSERV Lite.

In 1.8d and earlier, if set, causes LISTSERV to reject all messages to the list which exceed the number of lines (including all Internet header lines) indicated. This can be helpful in discouraging subscribers from posting long screeds or uuencoded files to your lists. It can also be set higher than the LISTSERV default if desired; check with your LISTSERV maintainer before changing this upward. (Generally "**Sizelim= 250**" is large enough for long posts but short enough to discourage postings of uuencoded binaries, but of course, your mileage may vary.)

The Sizelim= list header keyword has been enhanced in 1.8e to allow list owners to specify a maximum message size in either kilobytes or megabytes, rather than in lines, if preferred. For instance:

Sizelim= 100K	Reject messages over 100Kbytes
Sizelim= 1M	Reject messages over 1Mbyte

As before, the limit operates against the entire message file, including all Internet header lines.

Subject-Tag= *text*

LISTSERV 1.8c and higher supports "subject tags", i.e., the ability to insert a predefined text tag into the subject line of mail coming from a list. For instance, your subscribers might want the subject lines of mail coming from your list to contain the name of your mailing list. Whereas the RFC822 subject line of a typical list posting without a "subject tag" would look like this:

```
Subject: I think ID4 is a great movie, don't you?
```

if you were to define

```
* Subject-Tag= SCI-FI
```

the subject would look like this for all users who are set to the new "SUBJheader" personal option:

```
Subject: [SCI-FI] I think ID4 is a great movie, don't you?
```

Note that this option may be toggled on and off by the user by use of the new "**SET *listname* SUBJecthdr**" option. It is turned off by default.

The normal default for "**Subject-Tag=**" is the name of the list, for example,

SCIFI-L. If "**List-Address=**" is defined for your list, the default is either the name of the list or the list ID, whichever is listed in "**List-Address=**". A subject tag can be only a single word; in other words, you cannot define a sentence to be used as a subject tag.

Starting with LISTSERV 1.8d, if a user sends a message with a blank RFC822 "Subject:" header, LISTSERV will create a "Subject:" header and place the subject tag into it (but only for subscribers with the **SUBJecthdr** option set.) Under 1.8c, subject tags worked only when posters defined a subject for their message.

Setting this keyword does *not* automatically reset users to the **SUBJecthdr** option. This must be done manually for existing users and may be specified by default for new subscribers by use of the **Default-Options=** keyword.

X-Tags= Comment | Yes | No

This keyword is not available in LISTSERV Lite.

Indicates whether "X-To:" and "X-cc:" tags are to be included in the output mail files to list recipients of the original mail file (other than the list userid) or not, and how they should appear in the RFC822 header.

Yes:	This information must be provided in the form of "X-To:" and "X-cc:" tags in the RFC822 header (similar to the "To:" and "cc:" tags). This is the default.
Comment:	This information must be provided in the form of "Comment:" tags, for example, "Comment: X-To:" and "Comment: X-cc:".
No:	This information must not appear at all in the mail header.

Security Keywords

Change-Log= No | Yes[,Yearly|Monthly|Weekly|Daily|Single]

This keyword is not available in LISTSERV Lite.

When set to YES, causes LISTSERV to write a ***listname*.CHANGELOG** file (***listname*** **CHANGLG** on VM) in the "A" disk or directory which contains information about all changes made to individual subscriptions. Commands tracked include **SUBscribe/JOIN**, **SIGNOFF/UNSUBscribe**, auto-deletions, and all changes to users' personal options. A CHANGELOG file can be retrieved by list owners and site maintainers with the **GET** command and deleted with the **PUT** command like any other file (it is not necessary to make catalog entries for CHANGELOG files).

In 1.8d, change-logs could be only enabled or disabled. There was no facility to rotate change-logs at all, so once enabled, a change-log simply kept growing until it was deleted. Since it can grow quite large, it was recommended that this option be enabled under 1.8d only if the list owner(s) kept it pruned on a regular basis. Alternately the option could be enabled for problem resolution, and disabled after debugging.

Non-VM LISTSERV 1.8e and later has an enhanced **Change-Log=** list header keyword that allows list owners to rotate change-logs along the same lines as they rotate list notebook logs.

Rotated logs are renamed to ***listname*.CHANGELOG-*yyyy*[*mm*[*dd*|*w*]]** and can be retrieved as usual with the GET command, which was changed to recognize these as change-logs.

When the feature is introduced (ie when upgrading to 1.8e or later), old logs will be renamed based on the date in the last entry. If the last entry is compatible with the newly introduced period (eg logs are YEARLY and the last entry is 2001), the log will continue, even if there were also entries for 2000. The definition of the 2001 log is that it contains all the entries for 2001, not that it is the first log started in 2001. Logs are not split as the feature is introduced.

If the rename fails (eg because you keep changing the period back and forth), the current log continues.

While LISTSERV on VM will accept all of the above settings without complaint, everything will behave as if you had specified SINGLE. This is because filetypes are limited to 8 characters on VM and it would be difficult to support a different naming convention just for VM.

The default is "Change-Log= No". For backward compatibility, "Change-Log= Yes" is equivalent to "Change-Log= Yes,Single".

Confidential= No | Yes | Service

Indicates whether the list should be hidden from users or not. A confidential list will not appear on the "**Lists**" command output. "**Confidential= No**" is the default value and indicates that the list is not confidential. "**Confidential= Service**" indicates that the list is to be hidden from users who are not in the list's service area (see "**Service=**" keyword) but not from other users.

"**Confidential= Yes**" means that the list is unconditionally confidential.

Please note that in LISTSERV 1.8c and following, the local list of (public) lists can be retrieved only by those users who are considered local, per the setting of the server-wide **LOCAL=** variable in LISTSERV's site configuration file. All other users will be told that none of the lists on the server are visible via the LISTS command, and will be referred to the use of the **LISTS GLOBAL *search-text*** command or to the CataList. This is regardless of the setting of **Confidential=** as outlined below.

Exit= *filename*

This feature and keyword are not available in LISTSERV Lite.

Background for non-technical users: an "exit" is a program supplied by the customer to modify the behavior of a product (such as LISTSERV) in ways that the supplier of the product could not anticipate, or could not afford to support via standard commands or options. The product checks for the presence of the "exit" program and calls it on a number of occasions, called "exit points". In some cases, the "exit" program supplies an answer ("return code") to the main program, which adjusts its behavior accordingly. For instance, LISTSERV may ask an exit program "Is it OK to add JOE@XYZ.EDU to the ABC-L list?", and the program will answer yes or no, and possibly send a message to the user explaining why his subscription was accepted or rejected. In other cases, the "exit point" call is purely informative: the exit program gets a chance to do something, such as sending an informational message to a user, but does not return any answer. Because this "exit" is a computer program, it must be prepared by a technical person and installed by the LISTSERV maintainer.

LISTSERV version 1.8a and higher support list "exits". List "exits" allow you to control the major events associated with list maintenance. This makes it easier to tailor the behavior of LISTSERV to local requirements that are too specific to be addressed through standard facilities.

An exit is enabled by adding "**Exit= *filename***" to the list header. For security reasons, all exits must be explicitly declared in the **LIST_EXITS** configuration variable (in the LISTSERV site configuration file). This prevents list owners from causing the invocation of arbitrary executable files through the use of the "**Exit=**" keyword.

This keyword is not generally usable by list owners without specific intervention by the LISTSERV maintainer, and thus is not otherwise documented here.

Local= *node1,node2,...*

This keyword is not available in LISTSERV Lite.

Defines the nodes which are to be considered as 'local nodes' for service area checking. The LISTSERV machine is automatically considered as a 'local node' and does not have to appear in the list. Subscribers from any of the local nodes will receive separate pieces of mail with a single recipient in the "To:" field – in other words, they will never receive a grouped piece of mail as non-local recipients would if there are more than one recipient in their node. Note that 'node' is a generic term that means "anything after the '@' sign in the network address". For instance, "SEARN" and "SEARN.SUNET.SE" are both valid node

names.

By default, this keyword takes its value from the **LOCAL** variable in LISTSERV's site configuration file.

PW= ***list-password***
(Obsolete since version 1.8c [except for peered lists]; included for backwards compatibility)

Defines the list password. When sending the list back to the server, the password is prefixed to the list file for validation (see the **Validate=** command for more specifics). The **PW=** parameter is "invisible" once it is defined; that is, for security reasons, it does not appear either when the list is reviewed or when it is retrieved with a **GET** command by the list owner.

LISTSERV version 1.8c and higher generate a 16-character random password for lists at list creation time if this keyword is not explicitly defined, making such lists more secure from random hackers. List owners are now encouraged to use personal passwords (defined with the **PW ADD** command, q.q.v.) in preference to list passwords for this reason.

The one exception to this keyword's obsolescence is when you are creating peer lists. Peers must have the same **PW=** setting, so you cannot use the LISTSERV-generated random password when creating peers.

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

This keyword is not available in LISTSERV Lite.

Defines the 'service area' outside of which subscription requests must not be accepted. When a **SUBSCRIBE** command is received, the "**Peers=**" keyword (if set) is checked first to see if there is a nearer peer list in the network. If this is the case, the command is forwarded to this nearer peer server. If not, the service area is checked to ensure that the recipient is acceptable; if it is not, the subscription request is denied. When the command is forwarded to a peer, the destination peer server might still deny access to the list if the subscriber is outside its own service area, if any.

It is important to note that the service area check is made only after the "best placement" check. This allows several servers in the same country to share an identical service area, for example "**Service= Germany**", and still have users subscribed to the best possible server.

For lists running the web archive interface: Starting with LISTSERV 1.8d it is possible to define "Service=" in terms of IP address blocks in order to limit access to list archive notebooks via the web archive interface. This is implemented as follows:

1. **Notebook= ...,Service**

2. "Service=" can contain entries of the form:

[^]IP(a.b.c.d[/e])

3. For any other keyword (call it "xxx") in "Service=" which contains neither

period, wildcard nor parentheses, if a site configuration variable called **IP_xxx** is defined, it is processed using the syntax in #2, except that the IP() is implicit, i.e., the syntax would be (for unix; no quotes for NT as usual):

```
IP_MYNETWORK="192.36.125.0/24 ^192.36.125.199"
```

The corresponding setting in "Service=" would then be something like

```
Service= example.com,*.example.com,MYNETWORK
```

("IP_" must not be specified in "Service=". LISTSERV understands that when you specify "Service= MYNETWORK", you mean for it to look at the value set in the IP_MYNETWORK site configuration variable.)

If both #2 and #3 are present, they are combined. Likewise, you can have multiple occurrences of #2 or #3 and they will just be combined.

Access will be granted if the IP address matches at least one of the entries that do not begin with a **^** (you can also use a minus sign if you prefer) AND the IP address does not match any of the negative entries. Otherwise you get a normal login request without any further comment.

Note carefully that LISTSERV does not do a reverse lookup on the IP addresses you code into the Service= keyword! When coding IPs into **Service=** you must also code in FQDN values for allowed hostnames. Thus if you have a list that should be restricted to the 192.36.0.0/16 subnet, which belongs to a domain called FOO.COM, you really have to code something like

```
* Service= FOO.COM,*.FOO.COM,IP(192.36.0.0/16)
```

in order for everyone in the FOO.COM domain who needs access to be able to have it.

The default value is "**Service= ***" (e.g., any host).

Validate= No | Yes[,Confirm[,NoPW]] | All,Confirm[,NoPW]]

The **Validate=** keyword determines what level of validation (if any) is performed for various LISTSERV commands that apply to individual lists. There are six different settings ranging from very basic to very strict. The two most common settings are (arguably) "**Validate= No**" and "**Validate= Yes**".

Lists are protected from hackers at the most basic level by the fact that a list **PUT** operation always requires validation, regardless of the setting of the **Validate=** keyword. In other words, the list owner or LISTSERV postmaster must *always* use a personal password (set with the **PW ADD** command, q.q.v.) when he sends an updated version either of the list header or of the entire list back to the server, even if "**Validate= No**". This is to protect you from network hackers who might issue a command "from" your userid@host address to change list settings, such as who has the ability to **GET** and **PUT** the list, review concealed subscribers, etc. The default for this keyword is "**Validate= No**", but it is recommended that "serious" or "important" lists be changed to at least "**Validate= Yes**".

When "**Validate= Yes**", password validation applies to so-called "protected" commands (all of the commands that modify the contents of the list, for example **ADD**, **DELETE**, **SIGNOFF**, etc.--but *not* **SUBscribe** or **SET**). This implies that

hackers cannot use these "protected" commands since they do not know the list owner's or LISTSERV postmaster's personal password. While at first glance this would also seem to mean that legitimate subscribers cannot use the **SIGNOFF** command, that is not the case, because for lists operating with "**Validate= Yes**" (i.e., without the "Confirm" option), LISTSERV may still use the "OK" mechanism in certain cases if it is deemed appropriate. LISTSERV's rationale is that the "**Validate=**" keyword describes the desired behavior for interaction with the list owner and people who can be expected to use the list on a regular basis. **SIGNOFF** requests from legitimate subscribers and **DELETE** requests from registered node administrators (NADs) on behalf of a user on their machine, for instance, may be validated using the "OK" mechanism even though that was not requested, because users and node administrators are not generally expected to have a password with which to validate such requests.

A notable exception to the list of "protected" commands is the **SUBscribe** command, which can still be used (if enabled, for example, if "**Subscription= Open**") to get on the list; however, when "**Validate= Yes**", sending a second **SUBscribe** command for the same list (for instance, to correct a spelling error in your name) would result in the command being forwarded to the list owner and not immediately executed. Also note that the **SET** command used to set various personal subscription options is not a "protected" command and may be issued without need for validation even when "**Validate= Yes**".

A rundown of the six different settings and what they mean follows:

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

- "Validate= Yes": "protected" commands, such as **DELETE** or **ADD**, require password validation. For list owner commands, personal passwords set with the **PW ADD** command are accepted. Some user commands may accept a personal password, while others will cause the request to be forwarded to the list owners for verification. Other "protected" commands include **GET**, but do not include **SUB** or **SET**.

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

- "Validate= Yes,Confirm,NoPW": protected commands are always validated using the "OK" mechanism. Passwords are not accepted, as they are not as safe as "cookies". This is the recommended setting for secure lists. Note that lists with this setting cannot be managed via the Web Management Interface.

- "Validate= All,Confirm": all commands causing a change in state, except the **PUT** command (which is always password-validated), are validated using the "OK" mechanism by default, although personal passwords are also accepted where appropriate. "Protected" commands (see above) are included in the class of commands that cause a change of state. Non-"protected" commands that cause a change in state include **SUB** and **SET**.

- "Validate= All,Confirm,NoPW": all commands causing a change in state (except **PUT**, as noted above) are always validated using the "OK" mechanism; passwords are not accepted, as with "Validate= Yes,Confirm,NoPW". Note that lists with this setting cannot be managed via the Web Management Interface.

Warning regarding obsolete values: Under Revised LISTSERV (that is, LISTSERV for VM prior to version 1.8a), "Validate= All commands" (or "Validate= All") and "Validate= Store only" (or "Validate= Store") were the only acceptable values for this keyword. These old values are still accepted for compatibility reasons, but generate a warning with an explanatory message when you update the list header. Since these values are obsolete and may not be supported in the future, you should change any instance of these settings in your lists to the current equivalent values (or to other currently-acceptable values as you see fit):

"Validate= Store only" is now "Validate= No"
"Validate= All commands" is now "Validate= Yes"

Informational commands such as **QUERY**, **SHOW**, **INDEX** and **REVIEW** do not require any validation, regardless of the setting of **Validate=**.

Requests originating on the local machine via CP MSG or CP SMSG (on VM systems) or originating on the local machine via LCMD (on VMS, Unix, and Windows 95/NT systems) never require validation, as they cannot be forged.

In all cases save one, the **PUT** command must always be validated with the personal password of the list owner or LISTSERV postmaster who is executing the **PUT** operation. This is because LISTSERV is not currently able to (1) suspend the execution of your **PUT** command, (2) store your list header or other file in a temporary file, and (3) wait for your "OK" before executing the **PUT**. If your password is used only for the purpose of validating **PUT** commands, any password exposure is minimal as **PUT** operations are not part of everyday list management routine. VM users should note that **PUT** requests require no validation when submitted via CMS SENDFILE from the machine on which LISTSERV is running, as the operating system itself guarantees the authenticity of the transaction (and thus there is no need to store the file you are **PUT**ting and wait for an "OK"). This is the only case in which a **PUT** operation does not require a password.

Table B.1 shows how LISTSERV commands are influenced by the **Validate=** keyword under different settings. Some redundant commands (e.g., **JOIN**, **LEAVE**, and **UNSUBscribe**) are not documented in the table, but behave exactly as do their "official" counterparts (e.g., **JOIN** behaves exactly as does **SUBSCRIBE**). Some commands never require validation but are included for completeness because they are specifically "list-level" commands. If a command is not otherwise listed below, it is not influenced in any way by the **Validate=** keyword.

NONE = does not require any validation
PW = requires password validation
OK = requires OK validation, will not accept PW validation
OK/PW = requires OK validation by default but will also accept PW validation

	Validate= setting is:					
Command	No	Yes	Yes, Confirm	Yes, Confirm, NoPW	All, Confirm	All, Confirm, NoPW
ADD(*)	NONE	PW	OK/PW	OK	OK/PW	OK
CHANGE(**)	NONE	PW	OK/PW	OK	OK/PW	OK
CONFIRM	NONE	NONE	NONE	NONE	NONE	NONE
DELETE(*)	NONE	PW	OK/PW	OK	OK/PW	OK
FREE(*)	NONE	PW	OK/PW	OK	OK/PW	OK
GET(***)	NONE	PW	OK/PW	OK	OK/PW	OK
GETPOST	NONE	NONE	NONE	NONE	NONE	NONE
HOLD(*)	NONE	PW	OK/PW	OK	OK/PW	OK
INFO	NONE	NONE	NONE	NONE	NONE	NONE
PUT(*)	PW	PW	PW	PW	PW	PW
QUERY	NONE	NONE	NONE	NONE	NONE	NONE
REVIEW	NONE	NONE	NONE	NONE	OK/PW	OK
SCAN	NONE	NONE	NONE	NONE	NONE	NONE
SEARCH	NONE	NONE	NONE	NONE	NONE	NONE
SET	NONE	PW	OK/PW	OK	OK/PW	OK
SIGNOFF	NONE	NONE	NONE	NONE	OK/PW	OK
SUBSCRIBE	NONE	NONE	NONE	NONE	OK/PW	OK
UNLOCK(*)	NONE	PW	OK/PW	OK	OK/PW	OK

Table B.1. LISTSERV list-level commands and how they are affected by Validate=.

(*) All commands so marked may be issued only by a list owner or LISTSERV postmaster.

(**) The **CHANGE** command has two syntaxes, one for general users, one for list owners/postmasters. General users will always be required to use "OK" confirmation, regardless of the **Validate=** setting. The values in the table above are for the syntax issued by list owners or the LISTSERV postmaster(s).

(***) '**GET** ***listname***' may be issued only by a list owner or LISTSERV postmaster. General users may issue **GET** commands for notebook archives and/or files listed in the list's file catalog and with appropriate **GET** FACs only.

Please note carefully that Table B.1 assumes that you have defined default values for most keywords. For instance, the **SUBSCRIBE** command will require an "OK" confirmation if you have "**Subscription= Open,Confirm**" and "**Validate= No**"; **REVIEW** will only be available depending on how you have the "**Review=**" keyword set; **GETPOST** and **SEARCH** may require passwords depending on the "**Notebook=**" setting; etc. However none of these conditions are influenced directly by "**Validate=**" except as noted in the table.

In some cases, "**Validate= Yes**" will cause an "OK" request to go out instead of requiring a password (i.e., if no password was included with the command). This is to avoid confusion on the part of a subscriber who may or may not have a LISTSERV password and who may not understand why he is being asked to provide a password before a given command will work.

Note also that even with "**Validate= No**" some users may be required to confirm commands with the "OK" method if they are sending commands via a web browser and **WEB_BROWSER_CONFIRM=** is set to 1 (the 1.8c default; under 1.8d the default is 0, or disabled) in the site configuration file.

History: This keyword was revised substantially in versions 1.7f and 1.8a. The "OK" command confirmation mechanism was introduced in version 1.7f, where it was used to implement the "Subscription= Open,Confirm" address verification

mechanism. When a user tries to subscribe to a mailing list with that setting, he is mailed a confirmation request with a randomly generated confirmation key, also known as "magic cookie". The user replies to the message, types "OK" in the message body, and the command is confirmed. If for any reason the user's address cannot be replied to, the confirmation request is never received (or the "OK" message never arrives) and the user is not added. In versions 1.8b and following, this procedure is also used for authentication purposes. Since the confirmation codes are valid only for a single command, this actually provides better security than personal passwords, while simplifying book-keeping.

As before, the security level of the mailing list is controlled through the "Validate=" keyword. The contents of this keyword, however, have changed from earlier versions (the old values are still accepted for compatibility reasons, but generate a warning with an explanatory message when you update the list header. This may change in subsequent versions, so it is advisable to use the new values).

Subscription Keywords

Confirm-Delay= *number*

This keyword is not available in LISTSERV Lite.

This parameter is an integer representing the number of hours LISTSERV will hold subscription jobs requiring confirmation before flushing them from its queue. For instance, if **`Subscription= Open,Confirm`** and **`Confirm-Delay= 72`**, LISTSERV will accept a subscription request pending confirmation, send the "cookie" command confirmation request, and will wait 3 days (72 hours) for that confirmation to be received. If the period expires before the "cookie" is received, the subscription request is deleted and the subscriber must resubmit his or her request. The default setting is 48 hours (2 days).

Many unreliable gateways have a turnaround time of several days, and this is another way to filter them: if the confirmation delay is long enough, they will never manage to subscribe and you will not have to put up with gateways that take a week to realize that the subscriber's account has expired and return a week's worth of delivery errors. On the other hand, if you do want to let these people in, you will have to increase the confirmation delay to a week or so (1 week=168 hours).

In LISTSERV 1.8b and later, this keyword can also extend the period of time during which postings to a list coded "**`Send= Editor,Hold`**" are held before they are flushed. The default (and minimum) for holding such postings is 7 days (168 hours). Note that you can only increase this period with "**`Confirm-Delay=`**", not decrease it. Thus for a list with "**`Send= Editor,Hold`**" and "**`Confirm-Delay=48`**", the holding period would still be 7 days. But for a list coded "**`Send= Editor,Hold`**" and "**`Confirm-Delay=240`**", the holding period would be 10 days (240 hours).

Please inform the LISTSERV maintainer before any significant increase to the value of "**`Confirm-Delay=`**", particularly if your list is coded "**`Send= Editor,Hold`**" or "**`Send= Editor,Hold,Confirm`**", as the increased delay could cause a problem with disk space availability.

Note that if you increase "**`Confirm-Delay=`**" to extend the holding period for postings, you also are increasing the period during which LISTSERV will hold subscription jobs requiring confirmation.

See also **`Subscription=`**.

Default-Options= *option1,option2,...*

A "Default-Options" keyword is available to define initial personal options for new subscribers. The syntax is the same as for the **`SET`** command, except that options are separated by commas in the usual fashion. Setting **`Default-Options=`** does not affect existing subscribers. If you want existing subscribers to have these settings, you must update them manually with a **`SET`** ***`listname options`*** **`FOR *@*`** command.

A typical **`Default-Options=`** setting might be:

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

Note that if you have "**Default-Options= NOPOST**" for your list and you add an editor or a moderator as a subscriber, you will have to manually reset the editor to **POST** (with "**SET *listname* POST FOR *userid@host***") before things will work properly. You will know that this is necessary if your editor or moderator can successfully approve postings but is then told that he or she cannot post to the list.

Starting with LISTSERV 1.8d, all default options are applied to non-subscribers, so it is possible to force even non-subscribers to post through a moderator by simply setting "**Default-Options= REVIEW**", or lock them out altogether by setting "**Default-Options= NOPOST**". This works even if your list is set "**Send= Public**", in which case there is a side benefit: the setting will stop people whom you have set to REVIEW or NOPOST from signing off the list and being able to post.

Two caveats regarding the use of this keyword under 1.8d:

Default-Options= REVIEW is overridden for addresses defined in **Editor=** or **Moderator=**.

Default-Options= NOPOST in conjunction with **Send= Editor** causes mail from non-subscribers to be forwarded to the appropriate Editor for approval rather than simply rejecting it with a "you are not allowed to post" message.

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

This keyword is not available in LISTSERV Lite.

A "**Default-Topics=**" list header keyword is available to define the initial topics for new subscribers. The syntax is the same as for the **SET TOPICS** command, except that topic names are separated by commas in the usual fashion and that the first topic may not start with a + or - sign (there is nothing to add to, as this is a new subscription). This is similar to "**Default-Options=**" in that it does not affect existing subscribers. Users who signed up before topics were enabled on the list are automatically subscribed to all topics.

As with **Default-Options=**, setting this keyword affects only subscribers who sign up after you set it. If you want existing subscribers to be set to these topics, you must update them manually with a **SET *listname* TOPICS: *topics* FOR *@*** command.

Subscription= Open [,Confirm]
Subscription= By_Owner[,Confirm]
Subscription= Closed

This keyword defines whether or not new users are allowed to subscribe to the list, and if not, whether their subscription requests are to be forwarded to the list owner or not.

Open:	New users are allowed to subscribe to the list.
Open,Confirm:	Before new users are allowed to subscribe to the list they must confirm their address with an "OK" response. No list owner intervention is required.
By_Owner:	New users are not allowed to subscribe, but their requests

will be forwarded to the list owner. This is the default.

By_Owner,Confirm: (1.8e) Similar to "By_Owner", but before the request is forwarded to the list owner, the would-be subscriber must first respond to an "OK" confirmation request. NOTE CAREFULLY that you MUST specify "By_Owner" rather than "By Owner" -- the underscore is required for this syntax.

Closed: New users are not allowed to subscribe, and their requests are not to be forwarded to the list owner.

Note that "Subscription= By Owner" is still supported but is deprecated, and as noted, the underscore must be supplied if ",Confirm" is used.

(The ",Confirm" option is used in conjunction with "Subscription= Open" and "Subscription= By_Owner" only. It has no effect with "Subscription= Closed".)

One problem plaguing some mailing lists is one-way or non-repliable addresses. Most of the time this is due to a small number of faulty gateways, which one can just ban from the list until their maintainers address the problem. But sometimes the very topic of the list is bound to attract a large number of people connected through such gateways, and the amount of domains to filter out becomes unmanageable. This is particularly problematic when the gateway keeps quiet for a few days, and suddenly returns hundreds of delivery errors with a promise to keep doing so every day for another 6 days.

This problem can be avoided by probing the return address before accepting the subscription. If the address cannot be replied to, only one delivery error will be bounced to the list owner (perhaps for several days, but there will be a single undeliverable message). With a gateway that waits 3 days before sending its first delivery error, however, there can be hundreds of messages "in the pipe" if the subscription is accepted directly. This probing is activated by specifying "**Subscription= Open,Confirm**" in the list header. When a user attempts to subscribe to the list, he is mailed a confirmation request with a randomly generated "confirmation code". The procedure for confirming the subscription is simple - you just reply to the message, type "OK", and send. If the return address does not work, the request will never be confirmed and the user will not be subscribed. And since the user cannot guess the confirmation code he will be assigned, he cannot "cheat" by sending the confirmation along with his request.

The subscription request also expires after a certain amount of time, as determined by the "**Confirm-Delay=**" keyword (the default is 48h).

Similarly, starting with LISTSERV 1.8e, it is possible to code "**Subscription= By_Owner,Confirm**". This adds an address probe into the usual procedure for subscriptions that must be approved by the list owner. The behaviour with "**Subscription= By_Owner**" is

1. User sends SUBSCRIBE command.
2. LISTSERV forwards to list owner.

and no check is done to verify that the user's address is viable. Adding the "**,Confirm**" parameter changes the behaviour to

1. User sends SUBSCRIBE command.
2. LISTSERV asks for OK confirmation (new).

3. User confirms.
4. LISTSERV forwards to list owner.

With this setting in place, list owners who run restricted-subscription lists will no longer have to discover for themselves whether or not a potential subscriber's address has been spoofed or is otherwise non-viable, since they will never see subscription requests that have not been confirmed by the subscriber.

Other Keywords

Categories= *category1,category2,...categoryn*

Note: the full list of categories may not be available when version 1.8d is released.

Sets search categories for this list (by default, none are defined) for the CataList service (see Chapter 3.3 of the *List Owner's Manual* for details). For instance, you might have a list on the topic of great opera tenors of the 20th century, and want anyone searching the CataList based on certain topics to find your list. You might therefore code:

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

and so forth.

DBMS= No
DBMS=Yes[,Table(*xxx*)][,Email(*xxx*)][,Name(*xxx*)][,Uemail(*xxx*)][,Options(*xxx*)] [,Server(*server_alias*)]

This keyword is not available in LISTSERV Lite.

This functionality is not available under VM. Under non-VM it requires an appropriate DBMS application (not provided by L-Soft) be installed on the LISTSERV machine.

"DBMS=" (introduced in 1.8d) is used to tell LISTSERV that the list of subscribers is kept in an ODBC or OCI database rather than in the traditional **`*.LIST`** file. In order to set this keyword to anything but the default, DBMS support must be installed on the LISTSERV server. Please see the *Developer's Guide for LISTSERV* (available separately) for more information on installing support for and using the DBMS and Mail Merge functions.

Warning: List owners should NOT add this keyword to their list header as the use of the DBMS= keyword presupposes existing DBMS support that has been configured to work with LISTSERV. Further, list owners should NOT change the value of this keyword if it exists in their list header, as changing any of the parameters could lead to unexpected results.

Windows OCI Support Withdrawn: In LISTSERV 1.8e, it is possible to define multiple simultaneous connections to DBMS tables, and as a result, previous OCI support for Windows NT/2000 (which was provided to work around the earlier limitation of only a single ODBC connection at a time) has been withdrawn. OCI databases may still be accessed via ODBC.

When migrating an existing list to use a DBMS, you are responsible for migrating the subscriber data to the DBMS, if necessary (in many cases, the subscriber data will already be in the DBMS, possibly in a slightly different format). Once you add the "DBMS= Yes" keyword, LISTSERV stops accessing subscriber data from the xxx.LIST file and uses the DBMS instead.

The default is "DBMS= No", i.e., keep subscriber information in a traditional **`*.LIST`** file.

Indent= *number*

This keyword is not available in LISTSERV Lite.

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

Language= *idiom*|*option*[,*option*]

This keyword is not available in LISTSERV Lite, except that it can be set to "Language= Exchange" to avoid suppression of application/ms-tnef attachments as noted below.

Defines the language in which information mail and messages are to be sent to subscribers of the list. The postmaster must have provided the required data file (called ***idiom*.MAILTPL**, where ***idiom*** is the name of the language specified by this keyword) to the server. The default is "**Language= English**", which uses **DEFAULT.MAILTPL**.

L-Soft does not provide non-English templates.

Starting with 1.8d, this keyword was expanded to cover the following functionality:

Language= HTML: This setting allows you to specify that LISTSERV's administrative messages (i.e., those specified in the MAILTPL) be sent out in HTML format. You specify either **Language= HTML** or **Language= *idiom*,HTML** to enable this feature. *Note carefully that this setting does not affect the WELCOME or FAREWELL files.*

Language= NOHTML: This setting allows you to specify that LISTSERV strip any HTML attachments from postings (while retaining HTML tags sent in the body of plain text messages). You specify either **Language= NOHTML** or **Language= *idiom*,NOHTML** to enable this feature.

The actual function of this setting is to remove the attachment that contains the HTML mail from the message. It does not remove HTML tags from plain text messages. This means that setting this option will not suppress HTML in messages sent from (for instance) Eudora Pro 3.x (since Eudora Pro 3.x does not send the HTML message as a MIME attachment with a plain text alternative).

In 1.8e and following, if an HTML message does not contain a plain text alternative, and **Language= NoHTML** is set, the message will be rejected. This is a change from the behavior in 1.8d, which would allow such messages through.

Language= EXCHANGE: This setting allows you to specify that LISTSERV keep Microsoft Exchange attachments in postings (the default is to remove them). You specify either **Language= EXCHANGE** or **Language= *idiom*,EXCHANGE** to enable this feature. Note that this affects "application/ms-tnef" attachments only--LISTSERV does not currently strip WINMAIL.DAT attachments.

These three formatting options are not mutually exclusive and may be defined in any grouping (in other words, **Language= HTML,NOHTML,EXCHANGE** is legal although it is unlikely anyone would want to use it).

Limits= Sub(*number*),...

This keyword is not available in LISTSERV Lite.

This keyword is available only with the ISP add-on, and may only be added or changed by the LISTSERV Maintainer. Defines specific limits for a list. Currently only the number of subscribers can be limited, for example,

```
* Limits= Sub(100)
```

This keyword may only be added or changed by the LISTSERV postmaster, and the list creation password is required for the list **PUT** operation when the keyword is added or changed. The list owner may execute a **PUT** operation with this keyword defined in the header as long as the values for the keyword are not changed.

Long-Lines= Yes | No

This keyword is not available in LISTSERV Lite.

Enables or disables "long-lines" support. This keyword was added to maintain compatibility with LISTEARN and will be removed in a future version of LISTSERV. The default is "**Long-Lines= Yes**". It is unlikely that this keyword will need to be set for any list.

Mail-Merge= Yes | No

This keyword is not available in LISTSERV Lite.

Documented Restriction: Note that LISTSERV's mail merge functionality REQUIRES the use of LSMTP Classic as the outgoing MTA. Mail merge does not work with sendmail, qmail, Post.Office, Netscape Mail Server, Microsoft Exchange, PMDF, MX, or any other MTA except L-Soft's LSMTP Classic mailer.

Under unixes not supported by LSMTP Classic this may require that you set **SMTP_FORWARD=** accordingly in **go.user**, to point to a separate machine running LSMTP (for instance, a dedicated Windows NT LSMTP machine). Under OpenVMS or Windows NT can run LSMTP Classic either on the same machine (the preferred method), or on a separate machine if desired. The main point is that the outgoing mail-merge postings MUST be handled by LSMTP Classic. (LSMTP Lite does not support mail-merge.)

Mail merge functions are documented fully in the *Developer's Guide for LISTSERV*, available separately.

Mail-Merge= Yes makes every posting to the list a mail-merge operation (see the *Developer's Guide to LISTSERV* for more information about formatting mail merge jobs). Digests and indexes also become mail-merge jobs. Header modifications are done in the same way as with a normal list, you confirm with OK as usual, etc. The text will then follow the same rules as when entered in a mail-merge job using the web interface--ASSUMING that your mail program did not do anything fancy to the text, such as replace all ampersands with =xx or rewrite everything in rich-text format. It is fundamentally difficult and unreliable to

use a mail program as the client for mail-merge since mail programs are engineered to encode information in fancy and unpredictable ways that were not used in the previous version, whereas mail servers are engineered to only look at the information they positively need to look at to get the job done, so that they do not inadvertently get in the way of new, fancier encoding methods. Therefore it is important when using a mail program to send mail-merge jobs that you ensure that the mail program sends plain text rather than any kind of encoded text.

This keyword should be used only on lists that are set up as announce-only mailing lists (that is to say, posting should be restricted to the list owner(s) only for security purposes). Because this keyword alone does not restrict the ability to post, the list owner(s) should ensure that the "**Send=**" keyword is set appropriately. Note that if you do not restrict the ability to post, anyone who is otherwise permitted to post to the list will also be able to send mail-merge jobs to it.

This method can be used to attach files unconditionally. It cannot be used to send file 1 to some people and file 2 to others because the mail program knows nothing about mail-merge and, as such, does not mark them as conditional blocks. Any markings you provide are part of the text attachment you are working in.

The default is to send mail normally, in other words, **`Mail-Merge= No`** . As noted above, if your LISTSERV installation does not use LSMTP as its outbound mail server, you should never change this setting from the default.

Misc-Options= ***option1,option2...optionn***

This keyword is not available in LISTSERV Lite.

This keyword is available in non-Lite 1.8e versions and later with build dates of September 2002 forward (issue the command SHOW LICENSE to LISTSERV to determine build date). It is not available in original release 1.8e.

This keyword is a catch-all for certain behavior-modifying options that are not otherwise covered by other, more specific keyword settings. Currently the only options available are as follows:

IETF_SUBJECT_TAG
IGNORE_EMAIL_CASE
KEEP_DKIM_SIGNATURE
NO_DKIM_SIGNATURE
NO_RFC2369
NO_SPAM_CHECK
RESPECT_EMAIL_CASE
SUBJECTHDR_SEQUENCE
SUPPRESS_APPROVED_BY

`IETF_SUBJECT_TAG` (14.5)

Add subject tags to IETF headers (that is, for users who are set to the IETFHDR personal option). However, as this can be considered a violation of the standard for IETF-style headers, it can be prevented site-wide by the site administrator if desired.

Adding "Misc-Options= IETFHDR_SUBJECT_TAG" to the list header causes the IETFHDR option to always include subject tags. This is a per-list setting; in other words, either all subscribers to a list who are set to the IETFHDR option get the subject-tag, or no. The design consideration is that, at present, it is prohibitively expensive to switch the existing header types into flags as the number of combinations grows significantly.

IGNORE_EMAIL_CASE | RESPECT_EMAIL_CASE

These options are mutually-exclusive; only one can be defined at a time per list.

When set in a list header, "Misc-Options= IGNORE_EMAIL_CASE" causes the ADD command to ignore the case of the "local part" of list subscriber entries (that is, the part of the address that is to the left of the "@" sign). Although most modern mail clients are configured to ignore the case of the local-part, this behavior technically violates RFC821 which states that local-parts are considered case-sensitive.

If an entry whose "local part" differs only in case is found in the list during an ADD operation (for instance, JOE@EXAMPLE.COM vs. joe@EXAMPLE.COM), that entry will be assumed to be the entry that was sought, and the address field will be updated to the new case (that is, "JOE@" will be changed to "joe@"). No other change will be made to the entry unless there is a change in the name field, in which case the name field will also be updated.
If there is no change in the address field associated with the entry, no change will be made to the entry (again, unless the name field changes, in which case the entry will be updated).

In either case, when this option is set, a new entry with a different case will NOT be added.

Note the following caveats:

1. Pre-existing duplicates are not automatically removed from lists when this option is set.
2. Because ADD updates the case of entries, it is possible to end up with multiple entries that have exactly the same case.
3. The only real way to de-dupe a given address is to DELETE and then re-ADD it.

Other than this, existing duplicate entries work exactly as they did before the option was enabled. Commands that do not add new entries ignore the option.

And finally, it should be carefully noted that the PUT command also ignores the option.

When a list is set to "Misc-Options= RESPECT_EMAIL_CASE", this tells LISTSERV to operate per RFC821 and treat address fields with differently-cased local parts as different addresses. The option is provided as an override to the site-level IGNORE_EMAIL_CASE configuration variable and does not need to be set to preserve the default unless the site setting has been changed to make IGNORE_EMAIL_CASE the default.

KEEP_DKIM_SIGNATURE (14.5)

Incoming DomainKeys signatures in messages submitted to a mailing list will be suppressed unless "`Misc-Options= KEEP_DKIM_SIGNATURE`" is set in the list configuration.

The KEEP_DKIM_SIGNATURE option is experimental and not meant for general use. As DomainKeys is specified today, signatures DO NOT survive posting to mailing lists (LISTSERV or otherwise), so LISTSERV removes them by default to avoid triggering alerts for subscribers on systems that have implemented the client side of DomainKeys.

`NO_DKIM_SIGNATURE` (14.5)

"`Misc-Options= NO_DKIM_SIGNATURE`" is available at the list level to override LISTSERV's default DomainKeys message signing if desired.

`NO_RFC2369` (14.5)

In LISTSERV 14.5, support has been added for RFC2369, which calls for the use of message headers such as "List-Help", "List-Subscribe", and "List-Unsubscribe". A list posting using these headers will look like this:

```
Date: Fri, 21 Oct 2005 14:21:03 -0500
Reply-To: Test list <TEST@LISTSERV.EXAMPLE.COM>
Sender: Test list <TEST@LISTSERV.EXAMPLE.COM>
From: Some User <someuser@EXAMPLE.COM>
Subject: What's all this RFC2369 stuff?
To: TEST@LISTSERV.EXAMPLE.COM
Precedence: list
List-Help: <http://listserv.example.com/scripts/wa.exe?LIST=TEST>,
           <mailto:LISTSERV.EXAMPLE.COM?body=INFO+TEST>
List-Unsubscribe: <mailto:TEST-unsubscribe-
request@LISTSERV.EXAMPLE.COM>
List-Subscribe: <mailto:TEST-subscribe-
request@LISTSERV.EXAMPLE.COM>
List-Owner: <mailto:TEST-request@LISTSERV.EXAMPLE.COM>
List-Archive:
<http://listserv.example.com/scripts/wa.exe?LIST=TEST>

I was curious about these new headers, can someone enlighten me?
```

RFC2369 support is activated by default and supplies all of the headers specified in the standard except "List-Post:", which L-Soft considers to be redundant.

In compliance with RFC2369, LISTSERV discards any pre-existing List-xxx tags.

RFC2369 compliance can be disabled using:

```
* Misc-Options= NO_RFC2369
```

and this can also be specified in the site-wide DEFAULT_MISC_OPTIONS variable. When RFC2369 support is disabled, you get the old behavior; that is, the tags are neither added nor removed.

`NO_SPAM_CHECK` (14.3)

Use this option to disable spam scans for a particular list and its associated **`xxx-request`** address. (This is only useful if the LISTSERV maintainer has enabled spam-scanning via the SPAM_EXIT feature.)

RESPECT_EMAIL_CASE

See above at IGNORE_EMAIL_CASE.

SUBJECTHDR_SEQUENCE (14.5)

For LISTSERV 14.5, it is possible for each list posting to have a sequence number attributed to it, which can be seen by subscribers who are set to the SUBJECTHDR personal option. This new feature is enabled by adding "Misc-Options= SUBJECTHDR_SEQUENCE" to the list header. Site administrators can enable it server-wide by adding the value SUBJECTHDR_SEQUENCE to DEFAULT_MISC_OPTIONS in the site configuration. The format of the new subject tag is

```
[listname - number]
```

For example:

```
Subject: [TEST - 256] Test of SUBJECTHDR_SEQUENCE
```

where 'number' is a sequence number, starting from 1 and increasing indefinitely for all practical purposes (the counter will accommodate over 2 billion postings per list).

"Subject-Tag=" is still used to change the first item. In order to allow server administrators to set a server-wide default for the new feature, it was not possible to implement the sequence number feature as an extension to "Subject-Tag=".

Note: Both the new [listname - number] and the traditional [listname] tags are removed from the subject on incoming messages. This happens whether or not the option is set, because people might be replying to old messages before the option was changed.

LISTSERV tries very hard never to skip a sequence number. The only plausible scenario in which this could ever happen would be when the MTA (not LISTSERV) fails to deliver the message. However, there are several cases where a sequence number will be reused:

- If the site administrator deletes or temporarily renames PERMVARS.FILE in an attempt to solve an unrelated problem. Deleting PERMVARS.FILE is *not supported* as a troubleshooting method.
- If the list is migrated to a new host.
- If there is a crash, disk full error, etc., when updating the counter.

SUPPRESS_APPROVED_BY (14.3)

Use this option to suppress RFC822 "Approved-By:" headers that would normally be generated by LISTSERV in messages posted through moderated lists.

Translate= Yes | No

Determines whether LISTSERV keeps or removes control characters from files which it distributes. "**`Translate= Yes`**" removes control characters;

"**Translate= No**" keeps them. The default setting is "**Translate= Yes**".

Setting "**Translate= No**" may be required for accurately passing messages written using double-byte character systems such as those available for the Japanese language.

Default Values for all keywords

Ack= No
Attachments= Yes
Auto-Delete= No if "Validate= Yes", Yes,Semi-Auto,Delay(4),Max(100) otherwise
Categories= <none>
Change-Log= No
Confidential= No
Configuration-Owner= Owner
Confirm-Delay= 48
Daily-Threshold= 50
DBMS= No
Default-Options= <none>
Default-Topics= <none>
Digest= Yes,Same,Daily if "Notebook= Yes", No otherwise
Editor= <none>
Editor-Header= Yes
Errors-To= Owners
Exit= <none>
Files= No
Filter= <built-in>
Indent= 40
Internet-Via= <none>
Language= English
Limits= <not set, ISP option only>
List-Address= <none> (per LIST_ADDRESS system default)
List-ID= <none>
Local= <none> (per LOCAL system default)
Long-Lines= Yes
Loopcheck= Normal
Mail-Merge= No
Mail-Via= DISTRIBUTE
Misc-Options= <none>
Moderator= <none> (defaults to first Editor if "Editor=" is defined))
New-List= <none>
Newsgroups= <none>
NJE-Via= <none>
Notebook= No,A,Single,Private
Notebook-Header= Short
Notify= Yes
Owner= (This is a mandatory parameter which must be filled with at least one person's network address in userid@node or userid@fqdn format)
Peers= <none>
Prime= Yes
PW= (randomly generated at list creation time if not specifically defined)
Renewal= <none, disabled>
Reply-To= List,Respect
Review= Private
Safe= Yes
Send= Public
Sender= List
Service= *
Sizelim= <none>
Stats= Normal,Private
Sub-lists= <none>
Subject-Tag= short name of the list, for example, MYLIST-L
Subscription= By_Owner
Topics= <none>
Translate= Yes
Validate= No
X-Tags= Yes

Appendix C: Site Configuration Keyword Reference for LISTSERV® 14.5

Last updated 22 February 2006

Defining site configuration keyword values

The syntax used to define these values differs from platform to platform. For each site configuration keyword we have provided examples for all systems affected by the keyword. A general syntax guide follows:

VM

```
KEYWORD = 'somevalue'
```

Note that for VM the space on either side of the "=" sign is required. For substitutions, follow the REXX language syntax. For instance,

```
MAILER = 'mailer@'NODE
MYDOMAIN = NODE 'some.host.com'
```

OpenVMS

OpenVMS users should use the **LISTSERV_CONFIGURE.COM** application, provided with the software, to modify the site configuration file, rather than editing it by hand.

For reference, however, SITE_CONFIG.DAT entries look like this:

```
NODE      "LISTSERV.EXAMPLE.COM"
MYDOMAIN    "LISTSERV.EXAMPLE.COM"
LOCAL      "*.EXAMPLE.COM"
DBRINDEX    "1"
```

All values must be in double quotes.

Unix (all)

Keywords requiring text values:

```
KEYWORD="somevalue"
```

Keywords requiring numeric values (ie, Boolean settings):

```
KEYWORD=number
```

eg, **NODE="listserv.mynode.com"**, but **DBRINDEX=0** .

For substitutions, follow this example:

```
MYDOMAIN="$NODE some.host.com"
```

Note that under unix, all site configuration variables must be exported. The commonly-used variables are exported for you in the **go** script. Anything else must be exported at the end of **go.user**.

Windows (all)

Windows users should use the **SITE.EXE** application, provided with the software, to modify the site configuration file, rather than editing it by hand. However, it is possible to edit the file by hand if necessary.

```
KEYWORD=somevalue
```

For substitutions, follow this example:

```
MYDOMAIN=%NODE% some.host.com
```

After Making Changes, Restart the Server

Whenever you change the contents of the site configuration file, you must stop and restart the LISTSERV server before the changes will take effect. Windows sites using the SMTPL.EXE "listener" should also stop and restart the listener as some changes will affect it as well.

Alphabetical Keyword Reference

Please see the full documentation of each keyword for platform availability.

Keywords not hyperlinked are obscure, undocumented VM-only settings. If you are a VM site and need help with these settings, please contact L-Soft customer support.

If the "Introduced" column is blank, the keyword existed prior to LISTSERV 1.8d (13).

Keyword	Abstract	Introduced
ALL_REQUEST_ALLOWED_USERS	Specifies non-POSTMASTER users who are allowed to post to the ALL-REQUEST alias.	14.0
ANTI_VIRUS	Turns LISTSERV's real-time anti-virus scanning on or off.	14.0
BITNET_ROUTE	Defines the hostname of a machine that knows how to route mail to BITNET addresses	
AVS_DROP_SPAM	One of two variables that may be set on servers that submit mail to the external LISTSERV AVS for virus and spam scanning, which can help deal with DoS mail-bombing attacks.	14.4
AVS_DROP_VIRUS	One of two variables that may be set on servers that submit mail to the external LISTSERV AVS for virus and spam scanning, which can help deal with DoS mail-bombing attacks.	14.4
BOUNCES_TO	Tells LISTSERV where to send bounces not related to any particular list	1.8d (13)
CHANGELOG_DBMS	Tells LISTSERV to mirror a copy of all or selected changelogs to DBMS.	14.0
CHANGELOG_DBMS_CONNECTION	Used in conjunction with CHANGELOG_DBMS. Sets the DBMS connection characteristics for mirroring changelog data to DBMS.	14.0
CHANGELOG_DBMS_TABLE	Used in conjunction with CHANGELOG_DBMS.	14.0

	Sets the DBMS table characteristics for mirroring changelog data to DBMS.	
CHECKMDISK	List of library minidisks to be checked at startup	
CLI_*	Three configuration variables under the CLI_* rubric are available for use with DBMS/Mail Merge.	14.0
CMDPIPE_HOSTNAME	Defines the hostname used by the LCMD utility	
CMSNAME	The name of the CMS system to be used on IPL commands	
CRASH_MONITOR	Where to send VMS or NT crash reports	1.8c (12)
CREATEPW	The password required to create new lists	
DATABASE	Indicates whether the (old) VM database functions are enabled or not	
DBMS_*	Several configuration variables under the DBMS_* rubric are available for use with the DBMS/Mail Merge.	1.8d (13)
DBRINDEX	Determines whether or not the new LISTSERV database functions use reverse indexing	1.8c (12)
DEBUG_LOG_TCPGUI	Show fully-logged TCPGUI commands for debugging purposes.	14.3
DEBUG_SHOW_PW	Debug setting to display passwords in LISTSERV's log file	14.3
DEFAULT_CHANGELOG_PERIOD	Sets a default period for rotating change-logs.	14.0
DEFAULT_LANGUAGE	Sets the default national language template for use by all lists on the server	1.8d (13)
DEFAULT_LOOPCHECK	Sets the server default for the list-level Loopcheck keyword	14.5
DEFAULT_MISC_OPTIONS	Sets the server default for the list-level Misc-Options keyword	14.5
DEFAULT_PROBE	Sets defaults for passive probing	
DEFAULT_SPLIT	Provides a default value for the "SPLIT=" command line keyword	1.8b (11)
DELAY	The delay between two reader-scan operations	
DIAGD4	Indicates whether LISTSERV should use diagnose X'D4' to mimic the RSCS origin on files it DISTRIBUTEs	
DIST_ALLOWED_USERS	Space-separated list of non-POSTMASTER users who are to be allowed to use DISTRIBUTE	1.8d (13)
DIST_FORWARD	Enables and tunes DISTRIBUTE "workers". This is a tuning feature for embedded mail merge.	14.4
DIST_FORWARD_THRESHOLD	Enables and tunes DISTRIBUTE "workers". This is a tuning feature for embedded mail merge.	14.4
DIST_OWNER_MAIL_MERGE	Determines whether or not list owners may use the mail-merge feature for their lists.	1.8d-2000b (13.2)
DIST_SECURITY	Boolean value which controls security validation feature for the DISTRIBUTE command. **WARNING:** See Appendix C before changing this value.	1.8d (13)
DKIM_SIGN_ALL	Directs LISTSERV to sign all messages for which it has a suitable DomainKeys private key.	14.5
EMBEDDED_MAIL_MERGE	Boolean value which determines whether or not LISTSERV may use an SMTP mailer other than LSMTP to send mail-merge messages.	14.4
FILEDISK	The filemode of the DEFAULT disk to be used for storing files via a PUT command	
FILEMAXL	The maximum number of lines for any incoming *non-mail* file to be accepted	
FILTER_ALLOW	Defines users or classes of users who should be exempt from LISTSERV's standard filter	1.8d (13)
FILTER_ALSO	Defines users or classes of users who should not be allowed to post to any list on the server.	1.8b (12)
FIOC_INUSE_RETRY	Defines the number of seconds LISTSERV will try to	

	open a file locked by an external process	
FIOC_TARGET	Defines (in kilobytes) the "target size" for LISTSERV's file cache.	
FIOC_TRIM	Defines (in kilobytes) the threshold at which point LISTSERV should start aggressively trimming the cache.	
FIOC_WARNING	Defines (in kilobytes) the cache size at which LISTSERV should write a warning to the console log.	
FOREIGN_ANTI_VIRUS	Add anti-virus protection for those Windows sites that for policy or other reasons must run anti-virus systems other than F-Secure on their servers.	14.4
GETQWAIVE	Internet addresses of persons to be granted an "infinite" GET quota	
HIDE_TRACEBACK	Determines whether or not error tracebacks are shown to non-postmasters.	14.3
IGNORE	(VM only) A list of userid@nodes whose messages and files are to be ignored	
IGNORE_EMAIL_CASE	Provided for DBMS lists with UEMAIL fields to work around RFC821 requirement that an email address local-part must be evaluated case-sensitively.	14.1
IGNORE_EXTERNAL_PRIME	Boolean value determining whether or not LISTSERV will ignore the PRIME setting on incoming DISTRIBUTE jobs.	1.8d (13)
INDEX_VIA_GETPOST	On VM, determines whether INDEX subscriptions use **GETPOST** or old-style database jobs by default.	1.8c (12)
INSTPW	The optional local "installation password" associated with the INSTALL command	
JOB_STAT_DEFAULT	Boolean value determining whether or not the "Summary of resource utilization" is generated or suppressed in a CJLI JOB command response.	1.8d (13)
LICENSE_WARNING	Toggles license warnings on and off. **WARNING:** See Appendix C before changing this value.	
LIST_ADDRESS	Default value for the "List-Address=" keyword	
LIST_EXITS	Filenames of executable files that can be defined as exits by an "Exit=" list header keyword	1.8a (10)
LMCPUT	Boolean variable indicating how PUT commands for datafiles associated with the LMC FAC are handled	
LOCAL	a list of nodes to be associated with the hardcoded LCL FAC. Also used as the default for the "Local=" list keyword	
MAILER	Internet address of the local mailer	
MAILMAXL	the maximum number of lines for an incoming MAIL file to be accepted	
MAXBSMTP	Maximum number of 'RCPT TO:' lines per BSMTP file sent to the mailer	
MAX_CONSECUTIVE_SUBS	The maximum number of lists to which consecutive subscription attempts will be accepted from a given subscriber, to prevent "spoofing" attacks.	14.3
MAXDISTL	(HPO only) The maximum number of recipients to be listed in the LISTSERV console log as recipients of an SMTP job	
MAXDISTN	Maximum number of recipients in forwarded DISTRIBUTE jobs	
MAXGET	Maximum number of GET requests a user can make per day	
MAXGETK	Maximum number of kilobytes of data a user may obtain a day via GET requests.	
MDISK.cuu	Information about library minidisks	
MSGD	Userid of the virtual machine running a RFC1312/MSP server, if "Internet TELL" support is	

	desired	
MYDOMAIN	The list of domain names which are equivalent to NODE--e.g., MX addresses, CNAMEs, etc.	
MYORG	Short organization name that appears in the RFC-822 "Sender:" line.	
NDMAIL	Whether to send mail to the local MAILER in Netdata format	
NODE	Internet address of this LISTSERV host	
NO_NJE_JOBS	Directs a VMS™ server running in NJE mode to send all outgoing server-to-server requests via the Internet	
OCI_*	Three configuration variables under the **OCI_*** rubric are available for use with the DBMS/Mail Merge.	1.8d (13)
ODBC_*	Three configuration variables under the **ODBC_*** rubric are available for use with the DBMS/Mail Merge.	1.8d (13)
OFFLINETHR	Defines the system and RSCS spool thresholds for automatic offline/online control	
PLAIN_FROMLINE	Controls the "verboseness" of LISTSERV's administrative From: line.	14.3
POSTMASTER	A list of userid@nodes, of which the first one is the "main" postmaster (to receive transferred files).	
PRIMETIME	Defines the "prime time" for your node	
QUALIFY_DOMAIN	Defines domain to be appended to all non-qualified addresses	1.8b (11)
RESMODES	Defines a list of filemodes which are to be considered as "reserved" and never available for dynamic ACCESS	
RSCS	A list of local userids which must be treated as RSCS virtual machines	
RUNMODE	The mode (NETWORKED, STANDALONE, or TABLELESS) that LISTSERV runs in.	
SEARCH_DISABLED	Determines whether or not the (new) SEARCH command is enabled.	1.8d (13)
SMTP_FORWARD	The Internet hostname of the server to which all outgoing SMTP mail should be forwarded for delivery	1.8b (11)
SMTP_FORWARD_n	Defines n number of "SMTP workers" used to split up the SMTP forwarding load	1.8b (11)
SMTP_LISTENER_IP	Dotted-decimal IP address which sets the IP address to which the SMTPL.EXE "listener" will bind at boot time.	
SMTP_LISTENER_PORT	Integer value which sets the port number to which the SMTPL.EXE "listener" will bind at boot time	
SMTP_RESET_EVERY	Directs LISTSERV to reset open SMTP connections every n minutes	1.8b (11)
SORT_RECIPIENTS	Determines whether or not to sort recipients in the RFC821 mail envelope	1.8c (12)
SPAM_ALERT	Determines whether or not spam reports are sent to the postmaster	14.2
SPAM_DELAY	Sets the server-wide value (in minutes) for the anti-spam quarantine period	1.8c (12)
SPAM_EXIT	Sets the name of the user-provided exit program used to call a third-party spam scanner.	14.3
SPAM_MAXSIZE	Sets the maximum size, in kilobytes, of any message to be handled by the spam scanner. Messages over the specified size are not scanned.	14.3
Spam Blackists and Whitelists	How to set up spam blacklists and whitelists (14.3)	14.3
SSI	Flag telling LISTSERV that it runs in a SSI system	
STARTMSG	Recipients of start and stop messages	

STOREPW	The password to be used by postmasters when executing CP/CMS commands and when storing files in the server by means of the PUTC command	
SYSTEM_CHANGELOG	Enables a system-level changelog	1.8d-2000b (13.2)
TCPGUI_IPADDR	Defines the IP address used by the TCPGUI interface	
TCPGUI_PORT	Defines the port number used by the TCPGUI interface	
TRAPIN	List of userid@node templates from whom LISTSERV should never accept mail	
TRAPOUT	List of userid@node templates to whom LISTSERV should never send mail	
TUNE_MANY_LISTS	(HPO only) Enables a suite of HPO functions that can markedly speed up operations on servers with many lists (>100).	14.3
UODBC_*	Three configuration variables under the UODBC_* rubric are available for use with DBMS/Mail Merge.	14.5
USE_LSMTP_MAIL_MERGE	Determines whether or not LISTSERV will use LSMTP's mail-merge functions for enhanced performance (requires LSMTP Classic version 1.1b or higher). OBSOLETE as of 14.5, see EMBEDDED_MAIL_MERGE instead.	
VM30091	Indicates whether or not the VM30091 message suppression functions are available	
WEB_BROWSER_CONFIRM	Indicates whether or not LISTSERV should require "OK" confirmation for commands sent from WWW browsers.	1.8c (12)
WWW_ARCHIVE_CGI	The (preferably) relative URL that leads to the WWW archive CGI script. (This is a URL, not an OS path name.)	1.8c (12)
WWW_ARCHIVE_DIR	The full OS path name to the WWW archive directory	1.8c (12)
WWW_AUTHINFO_DISABLE	Disable/enable the web archive interface's IP address verification function	1.8d (13)
XFERTO	Userid of the virtual machine to which files found in the lists readers should be transferred	

ALL_REQUEST_ALLOWED_USERS

Platforms

LISTSERV 1.8e and following.

Abstract

Specifies non-POSTMASTER users who are allowed to post to the ALL-REQUEST alias.

Example

```
VM:    ALL_REQUEST_ALLOWED_USERS = 'joe@example.com pete@abc.unix.example.org'
VMS:   ALL_REQUEST_ALLOWED_USERS "joe@example.com pete@abc.unix.example.org"
Unix:  ALL_REQUEST_ALLOWED_USERS="joe@example.com pete@abc.unix.example.org"
Win:   ALL_REQUEST_ALLOWED_USERS=joe@example.com pete@abc.unix.example.org
```

Details

In LISTSERV 1.8e and following, the ability to post to the ALL-REQUEST alias, which expands to all non-quiet list owners, has been restricted as follows:

1. The site configuration variable, ALL_REQUEST_ALLOWED_USERS, can be used to specify who can mail to ALL-REQUEST. This variable uses the same syntax as POSTMASTER=, in other words, a space-separated list of userid@host specifications.

2. Postmasters are always allowed to mail to ALL-REQUEST, even when not listed in ALL_REQUEST_ALLOWED_USERS.

3. The determination is made conclusively on the RFC821 MAIL FROM because:

   a. This avoids dealing with header errors. Header error = don't know who sent this = discard silently = unhappy admin who had to send an urgent notice to all his owners.

   b. The main point of this change is to block spammers, and spammers typically have non-working or null MAIL FROM: addresses. Needless to say, null doesn't pass the test.

4. When the MAIL FROM is not null, the REQNAK1 mail template form is sent when the message is rejected.

Default Value

Null string.

Wildcards

Not allowed.

ANTI_VIRUS

Platforms

LISTSERV 1.8e Classic and later running on Windows NT/2000 or Linux (Intel). Other OS platforms may be supported later.

Abstract

Turns LISTSERV's real-time anti-virus scanning on or off. Anti-virus scanning of all messages passing through LISTSERV mailing lists was introduced in version 1.8e. See the relevant section in the Site Manager's Manual for LISTSERV for further details.

Example

```
VM:   ANTI_VIRUS = 0
VMS:  ANTI_VIRUS 0
Unix: ANTI_VIRUS=0
Win:  ANTI_VIRUS=0
```

Default Value

If the supported anti-virus system is detected, defaults to 1 (enabled), otherwise to 0 (disabled).

Wildcards

Not allowed.

AVS_DROP_SPAM
AVS_DROP_VIRUS

Platforms

All version 14.4 and later using the LISTSERV AVS to scan mail for spam or viruses

Abstract

Two variables that may be set on servers that submit mail to the external LISTSERV AVS for virus and spam scanning, which can help deal with DoS mail-bombing attacks.

Examples (using AVS_DROP_SPAM)

```
VM:   AVS_DROP_SPAM = 1
VMS:  AVS_DROP_SPAM "1"
Unix: AVS_DROP_SPAM=1
Win:  AVS_DROP_SPAM=1
```

Details

Two AVS-specific options, AVS_DROP_SPAM and AVS_DROP_VIRUS, are available to deal with mail flooding that may result from a mail-bombing attack on a LISTSERV site protected by an external LISTSERV Anti-Virus Station (AVS).

Both options, which default to 0, can be set to 1 to tell an AVS server to drop messages on the floor if they turn out to be spam or infected with a virus. The normal behavior is to return the message to the primary LISTSERV instance, which can then log it and decide what to do with it. In the case where the primary instance is configured to drop the message, it may be more efficient to drop it at the AVS.

Please note carefully that this works only with mail submitted to an AVS server. It does not have any effect on mail scanned locally.

Default Value

`0` (that is, disabled)

BITNET_ROUTE

Platforms

All

Abstract

Defines the Internet hostname of a machine that will be used to route mail to BITNET addresses. If your organization is connected to BITNET, you may want to use the hostname of your main BITNET system for best turnaround time. Otherwise the default value is suitable.

Example

```
VM:   BITNET_ROUTE = 'BITMAIL.LSOFT.COM'
VMS:  BITNET_ROUTE "BITMAIL.LSOFT.COM"
Unix: BITNET_ROUTE="BITMAIL.LSOFT.COM"
Win:  BITNET_ROUTE=BITMAIL.LSOFT.COM
```

Default Value

If not explicitly defined, this keyword defaults to BITMAIL.LSOFT.COM.

Wildcards

Not allowed.

BOUNCES_TO

Platforms

All

Abstract

Tells LISTSERV where to send bounces that are not related to any specific list (i.e., bounces that should never have been sent to LISTSERV to begin with). This includes almost every situation in which LISTSERV thinks it has detected a bounce and cannot determine which list it is for, along with messages about served-out users. It does *not* include mail sent to the `owner-listserv` mailbox, which will still go to the LISTSERV maintainer(s).

Example

```
VM:   BOUNCES_TO = 'FOO@VM.EXAMPLE.EDU'
VMS:  BOUNCES_TO "FOO@VM.EXAMPLE.EDU"
Unix: BOUNCES_TO="FOO@VM.EXAMPLE.EDU"
Win:  BOUNCES_TO=FOO@VM.EXAMPLE.EDU
```

Default Value

Defaults to the value of the `POSTMASTER` variable.

Wildcards

Not allowed.

CHANGELOG_DBMS

Platforms

All platforms under which LISTSERV's DBMS features are supported, starting with LISTSERV 1.8e.

Abstract

In LISTSERV 1.8e and later, tells LISTSERV to mirror a copy of all or selected changelogs to DBMS.

Examples

```
VMS:     CHANGELOG_DBMS "ALL"
         CHANGELOG_DBMS "SYSTEM BIGLIST-L"
Unix:    CHANGELOG_DBMS="ALL"
         CHANGELOG_DBMS="SYSTEM BIGLIST-L"
Win:     CHANGELOG_DBMS=ALL
         CHANGELOG_DBMS=SYSTEM BIGLIST-L
```

Details

LISTSERV 1.8e allows a copy of change logs to be stored in a DBMS. This requires a pre-existing DBMS connection (see the Developer's Guide for LISTSERV, chapter 4, if you need guidance), and is controlled by three new site configuration parameters, and a table that must be created manually. The parameters, which are defined in LISTSERV's site configuration file, are `CHANGELOG_DBMS`, `CHANGELOG_DBMS_TABLE`, and `CHANGELOG_DBMS_CONNECTION` . The first two are mandatory while the third is optional.

Note: It is not possible for LISTSERV to write the changelog in multiple different tables based on various combinations of parameters. This can be accomplished on the DBMS side with a stored procedure if required.

Important: The DBMS copy is just that, a copy. The disk files are still generated. Naturally if they are not needed they may periodically be erased with a script.

`CHANGELOG_DBMS` controls which entries are to be copied to the DBMS. Note that only entries that are actually generated can be copied. If a given change-log is disabled, it will not go to the DBMS even if you request it in this variable.

The value can be ALL or a space-separated combination of SYSTEM, NOLIST (matches any NOLIST-xxx), LISTS (matches any list) or the names of individual lists.

Default Value

Not set (null string).

Wildcards

Not allowed.

See also

`CHANGELOG_DBMS_TABLE`, `CHANGELOG_DBMS_CONNECTION`

CHANGELOG_DBMS_CONNECTION

Platforms

All platforms under which LISTSERV's DBMS features are supported, starting with LISTSERV 1.8e.

Abstract

Used in conjunction with CHANGELOG_DBMS. Sets the DBMS connection characteristics for mirroring changelog data to DBMS. This parameter is optional if CHANGELOG_DBMS is used and does not normally need to be set.

Example

```
VMS:     CHANGELOG_DBMS_CONNECTION "ODBC MYSERVER"
Unix:    CHANGELOG_DBMS_CONNECTION="ODBC MYSERVER"
Win:     CHANGELOG_DBMS_CONNECTION=ODBC MYSERVER
```

Details

This contains two optional space separated words:

1. The type of driver to be used (CLI, OCI, ODBC). This defaults to your system default driver type.
2. The server to connect to (similar to SERVER=). This defaults to the empty string ie the default server.

Default Value

Not set (the defaults are as noted in the details).

See also

CHANGELOG_DBMS

CHANGELOG_DBMS_TABLE

Platforms

All platforms under which LISTSERV's DBMS features are supported, starting with LISTSERV 1.8e.

Abstract

Used in conjunction with CHANGELOG_DBMS. Sets the DBMS table characteristics for mirroring changelog data to DBMS. This parameter is required to be set if CHANGELOG_DBMS is used.

Example

```
VMS:     CHANGELOG_DBMS_TABLE "CHANGELOG TIMESTAMP LISTNAME RECORDTYPE PARAMETERS"
Unix:    CHANGELOG_DBMS_TABLE="CHANGELOG TIMESTAMP LISTNAME RECORDTYPE PARAMETERS"
Win:     CHANGELOG_DBMS_TABLE=CHANGELOG TIMESTAMP LISTNAME RECORDTYPE PARAMETERS
```

Details

This contains five space-separated names:

1. The name of the table in which to store changelog entries.
2. The name of a time-stamp column in which to write the current date and time. This must be a DATE (Oracle), TIMESTAMP (DB2), DATETIME (SQL Server), or whatever else can store both date and time. This cannot be a character string.
3. The name of a VARCHAR or equivalent column storing the name of the list. The maximum size depends on the list names you choose, but it should be at least 40.
4. The name of a VARCHAR column storing the record type (BOUNCE, etc). This should probably be around 40.
5. The name of a VARCHAR column storing the parameters. This ought to be 256 or so.

If any of these parameters is missing, the setting is ignored.

Default Value

Not set (null string).

See also

CHANGELOG_DBMS

CHECKMDISK

Platforms

VM only

Abstract

Defines the list of library minidisks to be checked at startup.

Example

```
CHECKMDISK = '191 192'
```

Details

This parameter defines the LISTSERV "library" minidisks, i.e., minidisks on which LISTSERV might have to read or write list archives or library files (as opposed to "system" files like PROFILE EXEC or DATABASE FILE which are manipulated by LISTSERV but are not apparent to the users).

Default Value

None. This parameter must be set explicitly.

Wildcards

Not allowed.

See also

MDISK.cuu

CLI_*

Platforms

AIX, HP-UX, Linux (x86 only), Solaris. LISTSERV Classic only.

Details

Three configuration variables under the `CLI_*` rubric are available for use with the DBMS/Mail Merge functionality introduced in LISTSERV 1.8d. Please see the *Developer's Guide for LISTSERV* (available separately) for documentation.

CMDPIPE_HOSTNAME

Platforms

Windows NT/2000 only

Abstract

Defines the default hostname used by the LCMD interface

Example

```
CMDPIPE_HOSTNAME=EXAMPLE.COM
```

Details

This variable allows you to specify the hostname used when you communicate with LISTSERV via the LCMD interface. For instance your LISTSERV host may be named LISTSERV.MYCORP.COM, in which case LCMD will identify you to LISTSERV as *userid*@LISTSERV.MYCORP.COM. This is fine as long as you have added this address to the POSTMASTER variable, but privileged commands will probably fail if you haven't (for instance you may be listed as *userid*@MYCORP.COM instead, which works fine via mail).

In order to simplify the issue and not have to code extra userids into the POSTMASTER variable just to be able to use LCMD, you can simply set CMDPIPE_HOSTNAME=MYCORP.COM and LCMD will issue commands from userid@MYCORP.COM instead of the default.

This setting is particularly useful for large list providers such as L-Soft itself, where there are many different nodes all operated by the same small group of people, and it would otherwise be difficult to keep track of personnel changes across the "farm" of machines.

Default Value

Defaults to the DNS A name of the LISTSERV host.

Wildcards

Not allowed.

CMSNAME

Platforms

VM only

Abstract

The name of the CMS system to be used on IPL commands.

Example

```
CMSNAME = 'CMS'
```

Details

This parameter defines the LISTSERV "library" minidisks, i.e., minidisks on which LISTSERV might have to read or write list archives or library files (as opposed to "system" files like PROFILE EXEC or DATABASE FILE which are manipulated by LISTSERV but are not apparent to the users).

Default Value

The standard value for your system.

Wildcards

Not allowed.

CRASH_MONITOR

Platforms

OpenVMS, Windows NT/2000

Abstract

Defines the address to which LISTSERV will send crash reports upon restart.

Example

```
VMS:   CRASH_MONITOR "CRASHLST@LISTSERV.EXAMPLE.COM"
NT:    CRASH_MONITOR=CRASHLST@LISTSERV.EXAMPLE.COM
```

Default Value

Defaults to all non-quiet POSTMASTERs.

Details

By default, LISTSERV crash reports are mailed to the LISTSERV maintainer(s). You can change the destination of these reports by adding a CRASH_MONITOR variable to your site configuration file (SITE.CFG for NT, SITE_CONFIG.DAT for VMS) with the e-mail addresses to which the report should be mailed. Note that CRASH_MONITOR replaces the entire recipient list, so make sure that all the necessary addresses are listed. This configuration variable follows the same syntax rules as POSTMASTER.

Please do not add L-Soft mailboxes to CRASH_MONITOR without checking with our support group first. While we will be happy to receive these reports, we want to make sure that they are sent to the addresses where we can process them most efficiently. In particular, these reports should never be mailed to a support engineer's personal mailbox. Instead, we use special addresses where these reports are logged for future reference.

Wildcards

Not allowed.

CREATEPW

Platforms

All

Abstract

Defines the password used to validate the creation of new lists.

Example

```
VM:   CREATEPW = 'SECRET'
VMS:  CREATEPW "SECRET"
Unix: CREATEPW="SECRET"
Win:  CREATEPW=SECRET
```

Details

Starting with LISTSERV 14.3, this setting is obsolete and deprecated, as all POSTMASTER-restricted commands can now be validated with the postmaster's personal password. Also starting with LISTSERV 14.3, setting CREATEPW to the special value `*NOPW*` tells LISTSERV to disable CREATEPW entirely.

Default Value

None. This parameter must be set explicitly.

Wildcards

Not allowed.

DATABASE

Platforms

VM only

Abstract

Boolean value that determines whether or not the database (archive search) functions are enabled. 0= no, 1= yes.

Example

```
DATABASE = 0
```

Default Value

```
DATABASE = 1
```

DBMS_*

Platforms

VMS, unix (some), Windows NT/2000. LISTSERV Classic only.

Details

Several configuration variables under the `DBMS_*` rubric are available for use with the DBMS/Mail Merge functionality introduced in LISTSERV 1.8d. Please see the *Developer's Guide for LISTSERV* (available separately) for documentation.

DBRINDEX

Platforms

All

Abstract

Determines whether or not LISTSERV's database functions use reverse indexing. Please see the 1.8c release notes for a full discussion of whether or not you should use reverse indexing.

Example

```
VM:   DBRINDEX = 0
VMS:  DBRINDEX "0"
Unix: DBRINDEX=0
Win:  DBRINDEX=0
```

Default Value

For VM and VMS: `DBRINDEX=0`
For unix and Windows: `DBRINDEX=1`

DEBUG_LOG_TCPGUI

Platforms

All non-VM, starting with LISTSERV 14.3

Abstract

Show fully-logged TCPGUI commands for debugging purposes.

Example

```
VMS:  DEBUG_LOG_TCPGUI "1"
unix: DEBUG_LOG_TCPGUI=1
Win:  DEBUG_LOG_TCPGUI=1
```

Details

By default, LISTSERV truncates overly long TCPGUI command lines to make logs less cumbersome. A new site configuration file variable called DEBUG_LOG_TCPGUI is available, defaulting to 0.

When set to 1, all commands starting with 'X-' are logged with no size limit.

Enabling TCPGUI debugging makes logs very large. Therefore L-Soft does not recommend leaving this option enabled for long periods. It should be used for debugging only and disabled after troubleshooting has been completed.

Default Value

The default is 0, that is, do not log the full TCPGUI command line.

DEBUG_SHOW_PW

Platforms

All, starting with LISTSERV 14.3

Abstract

Debug setting to display passwords in LISTSERV's log file

Example

```
VM:   DEBUG_SHOW_PW = 1
VMS:  DEBUG_SHOW_PW "1"
unix: DEBUG_SHOW_PW=1
Win:  DEBUG_SHOW_PW=1
```

Details

Prior to LISTSERV 14.3, LISTSERV always logged passwords received in plain text along with the commands they authenticated. To prevent unintentional password disclosure when e-mailing logs for troubleshooting purposes, LISTSERV now suppresses all passwords in its logs. Passwords are replaced with the text "[redacted]". This applies not only to the **PW=** keyword, but to the **PW** and **PWC** commands, and the internal **X-PWADD** command used by the web interface.

As it is recognized that sometimes passwords need to be seen for debugging purposes, especially on development servers, this debugging option has been added to override the new behavior.

Default Value

The default is 0, that is, obfuscate passwords in the log.

DEFAULT_CHANGELOG_PERIOD

Platforms

Non-VM

Abstract

Sets the default change-log rotation period in LISTSERV 1.8e and later.

Example

```
VMS:  DEFAULT_CHANGELOG_PERIOD "MONTHLY"
Unix: DEFAULT_CHANGELOG_PERIOD="MONTHLY"
Win:  DEFAULT_CHANGELOG_PERIOD=MONTHLY
```

Details

In LISTSERV 1.8e and following it is possible to automatically rotate list and system change-log files on a periodic basis. This site configuration variable allows site administrators to set a server-default rotation period, which can be one of the following: `SINGLE` (the 1.8d behavior), `YEARLY`, `MONTHLY`, `DAILY`, or `WEEKLY`.

Default Value

For backward compatibility, the default is `SINGLE`.

DEFAULT_LANGUAGE

Platforms

All

Abstract

This variable sets the defaults for the Language= keyword for all lists on the server.

Examples

```
VM:   DEFAULT_LANGUAGE = 'ESPANOL'
      DEFAULT_LANGUAGE = 'NOHTML'
      DEFAULT_LANGUAGE = 'FRANCAIS EXCHANGE'
VMS:  DEFAULT_LANGUAGE "ESPANOL"
unix: DEFAULT_LANGUAGE="ESPANOL"
Win:  DEFAULT_LANGUAGE=ESPANOL
```

Details

As with other site-level configuration variables, this is a space-separated list of parameters. DEFAULT_LANGUAGE is particularly useful for servers running in non-English-speaking countries and/or with few lists in English. In 1.8d and following, this variable allows you to define a default national language mail template file for all lists (thereby eliminating the need to code "Language=" in all lists in order to make them use the national language template). Setting this variable to any value other than "ENGLISH" assumes that you have provided a national language mail template file (see the entry for "Language=" in Appendix B for details). If this variable is left unset or is set to "ENGLISH", LISTSERV uses the DEFAULT MAILTPL file (in other words there is no ENGLISH MAILTPL).

You can also change the list-level defaults for the NOHTML, HTML, and EXCHANGE options if desired.

Default Value

Not set (the default settings for the Language= list header keyword are in force).

DEFAULT_LOOPCHECK

Platforms

All

Abstract

This variable sets the default for the Loopcheck= keyword for all lists on the server.

Examples

```
VM:    DEFAULT_LOOPCHECK = 'FULL'
VMS:   DEFAULT_LOOPCHECK "FULL"
unix:  DEFAULT_LOOPCHECK="FULL"
       export DEFAULT_LOOPCHECK
Win:   DEFAULT_LOOPCHECK=FULL
```

Details

In LISTSERV 14.5, the loopcheck heuristic defaults to "Loopcheck= Normal" (which is a new option to that keyword setting) if there is no explicit "Loopcheck=" setting in the list header. This is a backward-compatible change that eliminates from the default test suite the test that results in the error *"Sender:", "From:" or "Reply-To:" field pointing to the list*.

This test was eliminated because the kinds of loops it was designed to prevent no longer occur on a typical discussion list; most MTAs that were responsible for them have long since been fixed. On the other hand, there are still exceptions, and this test remains the only reliable way to catch these kinds of loops. To put things in perspective, whenever you post something to a discussion list and you, the poster, receive a notice that the mailbox does not exist, it is a signal that the list may be at risk for this type of loop. Nowadays, this almost never happens, but there are still lists where it does. In addition, the larger the list, the higher the risk.

To revert to the pre-14.5 behavior at the server level, set DEFAULT_LOOPCHECK=FULL as shown in the examples above. This setting may then be overridden at the list level if desired.

In the future, it is anticipated that other tests found to be obsolete will be moved from the "Normal" suite to the "Full" suite, for the same backward compatibility.

Default Value

NORMAL

DEFAULT_MISC_OPTIONS

Platforms

All

Abstract

This variable sets the default for the Misc-Options= keyword for all lists on the server.

Examples

```
VM:    DEFAULT_MISC_OPTIONS = '+IGNORE_EMAIL_CASE SUPPRESS_APPROVED_BY'
VMS:   DEFAULT_MISC_OPTIONS "+IGNORE_EMAIL_CASE SUPPRESS_APPROVED_BY"
unix:  DEFAULT_MISC_OPTIONS="+IGNORE_EMAIL_CASE SUPPRESS_APPROVED_BY"
       export DEFAULT_MISC_OPTIONS
Win:   DEFAULT_MISC_OPTIONS=+IGNORE_EMAIL_CASE SUPPRESS_APPROVED_BY
```

Details

A DEFAULT_MISC_OPTIONS variable is now available for use in the site configuration file. When populated with one or more of the available values for the Misc-Options= list header keyword, it sets a default for all lists that do not already have a Misc-Options= setting. If a list has a Misc-Options= keyword, the site-level setting is normally overridden. For instance, if you have (using Windows site.cfg for an example)

```
DEFAULT_MISC_OPTIONS=IGNORE_EMAIL_CASE
```

and a given list has

```
* Misc-Options= RESPECT_EMAIL_CASE
```

then the list-level option overrides the site-level option.

You may also prefix options in DEFAULT_MISC_OPTIONS with either a plus or minus sign (no space between the +/- sign and the name of the option). As before, an unprefixed option is active if the list owner did not specify any "Misc-Options=" keyword, and is otherwise ignored.

An option prefixed with a plus sign is always active, for every list. There is no way for the list owner to turn it off. Therefore, if we modified our example above to read

```
DEFAULT_MISC_OPTIONS=+IGNORE_EMAIL_CASE
```

then that would always override any list-level setting.

Likewise, an option prefixed with a minus signed is prohibited and will be ignored if specified in the list header. Therefore something like

```
DEFAULT_MISC_OPTIONS=-NO_SPAM_CHECK
```

would disallow list owners from disabling the spam check for their list by setting "Misc-Options= NO_SPAM_CHECK".

DOCUMENTED RESTRICTION: Using the plus and minus prefixes together for the same option (for instance, "DEFAULT_MISC_OPTIONS=+-NO_SPAM_CHECK") will produce unpredictable results and is not supported or recommended. In the example case, there is no good reason to both (a) force no spam check for all lists and also (b) ignore it if specified in a given list header; all that's needed in that case is the plus prefix.

Multiple options for DEFAULT_MISC_OPTIONS are specified in the usual space-separated manner.

Default Value

Not set (the default settings for the `Misc-Options=` list header keyword are in force).

DEFAULT_PROBE

Platforms

All

Abstract

This variable sets defaults for passive probing.

Examples

```
VM:   DEFAULT_PROBE = '10 500'
      DEFAULT_PROBE = 21
VMS:  DEFAULT_PROBE "10 500"
      DEFAULT_PROBE "21"
unix: DEFAULT_PROBE="10 500"
      DEFAULT_PROBE=21
Win:  DEFAULT_PROBE=10 500
      DEFAULT_PROBE=21
```

Details

In 1.8d and following, a feature called "passive probing" is enabled by default for lists under a certain size. This variable controls two aspects of passive probing. The first parameter to this variable is the length of the default passive probing cycle in days.

Because passive probing is very resource-intensive, above a certain list size it is disabled by default. The second (and optional) parameter to this variable is the list size beyond which passive probing is disabled unless you explicitly enable it in the Auto-Delete= list header keyword.

Since probes do not work with sendmail, the default under unix is to disable passive probing altogether.

Default Value

All platforms except unix:	`DEFAULT_PROBE=30 1000`
Unix only:	`DEFAULT_PROBE=0`

DEFAULT_SPLIT

Platforms

All

Abstract

Provides a default value for the "SPLIT=" command line keyword, causing files ordered through the GET command to be automatically split into smaller "chunks". This option can be useful if LISTSERV is behind a firewall or other central mail gateway with an unusually low maximum message size limit. Unit: kilobytes.

Example

```
VM:   DEFAULT_SPLIT = 250
VMS:  DEFAULT_SPLIT "250"
unix: DEFAULT_SPLIT=250
Win:  DEFAULT_SPLIT=250
```

Default Value

Not set - messages are not split unless specifically requested by the user.

DELAY

Platforms

VM only

Abstract

The delay between two reader-scan operations. Increasing it will reduce CPU time consumption but will increase response time.

Example

```
DELAY = 250
```

Default Value

Standard value.

DIAGD4

Platforms

VM only

Abstract

Indicates whether LISTSERV should use diagnose **`X'D4'`** to mimic the RSCS origin on files it DISTRIBUTEs. Boolean value. Requires VM/SP release 5 and privilege class B.

Example

```
DIAGD4 = 1
```

Default Value

```
DIAGD4 = 0
```

DIST_ALLOWED_USERS

Platforms

All

Abstract

In 1.8d and following, space-separated list of non-POSTMASTER users who are to be allowed to use DISTRIBUTE. SEE DETAILS.

Example

```
VM:   DIST_ALLOWED_USERS = 'joe@unix.host.com alma@univ.edu'
VMS:  DIST_ALLOWED_USERS "joe@unix.host.com alma@univ.edu"
unix: DIST_ALLOWED_USERS="joe@unix.host.com alma@univ.edu"
Win:  DIST_ALLOWED_USERS=joe@unix.host.com alma@univ.edu
```

Details

In 1.8d, to secure the DISTRIBUTE command from use by spammers, a security validation feature was added. Only a user defined in POSTMASTER or in DIST_ALLOWED_USERS may issue the DISTRIBUTE command, and the DISTRIBUTE command must be validated with that user's personal LISTSERV password.

Default Value

Not set (null value).

Wildcards

Not allowed.

See also

DIST_SECURITY

DIST_FORWARD
DIST_FORWARD_THRESHOLD

Abstract

In 14.4 and following, enables and tunes DISTRIBUTE "workers". This is a tuning feature for embedded mail merge, also introduced in 14.4.

Example

```
DIST_FORWARD=smtp.example.com 2*smtp2.example.com
DIST_FORWARD_THRESHOLD=200
```

Details

Configuring many SMTP workers through the SMTP_FORWARD_n site configuration parameter will help speed up the delivery of the many individual mail files generated with the embedded mail-merge feature. However, in very high volume situations, it may also be desirable to off-load the work of *generating* the many individual mail files to one or more separate servers.

This can be done in LISTSERV 14.4 by configuring additional LISTSERV instances to be “DISTRIBUTE workers”. The primary LISTSERV instance can quickly divide and delegate the mail-merge responsibility among its workers and get on with its own work while the “workers” take care of the brute-force generation of the individual mail messages, which are then delivered by the workers’ SMTP workers. By using several DISTRIBUTE workers, not only does the primary instance’s availability increase, but the deliveries can be forwarded to the SMTP server(s) much faster because multiple messages are being generated in parallel, instead of one at time.

In most cases, each instance will run on a separate, dedicated server, but this is not a requirement. On a very large server with many CPUs and as many independent disk drives, it could make sense to run multiple instances on the same server.

Each instance requires a separate license. Since this feature is intended for processing large volumes, the primary instance must be running with an HPO license. The “worker” instances must have either HPO licenses (in which case they can also act as standalone LISTSERV hosts), or special SCOPE=DISTWORKER licenses (in which case they are dedicated DISTRIBUTE workers).

DISTRIBUTE workers are enabled with two options set in the primary instance’s site configuration:

1. **DIST_FORWARD_THRESHOLD=nnn**
This setting is optional, and defaults to 100. It defines the minimum number of total recipients to make use of the workers. It would be inefficient to forward a message with one recipient to a worker so that it could deliver it for you.

2. **DIST_FORWARD=[[n*]hostname [[n*]hostname [...]]**
This is a space-separated list of all your DISTRIBUTE workers.

For "hostname", you can specify just the hostname if the userid is LISTSERV. If the userid is not LISTSERV, then you must specify a full RFC822 e-mail address pointing to the USERID= address for that server.

Optionally, you can specify a weight (the "n" parameter), which defaults to 1. The number of recipients sent to a particular server is proportional to its weight. The weight must be greater than zero. If a particular server is specified multiple times, the weights are added.

By default, the primary server also takes one share of recipients. In other words, it has a weight of 1. You can change that using the special hostname “SELF”. So, if you want a double share for your primary LISTSERV because it runs on a bigger server, just add 2*SELF to the list. SELF is the only worker allowed to have a weight of zero. In this case, nothing is processed locally (except for jobs that are below the threshold). For very large mail-merge volumes, 0*SELF is recommended.

A worker cannot delegate jobs to sub-workers, but the primary instance can drive any number of workers.

See also

EMBEDDED_MAIL_MERGE

DIST_OWNER_MAIL_MERGE

Platforms

Non-VM

Abstract

In 1.8d and following, a Boolean value that determines whether or not list owners may use the mail-merge feature for their lists. BE SURE TO READ DETAILS BELOW!

Example

```
VMS:  DIST_OWNER_MAIL_MERGE "1"
unix: DIST_OWNER_MAIL_MERGE=1
Win:  DIST_OWNER_MAIL_MERGE=1
```

Details

DO NOT ENABLE THIS FEATURE UNLESS YOU ARE USING LSMTP AS YOUR OUTBOUND MTA, OR UNLESS YOU HAVE ENABLED EMBEDDED MAIL MERGE!!!!

By default, list owners who are not LISTSERV postmasters may not use the mail-merge features against their own lists (ie, via the "Mail-Merge" button in the web administration interface). Setting this variable to 1 enables this feature for all list owners on the server.

LISTSERV postmasters may always use mail-merge features regardless of the setting of this variable, as it is assumed that they know whether or not LSMTP Classic 1.1b or higher is installed and/or whether or not LISTSERV's embedded mail merge feature is available and enabled, whereas individual list owners may not have this information.

Note that if embedded mail merge is not available (pre-14.4) or is not enabled, the SMTP_FORWARD* variables MUST IMPERATIVELY be pointed to a host which is running LSMTP Classic version 1.1b or higher, since LISTSERV's mail-merge features require LSMTP Classic version 1.1b or higher.

NOTE CAREFULLY that mail-merge operations will fail if the prerequisites are not met, as mail-merge operations use L-Soft proprietary extensions to the BSMTP protocol. If embedded mail merge is not available or enabled, LSMTP MUST be used with mail-merge.

Default Value

0

Wildcards

Not allowed.

DIST_SECURITY

Platforms

All

Abstract

In 1.8d and following, Boolean value which turns off new security validation feature for the DISTRIBUTE command. SEE DETAILS.

Example

```
VM:   DIST_SECURITY = 0
VMS:  DIST_SECURITY "0"
unix: DIST_SECURITY=0
Win:  DIST_SECURITY=0
```

Details

In 1.8d, to secure the DISTRIBUTE command from use by spammers, a security validation feature was added. Only a user defined in POSTMASTER or in DIST_ALLOWED_USERS may issue the DISTRIBUTE command, and the DISTRIBUTE command must be validated with that user's personal LISTSERV password. This feature is turned on by default.

If it is deemed necessary to revert to the pre-1.8d behavior, simply set this variable to zero and restart the server. However this is NOT RECOMMENDED as the original behavior would continue to allow the indiscriminate use of DISTRIBUTE by spammers. L-Soft will not be responsible for any consequences deriving from the disabling of this feature.

Default Value

1

See also

DIST_ALLOWED_USERS

DKIM_SIGN_ALL

Platforms

All

Abstract

In 14.5 and following, Boolean value which tells LISTSERV whether or not to sign all messages for which it has a suitable DomainKeys private key. SEE DETAILS.

Example

```
VM:   DKIM_SIGN_ALL = 0
VMS:  DKIM_SIGN_ALL "0"
unix: DKIM_SIGN_ALL=0
Win:  DKIM_SIGN_ALL=0
```

Details

Starting with version 14.5, LISTSERV supports DomainKeys for outbound mail as described in the Internet draft draft-delany-domainkeys-base-03 . Your LISTSERV Classic or Classic HPO installation must have current maintenance in order to enable this support.

Additionally, LISTSERV HPO for Windows, Linux, and FreeBSD incorporates a DomainKeys Accellerator for faster signing of messages. The accellerator is part of HPO and cannot be disabled without disabling HPO.

Once you have become comfortable with DomainKeys signatures, you may want to have LISTSERV sign *every* message that it generates, regardless of its source. Setting `DKIM_SIGN_ALL=1` in the site configuration file tells LISTSERV to try to sign every message for which it has a suitable private key, as defined in the DKIM_SIGN configuration parameter.

IMPORTANT: Please review *Using LISTSERV with DomainKeys* before enabling this feature. DomainKeys support **MUST** be configured in LISTSERV or message signing will fail.

Default Value

0

See also

Using LISTSERV with DomainKeys

EMBEDDED_MAIL_MERGE

Platforms

All

Abstract

Starting with LISTSERV 14.4, any high-capacity SMTP server can be used to deliver e-mail that uses mail-merge. When operating in this mode, the proprietary mail-merge BSMTP features of LSMTP are *not* required for mail-merge.

Embedded mail-merge is activated in LISTSERV by adding the EMBEDDED_MAIL_MERGE= site configuration variable to the site configuration file and restarting LISTSERV.

Example

```
EMBEDDED_MAIL_MERGE=1
```

Details

Embedded mail-merge is a "brute-force" method. In this mode, each individual merged message is written to LISTSERV's spool as a single-recipient MAIL file. In turn, each individual MAIL file is then delivered by SMTP to the outbound SMTP_FORWARD host.

Since Batch SMTP (BSMTP) cannot be used in the embedded mode, sites with very high volumes of mail may wish to split the load across multiple outbound mail servers using SMTP "workers" (defined with SMTP_FORWARD_n variable settings in the site configuration file). In addition, the new DISTRIBUTE "workers" (detailed in the next section) can be used in very high volume situations to move mail smoothly.

Default Value

`0` (that is, disabled)

FILEDISK

Platforms

VM only

Abstract

The filemode of the DEFAULT disk to be used for storing files with a PUT command.

Example

```
FILEDISK = 'L5'
```

Default Value

```
FILEDISK = 'A5'
```

Wildcards

Not allowed.

FILEMAXL

Platforms

All

Abstract

The maximum number of lines for any incoming *non-mail* file to be accepted. A non-mail file is defined as a piece of mail that contains LISTSERV commands (eg, a bulk `ADD` or `DELETE` job) as opposed to a piece of mail that contains a message (eg, a list posting).

Example

```
VM:   FILEMAXL = 25000
VMS:  FILEMAXL "25000"
unix: FILEMAXL=25000
Win:  FILEMAXL=25000
```

Default Value

`100000`

See also

`MAILMAXL`

FILTER_ALLOW

Platforms

All

Abstract

Blank-delimited list of addresses or wildcards which will be exempted from LISTSERV's standard loop-detection filter.

Example

```
VM:   FILTER_ALLOW = '*@GOODNODE.COM POSTMASTER@SOMEHOST.NET'
VMS:  FILTER_ALLOW "*@GOODNODE.COM POSTMASTER@SOMEHOST.NET"
unix: FILTER_ALLOW="*@GOODNODE.COM POSTMASTER@SOMEHOST.NET"
Win:  FILTER_ALLOW=*@GOODNODE.COM POSTMASTER@SOMEHOST.NET
```

Details

This variable determines what, if any, exemptions are made to the standard set of addresses normally bounced by LISTSERV as part of its loop-checking heuristic. For instances, if you want to allow `POSTMASTER@SOMEHOST.NET` to post to lists like a normal user, you can add that value to this variable. Or if you should decide that any postmaster account anywhere should be allowed to post to lists, you can add the value `POSTMASTER@*` (but please note that L-Soft does not recommend this). `FILTER_ALLOW` works in conjunction with enhancements to the `Filter=` list header keyword; for more information on this interaction please see Appendix B.

Default Value

Not set; adds to LISTSERV's built-in filter.

Wildcards

Allowed.

FILTER_ALSO

Platforms

All

Abstract

Blank-delimited list of problem users who should not be allowed to post or subscribe to any list. This is similar to the "Filter=" list header keyword, but applies to all the lists. The FILTER_ALSO option is usually used to filter out problematic mail gateways, anonymous remailers (if anonymous postings are not desired), troublesome users, *etc.* Please refer to the list owner's guide for more information.

Example

```
VM:   FILTER_ALSO = '*@BADNODE.COM OBNOXUSR@SOMEHOST.NET'
VMS:  FILTER_ALSO "*@BADNODE.COM OBNOXUSR@SOMEHOST.NET"
unix: FILTER_ALSO="*@BADNODE.COM OBNOXUSR@SOMEHOST.NET"
Win:  FILTER_ALSO=*@BADNODE.COM OBNOXUSR@SOMEHOST.NET
```

Details

Sometimes it may be necessary to deny a specific user (or a class of users) access to all the lists hosted by your server. This may be due to policies internal to your organization, technical problems, or simply to lock out an obnoxious user. FILTER_ALSO adds to the standard LISTSERV filter and denies access to all lists on the server.

Default Value

Not set; adds to LISTSERV's built-in filter.

Wildcards

Allowed.

FIOC_INUSE_RETRY

Platforms

VM, OpenVMS, Windows

Abstract

Defines the number of seconds LISTSERV will try to open a file which is locked by an external process.

Example

```
VM:   FIOC_INUSE_RETRY = 15
VMS:  FIOC_INUSE_RETRY "15"
Win:  FIOC_INUSE_RETRY=15
```

Details

Sometimes LISTSERV may try to open a file (such as a list file) while that file is locked by an external process (for instance, a nightly backup program). If the open attempt times out, an error is generated. This functionality is not available on unix because file locking is not implemented.

Default Value

```
10
```

FIOC_TARGET

Platforms

All

Abstract

Defines the "target size" for LISTSERV's file cache (in kilobytes). (Windows sites can set this option through the various "Optimize for..." buttons in the LISTSERV configuration program.) See the LISTSERV Tuning Guide (available from L-Soft at no extra charge) for more information.

Example

```
VM:   FIOC_TARGET = 15000
VMS:  FIOC_TARGET "15000"
unix: FIOC_TARGET=15000
Win:  FIOC_TARGET=15000
```

Details

One of three variables that control how LISTSERV handles its built-in data cache. FIOC_TARGET is the "target size" for the cache. LISTSERV will strive to keep the cache to about that size, but will allow it to grow past this value for short periods of time. LISTSERV expects that it will have fast access (low paging rate) to these FIOC_TARGET kilobytes of cache memory; it does not help to increase this value if your system is memory-constrained.

Default Value

System dependent.

FIOC_TRIM

Platforms

All

Abstract

Defines the threshold (in kilobytes) where LISTSERV should start aggressively trimming the cache. (Windows sites can set this option through the various "Optimize for..." buttons in the LISTSERV configuration program.) See the LISTSERV Tuning Guide (available from L-Soft at no extra charge) for more information.

Example

```
VM:   FIOC_TRIM = 17000
VMS:  FIOC_TRIM "17000"
unix: FIOC_TRIM=17000
Win:  FIOC_TRIM=17000
```

Details

One of three variables that control how LISTSERV handles its cache. `FIOC_TRIM` is the point at which LISTSERV should start aggressively trimming the cache in order to free up virtual storage. Typically this value should be set to `FIOC_TARGET` plus 20% or 512KB (whichever is larger).

Default Value

System dependent.

FIOC_WARNING

Platforms

All

Abstract

Defines the cache size (in kilobytes) at which LISTSERV should write a warning to the console log. (Windows sites can set this option through the various "Optimize for..." buttons in the LISTSERV configuration program.) See the LISTSERV Tuning Guide (available from L-Soft at no extra charge) for more information.

Example

```
VM:   FIOC_WARNING = 20000
VMS:  FIOC_WARNING "20000"
unix: FIOC_WARNING=20000
Win:  FIOC_WARNING=20000
```

Details

One of three variables that control how LISTSERV handles its cache. Under certain circumstances, LISTSERV may not be able to trim the cache right away, either because a cache entry is locked by a routine that maintains pointers to it or because the file is currently open and thus it would be counter-productive to flush the cache entry right away. In such cases, the cache size can continue to grow past `FIOC_TRIM`. When it reaches `FIOC_WARNING`, a warning is displayed on the LISTSERV log. This probably indicates a programming error, or a value of `FIOC_TARGET` which is significantly below the "correct" value for your workload. Typically this value should be set to `FIOC_TARGET` plus 75% or 1MB (whichever is larger).

Default Value

System dependent.

FOREIGN_ANTI_VIRUS

Platforms

Windows only, AVS systems excluded.

Abstract

In 14.4 and following, add anti-virus protection for those Windows sites that for policy or other reasons must run anti-virus systems other than F-Secure on their servers.

Example

```
FOREIGN_ANTI_VIRUS=1
```

Details

Version 14.4 adds anti-virus protection for those sites that for policy or other reasons must run anti-virus systems other than F-Secure on their servers. *This feature is not available on anti-virus stations (AVS).* Support for these “foreign” anti-virus systems is not fully integrated with LISTSERV in the same way that FSAV is, but LISTSERV will give the AV system the opportunity to disinfect messages.

To have LISTSERV messages disinfected by an AV system other than FSAV, add the FOREIGN_ANTI_VIRUS parameter to your site configuration as shown in the example.

The anti-virus system must be installed on the LISTSERV server, and must block creation of a .EXE file containing a virus with an error of “access is denied“.

Use of this feature is *at your own risk*. Because the foreign AV system is not fully integrated with LISTSERV as is FSAV, LISTSERV cannot check that the AV system is up and running and has up-to-date virus definitions. It can only assume that any file that is not denied access does not contain a virus. If you use this feature, it is recommended that you test it thoroughly and set up an external monitoring system to make sure that the anti-virus system is always active when LISTSERV is.

Since the foreign AV system is not integrated with LISTSERV, LISTSERV does not know which viruses have been detected. Anti-virus reports will identify every virus detected as "unknown virus”.

This feature is part of the LISTSERV message scanner, and therefore requires a valid and current maintenance LAK. See http://www.lsoft.com/products/listserv_av.asp for more information.

Default Value

`0` (that is, disabled)

GETQWAIVE

Platforms

VM only; there is no GET quota on any other platform.

Abstract

Internet addresses (space-delimited) of persons to be granted an "infinite" GET quota.

Example

```
GETQWAIVE = 'nathan@example.com holly@example.edu lowell@example.bitnet'
```

Default Value

Empty string.

Wildcards

Not allowed.

HIDE_TRACEBACK

Platforms

All LISTSERV 14.3 and following

Abstract

Boolean value determining whether or not error tracebacks are shown to non-postmasters.

Examples

```
VM:   HIDE_TRACEBACK = 1
VMS:  HIDE_TRACEBACK "1"
unix: HIDE_TRACEBACK=1
Win:  HIDE_TRACEBACK=1
```

Details

It is now possible to suppress tracebacks in error messages by setting the Boolean site configuration variable HIDE_TRACEBACK. When set, the tracebacks are no longer included in error messages, except in certain postmaster-only e-mail messages.

This has no impact on what is written to the LISTSERV log. The traceback is always written to the LISTSERV log.

Default Value

0, that is, do not suppress the tracebacks.

IGNORE

Platforms

VM only

Abstract

A list of Internet addresses (space-delimited) whose messages and files are to be ignored. Usually these are addresses that generate auto-responses that should simply be discarded.

Example

```
IGNORE = 'response@* dirmaint@'node 'reslim@'node
```

Default Value

Empty string.

Wildcards

Allowed.

IGNORE_EMAIL_CASE

Platforms

Non-VM

Abstract

Provided for DBMS lists with UEMAIL fields to work around RFC821 requirement that an email address local-part must be evaluated case-sensitively.

Examples

```
VMS:  IGNORE_EMAIL_CASE "0"
unix: IGNORE_EMAIL_CASE=0
Win:  IGNORE_EMAIL_CASE=0
```

Details

IGNORE_EMAIL_CASE=0 (the default) tells LISTSERV to operate per RFC821 and treat address fields with differently-cased local parts as different addresses.

IGNORE_EMAIL_CASE=1 causes the ADD command to ignore the case of the "local part" of list subscriber entries (that is, the part of the address that is to the left of the "@" sign). Although most modern mail clients are configured to ignore the case of the local-part, this behavior technically violates RFC821 which states that local-parts are considered case-sensitive.

If an entry whose "local part" differs only in case is found in the list during an ADD operation (for instance, JOE@EXAMPLE.COM vs. joe@EXAMPLE.COM), that entry will be assumed to be the entry that was sought, and the address field will be updated to the new case (that is, "JOE@" will be changed to "joe@"). No other change will be made to the entry unless there is a change in the name field, in which case the name field will also be updated.

If there is no change in the address field associated with the entry, no change will be made to the entry (again, unless the name field changes, in which case the entry will be updated).

In either case, when this option is set, a new entry with a different case will NOT be added.

Note the following caveats:

1. Pre-existing duplicates are not automatically removed from lists when this option is set.
2. Because ADD updates the case of entries, it is possible to end up with multiple entries that have exactly the same case.
3. The only real way to de-dupe a given address is to DELETE and then re-ADD it.

Other than this, existing duplicate entries work exactly as they did before the option was enabled. Commands that do not add new entries ignore the option.

And finally, it should be carefully noted that the PUT command also ignores the option.

IGNORE_EXTERNAL_PRIME

Platforms

All

Abstract

A Boolean value which determines whether or not LISTSERV will ignore the PRIME setting on incoming DISTRIBUTE jobs.

Example

```
VM:   IGNORE_EXTERNAL_PRIME = 1
VMS:  IGNORE_EXTERNAL_PRIME "1"
unix: IGNORE_EXTERNAL_PRIME=1
Win:  IGNORE_EXTERNAL_PRIME=1
```

Details

This parameter can be used to expedite the delivery of DISTRIBUTE jobs which have an explicit PRIME setting. The point of this is simply to keep the spool clear of DISTRIBUTE jobs that might have a very narrow window (e.g., jobs set for weekend delivery only). Most sites will not need to use this setting, and it is disabled by default.

Default Value

0

INDEX_VIA_GETPOST

Platforms

VM only

Abstract

Starting with 1.8c, the INDEX subscription option now uses the GETPOST command rather than a database job by default. INDEX_VIA_GETPOST can be used to control this default.

Example

```
INDEX_VIA_GETPOST = 0
```

Details

This change was made because the GETPOST command requires fewer resources and is easier to use for non-technical users. Users may of course continue to use their old database jobs to access the list archives. The old behavior can be restored by adding 'INDEX_VIA_GETPOST = 0' to LOCAL SYSVARS.

Default Value

1

INSTPW

Platforms

VM-NJE only

Abstract

The optional local "installation password" associated with the INSTALL command.

Example

```
INSTPW = 'fixmelater'
```

Details

When INSTPW is set to `''`, it indicates that there is no local password and that only the global password is required to install automatic shipments. The server can then be updated by the LISTSERV Master Coordinator without requiring any intervention by the local operation staff (provided that this has not been disabled from the LSV$INST EXEC user exit). A non-null value indicates that the installation of each shipment must be confirmed by means of an "`INSTALL PASSWORD` *`instpw`*" command from either the LMC or the local operation staff.

Default Value

Empty string.

Wildcards

Not allowed.

JOB_STAT_DEFAULT

Platforms

All Classic (no effect under Lite)

Abstract

Boolean value determining whether or not the "Summary of resource utilization" is generated or suppressed in a CJLI JOB command response.

Example

```
VM:   JOB_STAT_DEFAULT = 0
VMS:  JOB_STAT_DEFAULT "0"
unix: JOB_STAT_DEFAULT=0
Win:  JOB_STAT_DEFAULT=0
```

Details

Sets a server-wide default which can be overridden on a single-job basis with the STAT= JOB card option (see the *Developer's Guide for LISTSERV* for more information on JOB cards). L-Soft does not recommend changing this value from the default but recognizes that some sites do not want this information to be sent out. (Under Lite from 1.8d, the summary is suppressed and this variable setting has no effect.)

Default Value

1

Wildcards

Not allowed.

LICENSE_WARNING

Platforms

All

Abstract

Disable LISTSERV's automatic license warnings. **WARNING: SEE DETAILS AND DISCLAIMER BELOW. IT IS NOT RECOMMENDED TO DISABLE LICENSE WARNINGS.**

Example

```
VM:   LICENSE_WARNING = 0
VMS:  LICENSE_WARNING "0"
unix: LICENSE_WARNING=0
Win:  LICENSE_WARNING=0
```

Details

LISTSERV issues a warning to the console and to all non-quiet POSTMASTERs when you reach a usage level corresponding to about 80% of your license points, and issues a similar warning every day for 45 days prior to the license expiration date. Warnings are automatically disabled for small licenses (9 points or less). Warnings can be turned off for higher license point values by using this Boolean variable.

DISCLAIMER: Setting this variable to 0 turns off all license warnings, *including expiration warnings*. L-Soft does not recommend turning this feature off as it is possible for your license to expire or to be at capacity without any warning. ***L-Soft is not responsible for delays or other problems arising from the deactivation of license warnings.***

Default Value

1

LIST_ADDRESS

Platforms

All (obsolete)

Abstract

Obsolete. Default value for the "List-Address=" list header keyword. This keyword does not normally need to be changed on non-VM systems.

Example

```
VM:   LIST_ADDRESS = 'LIST-ID@NJE'
VMS:  LIST_ADDRESS "LIST-ID@NJE"
unix: LIST_ADDRESS="LIST-ID@FQDN"
Win:  LIST_ADDRESS=LIST-ID@FQDN
```

Details

LIST_ADDRESS defines how mailing lists will identify themselves by default. The main purpose of this keyword is to allow BITNET sites to select their NJE address as the primary address, for compatibility. There is no practical application for this keyword under non-VM systems, other than as a migration aid for mainframe BITNET sites moving off of VM.

Default Value

```
LIST-ID@FQDN
```

Wildcards

Not allowed.

LIST_EXITS

Platforms

All

Abstract

A list of filenames of files that can be activated as list exits through the "Exit=" list header keyword. Any suffixes (CMD for Windows NT and BAT for Windows 95, for example) should not be included.

Example

```
VM:   LIST_EXITS = 'EXIT1 EXIT2'
VMS:  LIST_EXITS "EXIT1 EXIT2"
unix: LIST_EXITS="EXIT1 EXIT2"
Win:  LIST_EXITS=EXIT1 EXIT2
```

Details

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.

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). This prevents list owners from causing the invocation of arbitrary executable files through the use of the "Exit=" keyword.

See the *Deveoper's Guide to LISTSERV* for more information on list exits.

Default Value

Empty string (no exits enabled).

Wildcards

Not allowed.

LMCPUT

Platforms

VM only

Abstract

A boolean variable indicating how PUT commands for datafiles associated with the LMC FAC are handled.

Example

```
LMCPUT = 0
```

Details

This variable indicates whether PUT commands for datafiles associated with the LMC FAC must be honored immediately (1) or transferred to the postmaster (0). Note that in the latter case, the postmaster is responsible for updating the file.

Default Value

```
LMCPUT = 1
```

LOCAL

Platforms

All

Abstract

A *space-separated* list of hosts and nodes to be associated with the hard-coded LCL FAC. Also used as the default for the "Local=" list header keyword. You usually want to set this variable to a wildcard pattern matching all the hosts in your organization (or department for large organizations).

Examples

```
VM:   LOCAL = 'PDQ.COM *.PDQ.COM PDQ.NET *.DARNQUICK.ORG'
VMS:  LOCAL "PDQ.COM *.PDQ.COM PDQ.NET *.DARNQUICK.ORG"
unix: LOCAL="PDQ.COM *.PDQ.COM PDQ.NET *.DARNQUICK.ORG"
Win:  LOCAL=PDQ.COM *.PDQ.COM PDQ.NET *.DARNQUICK.ORG
```

Details

Note carefully that LISTSERV evaluates the values associated with LOCAL literally. If you code "`LOCAL = 'XYZ.EDU'`", then users on (for instance) UNIX.XYZ.EDU will not be treated as local users. Likewise, if you code "`LOCAL = '*.XYZ.EDU'`", then users who are addressed as userid@XYZ.EDU will not be considered local users. In order for all users in a given domain to be considered "local" by LISTSERV, you must imperatively code a value for LOCAL that covers all bases--ie, "`LOCAL = 'XYZ.EDU *.XYZ.EDU'`".

Default Value

None. This parameter must be set explicitly.

Wildcards

Allowed.

MAILER

Platforms

VM only

Abstract

RFC822 address of the local mailer.

Example

```
MAILER = 'MAILERT@'node
```

Default Value

```
'MAILER@'node
```

Wildcards

Not allowed.

MAILMAXL

Platforms

All (obsolete except for VM BITNET servers)

Abstract

The maximum acceptable size, in lines, of an incoming mail message. Obsolete; see Details.

Example

```
VM:   MAILMAXL = 15000
VMS:  MAILMAXL "15000"
unix: MAILMAXL=15000
Win:  MAILMAXL=15000
```

Details

Messages larger than MAILMAXL lines are not accepted for processing. For all practical purposes this option is obsolete (it was intended to solve a problem specific to VM BITNET servers) and you should not set it (i.e., you should accept the default) unless you are experiencing this kind of problem. *In other words, if you aren't running VM on BITNET, don't set this! It is preferable to set a Sizelim= keyword in individual list headers if you must restrict postings on the basis of size. Do not use MAILMAXL as a server-level global replacement for Sizelim=.*

MAILMAXL is primarily intended for small machines where there are not enough resources to process very large messages. Rather than attempting to process them and then running out of resources anyway, you can use the MAILMAXL option to reject the message right away.

Note that if MAILMAXL is set, messages coming in that exceed MAILMAXL will be discarded without any notification to the sender. This is *not* the same as the list header keyword setting Sizelim=, which when exceeded *will* cause LISTSERV to send a notification. *However note carefully* that since MAILMAXL is evaluated before the message is actually processed, that is, before Sizelim= can be evaluated, a message that exceeds MAILMAXL (but which may or may not exceed Sizelim= for the list it is being posted to) will be discarded without warning to the sender. This is why it is preferable to restrict postings based on size at the list level rather than at the server level.

Default Value

MAILMAXL defaults to the value of FILEMAXL. Note that under VM this can be more complicated as there typically is an entry for MAILMAXL in one of the SYSVARS files which overrides the default in the MODULE. It is not recommended to set MAILMAXL to a value less than 10000, and LISTSERV will issue a warning at startup if MAILMAXL is set lower than that.

MAXBSMTP

Platforms

All

Abstract

The maximum number of recipients in each message forwarded to the mail delivery system.

Example

```
VM:   MAXBSMTP = 50
VMS:  MAXBSMTP "50"
unix: MAXBSMTP=50
Win:  MAXBSMTP=50
```

Details

If your mail delivery system runs on a unix machine, you should use values in the 50-100 range; smaller values will result in faster delivery, but will use up more system resources. If your mail delivery system runs LSMTP®, you should use a much larger value, such as 1000. With LSMTP®, larger values improve turnaround time and decrease system resource usage. Very large values, however, may exhaust available system storage.

On Windows servers, this is an optimization parameter that you would normally set through the various "Optimize for..." buttons in the configuration program.

Default Value

100

See also

SMTP_FORWARD

MAX_CONSECUTIVE_SUBS

Platforms

All LISTSERV 14.3 and later

Abstract

The maximum number of lists to which consecutive subscription attempts will be accepted from a given subscriber, to prevent "spoofing" attacks.

Example

```
VM:   MAXBSMTP = 50
VMS:  MAXBSMTP "50"
unix: MAXBSMTP=50
Win:  MAXBSMTP=50
```

Details

In previous releases, as a preventative against spoofers adding third parties to hundreds of lists without their knowledge, LISTSERV has had a hard-coded maximum for the number of local lists to which a user could subscribe at any one given time. The limit was set at 50, after which LISTSERV would assume that the subscription requests were coming from a spoofer and would cancel the last 50 subscription requests for the user in question. (To clarify, a user could be subscribed to more than 50 lists on the server, but could not issue more than 50 subscription requests in a row.)

MAX_CONSECUTIVE_SUBS allows site maintainers full control over the limit. The default remains 50. A setting of 0 disables the anti-spoofing filter (which is not recommended).

Default Value

50

MAXDISTL

Platforms

All; HPO only

Abstract

The maximum number of recipients to be listed in the LISTSERV console log as recipients of an SMTP job. When the number of recipients is greater than MAXDISTL, a shorter message is printed in the log.

Example

```
VM:   MAXDISTL = 100
VMS:  MAXDISTL "100"
unix: MAXDISTL=100
Win:  MAXDISTL=100
```

Default Value

```
1000
```

MAXDISTN

Platforms

All

Abstract

The maximum number of recipients in forwarded DISTRIBUTE jobs. You should not modify this value unless instructed to do so by L-Soft.

Example

```
VM:   MAXDISTN = 100
VMS:  MAXDISTN "100"
unix: MAXDISTN=100
Win:  MAXDISTN=100
```

Default Value

```
1000
```

MDISK.cuu

Platforms

VM only

Abstract

Information about library minidisks

Example

```
MDISK.191 = 'A- N L L 85 95 //A(191) is the LISTSERV system minidisk.'
MDISK.192 = 'D- N L L 100 100 //D(192) is LISTSERV''s scratch work disk.'
```

Details

The 'MDISK.' stem defines various parameters associated with LISTSERV "library" minidisks, that is, minidisks on which LISTSERV might have to read or write list archives or library files (as opposed to "system" files like PROFILE EXEC or DATABASE FILE which are manipulated by LISTSERV but are not apparent to the users). The syntax of the assignment is the following:

```
MDISK.cuu = 'acc rw ro nl th1 th2 / wlist / wmsg'
```

The slash signs are used as separators. They allow for more parameters to be specified in future releases while ensuring compatibility.

- CUU is the minidisk address (*three* hexadecimal digits), or the FULL name of the SFS directory (*filepool*:*dirid*), as returned by a QUERY ACCESSED command.

- ACC indicates how the minidisk should be accessed. If it should be accessed permanently at a fixed filemode, just specify its filemode letter (eg 'F') and LISTSERV will access it during startup. If the disk has already been accessed by the PROFILE, append a dash to the filemode letter (as in 'A-') to bypass the ACCESS command. If the disk should be accessed dynamically (ie as the needs arise), specify '*' to let LISTSERV choose the filemode, '*S' to force the filemode to be in the T-Y range (for security reasons) or '*fm' to force a given filemode to be used for the disk (note that A,D and Z are reserved and cannot be specified in this fashion).

- RW indicates what LISTSERV should do if the minidisk is found to be accessed in R/W status. 'N' means that this is a normal condition, 'W' means that a warning message should be issued (see below) and 'L' means that this is a fatal error and LISTSERV should log off.

- RO is similar to RW and describes what happens if the minidisk is R/O; NL specifies what should be done if the minidisk is not linked.

- TH1 is the %utilization threshold above which a warning message will be generated during startup. Specify 100 to bypass the test.

- TH2 is the %utilization threshold above which LISTSERV will log off. Specify 100 to bypass the test.

- WLIST is a blank-separated list of RFC822 addresses to be notified if something is wrong with the minidisk. Note that the postmasters are always notified, so that the default setting of WLIST (null string) actually means that the notification will be sent only to the postmasters.

- MSG is an optional message to be appended to the standard error or warning messages generated by LISTSERV. This could, for example, contain some description of the contents of the minidisk.

Please note that only those CUUs listed in the CHECKMDISK variable will be verified at startup time. However, some of the information in the MDISK stem will be used when dynamically accessing libraries. The default settings for library minidisks which have not been described here is shown below.

Default Value

```
MDISK.cuu = '*S N N N 100 100//'
```

Wildcards

Not allowed.

See also

CHECKMDISK

MSGD

Platforms

VM only

Abstract

Userid of the virtual machine running a RFC1312/MSP server, if "Internet TELL" support is desired. Leave this field blank if you are not running a MSGD virtual machine.

Example

```
MSGD = 'MSGD'
```

Default Value

Emtpy string.

Wildcards

Not allowed.

MYDOMAIN

Platforms

All

Abstract

The list of all the possible Internet host names and aliases for the machine on which LISTSERV is running.

Examples

```
VM:   MYDOMAIN = 'LISTSERV.XYZ.COM WWW.XYZ.COM VM.IGATE.XYZ.COM'
      MYDOMAIN = NODE 'WWW.XYZ.COM VM.IGATE.XYZ.COM'
VMS:  MYDOMAIN "LISTSERV.XYZ.COM WWW.XYZ.COM VMS.IGATE.XYZ.COM"
unix: MYDOMAIN="LISTSERV.XYZ.COM WWW.XYZ.COM UNIX.IGATE.XYZ.COM"
      MYDOMAIN="$NODE WWW.XYZ.COM UNIX.IGATE.XYZ.COM"
Win:  MYDOMAIN=LISTSERV.XYZ.COM WWW.XYZ.COM NT2.IGATE.XYZ.COM
      MYDOMAIN=%NODE% WWW.XYZ.COM NT2.IGATE.XYZ.COM
```

Details

Usually this is the same as `NODE`, however you can supply additional names if your machine operates several services under different host names. For instance, if your machine operates WWW and FTP servers in addition to LISTSERV, under the hostnames WWW.XYZ.COM and FTP.XYZ.COM, you may want to list these names in MYDOMAIN. Similarly, if you operate the LISTSERV service under a hostname such as LISTSERV.XYZ.COM, while the machine's real name is NT2.IGATE.XYZ.COM, you will want to list the real name in `MYDOMAIN` because some unix machines will automatically substitute it for the published name.

Default Value

None. This parameter must be set explicitly.

Wildcards

Not allowed.

MYORG

Platforms

All

Abstract

Short organization name that appears in the mail header of messages from LISTSERV itself (*i.e.* not in the header of messages from a mailing list).

Example

```
VM:   MYORG = 'XYZ, Inc.'
VMS:  MYORG "XYZ, Inc."
unix: MYORG="XYZ, Inc."
Win:  MYORG=XYZ, Inc.
```

Default Value

Not set. Generates the standard "L-Soft list server at *host*" organization name. Please note that the "L-Soft list server at" part of the string is not configurable, for trademark and copyright reasons.

Wildcards

Not allowed.

NODE

Platforms

All

Abstract

Defines the Internet hostname of this LISTSERV host.

Example

```
VM:   NODE = 'LISTSERV.MYHOST.NET'
VM:   NODE "LISTSERV.MYHOST.NET"
VM:   NODE="LISTSERV.MYHOST.NET"
VM:   NODE=LISTSERV.MYHOST.NET
```

Details

This must be a fully-qualified address, as noted in the example. It must not be an IP address or a non-qualified address such as ‘LISTSERV’.

For testing/evaluation purposes only you can define the NODE= as a square-bracketed, dotted-IP, for example, `NODE=[aaa.bbb.ccc.ddd]`, but this is not officially supported and L-Soft will not accept registrations for servers using this nomenclature.

Default Value

None. This parameter must be set explicitly.

Wildcards

Not allowed.

OCI_*

Platforms

OpenVMS, AIX, HP-UX, Linux (x86 only), Tru64 (aka Digital Unix, OSF/1), Solaris. LISTSERV Classic only.

Details

Three configuration variables under the `OCI_*` rubric are available for use with the DBMS/Mail Merge functionality introduced in LISTSERV 1.8d. Please see the *Developer's Guide for LISTSERV* (available separately) for documentation.

Due to the fact that multiple simultaneous ODBC connections are now supported, native OCI support for LISTSERV under Windows NT/2000 (which had been available as a workaround for the absence of multiple connection support for sites running both ODBC and Oracle databases) has been withdrawn in version 1.8e. Oracle databases may still be accessed via ODBC under Windows NT/2000.

ODBC_*

Platforms

Windows NT. LISTSERV Classic only.

Details

Three configuration variables under the `ODBC_*` rubric are available for use with the DBMS/Mail Merge functionality introduced in LISTSERV 1.8d. Please see the *Developer's Guide for LISTSERV* (available separately) for documentation.

PLAIN_FROMLINE

Platforms

All LISTSERV 14.3 and following.

Abstract

Boolean value controlling the "verboseness" of LISTSERV's administrative From: line.

Example

```
VM:   PLAIN_FROMLINE = 1
VMS:  PLAIN_FROMLINE "1"
Unix: PLAIN_FROMLINE=1
Win:  PLAIN_FROMLINE=1
```

Details

PLAIN_FROMLINE controls whether or not LISTSERV generates a “plain” From: line when sending administrative mail, eg, setting this variable to "1" would result in

```
From: LISTSERV@LISTSERV.EXAMPLE.COM
```

rather than

```
From: "Example Company LISTSERV Server (14.3)"
<LISTSERV@LISTSERV.EXAMPLE.COM>
```

Enabling PLAIN_FROMLINE overrides any setting made to MYORG=, MYNAME=, and/or MYNAME_HIDE_RELEASE=, since the organization name setting and release number will no longer be displayed.

Default Value

0 (that is, the original behavior)

POSTMASTER

Platforms

All

Abstract

The space-delimited list of Internet addresses of the "LISTSERV maintainers", that is, the people in charge of operating the LISTSERV service who are to be granted maintainer privileges and notified of problems with the operation of the server.

Example

```
VM:     POSTMASTER = 'NATHAN@EXAMPLE.COM Hide: LIAM@EXAMPLE.COM Quiet: ERIC@EXAMPLE.COM'
VMS:    POSTMASTER "NATHAN@EXAMPLE.COM Hide: LIAM@EXAMPLE.COM Quiet: ERIC@EXAMPLE.COM"
unix:   POSTMASTER="NATHAN@EXAMPLE.COM Hide: LIAM@EXAMPLE.COM Quiet: ERIC@EXAMPLE.COM"
Win:    POSTMASTER=NATHAN@EXAMPLE.COM Hide: LIAM@EXAMPLE.COM Quiet: ERIC@EXAMPLE.COM
```

Details

Please note that userids such as ‘postmaster@somehost.com’ or ‘listserv-maintainer@somehost.com’ cannot be defined as LISTSERV maintainers because they will be trapped by LISTSERV’s built-in loop-checking heuristics. If it is desired to have an "generic" address listed as the POSTMASTER address (for instance, in the output of the RELEASE or HELP commands), L-Soft suggests using a userid such as ‘lstmaint’.

The SHOW VERSION, RELEASE and HELP commands report the names and addresses of all the LISTSERV maintainers, allowing users to determine where to report problems. However, you can insert a "Hide:" keyword in the list, causing *all* the addresses that follow to be hidden from SHOW VERSION, RELEASE or HELP. Similarly, a "Quiet:" keyword indicates that *all* the addresses that follow should be granted privileges, but should not be notified of problems with the service.

Default Value

None. This parameter must be set explicitly.

Wildcards

Not allowed.

PRIMETIME

Platforms

All

Abstract

Defines the "prime time" for your node, during which mail to lists configured as "Prime= Yes" should not be processed. This option is mostly for small machines that are very busy during business hours. See Appendix B under the list header keyword "Prime=" for more information. Note that the value for PRIMETIME must always be enclosed in quotes, EXCEPT on Windows machines.

Do not confuse "prime time" for time that LISTSERV is allowed to process mail. "Prime time" is the time during which LISTSERV *is not allowed* to process mail, for example, it is the "prime time" of day on your machine during which other operations must take precedence.

Example

```
VM:   PRIMETIME = 'MON-FRI: 0800-1700; SAT-SUN: -'
VMS:  PRIMETIME "MON-FRI: 0800-1700; SAT-SUN: -"
unix: PRIMETIME="MON-FRI: 0800-1700; SAT-SUN: -"
Win:  PRIMETIME=MON-FRI: 0800-1700; SAT-SUN: -
```

NOTE CAREFULLY that under Windows, the `PRIMETIME=` value IS NOT QUOTED. For all other OS ports, the variable setting must be quoted as shown.

Default Value

`MON-SUN: -`

Wildcards

Not allowed.

QUALIFY_DOMAIN

Platforms

All

Abstract

Defines the Internet domain to be appended to all non-qualified Internet addresses in incoming mail.

Example

```
VM:   QUALIFY_DOMAIN = '.DC.LSOFT.COM'
VMS:  QUALIFY_DOMAIN ".DC.LSOFT.COM"
unix: QUALIFY_DOMAIN=".DC.LSOFT.COM"
Win:  QUALIFY_DOMAIN=.DC.LSOFT.COM
```

Details

This is mostly useful when dealing with unix systems, which often do use unqualified addresses (in violation of the Internet mail standards). In a typical non-unix-based network, this option does not need to be set. Note that this option applies only to unqualified addresses in the mail header, for example, with SUBSCRIBE. It will not qualify hostnames used with ADD or other third-party (list owner) commands.

Default Value

Determined from the Internet hostname; for instance, if your hostname is LISTSERV.XYZ.COM, the value of `QUALIFY_DOMAIN` will be .XYZ.COM.

Wildcards

Not allowed.

RESMODES

Platforms

VM

Abstract

Defines a list of filemodes which are to be considered as "reserved" and never available for dynamic ACCESS

Details

This variable defines a list of filemodes which are to be considered as "reserved" and never available for dynamic ACCESS. That is, LISTSERV will never ACCESS a library minidisk by itself under one of these filesmodes. You may of course use them to ACCESS minidisks from the PROFILE EXEC or from any local command/exec you may have written. Note that A,D,S and Z are always reserved, regardless of what you put in the RESMODES variable. Also, any filemode which happens to be in use at the time LISTSERV is started will be marked as reserved, so that LISTSERV will never release minidisks which were ACCESSed when it was started.

Note that the value for RESMODES= must be in UPPER CASE.

Example

```
RESMODES = 'BC'
```

Default Value

None (that is, no RESMODES are reserved by default).

Wildcards

Not allowed.

RUNMODE

Platforms

All

Abstract

Determines the server's mode of operation with respect to peer LISTSERV servers running on other Internet hosts.

Details

LISTSERV Classic or the non-free edition of LISTSERV Lite can operate in one of three modes:

- Networked: in this mode, your server will connect to the worldwide LISTSERV backbone operated over the Internet, exchanging information with these other servers on a regular basis. This allows you, for instance, to keep a local database of all the available LISTSERV lists, to act as a redistribution point for all LISTSERV mail directed to users on your campus, to advertise your lists in the worldwide list of lists, *etc.* Networked mode requires a number of special tables, which must be updated on a regular basis, and 24h uptime. Thus, this mode is not suitable for servers with dial-up connectivity.
- Tableless: in this mode, your server accesses the worldwide LISTSERV backbone through another LISTSERV site (which must be running in Networked mode with full backbone status). Your server still has access to the data available to backbone servers, but doesn't need to maintain any LISTSERV table (hence the name). This is the preferred mode for dial-up servers and for small servers where the overhead of maintaining the server should be kept to a minimum. Unless you know of a specific LISTSERV backbone site that is willing to allow you to do lookups, you should use SWGATE.LSOFT.COM for the lookup site.
- Standalone: this mode is for servers that are not connected to the Internet, or that operate in a "closed" environment where outside communication is not desired. The server will not communicate with any of the other LISTSERV servers on the Internet. As such, it will not have access to the list of lists or to other Internet LISTSERV resources.

The traditional academic servers operate in Networked mode. This is the default mode for the non-shareware versions.

Example

```
VM:   RUNMODE = 'TABLELESS SWGATE.LSOFT.COM'
VMS:  RUNMODE "TABLELESS SWGATE.LSOFT.COM"
unix: RUNMODE="TABLELESS SWGATE.LSOFT.COM"
Win:  RUNMODE=TABLELESS SWGATE.LSOFT.COM
```

Default Value

Tableless for the Windows 95 version and all Lite versions, networked for all others. The value cannot be changed under the Lite "Free Edition". Note that if a host is not defined when running in TABLELESS mode, LISTSERV defaults to using the host SWGATE.LSOFT.COM for lookups.

Wildcards

Not allowed.

SEARCH_DISABLED

Platforms

All

Abstract

A string variable which, if set to anything but the null string, disables the SEARCH command and returns the text in the string to anyone attempting an archive search. Similar in function to the (Boolean) DATABASE variable for VM, but allows you to customize the string returned to the invoker.

Example

```
VM:    SEARCH_DISABLED = 'The SEARCH command is disabled on this server.'
VMS:   SEARCH_DISABLED "The SEARCH command is disabled on this server."
unix:  SEARCH_DISABLED="The SEARCH command is disabled on this server."
Win:   SEARCH_DISABLED=The SEARCH command is disabled on this server.
```

Details

This variable is provided primarily for servers such as the IBM P/390 which may not have sufficient resources to offer database searching functionality. This variable should not be used unless there is a good reason to disable the SEARCH command altogether.

Default Value

Not set (null string).

Setting non-default directory paths with .SD

Platforms

Windows only

Abstract

It is possible (but not recommended) to set non-default directory paths for LISTSERV files by using the .SD parameter.

Example

```
Win:  .SD L E:\FTP\LOGS
```

Details

L-Soft does not recommend that the default directory configuration be changed without good reason. Such reasons might include putting LISTSERV logs on a shared network drive for testing or debugging purposes.

Default Value

Not set.

Wildcards

Not allowed. Note that this parameter must point to a valid, existing directory name. For security reasons, LISTSERV will not create directories that do not already exist.

SMTP_FORWARD

Platforms

OpenVMS, Windows, unix

Abstract

The Internet hostname (not the IP address; this variable must contain a fully-qualfied domain name [FQDN]) of the server to which all outgoing SMTP mail should be forwarded for delivery. This can be any machine with SMTP software that will accept mail from your machine.

Note: If you are running the MS Mail SMTP gateway product as your SMTP forwarding host, you should point SMTP_FORWARD to the "smart host" defined in the MS Mail configuration, rather than to the MS Mail gateway itself.

Note that for LISTSERV 14.3 and following, dotted-decimal IP addresses *are* supported for this keyword. The "no-IP" restriction still stands for versions inferior to 14.3.

Example

```
VMS:   SMTP_FORWARD "UNIX.XYZ.COM"
unix:  SMTP_FORWARD="UNIX.XYZ.COM"
Win:   SMTP_FORWARD=UNIX.XYZ.COM
```

Default Value

Non-unix: None. This parameter must be set explicitly.
Unix: The default is to point to the localhost; it is not normally necessary to set this variable under unix unless you are using an external host to deliver LISTSERV's mail.

Wildcards

Not allowed.

See also

SMTP_FORWARD_n

SMTP_FORWARD_n

Platforms

OpenVMS, Windows, unix

Abstract

Allows LISTSERV to deliver mail asynchronously via one or more SMTP "worker" processes.

Formal Syntax

There are several different ways to define the value for this variable. Here is the formal syntax (for unix the value must be enclosed in double-quotes, of course):

SMTP_FORWARD_*n*=[*x**]*hostname*[(*opentime-setting*)][;*failoverhostname*]

(Advanced tuning parameters typically used only by "huge" sites (millions of recipients/day) have been omitted.)

Examples

There are many different ways to set this keyword. A few examples are given below. Note carefully that the settings shown are Windows-specific; in order to set these on OpenVMS or unix servers, be sure to observe the syntax conventions specific to those platforms -- no "=" sign for VMS and double-quotes around the setting strings for both VMS and unix.

A typical VMS example: `SMTP_FORWARD_1 "UNIXSERVER.BAR.COM"`
A typical unix example: `SMTP_FORWARD_1="UNIXSERVER.BAR.COM"`

To save space, the rest of these examples are in Windows format.

```
SMTP_FORWARD_1=UNIXSERVER.BAR.COM
SMTP_FORWARD_2=SMTP.BAZ.NET
SMTP_FORWARD_3=UNIXSERVER.BAR.COM

SMTP_FORWARD_1=2*UNIXSERVER.BAR.COM
SMTP_FORWARD_2=SMTP.BAZ.NET

SMTP_FORWARD_1=2*UNIXSERVER.BAR.COM;SMTP.BAZ.NET
SMTP_FORWARD_2=SMTP.BAZ.NET

SMTP_FORWARD_1=UNIXSERVER.BAR.COM(00:00-05:00);SMTP.BAZ.NET
SMTP_FORWARD_2=SMTP.BAZ.NET
SMTP_FORWARD_3=SMTP.BAZ.NET(10:00-14:00)
```

Details

Defines a number of "SMTP workers" (`SMTPW.EXE` on VMS and Windows; `lsv` child processes on supported unix platforms) used to spread the mail delivery load across multiple machines (or multiple connections to the same machine). For most normal workloads, the recommendation is to define a single SMTP_FORWARD_*n* host (ie, SMTP_FORWARD_1=) which points to the same host as is set for the SMTP_FORWARD= synchronous delivery variable.

Multiple SMTP_FORWARD_n host definitions should be made only for large workloads

(30,000 daily deliveries or more), or when the mail delivery server being pointed to is very slow. In that case, opening multiple connections to the machine may improve throughput. A general rule is that you define multiple SMTP_FORWARD_n hosts only when xxx.MAIL files are consistently accumulating in the LISTSERV spool directory. Please be sure to read the comments below.

The value of each SMTP_FORWARD_*n* host must be a fully-qualified domain name (FQDN), not an IP address. (However, for LISTSERV 14.3 and following, dotted-decimal IP addresses *are* supported for this keyword. The "no-IP" restriction still stands for versions inferior to 14.3.)

Starting with 1.8d, this functionality is available on unix platforms. Under unix this keyword causes LISTSERV to spawn sub-processes of itself rather than requiring a separate executable.

Note: Under the Microsoft Windows OSes you MUST define an SMTP_FORWARD= parameter in the site configuration file which will tell LISTSERV where to deliver mail synchronously should asynchronous delivery fail for any reason. Under unix this isn't technically required since synchronous delivery always defaults to the local machine.

The first example is one of the most simple configurations: one SMTP_FORWARD_*n* variable is defined for each separate worker. In this case two workers are pointed at one host, meaning that that host will be used for two-thirds of the outgoing mail.

The second example is functionally identical to the first example but the syntax is different. SMTP_FORWARD_1=2*UNIXSERVER.BAR.COM says "define two workers pointed at UNIXSERVER.BAR.COM".

The third example is also functionally identical to the first example except that it tells worker #1 to use SMTP.BAZ.NET as a "failover" host if UNIXSERVER.BAR.COM cannot be contacted. Otherwise SMTP.BAZ.NET will handle only one-third of the outgoing mail as would be expected.

The fourth example tells worker #1 to use UNIXSERVER.BAR.COM only between midnight and 5AM, and use the failover host SMTP.BAZ.NET at all other times. It also tells worker #3 to operate only between the hours of 10AM and 2PM (with no failover). This is handy for (for instance) offloading large queues to other machines during heavy traffic periods when the other machines aren't being used for other things (like number-crunching).

Default Value

Not set (workers are not used and mail is delivered synchronously).

Comments

The more SMTP_FORWARD_*n* hosts you define, the more SMTP workers will be started. Since each SMTP worker takes up some resources on your machine, you should not define more workers than your workload requires.

Wildcards

Not allowed.

SMTP_LISTENER_IP

Platforms

Windows

Abstract

Dotted-decimal IP address which sets the IP address to which the SMTPL.EXE "listener" will bind at boot time.

Details

Under normal circumstances it is not advisable to set this parameter. It is only for use when it becomes necessary to force SMTPL.EXE to "listen" out on a specific IP address (for instance, on a multi-homed machine you might have another mail server running on one IP address and LISTSERV/SMTPL running on another). This parameter would normally be set only if another SMTP mail application were running on the same machine as LISTSERV and it were necessary to divide the mail up based on the IP address. However, please note that this assumes that the other SMTP mailer is also able to listen out on a designated IP address, and that separate DNS entries have been established for each of the IP addresses assigned to the machine.

Example

```
SMTP_LISTENER_IP=1.2.3.4
```

Default Value

Defaults to all valid IP addresses configured for the machine.

Wildcards

Not allowed.

SMTP_LISTENER_PORT

Platforms

Windows

Abstract

Integer value. Sets the port number to which the SMTPL.EXE "listener" will bind at boot time.

Details

Under normal circumstances it is not advisable to set this parameter. It is only for use when it becomes necessary to force SMTPL.EXE to "listen" out on a non-standard port (i.e., other than the standard SMTP port, port 25). This parameter would normally be set only if another SMTP mail application were running on the same machine as LISTSERV and it were necessary to avoid an SMTP port conflict. However, please note that this implies that the other SMTP mailer is able to forward LISTSERV's mail to the non-standard port you define.

Example

```
SMTP_LISTENER_PORT=10025
```

Default Value

Normally not set in `SITE.CFG` but the default would be

```
SMTP_LISTENER_PORT=25
```

Wildcards

Not allowed.

SMTP_RESET_EVERY

Platforms

All

Abstract

Directs LISTSERV to reset the SMTP connections to the SMTP delivery machines (see SMTP_FORWARD) at regular intervals (units: minutes). This parameter improves turnaround time on busy servers if the mail delivery server is a unix machine. It should not be used with other types of delivery servers. Under Windows this parameter is normally set through one of the "Optimize for..." buttons in the configuration program.

Example

```
VM:   SMTP_RESET_EVERY = 60
VMS:  SMTP_RESET_EVERY "60"
unix: SMTP_RESET_EVERY=60
Win:  SMTP_RESET_EVERY=60
```

Default Value

Not set; connections are not reset unless inactive.

SORT_RECIPIENTS

Platforms

All

Abstract

An integer value (0, 1, or 2) that determines whether or not LISTSERV sorts the recipient list in outgoing SMTP jobs. Under Windows this option is normally set through the "Optimize for..." buttons in the configuration program. It should be set to 1 for best performance if your mail delivery host is a unix machine. Other systems do not normally need a pre-sorted recipient list for optimal performance.

Example

```
VM:   SORT_RECIPIENTS = 1
VMS:  SORT_RECIPIENTS "1"
unix: SORT_RECIPIENTS=1
Win:  SORT_RECIPIENTS=1
```

Details

This variable setting has changed for 1.8d. Under 1.8c and earlier the variable was a Boolean value that simply determined whether or not the recipient list was sorted by hostname before being handed off to the outgoing MTA.

Under 1.8d and following, if this variable is set to the value 2, LISTSERV will not sort the recipient list if the number of recipients is less than or equal to your MAXBSMTP setting. Conversely, for jobs where the number of recipients is greater than MAXBSMTP, LISTSERV will sort the recipient list (keeping XYZ.COM together and thereby conserving resources). This enhancement is useful when you are using LSMTP as your outgoing MTA, since LSMTP doesn't care whether or not the recipient list is sorted.

Default Value

All platforms except Windows NT:	`SORT_RECIPIENTS = 0`
Windows NT:	`SORT_RECIPIENTS = 2`

SPAM_ALERT

Platforms

All LISTSERV 14.3 and later.

Abstract

Boolean value determining whether or not spam reports are sent to the postmaster.

Example

```
VM:   SPAM_ALERT = 1
VMS:  SPAM_ALERT "1"
unix: SPAM_ALERT=1
Win:  SPAM_ALERT=1
```

Details

SPAM_ALERT defaults to 0, meaning that spam alerts will not be sent to the LISTSERV postmaster (but will still be sent back to the message poster for his/her information).

The previous default behavior, in which the LISTSERV postmaster was cc'd on all spam alerts, can be reverted to by setting SPAM_ALERT=1.

Default Value

0, that is, enabled (postmaster will not receive cc's of spam reports).

SPAM_DELAY

Platforms

All

Abstract

Sets the server-wide delay period for the spam quarantine feature.

Example

```
VM:   SPAM_DELAY = 15
VMS:  SPAM_DELAY "15"
unix: SPAM_DELAY=15
Win:  SPAM_DELAY=15
```

Details

This variable sets the server-wide delay period (in minutes) during which LISTSERV holds certain "suspicious" messages before processing them. This gives LISTSERV's anti-spamming algorithms time in which to gather the necessary evidence to determine whether or not the message may be a spam. The value can be overridden on a list-by-list basis with the list header keyword setting "Loopcheck= Spam-Delay(xxx)" (the value is in minutes). `SPAM_DELAY=0` disables this "spam quarantine" feature for all lists on the server.

It should be carefully noted that setting `SPAM_DELAY=0` *does not* turn off LISTSERV's spam filter. It turns off only the spam quarantine part of the overall filter. There is no setting to disable the spam filter server-wide; it can be turned off only at the list level, with "Loopcheck= NoSPAM" in the list header.

Default Value

```
SPAM_DELAY = 10
```

SPAM_EXIT

Platforms

LISTSERV 14.3 on Windows and unix; also VM and OpenVMS when using the LISTSERV Anti-Virus Server option.

Abstract

Sets the name of the user-provided exit program used to call a third-party spam scanner.

Example

```
VM:   (no setting -- spam scanning is handled by the AVS)
VMS:  (no setting -- spam scanning is handled by the AVS)
unix: SPAM_EXIT="saexit"
Win:  SPAM_EXIT=SAEXIT
```

Details

Please see the full documentation (or the LISTSERV 14.3 release notes) for information on this feature.

Default Value

Not set, ie, disabled.

SPAM_MAXSIZE

Platforms

All, LISTSERV 14.3 and later.

Abstract

Sets the maximum size, in kilobytes, of any message to be handled by the spam scanner. Messages over the specified size are not scanned.

Example

```
VM:   SPAM_MAXSIZE = 512
VMS:  SPAM_MAXSIZE "512"
unix: SPAM_MAXSIZE=512
Win:  SPAM_MAXSIZE=512
```

Details

Related to SPAM_EXIT. Please see the full documentation (or the LISTSERV 14.3 release notes) for information on this feature.

Default Value

256 (ie, 256KB).

Spam Blacklists and Whitelists

Platforms

All, LISTSERV 14.3 and later.

Abstract

Sets up blacklists and whitelists to help combat spam.

Examples

See below, in Details.

Requirements

In order for LISTSERV to use the blacklists and whitelists, SPAM_EXIT must be enabled and pointed to an existing, external exit program. This is necessary because the white- and blacklisting feature is part of LISTSERV's overall anti-spam toolbox, which is only activated if SPAM_EXIT is enabled.

While you may of course use the exit program you write to submit inbound mail to an external spam filter for scanning, that is not mandatory. In that case, the exit program does not need to do anything other than exit immediately with a return code of zero. For example:

Windows:	site.cfg setting:
	SPAM_EXIT=SAEXIT
	
	x:\listserv\main\saexit.cmd :
	`rem don't do anything, just go back to LISTSERV`
	`exit 0`

Unix:	go.user setting:
	SPAM_EXIT="SAEXIT"
	export SPAM_EXIT
	
	~listserv/home/SAEXIT :
	`# don't do anything, just go back to LISTSERV`
	`exit 0`

Details

Starting with version 14.3, LISTSERV includes a whitelist/blacklist system that can be used either on its own, or in conjunction with the spam exit feature (also new in 14.3; please see the *Developers Guide to LISTSERV* for more information on the spam exit feature). This built-in whitelist/blacklist system is very efficient, which makes it advantageous to duplicate your spam filter's whitelist/blacklist rules so that the spam filter is bypassed for messages whose disposition can be determined simply on the basis of the origin or target address.

There are four separate lists (or in reality, site configuration variables that contain the lists), called:

```
SPAM_BLACKLIST_FROM
SPAM_BLACKLIST_TO
SPAM_WHITELIST_FROM
SPAM_WHITELIST_TO
```

The lists are defined in the site configuration file in the usual space-separated manner. A restart of LISTSERV is required after updating any of the lists, also as usual.

The FROM lists are applied to all the origin fields in the RFC822 header – `From:`, `Sender:`, `Resent-From:` and `Resent-Sender:` (note that the SMTP-level `MAIL FROM:` is not used).

The TO lists are applied to the various recipient fields in the RFC822 header. Please note that LISTSERV does not work at the SMTP transaction level and does not have access to the `RCPT TO:` field.

The listing system is based on a score that LISTSERV maintains as it goes through the four lists in sequence. If the final score is zero, the message is neither white- nor blacklisted, and processing continues normally (to the external spam filter, if one has been configured). If the final score is positive, the message is whitelisted and accepted, bypassing any additional tests, including the external spam filter. If negative, the message is immediately rejected. A negative final score is a conclusive determination that makes any further tests unnecessary.

Being whitelisted normally gives one point, and being blacklisted costs one point. This can be changed by using the following syntax:

```
SPAM_WHITELIST_FROM=*@EXAMPLE.COM *@*.EXAMPLE.COM
SPAM_BLACKLIST_FROM=*@PUBLIC.EXAMPLE.COM >JAMES@EXAMPLE.COM
```

The first entry whitelists all EXAMPLE.COM addresses. The second entry acts as a cancellation of this whitelist for the fictitious machine PUBLIC.EXAMPLE.COM, which is not part of the Intranet. It also blacklists JAMES@EXAMPLE.COM, a notorious source of spam, with a score of 2 (every '>' sign gives another score point). JAMES@EXAMPLE.COM receives 1 point from the whitelist and -2 from the blacklist, so he will be effectively blacklisted. It is possible to load an entry with up to 254 extra score points, although it is not expected that anyone would need to go that far.

Every list gives score points at most once. So if we also had JAMES@EXAMPLE.COM and JAMES@* in the whitelist, he would still get only one point from the whitelist. But, when applicable, you do get the highest possible number of points that you have matched. If we had JAMES@EXAMPLE.COM and >>JAMES@* in the whitelist, he would get 3 points.

Again, all the whitelists and blacklists scores are added. If you use both FROM and TO, you need to use a point system that gives the desired results. It is much easier if you only use FROM or only TO.

What you would put in TO is defunct addresses (former employees, “honeypots”, *etc.*) that are guaranteed to have come out of spam listings. The message is then bounced for all recipients, not just the defunct address.

In most cases you will not want bounces to be blacklisted, since they are useful to LISTSERV and are processed automatically. In addition to the white/blacklists themselves, another site configuration variable, SPAM_WHITELIST_BOUNCE, is available. This value is an integer, defaulting to 1, and is the number to be added to the message's whitelist score if it is a bounce. Set to 0 to disable. You can also use a higher value.

Note that bounces may not be run through the spam filter at all. If LISTSERV can immediately determine what the bounce is for and process it, it will not scan it for spam.

SYNTAX: The syntax for all of the white- and blacklists is identical, so we will use just one of them for the example.

White/Blacklisting example:

```
VM:    SPAM_BLACKLIST_FROM = '*@PUBLIC.EXAMPLE.COM >JAMES@EXAMPLE.COM'
VMS:   SPAM_BLACKLIST_FROM "*@PUBLIC.EXAMPLE.COM >JAMES@EXAMPLE.COM"
unix:  SPAM_BLACKLIST_FROM=*@PUBLIC.EXAMPLE.COM >JAMES@EXAMPLE.COM
       export SPAM_BLACKLIST_FROM
Win:   SPAM_BLACKLIST_FROM=*@PUBLIC.EXAMPLE.COM >JAMES@EXAMPLE.COM
```

SPAM_WHITELIST_BOUNCE examples:

```
VM:    SPAM_WHITELIST_BOUNCE = 1
VMS:   SPAM_WHITELIST_BOUNCE "1"
unix:  SPAM_WHITELIST_BOUNCE=1
       export SPAM_WHITELIST_BOUNCE
Win:   SPAM_WHITELIST_BOUNCE=1
```

See Also

SPAM_EXIT

STOREPW

Platforms

VM

Abstract

STOREPW is the password to be used by postmasters when executing CP/CMS commands and when storing files in the server by means of the PUTC command.

Example

```
STOREPW= 'almighty'
```

Details

If you do not want to allow remote file storing nor remote CP/CMS command execution, just set this variable to ''. Note that CP/CMS commands typed at the server's console will always be honored.

Although this variable is available under non-VM versions of LISTSERV, for non-VM it is functionally equivalent to `CREATEPW` and should simply be left unset.

Starting with LISTSERV 14.3, this setting is obsolete and deprecated, as all POSTMASTER-restricted commands can now be validated with the postmaster's personal password. Also starting with LISTSERV 14.3, setting STOREPW to the special value `*NOPW*` tells LISTSERV to disable STOREPW entirely.

See also

CREATEPW

SYSTEM_CHANGELOG

Platforms

All Classic and Classic HPO (requires LISTSERV 1.8d 2000b level set release or later)

Abstract

System changelogs are not available in LISTSERV Lite. Enable/disable a system-wide changelog that logs information about DISTRIBUTE/mail-merge jobs and postings to lists.

Example (for 1.8d)

```
VM:   SYSTEM_CHANGELOG = 1
VMS:  SYSTEM_CHANGELOG "1"
Unix: SYSTEM_CHANGELOG=1
Win:  SYSTEM_CHANGELOG=1
```

Example (for 1.8e non-VM, see changes noted in Details)

1.8d syntax continues to work, enhanced by (for example):

```
VMS: SYSTEM_CHANGELOG "SINGLE"
Unix: SYSTEM_CHANGELOG="YEARLY"
Win: SYSTEM_CHANGELOG=WEEKLY
```

Change-log rotation is not supported under VM.

Details

LISTSERV builds dated 16 July 2000 (the 2000b level set release) or later running with Classic or Classic HPO LAKs include this feature. If enabled, a file called `system.changelog` (`SYSTEM CHANGELG` on VM) is created and written to in LISTSERV's A directory (or on the A disk for VM) containing data for each DISTRIBUTE, mail-merge, and list posting sent through the server. The syntax of the data line is documented in chapter 10 of the site manager's manual.

Difference between 1.8d and 1.8e versions: In 1.8d, the SYSTEM_CHANGELOG value was a binary "on/off" setting. In non-VM 1.8e, the system change-log can now be rotated on a DAILY, WEEKLY, MONTHLY, YEARLY, or SINGLE basis, or explicitly turned off (the default) by setting the value to 0 as before. For backward compatibility the value 1 is the same as the value SINGLE.

Change-log rotation is not supported under VM due to the filesystem restriction of 8-character file extensions; while the command processor will not complain if you set `SYSTEM_CHANGELOG = 'MONTHLY'`, the behaviour will always be as if you had specified 1 or 'SINGLE'.

Rotated system change-logs are renamed with the format `SYSTEM.CHANGELOG-`*`yyyy`*`[`*`mm`*`[`*`dd`*`|`*`w`*`]]` (depending on the rotation period selected)

Default Value

`0` (ie, disabled)

TCPGUI_IPADDR

Platforms

Non-VM

Abstract

Sets the default IP address for use with LISTSERV's TCPGUI interface.

Example

```
VMS:  TCPGUI_IPADDR "101.231.8.1"
unix: TCPGUI_IPADDR="101.231.8.1"
Win:  TCPGUI_IPADDR=101.231.8.1
```

Details

If TCPGUI_IPADDR is not specified on a multi-homed machine, LISTSERV listens to port 2306 (by default) on every IP configured for use by the machine. This is primarily important on multi-homed machines where you want to use port 2306 on another IP for something else. It can be set on single-homed machines but is not necessary since this value will default to the machine's single IP address.

Default Value

Defaults to the primary IP address of the LISTSERV machine.

Wildcards

Not allowed.

See also

TCPGUI_PORT

TCPGUI_PORT

Platforms

Non-VM

Abstract

Sets the port number for use with LISTSERV's TCPGUI interface.

Example

```
VMS:  TCPGUI_PORT "23006"
unix: TCPGUI_PORT=23006
Win:  TCPGUI_PORT=23006
```

Details

`TCPGUI_PORT=` can be used to resolve potential port conflicts with other software on the LISTSERV machine by allowing you to change the port number used by the TCPGUI interface. It can also be useful on single-homed machines on which you wish to disable the LISTSERV TCPGUI interface altogether (for instance, to address local security issues). In order to disable the TCPGUI interface, simply set `TCPGUI_PORT=0` and restart the server.

Default Value

`TCPGUI_PORT = 2306`

Wildcards

Not allowed.

See also

TCPGUI_IPADDR

TRAPIN

Platforms

All

Abstract

A space-separated list of Internet addresses from which LISTSERV should never accept administrative mail.

Example

```
VM:   TRAPIN = 'OBNOX@SOMENODE.NET *@BADNODE.COM'
VMS:  TRAPIN "OBNOX@SOMENODE.NET *@BADNODE.COM"
unix: TRAPIN="OBNOX@SOMENODE.NET *@BADNODE.COM"
Win:  TRAPIN=OBNOX@SOMENODE.NET *@BADNODE.COM
```

Details

A space-separated list of Internet addresses from which LISTSERV should never accept mail. Mail and files from users matching these templates will not be processed, but rather simply transferred to the LISTSERV maintainer. This parameter is provided primarily for VM sites, but is available for the convenience of VM customers migrating to other platforms and should not need to be set by typical non-VM installations.

Note: L-Soft does *not* recommend using this keyword to filter out problem users, as it does not prevent people at the hosts in question from posting to mailing lists. It controls only how incoming administrative mail (e.g., containing LISTSERV commands) sent to the LISTSERV address is treated. You should use the FILTER_ALSO keyword to filter problem users.

Default Value

Built in.

Wildcards

Allowed.

See also

TRAPOUT

TRAPOUT

Platforms

All

Abstract

A space-separated list of Internet addresses to which LISTSERV should never send administrative mail.

Example

```
VM:   TRAPOUT = 'OBNOX@SOMENODE.NET *@BADNODE.COM'
VMS:  TRAPOUT "OBNOX@SOMENODE.NET *@BADNODE.COM"
unix: TRAPOUT="OBNOX@SOMENODE.NET *@BADNODE.COM"
Win:  TRAPOUT=OBNOX@SOMENODE.NET *@BADNODE.COM
```

Details

A space-separated list of Internet addresses to which LISTSERV should never send mail. Mail and files to users matching these templates will be sent to the postmaster instead. This parameter is provided primarily for VM sites, but is available for the convenience of VM customers migrating to other platforms and should not need to be set by typical non-VM installations.

Note: L-Soft does not recommend using this keyword to filter out problem users, as it does *not* prevent people at the hosts in question from posting to mailing lists. It controls only how outgoing administrative mail (e.g., containing responses to LISTSERV commands) sent to the LISTSERV address is treated. You should use the FILTER_ALSO keyword to filter problem users. If you wish to stop outgoing list mail to specific domains, you should simply do a global delete (`QUIET DELETE * *@*badnode.com`) and then add the domain to FILTER_ALSO to prevent future subscriptions from that domain.

Default Value

Built in.

Wildcards

Allowed.

See also

TRAPIN

TUNE_MANY_LISTS

Platforms

All 14.3 and later HPO

Abstract

(HPO only) In LISTSERV 14.3 and following, enables a suite of HPO functions that can markedly speed up operations on servers with many lists (>100).

Example

```
TUNE_MANY_LISTS=1
```

Details

Setting this Boolean value to 1 makes LISTSERV HPO work faster at the expense of memory. For sites of 100 lists or less, enabling TUNE_MANY_LISTS does nothing. Sites between 100 and 1000 lists may note some improvement if enabled. Sites of 1000 lists or more will want to turn this feature on.

Default Value

`0` (that is, disabled)

UODBC_*

Platforms

LISTSERV Classic/HPO under unix only.

Details

Starting with LISTSERV 14.5, three configuration variables under the `UODBC_*` rubric are available for use with LISTSERV's DBMS/Mail Merge functionality. Please see the *Developer's Guide for LISTSERV* (available separately) for documentation.

USE_LSMTP_MAIL_MERGE

Platforms

Non-VM

Abstract

Boolean value which determines whether or not LISTSERV will use its mail-merge functions for enhanced performance. Requires LSMTP version 1.1b or higher, or -- in 14.4 -- a high-capacity third-party SMTP mailer and LISTSERV's EMBEDDED_MAIL_MERGE feature enabled.

> This setting is obsolete as of LISTSERV 14.5. Sites requiring the performance improvements should use EMBEDDED_MAIL_MERGE instead.

Example

```
USE_LSMTP_MAIL_MERGE=1
```

Details

Prior to LISTSERV 14.4, if all your SMTP servers (as defined in the SMTP_FORWARD* variables) are, at all times and including backup and last resort, guaranteed to be LSMTP servers running version 1.1b or higher, you can set USE_LSMTP_MAIL_MERGE=1 to enable a number of delivery optimizations which REQUIRE the LSMTP 1.1b (or higher) mail merge functionality.

If you are running LISTSERV 14.4, LSMTP is no longer required for mail merge, but you must enable the LISTSERV site configuration variable EMBEDDED_MAIL_MERGE (q.v.) by setting it to 1.

For LISTSERV 14.3 and earlier: NOTE CAREFULLY that L-Soft makes absolutely no guarantees as to the efficacy of enabling this feature unless ALL of your SMTP_FORWARD* variables point to LSMTP 1.1b (or higher) servers. L-Soft STRONGLY DISCOURAGES changing this variable from the default unless it can be guaranteed that all of your SMTP_FORWARD* hosts are LSMTP 1.1b (or higher) servers. Since LISTSERV's mail-merge functionality works only in conjunction with LSMTP 1.1b or higher, you run considerable risk of non-delivery of mail if this feature is enabled and any of your SMTP_FORWARD* servers are not running LSMTP 1.1b or higher.

> This setting is obsolete as of LISTSERV 14.5. Sites requiring the performance improvements should use EMBEDDED_MAIL_MERGE instead.

Default Value

```
VMS:  USE_LSMTP_MAIL_MERGE "0"
unix: USE_LSMTP_MAIL_MERGE=0
Win:  USE_LSMTP_MAIL_MERGE=0
```

See also

EMBEDDED_MAIL_MERGE, SMTP_FORWARD, SMTP_FORWARD_n

WEB_BROWSER_CONFIRM

Platforms

All

Abstract

LISTSERV can be configured to require confirmation (through the "OK" mechanism) when commands are sent through a WWW browser, even if they apply to a list whose security level or "Subscription=" keyword does not require confirmation. This variable controls whether or not that functionality is enabled.

Example

```
VM:   WEB_BROWSER_CONFIRM = 1
VMS:  WEB_BROWSER_CONFIRM "1"
unix: WEB_BROWSER_CONFIRM=1
Win:  WEB_BROWSER_CONFIRM=1
```

Details

This change was made the default for the 1.8c release both to avoid abuse and because many occasional WWW users do not know their e-mail address or enter it incorrectly, whereas people who use e-mail regularly do of course have a working e-mail address in their configuration. To disable this feature, set `WEB_BROWSER_CONFIRM=0` in the site configuration file.

In 1.8d and following, this feature is disabled by default.

Default Value

```
WEB_BROWSER_CONFIRM = 0
```

WWW_ARCHIVE_CGI

Platforms

VMS, unix, Windows

Abstract

The (preferably) relative URL that leads to the WWW archive CGI script. (This is a URL, not an OS path name.)

Examples

```
VMS:  WWW_ARCHIVE_CGI "/htbin/wa.exe"
      WWW_ARCHIVE_CGI "http://listserv.example.com/htbin/wa.exe"
unix: WWW_ARCHIVE_CGI="/cgi-bin/wa"
      WWW_ARCHIVE_CGI="http://listserv.example.com/cgi-bin/wa"
Win:  WWW_ARCHIVE_CGI=/scripts/wa.exe
      WWW_ARCHIVE_CGI=http://listserv.example.com/scripts/wa.exe
```

Details

This value is preferably a relative URL, but in some cases (eg where the LISTSERV host name is an MX but not an A in DNS and thus cannot be resolved by a browser) it may be necessary to define this as an absolute URL including the A record hostname. If using SSL or a non-standard port it is also necessary to use the absolute URL.

Default Value

Null.

Wildcards

Not allowed.

See also

WWW_ARCHIVE_DIR

WWW_ARCHIVE_DIR

Platforms

VMS, unix, Windows

Abstract

The full OS path name to the WWW archive directory

Examples

```
VMS:  WWW_ARCHIVE_DIR "DISK2:[HTTPD.HTDOCS.ARCHIVES]"
unix: WWW_ARCHIVE_DIR="/usr/local/etc/httpd/htdocs/archives"
Win:  WWW_ARCHIVE_DIR=c:\inetpub\wwwroot\archives
```

Default Value

Null.

Wildcards

Not allowed.

See also

WWW_ARCHIVE_CGI

WWW_AUTHINFO_DISABLE

Platforms

OpenVMS, unix, Windows

Abstract

A Boolean value which can be set to 1 to disable the IP address verification function of the web archive interface ('wa'), or 0 to re-enable the function.

Example

```
VMS:  WWW_AUTHINFO_DISABLE "0"
unix: WWW_AUTHINFO_DISABLE=0
Win:  WWW_AUTHINFO_DISABLE=0
```

Details

The default setting allows you to bypass certain proxy problems that prevent the 'wa' interface from working properly. There is a certain minimum security exposure involved in that someone with a good memory can watch you using 'wa', memorize the URL, and then type it into another browser and use your login ticket. The ticket will still expire but until then the "thief" has full reign on your list. If the minimum security exposure noted above is unacceptable, this variable can be set to 0 (i.e., enable the authorization).

Default Value

```
WWW_AUTHINFO_DISABLE = 1
```

XFERTO

Platforms

VM

Abstract

Userid of the virtual machine to which files found in the lists readers should be transferred. This is part of the VM/ISF support and should NOT be changed on a regular SP or HPO system.

Example

```
XFERTO = 'LSTMAINT'
```

Default Value

Standard value.

Wildcards

Not allowed.

Appendix D: 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.)

D.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.

*D.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:

D.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.

D.4. 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.

D.5. 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------

D.6. 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

D.7. 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:

D.8. 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. :)

D.9. A sample "your list has been created" boilerplate

Mailing List Setup Confirmation

I have created the XXXXX-L list on LISTSERV.MYHOST.COM per your setup sheet.

If you are new to LISTSERV, you will probably want to download L-Soft's Quick Start manual for list owners. Simply point your web browser to

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

and view online or choose the version appropriate for your word processor or viewer.

Formal documentation of list owner commands and other list ownership issues can be found in the List Owner's Manual, which is available at the URL

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

Per your list service agreement, support for your list is handled through a mailing list, LIST-SUPPORT@LISTSERV.MYHOST.COM. You have been added to that list. Please direct all support questions to the LIST-SUPPORT list.

You may also be interested in subscribing to the LSTOWN-L mailing list for LISTSERV list owners. To do so, send a mail message to LISTSERV@LISTSERV.NET with the command

SUB LSTOWN-L Your Name

in the body (not the subject) of the message. There are a number of extremely experienced LISTSERV list owners subscribed there who are more than willing to share their expertise. Don't hesitate to ask for help.

You now need to instruct LISTSERV to add personal passwords for the list owner account(s). These passwords are used to validate privileged commands (such as the PUT command for storing your list "header" on the server after making changes to it). This is done by sending mail from each account to LISTSERV@LISTSERV.MYHOST.COM with the command

PW ADD password

(again, "password" is whatever you want it to be) in the body of the message. LISTSERV will request confirmation of this operation; simply reply to the confirmation request with the word "ok".

Adding these passwords will considerably lessen the chance that someone will "spoof" mail from you to make changes on your list. It is very unlikely that this will happen, but it never hurts to be cautious. :)

Sincerely,
Joe Smith
LISTSERV Maintainer

Appendix E: Related Documentation and Support

E.1. Official L-Soft Documentation

Subscribers to LISTSERV mailing lists will find answers to many of their questions in the *General User's Guide to LISTSERV*. This manual supersedes the old "NSC93" document originally written for LISTSERV 1.8a.

LISTSERV list owners will be interested in both the *List Owner's Quick Start Manual* and the *List Owner's Manual for LISTSERV*.)

New for LISTSERV 1.8d is the *Developer's Guide for LISTSERV*, which includes (in addition to some material formerly part of the *Site Manager's Operations Manual*) information on using the new LISTSERV API.

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

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

L-Soft invites comment on its manuals. Please feel free to send your comments via e-mail to MANUALS@LSOFT.COM, and mention which manual you are commenting on. (However, please do not send support questions to this address. See chapter 19 for appropriate support addresses.)

E.2. LISTSERV Support FAQ

An official LISTSERV Support FAQ is updated regularly and can be viewed at

`http://www.lsoft.com/lsv-faq.html`

There are actually four FAQs currently linked from this page: A list owner's FAQ, a site maintainer's FAQ, an LSMTP FAQ, and L-Soft's Year 2000 FAQ.

E.3. 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.

E.3.1. Documentation of new database functions

This document is in fact not dated at all; it is practically brand new. Norm Aleks has put together documentation of the new database functions (**`SEARCH`** and **`GETPOST`**) available beginning with version 1.8c. You can get a copy by sending the command

`INFO DATABASE`

to LISTSERV@LIBRARY.UMMED.EDU .

E.3.2. LISTSERV List Owner Information Area on LISTSERV.BUFFALO.EDU

This site replaces the old LSVOWNER package mentioned in earlier versions of this

manual. Here are the files currently (30 April 2001) available:

LISTSERV® "Ins & Outs"
Guide from Hugh Jarvis, Faculty of Social Sciences
LISTSERV® List Owner's Manual
LISTSERV® List Header and Configuration Keywords Information (from manual)
LISTSERV® Listserv Management and User Commands (from manual)
List Owner USENET News Gateway Information
List Owner New List Announcement
List Owner Welcome to the List Message
List Owner General Information
List Owner How To Maintain A List Header
WWW Gateway to LISTSERV® Lists

These files are found at the URL

http://listserv.buffalo.edu/owner/

(Please note that any links found there to list management or list creation are for local use only and remote users should avoid trying to use them :)

E.3.3. LISTSERV TIPS

LISTSERV TIPS is a document with the formal title "List Management Tips for LISTSERV Postmasters and List Owners". It was compiled in 1991 by Lisa Covi. LISTSERV TIPS can be retrieved from the LISTSERV hosts at SEARN and UBVM (among others) with the usual GET command.

The basic document has never been updated and is very BITNET-oriented, but is still quite useful as a basic information source on running a list. To view LISTSERV TIPS on the World Wide Web, see Holly Stowe's LISTSERV FAQ at

http://www.listserv.iupui.edu/

You can also go directly to LISTSERV TIPS at

http://www.iupui.edu/~lstmaint/listserv_tips.shtml

Please note that LISTSERV TIPS is not an official L-Soft publication and is therefore not maintained by L-Soft.

E.3.4. 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 "traditional" VM LISTSERV file server system. You can get a copy from LISTSERV@LISTSERV.NET or by FTP from

ftp://ftp.lsoft.com/documents/fsv.guide

Appendix F: Revision History

Note carefully that these revision notes are cumulative. For instance, if a revision note says that chapter x was renumbered to chapter y, or that a change was made to chapter y, and a later revision note says that chapter y was moved to another manual, obviously the later revision supersedes the earlier one (which is nonetheless retained for reference). So before complaining that you can't find a certain revision in the manual, please read the entire list of revisions to make sure something else didn't change in the meantime :) Revision notes from the release of the 1.8e manual through the beta stages will be removed when the manual for the next version is released and this process will begin again.

20050302-001 Released 14.5 manual.

Index