

---

---

## LAFITE UPDATE

---

---

This document describes changes to Lafite since the October, 1986 release.

### Lafite Modes

---

Lafite is now willing to operate in more than one mode "simultaneously". The mail watch background task checks all of your mailboxes in all the modes that are active, and **GetMail** retrieves mail from all mailboxes. The variable `LAFITE.USE.ALL.MODES` controls whether Lafite runs in more than one mode at once. The interesting values are:

- NIL This is the old way, where you must change modes manually. All commands operate only in the currently active mode.
- : POLL The mail watch task checks all modes, but **GetMail** only retrieves in the currently active mode.
- T Mail watch checks and **GetMail** retrieves in all modes.

The initial value is T. The flag has no effect on the **SendMail** command, which continues to bring up a message composition window in the current mode. It also does not affect the **Answer** command, which answers in the mode that the message was retrieved in. The retrieval mode is remembered in the table of contents file; in the absence of this knowledge, e.g., if the toc is deleted or the message was retrieved with an older Lafite, the **Answer** command uses simple rules that nearly always unambiguously distinguish GV from NS messages. Be aware that the value : POLL can be confusing to such hacks as automatic mail retrievers, since the status window can report new mail, yet **GetMail** won't retrieve it.

Ordinarily, Lafite only attempts to authenticate you in each mode when it starts up, and after logout. If, for example, the NS authentication service is down when you start up Lafite, or your NS password is incorrect, it does not repeatedly try to authenticate you, but merely reports the condition once and behaves as if the failing mode does not exist. If conditions later change, you can force another attempt by using the mode switching commands to explicitly switch to the mode. If your password is incorrect, you can change it using the NS Login subcommand of Quit (see below).

**GetMail** only attempts to retrieve mail from the servers that reported new mail in the most recent poll. This means that if one of your mail servers is not responding, you no longer have to wait during **GetMail** for it to time out. If you click **GetMail** at a time when the status window reports "No New Mail", then **GetMail** will attempt to contact all of your mailbox servers, just as it did in previous releases.

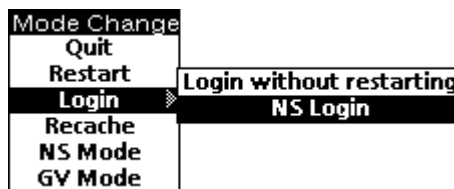
## Turning Lafite on and off

You can now turn Lafite on or switch modes from the background menu.



- Mail** This brings up a standard message sending window in the current mode, whether Lafite is on or not. If Lafite is not on, it will attempt to authenticate you in the current mode.
- Send Mail** This is the same as the main **Mail** command, except that you get to choose which kind of message form, just as middle-button on the Lafite status window's **Send Mail** command allows.
- Turn Lafite On** This is the same as calling (LAFITE :ON). It has no effect if Lafite is already on.
- Set Lafite Mode** This command allows you to change Lafite's mode. It offers a menu of the mode choices found also as part of the middle-button **Quit** command on the Lafite status window.

The middle-button **Quit** command on the Lafite status window has two new subcommands, **Recache** and **Login**. Here, for completeness, is a description of all the commands:



- Quit** This is the same as the main command, which is the same as calling (LAFITE :OFF). It closes all folders, asking about updating them if any need it, then turns off Lafite.
- Restart** This turns Lafite off, then immediately back on; same as (LAFITE :RESTART).
- Login** Changes the logged-in user for Lafite. This command first turns Lafite off, so that any updating required by closing out the previous user's folders happens under the previous user's login. It then prompts for user name and password, exactly as if you had called (LOGIN) yourself, and then for a new LAFITEDEFAULTHOST&DIR. Finally, it turns Lafite back on.
- Login without restarting** Performs a login without changing the state of Lafite. This is useful if you had previously logged in with an incorrect password, or if you have multiple user identities all sharing the same set of folders.
- NS Login** Prompts for a new login just for NS mail. You may need to use this if your global user name does not correspond to your NS user name or an alias of it in the default domain, or if you have different passwords for Grapevine and NS. There is actually

one of these commands for each mode that understands an independent login, but NS is the only current instantiation.

**Recache** In the case of several of the public interface variables, Lafite computes other structures based on them, and does not automatically recompute them when you change the public variable. The recache command forces Lafite to recompute these "caches" of the variables' values. A recache also occurs whenever Lafite is (re)started. The command is implemented by the function `LAFITE.COMPUTE.CACHED.VARS`, which you can advise if you have other caches that Lafite doesn't know about. The variables currently affected by this command (i.e., changes in whose values are not otherwise noticed) are:

```
LAFITE.DONT.DISPLAY.HEADERS
LAFITE.DONT.FORWARD.HEADERS
LAFITE.DONT.HARDCOPY.HEADERS
LAFITE.EXTRA.DISPLAY.COMMANDS
LAFITE.LOOKS.SUBCOMMANDS
LAFITE.GV.FROM.FIELD.
```

**NS Mode**

**GV Mode** Switch to the specified mode. There is one of these for each mode.

## Screen Appearance

There are several new variables and one function for controlling the appearance of your Lafite screen.

`LAFITE.BROWSER.LAYOUTS`

[Variable]

A list of browser "layouts" specifying where Lafite should place browsers, their display windows and their icons. Each layout is of the form (*BrowserRegion IconPosition DisplayRegion*). *IconPosition* and *DisplayRegion* are optional. The region variables are standard window system regions of the form (*left bottom width height*); *IconPosition* is a position of the form (*x . y*). When the **Browse** command creates a new browser window, it chooses the first element of `LAFITE.BROWSER.LAYOUTS` that is not already in use. If all are in use, it prompts for a region.

`LAFITE.DISPLAY.SIZE`

[Variable]

Specifies the default size of display windows, other than those already specified in a browser layout. If the value of this variable is a size (*width . height*), then Lafite prompts for a region with a box this size. If the value is `NIL`, Lafite prompts for a general region, requiring you to drag from one corner to the opposite. The initial value is `(500 . 300)`.

`LAFITE.EDITOR.LAYOUTS`

[Variable]

A list of editor "layouts" specifying where Lafite should place message composition editors and their icons. As message editors have only a single window, each layout consists of only two elements (*EditorRegion IconPosition*). When one of the message-sending commands creates a new editor window, it chooses the first element of `LAFITE.EDITOR.LAYOUTS` that is not already in use. If all are in use, it prompts for a region.

LAFITE.EDITOR.SIZE

[Variable]

Specifies the default size of message composition editors, other than those already specified in LAFITE.EDITOR.LAYOUTS. If the value of this variable is a size (*width . height*), then Lafite prompts for a region with a box this size. If the value is NIL, Lafite prompts for a general region, requiring you to drag from one corner to the opposite. The initial value is (470 . 300).

For backward compatibility, if LAFITE.BROWSER.LAYOUTS is NIL, it is considered to contain the single element (LIST LAFITEBROWSERREGION NIL LAFITEDISPLAYREGION). If LAFITE.EDITOR.LAYOUTS is NIL, it is considered to contain the single element (LIST LAFITEEDITORREGION). The variables LAFITEBROWSERREGION, LAFITEDISPLAYREGION, and LAFITEEDITORREGION are now obsolete and may be removed in a future release.

(LAFITE.BROWSE.FOLDER *foldername layout options* —)

[Function]

Programmatic interface to the **Browse** command. Browses the mail folder named *foldername* and returns a mailfolder object. If a browser already exists for *foldername* it is returned; otherwise, a new browser is created. The normal Lafite defaulting rules apply to *foldername*, so, for example, "active" defaults to the file "active.mail" on the directory specified by LAFITEDEFAULTHOST&DIR. The argument *layout* is a list of up to three elements describing the layout of the browser; it takes the same form as elements of LAFITE.BROWSER.LAYOUTS. *options* is a list of zero or more keywords from among the following:

- :ACTIVE The browser is considered an "active" mail browser. Currently the only import of this setting is to enable GetMail on expansion (see below).
- :GETMAIL After loading the mail folder, retrieve new mail, if any, into this folder.
- :SHRINK After loading the mail folder, shrink the browser window. This option is ignored if :GETMAIL was specified.
- :FORGET If *foldername* is new to Lafite, don't remember it; i.e., do not add it to the menu of mail folders.
- :CONFIRM If *foldername* does not exist, require confirmation before creating it.

LAFITE.BROWSER.ICON.PREFERENCE

[Variable]

Specifies where a browser's icon should be positioned if it has not already been specified in a browser layout as described above. The possible values are

- NIL Place the icon at the bottom left corner of the browser window. This is the default.
- :ASK Prompt for a position.
- a function* Call this function, passing the browser window as its sole argument. The function should return a position, or NIL to make Lafite prompt for a position.

LAFITE.FOLDER.MENU.FONT

[Variable]

The font used in the menu of folders displayed by the **Browse** and **MoveTo** commands. Initially NIL, which means use the default system MENUFONT.

LAFITE.FOLDER.ICON

[Variable]

An icon specification for the icon used by Lafite browsers. It is of the form expected by TITLEDICONW, i.e., (*IconBitmap MaskBitmap TitleRegion*). This variable replaces the undocumented variables MSGFOLDERTEMPLATE, MSGFOLDERICON, and MSGFOLDERMASK.

LAFITE.MSG.ICON

[Variable]

An icon specification for the icon used by Lafite message editors. This variable replaces the undocumented variables MSGUNSENTREGION, MSGUNSENTICON, and MSGUNSENTMASK.

The function LAFITE itself has been extended to take advantage of the same options available to LAFITE.BROWSE.FOLDER, so that you can set up your entire default mail screen in a single call. Here is the new documentation:

(LAFITE *on/off* &optional *folder* &rest *options*)

[Function]

Turns Lafite on or off according to the first argument, which is one of the keywords :ON, :OFF, or :RESTART. For backward compatibility, the same symbols are accepted in the Interlisp package. :ON turns Lafite on, and has no effect if it is already on. :OFF turns it off, and has no effect if it is already off. :RESTART is equivalent to (LAFITE :OFF) followed by (LAFITE :ON *folder . options*).

The argument *folder* specifies a mail folder to browse. If it is omitted, i.e., for a simple (LAFITE :ON), it defaults to the value of DEFAULTMAILFOLDERNAME. NIL specifies no folder. So that you can specify *options* and still browse the default folder, *folder* = T also defaults to DEFAULTMAILFOLDERNAME.

The remaining arguments are optional keywords specifying what to do with the folder being browsed. The possible values are :ACTIVE, :GETMAIL, :SHRINK, :FORGET, and :CONFIRM, which are handled as described under LAFITE.BROWSE.FOLDER. For example, (LAFITE :ON "active" :CONFIRM :SHRINK) turns Lafite on, browses the folder "active", requiring confirmation if the file does not exist, and then shrinks the browser.

If *folder* is a list, it is interpreted as a list of specifications for files to browse and their placement. Each element is of the form (*foldername BrowserRegion IconPosition DisplayRegion . options*), which specifies invoking (LAFITE.BROWSE.FOLDER *foldername* (LIST *BrowserRegion IconPosition DisplayRegion*) *options*). If there were any *options* specified in the call to LAFITE itself, they are appended to the *options* specified in each element of *folder*. See LAFITE.BROWSE.FOLDER for details.

## Message Display

---

The —*End of message*— indication is back in operation for those of you who missed it in Medley.

Middle-button on the **Display** command now always prompts for its own display region (a box if `LAFITE.DISPLAY.SIZE` is non-NIL), and this window survives Updates; i.e., it stays open until you explicitly close it.

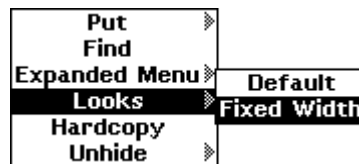
It is now possible to tell Lafite you do not care to see certain fields in the headers of messages, such as the numerous fields inserted by Arpanet mail transport.

<code>LAFITE.DONT.DISPLAY.HEADERS</code>	[Variable]
<code>LAFITE.DONT.FORWARD.HEADERS</code>	[Variable]
<code>LAFITE.DONT.HARDCOPY.HEADERS</code>	[Variable]

The value of each of these variables is a list of strings specifying the names of the fields you wish omitted by the **Display**, **Forward**, and **Hardcopy** commands, respectively. The alphabetic case of the strings is unimportant. When fields are omitted from a displayed message, an asterisk appears in the titlebar of the message. You can see the omitted fields by using the **Unhide** command from the titlebar menu (see below). There is no corresponding command to see the omitted headers in a forwarded message or, of course, in a hardcopied message.

The value ("Return-Path" "Redistributed" "Received" "Message-Id") is useful for filtering most of the rubbish from Arpanet headers.

Left- or Middle-button in the titlebar of a displayed message gives a modified TEdit menu of operations.



Those specific to Lafite are as follows:

**Looks** This command is used to change the appearance of the message. If you have selected a region of the message (longer than a single character), the command affects only that region; otherwise it applies to the whole message. This command affects only the display of this message this time; it has no effect on the message living in the mail folder.

The main **Looks** command brings up the same set of three menus as the normal TEdit Looks command, allowing you to choose any font and size. The subcommand **Fixed Width** changes the font to a fixed-width font, useful for viewing text formatted by folks in the fontless world. In addition, when changing to a fixed-width font, Lafite also sets the tab size to be the width of eight characters, something TEdit unfortunately doesn't do automatically. The subcommand **Default** sets the font back to Lafite's normal display font (the value of `LAFITEDISPLAYFONT`).

**Hardcopy** This command sends this single message to the default printer in whatever appearance it currently has. The hardcopy is sent immediately, independent of the setting of `LAFITEHARDCOPYBATCHFLG`.

**Unhide** If some fields of the message header have been omitted because they matched the strings in `LAFITE.DONT.DISPLAY.HEADERS`, this command reveals them. The subcommand **Hide** can be used to hide them again.

This menu can be further tailored using the following variables:

`LAFITE.EXTRA.DISPLAY.COMMANDS` [Variable]

This is the list of menu items added to the three possibly interesting TEdit commands offered in a message display window. Each item should be set up to return a function of one argument, which is applied to the message's text stream when the item is selected.

`LAFITE.LOOKS.SUBCOMMANDS` [Variable]

This is the list of menu subitems under the **Looks** command. It has the same format as `LAFITE.EXTRA.DISPLAY.COMMANDS`. At run time, Lafite inserts this list into the **Looks** item of `LAFITE.EXTRA.DISPLAY.COMMANDS`. The following function may be useful in implementing new Looks commands:

`(LAFITE.SET.LOOKS textstream newlooks paraloops)` [Function]

Changes the character looks of the current selection in *textstream*, or the whole stream if no more than one character is selected, to be *newlooks*, which can be a font descriptor or any argument acceptable to `TEDIT.LOOKS`. If *paraloops* is specified, it is a set of paragraph looks as expected by `TEDIT.PARALOOKS`. If *newlooks* is a fixed-width font and *paraloops* is `NIL`, this function also sets the default tab width of the selected text to be eight times the character width.

`LAFITEFIXEDWIDTHFONT` [Variable]

The font used by the **Fixed Width** command. It is initially the value of `DEFAULTFONT`, unless that font is variable-width, in which case it is set to Gacha 10.

`(LAFITE.HARDCOPY.TAB.WIDTH)` [Function]

When the **Fixed Width** command is used on a message, the tab width set for display is unlikely to be appropriate for hardcopy. If you subsequently issue the Hardcopy command on that window, Lafite has to guess what the tab width should be. Currently it does so by calling this function, whose initial definition computes a tab width based on `LAFITEFIXEDWIDTHFONT` coerced to Interpress. If you defaultly hardcopy to some other kind of printer, you may want to change this.

## Browser Commands ---

Middle-button on the **Update** command bypasses the normal Update options menu. Its behavior is controlled by the following variable:

`LAFITE.MIDDLE.UPDATE` [Variable]

A list of keywords specifying the normal way by which you like to update browsers. The choices are:

- :UPDATE Write out changes, but do not expunge deleted messages.
- :EXPUNGE Expunge deleted messages. If there are no deleted messages, this is equivalent to :UPDATE. Both :UPDATE and :EXPUNGE will also process any pending hardcopy (when LAFITEHARDCOPYBATCHFLG is T).
- :SHRINK After updating the folder, shrink the browser.
- :CLOSE After updating the folder, close the browser.
- :CONFIRM Require confirmation (via mouse) of the operation.

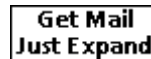
The keywords :UPDATE and :EXPUNGE are mutually exclusive, as are :SHRINK and :CLOSE. The default value is (:EXPUNGE :SHRINK :CONFIRM), which means that middle-button on **Update** prompts with a message like "Click LEFT to confirm Expunge and Shrink". The message specifies the operation implied by the setting of the variable, rather than always being "Expunge"; e.g., it might read "Click LEFT to confirm Hardcopy, Update table of contents and Shrink". As a special case, when the folder needs no updating at all and LAFITE.MIDDLE.UPDATE specifies :SHRINK, Lafite shrinks the browser immediately without asking for confirmation.

The menu of Update options offered by the Update, Shrink, and Close commands now more accurately reflects what needs to be done. In particular, there is no Expunge option if you have not deleted any messages.

The widest labels on the browser menu (**Undelete** and **Hardcopy**) have been shortened. With Lafite's default menu font (Helvetica 10 Bold), this allows browsers to be 54 points narrower than before, so that two browsers can fit side by side on the screen. (I personally prefer Modern 12 Bold (10 for users of NSDISPLAYSIZES), which yields a browser another 18 points narrower.) You can (awkwardly) change the labels by editing LAFITEBROWSERMENUITEMS.

LAFITEHARDCOPY.MIN.TOC now works even when LAFITEHARDCOPYBATCHFLG = T. (If more than LAFITEHARDCOPY.MIN.TOC messages are being hardcopied, the output is preceded by a table of contents.)

If you have ever performed **GetMail** on a browser (or browsed it with the :ACTIVE option), then the behavior of middle-button on the browser's icon behaves differently: instead of just expanding the window, it pops up a menu offering to get mail.



Choosing **GetMail** is equivalent to expanding the window, then choosing the regular **GetMail** command, but of course requires less interaction on your part.

Lafite browsers now support copy selection. Selecting with the mouse in a browser window while holding down the Copy or Shift keys selects the text of the field you are pointing at. You can select the Date, From, or Subject field. Selecting to the left of the Date field selects the whole message, producing a "summary line" such as "#185 14 Jun From: Carstairs.pa -- Meeting rescheduled". This use of Shift for copy selection means that only the CTRL key can be used to remove messages from the selection, an operation previously supported by either CTRL or Shift.



The title field of the browser has been rearranged so that the most interesting information appears first, and is thus less likely to fall off the right edge of the window. For example, the browser for {FS}<Carstairs>Mail>Active.mail might have a title reading "Browsing Active (Move To: Memos) on {FS}<Carstairs>Mail>".

LAFITEMOVETOCONFIRMFLG

[Variable]

This variable now only affects what happens when you choose the destination of a move from the menu (or using the middle-button accelerator), rather than typing it in (having selected "Other"). To summarize:

- NIL Do not require confirmation.
- LEFT Confirm only left-button MoveTo.
- MIDDLE Confirm only middle-button MoveTo.
- ALWAYS Require confirmation in all cases. This is the default.

Typing in a destination from the keyboard is enough evidence that you didn't slip on a mouse button, so confirmation is only required then if the destination file does not yet exist.

## Mail Sending

You can now control whether Lafite bothers to ask you about whether to send a message formatted or not:

LAFITE.SEND.FORMATTED

[Variable]

This variable is consulted when you ask to deliver a message that has what TEdit considers non-trivial formatting—font shifts, paragraph formatting, image objects, or NS characters. The choices are:

- NIL Send it unformatted.
- :ASK Prompt with a menu.
- T Send it formatted without asking.

The value can also be a list of elements (*formattingType answer*), specifying that when (TEDIT.FORMATTEDFILEP *message*) is *formattingType* the decision is *answer*. The initial value is

```
( (NSCHARS :ASK)
  (CHARLOOKS :ASK)
  (PARALOOKS :ASK)
  (IMAGEOBJ T) )
```

meaning to send messages with image objects formatted without asking (sending them unformatted is always a bad idea), but ask for all other kinds of formatted messages.

You can now send unformatted messages containing NS characters. However, note that the Koto version of TEdit did not understand NS characters in unformatted text, so you should not do this if you expect that some of your recipients have not yet upgraded to Lyric. If you never send

messages to Koto Lisp users (or don't care about this shortcoming), you might want to set the `NSCHARS` component of `LAFITE.SEND.FORMATTED` to `NIL`, i.e., have Lafite never send a message formatted merely because it contains NS characters.

Programs that compose message forms, such as those invoked from `LAFITESPECIALFORMS`, can specify that the message is to be delivered formatted without asking by giving the message (a textstream) the `TEXTPROP` property `LAFITEFORMAT`, value `TEDIT`.

Here are a couple more message customization variables:

`LAFITE.SIGNATURE`

[Variable]

A string with which to "sign" your messages. This string is appended to the standard message forms—those produced by the **SendMail**, **Answer**, and **Forward** commands—always starting it on a new line. There should be a carriage return at the beginning of the string if you want a blank line to set off the signature. The initial value is `NIL`, meaning no signature.

`LAFITE.GV.FROM.FIELD`

[Variable]

A string to use as the contents of the "From:" field of any Grapevine message you send that does not already have a From field. It must be a valid Grapevine address specification that parses down to your Grapevine identity, including registry. The usual syntax is "Human name <MailName.reg>", e.g., "Johann Amadeus Slonimsky III <Slo.pa>".

These two variables are examples of Lafite "personal" variables, in that they are closely bound to the logged-in user. The other is `LAFITEDEFAULTHOST&DIR`. If you change the logged in user (e.g., by calling `(LOGIN)`), Lafite resets all these variables to `NIL`, but remembers their values in case you log in again as yourself. This is designed to reduce confusion if you have more than one person using Lafite on the same machine. The values for each user are remembered as the value of the variable `LAFITE.USER.INFO`, an association list of user name and values, the latter being in property list format. Ordinarily you need not be aware of this variable, but you might, for example, want to initialize it with information about many users of a public machine.

---

## Grapevine

Lafite now does a more thorough job of parsing Grapevine messages according to the RFC 822 specification and objecting to malformed addresses. It also preserves full addresses when composing replies, rather than discarding all the commentary.

Lafite now supports "private distribution list" syntax in Grapevine messages. A private distribution list (dl) is an indirection mechanism: it allows you to name a possibly large group of addressees without explicitly enumerating them in the header of the message. Unlike a public dl, whose membership is maintained in the Grapevine database, a private dl is simply a file in which you have stored the real names. The names in the file must be in standard Grapevine syntax, just as you would type in the header of a message, but with two exceptions: (1) Names may be separated with one or more carriage returns, rather than commas, to enhance readability; and (2) The names must be fully qualified, i.e., the Grapevine registry must be included. The latter requirement eliminates confusion in the case where the current sender is in a different registry than the maintainer of the dl. The names can make use of all the standard Grapevine syntax, which allows you to include comments in the file. For example, you could have a line reading

<Carstairs.wbst> "Works with bj on the ACME project"

The file may be a TEdit file; Lafite strips out the formatting before attempting to parse the addresses in it.

The syntax for a private dl in a message is

To: *dlfilename*::;

i.e., the name of the file containing the real names, followed immediately by the two characters colon, semi-colon. If the file name contains any characters whose syntax is significant to Grapevine messages—colon, semi-colon, parentheses, brackets—the file name must be enclosed in double quotes. For example,

To: "{FS9:PARC:Xerox}<Carstairs>Mail>Friends"::;

The file name need not be complete; Lafite fills in defaults to locate the file according to the following two variables:

LAFITEDL.EXT

[Variable]

The default extension of private distribution list files. If a private dl is specified without an explicit extension, Lafite uses this extension when looking for the file. The value is initially "DL".

LAFITEDLDIRECTORIES

[Variable]

List of directories on which private distribution list files may be found. When you use a private dl in a message, Lafite searches LAFITEDEFAULTHOST&DIR, then this list, for the file containing the addressees. The value is initially NIL.

\*GV-SHOW-POSTMARK\*

[Variable]

When Lafite retrieves Grapevine mail, there is additional information about the transport of each message that Lafite normally discards as uninteresting, as it is for the majority of messages. When the variable \*GV-SHOW-POSTMARK\* is true, Lafite preserves this information in an extra line in the header of the message. The line reports the authenticated name of the sender, the time at which the message entered the Grapevine system, and the host from which it entered; for example,

GV-Info: Carstairs.pa at 4-Jun-88 23:38:00 from 204#16#

Note that the variable only affects the retrieval of mail, not its display. If the variable is NIL at the time of retrieval, the information is lost forever.

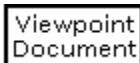
---

## NS Mail

The NS mail retrieval code correctly interprets messages that are really returned messages. It now turns such messages into a description of a failed delivery, with a mail server as the apparent sender, instead of just putting a cryptic "TransportProblem" header in the message and leaving the header looking as though the message was from you.

The NS mail watcher no longer maintains a "session" on the mail server. This reduces server load a bit, and means that there is no confusion if another client (either Lafite or Viewpoint) tries to retrieve the mail of a user who happens to be running Lafite on another machine at the same time.

NS Mail now handles "attachments" better. This change was actually made in a special release of NS mail in September, 1987, but is included here for completeness, and to explain the extensions that handle reference objects. Attachments are no longer left in your mailbox to be read later with, for example, Viewpoint. Instead, Lafite retrieves the entire attachment and encapsulates it into an image object that is enclosed as part of the text message, immediately following the header. A typical attachment appears in a mail message as:

Attachment: 

If you click inside the object with any mouse button, you are offered a menu of things you can do with the attachment. The choices vary according to the type of attachment:

- View as text** This brings up a window in which is displayed the raw content of the attachment as ascii bytes. Runs of non-ascii bytes are replaced by nulls to reduce the amount of garbage. Some attachments are utter gibberish, but some, such as Viewpoint documents and Interpress masters, contain sections that are plain text. With this command, you may be able to decide whether you care to do anything further with the attachment. (Sorry, there is no Viewpoint to TEdit converter, nor are there plans for one.)
- Put to file** This prompts you for a file name, and creates a file to contain the attachment. The file must be on an NS file server for this command to be very useful; otherwise, information will be lost. Once the file is so stored, you can retrieve it from Viewpoint and manipulate it just as if you had originally retrieved it as mail in Viewpoint. If you are running in Lyric, you must have the module NSRANDOM loaded for this command to work.
- Send to Printer** This command is only available for attachments that are in the form of an Interpress master. The command prompts you for a printer (which must be one that accepts Interpress, of course), and sends the attachment to it for printing.
- Expand folder** This command is only available for attachments that are in the form of a "folder". A folder is a mechanism for collecting several objects into a single one. The **Expand folder** command splits the attachment up into its component objects, each of which can be manipulated in the same way as a top-level attachment. For example, if the folder contains an Interpress master, you can print it.

If you use the **Put to file** command on a folder, the name component of the file name you type will be treated as the name of a new subdirectory, and the components of the folder will appear as files in that subdirectory. For other types of attachments, **Put to file** (usually) produces an ordinary (non-directory) file.

If the attachment is a Viewpoint "Reference" object, the **View as text** and **Send to Printer** commands apply to the file the reference mentions, not the attachment itself. The **Put to file** command is renamed **Store reference**, to make it clear it applies to the reference object itself,

rather than the file being referred to (which there is no point in copying, since Viewpoint can deal with the reference object itself just fine). References to folders also offer the command **FileBrowse**, which brings up a FileBrowser (if you have it loaded) browsing the directory referred to.

Messages containing attachments are otherwise just like formatted messages—you can move them to other folders, and you can forward them (assuming the mail is received by another Lafite recipient and did not have to pass through an information-losing mail gateway).

There is currently no mechanism for creating your own attachments to send to other users.

## Miscellaneous

---

The middle-button **Browse** command has a new subcommand **Browse & Forget**. This command prompts for a file name and browses it with the `:FORGET` option, i.e., it does not add it to the folder menu. Note that this command does not remove an existing folder from the menu, it merely refrains from adding a new one.

Lafite's table of contents code now handles messages containing NS strings in the Subject field. This fixes the bug where browsing such a file would claim that the toc was inconsistent with the folder, throw away the toc file and reparse from scratch. Also, the bug where Lafite reported "Folder is Empty" when discarding a malformed toc has been fixed.

Lafite now treats folder names as strings, and does its best to (a) ignore alphabetic case within folder names and (b) preserve the case that you type when creating a folder. This means, among other things, that you can now have mixed case in your `LAFITEDEFAULTHOST&DIR`, and that you can have folder names that are entirely numeric, such as "1186".

Lafite also keeps in its folder menu what the server considers the "canonical" name of each file to be. This means that if your folders include a file not on your default directory with a host not specified by its canonical name, then the next time you browse this folder, Lafite will think it is a new folder, leaving you with a menu containing both the old and the new names. Use the **Forget Folders** subcommand of **Browse** to dispose of such duplicates.

The information about your folders and message forms, formerly maintained in a file named `LAFITE.PROFILE` on your `LAFITEDEFAULTHOST&DIR`, is now kept in a file named `LAFITE.INFO` in a more extensible format. When you first run the new Lafite, if you have no `LAFITE.INFO` file, it will read `LAFITE.PROFILE` instead and convert to the new format.

Lafite does a better job now of noticing whether a mail file has been changed out from under you. It checks every time it opens the file, not just after logout. If the mail file has been changed but you have unsaved changes to the browser, Lafite will offer to save your changes if the mail file has merely been appended to. If, however, it has been expunged, so that the messages all live in different locations in the file, this is not possible, and the changes in your browser must be discarded.

When you return from logout and Lafite can't locate the file behind one or more of your browsers, it no longer closes the browser. The next time you try to do anything with the browser, such as display a message, Lafite will again try to locate the file. Thus, if the file server holding

the file is down, you need only wait for it to come back up before using the browser, rather than having to rebrowse the folder.

The after-logout code now runs mostly in a separate process, which allows the call to LOGOUT to return faster.

Lafite files are now of type LAFITE, rather than TEXT, since they typically contain non-text content, such as NS characters and TEdit formatting. On any kind of server but an NS file server, type LAFITE is usually coerced to BINARY.

## **Changes to Programmer Interface**

---

Well, actually, there is no documented programmer's interface. Numerous internals have changed; this section lists but a few.

With the new handling of modes, nearly everything surrounding the use of \LAFITEUSERDATA and the record LAFITEOPS (and hence the variable LAFITEMODELST) has changed. POLLNEWMAIL has been restructured; PRINTLAFITESTATUS behaves differently. There is a new hook LAFITENEWMAILFN. When this variable is non-NIL, its value is called, with no arguments, when the Lafite status window notes new mail.

The MAILFOLDER record has changed slightly. The use of its FOLDERDISPLAYWINDOWS field is different. The new field FOLDERDISPLAYREGIONS replaces BROWSERSELECTIONREGION.

The whole family of functions around \LAFITE.BROWSE has substantially changed.

Several functions have been deleted or renamed including LA.REMOVEDUPPLICATES, LA.SETDIFFERENCE, PROFILEFILENAME, \LAFITE.MERGE.PROFILES (now \LAFITE.MERGE.NAMELISTS), CHANGEFLAGINFOLDER, \LAFITE.GETMAILFOLDER.

## **Partial List of Fixed or Obsolete ARs**

---

- 267 Easy font changing
- 307 Filter selected fields out of displayed msg?
- 552 Make it easier to change logged in user
- 1239 Recover from "Lafite is confused..."
- 1249 Want automatic GetMail on icon expansion
- 1273 Prompt for GV login if BadPassword error => Declined: use Login
- 1389 Answer command should preserve entire address
- 1644 Private dl's
- 2143 Want detachable display windows that survive Update
- 2421 Preserve case when creating files
- 2433 Use GETBOXREGION to place window?
- 2597 Check that GV headers conform to RFC822
- 3168 Want user-specifiable browser/menu layout
- 3174 List of regions for browser, display, etc
- 3269 Update, GetMail etc should check creationdate/eof and rebrowse if needed
- 4420 LAFITEHARDCOPY.MIN.TOC doesn't work with LAFITEHARDCOPYBATCHFLG
- 4628 addresses lacking a closing double quote break with illegal arg 65536

5109 Don't loop if bad NS password  
5110 Blank name (e.g. in Reply-to) completes to \*:domain:org  
5111 Want name parser that doesn't add default registries  
5116 Parser loops forever if header address has unmatched paren or quote  
5117 Want way of dealing with NS Mail attachments without running Star  
5121 Close/Shrink menu includes "Expunge" even if no deleted messages  
5122 Simultaneously watch for both NS and GV mail  
5125 Case sensitivity in LAFITEDEFAULTHOST&DIR screws up MoveTo  
5129 Want browser title rearranged to avoid clipping important info  
5134 Message from self not noted if user not authenticated when folder browsed  
5137 Want to be able to keep display window open after browser shrunk  
5300 Answer NS message in GV mode does not give To field  
5541 Default status window off screen on small 1186 screen  
5638 Prune headers on forwarded messages  
6372 LAFITENEWMMAILTUNE not on daybreak  
6260 Blank To/cc line parsed incorrectly  
6547 Can't put Arpa address in From field  
6918 Send unformatted dies in COPYBYTES illegal arg  
7244 Numeric folder names die in string-equal  
8643 (il:lafite 'il:off) breaks in SHADEITEM, menu = nil  
8679 Changing modes with Lafite off doesn't reset \LAFITEUSERDATA?  
8995 When toc discarded, says "Folder empty"