LAFITEABBREV

By: Lennart Lövstrand
(Lovstrand.EuroPARC@Xerox.COM)

Requires: LAFITE, NSMAIL

INTERNAL

This document last edited on December 21, 1988.

INTRODUCTION

As the name suggests, LAFITEABBREV gives you the ability to refer to mail addresses using personal, easy to remember abbreviations. If you define (say) "Joe" to stand for "John Doe:Xyzzy North:ACME Corporation", then all you need to type to send a message to Joe is *Joe*. If you don't want to be bothered with his long address even when you read messages from him, you can optionally let the abbreviation apply both ways. When receiving messages from him (or where you both are recipients), his full address will be replaced by your abbreviation and thus only show as simple *Joe* in all header lines and the Lafite browsers.

In addition to this rather straight forward string-to-string text substitution, LAFITEABBREV will also do pattern matching using multiple wildcards in both the abbreviation and fully expanded strings. This is of course particularly useful for cases when many people share the same substring in their addresses, such as the NS domain. Say for example that Jane Doe worked at the same place as her brother. Then we could abbreviate "*:Xyzzy North:ACME Corporation" as "*:X:ACME" and refer to John as "John Doe:X:ACME" and Jane as "Jane Doe:X:ACME". In fact, we could even create the abbreviation "* Doe" ⇒ "* Doe:Xyzzy North:ACME Corporation" and refer to our friends as just "John Doe" and "Jane Doe" with the obvious expansions.

Abbreviations are purely personal; all other recipients of the message will see the real, expanded addresses. The abbreviations are on a strictly one-to-one mapping basis, ie. you can't implement private DLs by making an abbreviation expand into more than one address. As is the case with normal mail addresses, all abbreviations and expansions are case-insensitive.

EXAMPLES

An individual abbreviation:	Sir	⇒	Tom Moran:EuroPARC:RX
A group abbreviation:	* (APU)	⇒	*@MRC-APU.CAM.AC.UK:GV:Xerox
Hiding the local domain:	*	⇐	*:EuroPARC:RX
Hiding gateway syntax:	*@*.EDU	⇔	*%*:EDU:Xerox

MODULE EXPLANATION

LAFITEABBREV advises the Lafite NSMAIL routines that encode and decode string addresses to NSNAMEs. It operates transparently and automatically with respect to the rest of the mail system while being controlled by the variables mentioned below. The most central of these is the list of abbreviations itself, LAFITE.ABBREVS:

LAFITE.ABBREVS [Variable]

This variable contains all abbreviations and their corresponding real addresses. It is formed as a list of translations, where each translation is a list of (up to) four elements: the abbreviation, the full address, an optional direction, and an optional comment. Both the abbreviation and full address should be strings; the direction may be either of `:IN`, `:OUT`, `:BOTH`, `:NONE`, or left empty in which case it defaults to `:BOTH`. Finally, the comment, if present, should be in normal Interlisp format, ie. `(*` *text*`)`. It has no functional purpose and serves only as a memory aid. A full translation thus looks like:

`(`*abbrevation full-address direction comment*`)`

eg: `("Pete" "Piotr Kropotkin:RusskiPARK:ZeRoks" :OUT (* ; "Old chap"))`

which means that an address of *Pete* will be expanded to *Piotr Kropotkin:RusskiPARK:ZeRoks* whenever you send a message to him, but his full address will always be presented as such when receiving mail from him or whenever his address is mentioned in the header lines of a message you receive. For an explanation on translation directions, read more below.

The default list of translations is:

```
*@*     "*%*":GV:Xerox    :IN
*@*      *%*:GV:Xerox
*@*.*    *%*:*:Xerox      :IN
*.pa      *:PA:Xerox
```

This will present most ARPA Internet addresses in their proper form and also allow you to see Palo Alto Grapevine addresses as if you were on Grapevine. Note that the first rule above is necessary because some external GV addresses are being enclosed in double quotes by the GV/XNS Mail Gateway (for no appearant reason, one might add, since there exist no common conventions for parsing such constructs and it technically is malforming the address).

LAFITE.ABBREV.DIRECTIONS [Variable]

This variable controls the overall behaviour of LAFITEABBREV. It can take either of the following values:

`:IN`	Translate full addresses to abbreviations when receiving messages.
`:OUT`	Translate abbreviations to full addresses when sending messages.
`:BOTH`	Both of the above.
`:NONE`	Neither of the above, ie. don't ever do any translations at all.

The setting of LAFITE.ABBREV.DIRECTIONS limits the scope of individual translation rules, so if you for example set LAFITE.ABBREV.DIRECTIONS to `:OUT`, no translations will ever be done on incoming messages, not even if a rule has an explict direction of `:IN` or `:BOTH`.

The default value of LAFITE.ABBREV.DIRECTIONS is `:BOTH`.

LAFITE.ABBREV.MOVE.GAZE.RIGHT [Variable]

A quite handy application of LAFITEABBREV is to present ARPA Internet (RFC-822) addresses in their proper form, ie. translating addresses like Jakobsson%Score.Stanford:EDU:Xerox to Jakobsson@Score.Stanford.EDU and vice versa. Unfortunately, since the pattern matching algorithm uses a strict left-to-right order, an address like user%host%foo.bar:EDU:Xerox would end up translated as user@host%foo.bar.EDU, which is not quite what was inteded. However, if LAFITE.ABBREV.MOVE.GAZE.RIGHT is set to a non-NIL value, translations resulting in a percent sign coming anywhere after an atsign will be automatically changed so that the rightmost percent sign is substituted for an atsign instead. Don't worry too much if you didn't understand any of the above; just think of it as applied magic (aka a *kludge*).

The default value of LAFITE.ABBREV.MOVE.GAZE.RIGHT is `T`.

LAFITE.ABBREV.TRACE [Variable]

Set this variable to T or a window of your choice (eg the PROMPTWINDOW) to trace all translations that are done for you. Reset to NIL to stop tracing.

The default value of LAFITE.ABBREV.TRACE is `NIL`.

(LAFITE.ABBREV *ADDRESS DIRECTION*) [Function]

This is the function that does the actual translations; it is used automatically by LAFITEABBREV's advices, but mentioned here if you want to test it manually or use it elsewhere. The address should be the string to expand/abbreviate and the direction either of `:IN` or `:OUT`, indicating whether this address is about to be sent or received. The result is the translated address.

(LAFITE.ABBREV.MATCH *PATTERN STRING TEMPLATE*) [Function]

This is a small, case insensitive pattern matching algorithm supporting multiple wildcards on both the pattern and the template side. Asterisks will match substrings of zero or more characters, anything else has to match literally for the comparison to succeed. The result is constructed using the template, if supplied. Some examples:

PATTERN	STRING	TEMPLATE	RESULT
"Foo"	"FOO"	NIL	"FOO"
"Foo"	"FOO"	"Bar"	"Bar"
"Foo*Bar"	"FOOBAZBAR"	NIL	"FOOBAZBAR"
"Foo*Bar"	"FOOBAZBAR"	"1*2"	"1BAZ2"
"Foo*B*ar"	"FOOBAZBAR"	"1*2*3"	"12AZB3"
"Foo*B*ar"	"FOOBAZBA"	"1*2*3"	NIL

BUGS

Only XNS is supported (ie. GV addresses aren't touched). Messages from you won't be presented as "To: <Recipients>" in the browser if you abbreviate your own name. No nice'n'cuddly user interface included.