Message 5.9.0 Manual

by Lars Magne Ingebrigtsen

Message

All message composition from Gnus (both mail and news) takes place in Message mode buffers.

This manual corresponds to Message 5.9.0. Message is distributed with the Gnus distribution bearing the same version number as this manual.

1 Interface

When a program (or a person) wants to respond to a message – reply, follow up, forward, cancel – the program (or person) should just put point in the buffer where the message is and call the required command. `Message` will then pop up a new `message` mode buffer with appropriate headers filled out, and the user can edit the message before sending it.

1.1 New Mail Message

The `message-mail` command pops up a new message buffer.

Two optional parameters are accepted: The first will be used as the `To` header and the second as the `Subject` header. If these are `nil`, those two headers will be empty.

1.2 New News Message

The `message-news` command pops up a new message buffer.

This function accepts two optional parameters. The first will be used as the `Newsgroups` header and the second as the `Subject` header. If these are `nil`, those two headers will be empty.

1.3 Reply

The `message-reply` function pops up a message buffer that's a reply to the message in the current buffer.

Message uses the normal methods to determine where replies are to go (see Section 5.1 [Responses], page 23), but you can change the behavior to suit your needs by fiddling with the `message-reply-to-function` variable.

If you want the replies to go to the `Sender` instead of the `From`, you could do something like this:

```
(setq message-reply-to-function
      (lambda ()
       (cond ((equal (mail-fetch-field "from") "somebody")
              (list (cons 'To (mail-fetch-field "sender"))))
             (t
              nil))))
```

This function will be called narrowed to the head of the article that is being replied to.

As you can see, this function should return a string if it has an opinion as to what the To header should be. If it does not, it should just return `nil`, and the normal methods for determining the To header will be used.

This function can also return a list. In that case, each list element should be a cons, where the car should be the name of an header (eg. `Cc`) and the cdr should be the header value (eg. '`larsi@ifi.uio.no`'). All these headers will be inserted into the head of the outgoing mail.

1.4 Wide Reply

The `message-wide-reply` pops up a message buffer that's a wide reply to the message in the current buffer. A *wide reply* is a reply that goes out to all people listed in the `To`, `From` (or `Reply-to`) and `Cc` headers.

Message uses the normal methods to determine where wide replies are to go, but you can change the behavior to suit your needs by fiddling with the `message-wide-reply-to-function`. It is used in the same way as `message-reply-to-function` (see Section 1.3 [Reply], page 3).

Addresses that match the `message-dont-reply-to-names` regular expression will be removed from the `Cc` header.

1.5 Followup

The `message-followup` command pops up a message buffer that's a followup to the message in the current buffer.

Message uses the normal methods to determine where followups are to go, but you can change the behavior to suit your needs by fiddling with the `message-followup-to-function`. It is used in the same way as `message-reply-to-function` (see Section 1.3 [Reply], page 3).

The `message-use-followup-to` variable says what to do about `Followup-To` headers. If it is `use`, always use the value. If it is `ask` (which is the default), ask whether to use the value. If it is `t`, use the value unless it is '`poster`'. If it is `nil`, don't use the value.

1.6 Canceling News

The `message-cancel-news` command cancels the article in the current buffer.

1.7 Superseding

The `message-supersede` command pops up a message buffer that will supersede the message in the current buffer.

Headers matching the `message-ignored-supersedes-headers` are removed before popping up the new message buffer. The default is
'`^Path:\\|^Date\\|^NNTP-Posting-Host:\\|^Xref:\\|^Lines:\\|
^Received:\\|^X-From-Line:\\|Return-Path:\\|^Supersedes:`'.

1.8 Forwarding

The `message-forward` command pops up a message buffer to forward the message in the current buffer. If given a prefix, forward using news.

`message-forward-ignored-headers`
: All headers that match this regexp will be deleted when forwarding a message.

`message-make-forward-subject-function`
: A list of functions that are called to generate a subject header for forwarded messages. The subject generated by the previous function is passed into each successive function.

  The provided functions are:

  `message-forward-subject-author-subject`
  : Source of article (author or newsgroup), in brackets followed by the subject.

  `message-forward-subject-fwd`
  : Subject of article with '`Fwd:`' prepended to it.

`message-wash-forwarded-subjects`
: If this variable is `t`, the subjects of forwarded messages have the evidence of previous forwards (such as '`Fwd:`', '`Re:`', '`(fwd)`') removed before the new subject is constructed. The default value is `nil`.

`message-forward-as-mime`
: If this variable is `t` (the default), forwarded messages are included as inline MIME RFC822 parts. If it's `nil`, forwarded messages will just be copied inline to the new message, like previous, non MIME-savvy versions of gnus would do.

1.9 Resending

The `message-resend` command will prompt the user for an address and resend the message in the current buffer to that address.

Headers that match the `message-ignored-resent-headers` regexp will be removed before sending the message. The default is '`^Return-receipt`'.

1.10 Bouncing

The `message-bounce` command will, if the current buffer contains a bounced mail message, pop up a message buffer stripped of the bounce information. A *bounced message* is typically a mail you've sent out that has been returned by some `mailer-daemon` as undeliverable.

Headers that match the `message-ignored-bounced-headers` regexp will be removed before popping up the buffer. The default is '`^\\(Received\\|Return-Path\\):`'.

2 Commands

2.1 Buffer Entry

You most often end up in a Message buffer when responding to some other message of some sort. Message does lots of handling of quoted text, and may remove signatures, reformat the text, or the like—depending on which used settings you're using. Message usually gets things right, but sometimes it stumbles. To help the user unwind these stumblings, Message sets the undo boundary before each major automatic action it takes. If you press the undo key (usually located at `C-_`) a few times, you will get back the un-edited message you're responding to.

2.2 Header Commands

All these commands move to the header in question. If it doesn't exist, it will be inserted.

`C-c ?` Describe the message mode.

`C-c C-f C-t`
: Go to the `To` header (`message-goto-to`).

`C-c C-f C-b`
: Go to the `Bcc` header (`message-goto-bcc`).

`C-c C-f C-w`
: Go to the `Fcc` header (`message-goto-fcc`).

`C-c C-f C-c`
: Go to the `Cc` header (`message-goto-cc`).

`C-c C-f C-s`
: Go to the `Subject` header (`message-goto-subject`).

`C-c C-f C-r`
: Go to the `Reply-To` header (`message-goto-reply-to`).

`C-c C-f C-n`
: Go to the `Newsgroups` header (`message-goto-newsgroups`).

`C-c C-f C-d`
: Go to the `Distribution` header (`message-goto-distribution`).

`C-c C-f C-f`
: Go to the `Followup-To` header (`message-goto-followup-to`).

`C-c C-f C-k`
: Go to the `Keywords` header (`message-goto-keywords`).

`C-c C-f C-u`
: Go to the `Summary` header (`message-goto-summary`).

2.3 Movement

`C-c C-b` Move to the beginning of the body of the message (`message-goto-body`).

`C-c C-i` Move to the signature of the message (`message-goto-signature`).

2.4 Insertion

`C-c C-y` Yank the message that's being replied to into the message buffer (`message-yank-original`).

`C-c M-C-y` Prompt for a buffer name and yank the contents of that buffer into the message buffer (`message-yank-buffer`).

`C-c C-q` Fill the yanked message (`message-fill-yanked-message`). Warning: Can severely mess up the yanked text if its quoting conventions are strange. You'll quickly get a feel for when it's safe, though. Anyway, just remember that `C-x u` (`undo`) is available and you'll be all right.

`C-c C-w` Insert a signature at the end of the buffer (`message-insert-signature`).

`C-c M-h` Insert the message headers (`message-insert-headers`).

`message-ignored-cited-headers`
: All headers that match this regexp will be removed from yanked messages. The default is '`.`', which means that all headers will be removed.

`message-citation-line-function`
: Function called to insert the citation line. The default is `message-insert-citation-line`, which will lead to citation lines that look like:

```
Hallvard B Furuseth <h.b.furuseth@usit.uio.no> writes:
```

: Point will be at the beginning of the body of the message when this function is called.

`message-yank-prefix`
: When you are replying to or following up an article, you normally want to quote the person you are answering. Inserting quoted text is done by *yanking*, and each quoted line you yank will have `message-yank-prefix` prepended to it. The default is '`> `'.

`message-indentation-spaces`
: Number of spaces to indent yanked messages.

`message-cite-function`
: Function for citing an original message. The default is `message-cite-original`, which simply inserts the original message and prepends '`> `' to each line. `message-cite-original-without-signature` does the same, but elides the signature. You can also set it to `sc-cite-original` to use Supercite.

`message-indent-citation-function`
: Function for modifying a citation just inserted in the mail buffer. This can also be a list of functions. Each function can find the citation between `(point)` and `(mark t)`. And each function should leave point and mark around the citation text as modified.

`message-signature`
: String to be inserted at the end of the message buffer. If `t` (which is the default), the `message-signature-file` file will be inserted instead. If a function, the result from the function will be used instead. If a form, the result from the form will be used instead. If this variable is `nil`, no signature will be inserted at all.

`message-signature-file`
: If non-`nil` the name of a file containing the signature to be inserted at the end of the buffer. This is ignored if the file doesn't exist. The default is '`~/.signature`'.

Note that RFC1036bis says that a signature should be preceded by the three characters '`-- `' on a line by themselves. This is to make it easier for the recipient to automatically recognize and process the signature. So don't remove those characters, even though you might feel that they ruin your beautiful design, like, totally.

Also note that no signature should be more than four lines long. Including ASCII graphics is an efficient way to get everybody to believe that you are silly and have nothing important to say.

2.5 MIME

Message is a MIME-compliant posting agent. The user generally doesn't have to do anything to make the MIME happen—Message will automatically add the `Content-Type` and `Content-Transfer-Encoding` headers.

The most typical thing users want to use the multipart things in MIME for is to add "attachments" to mail they send out. This can be done with the `C-c C-a` command, which will prompt for a file name and a MIME type.

You can also create arbitrarily complex multiparts using the MML language (see section "Composing" in *The Emacs MIME Manual*).

2.6 Various Commands

C-c C-r
: Caesar rotate (aka. rot13) the current message (`message-caesar-buffer-body`). If narrowing is in effect, just rotate the visible portion of the buffer. A numerical prefix says how many places to rotate the text. The default is 13.

C-c C-e
: Elide the text between point and mark (`message-elide-region`). The text is killed and replaced with the contents of the variable `message-elide-ellipsis`. The default value is to use an ellipsis ('`[...]`').

C-c C-z
: Kill all the text up to the signature, or if that's missing, up to the end of the message (`message-kill-to-signature`).

C-c C-v
: Delete all text in the body of the message that is outside the region (`message-delete-not-region`).

M-RET
: Insert four newlines, and then reformat if inside quoted text.

  Here's an example:

```
> This is some quoted text.  And here's more quoted text.
```

If point is before '`And`' and you press *M-RET*, you'll get:

```
> This is some quoted text.

*

> And here's more quoted text.
```

'`*`' says where point will be placed.

C-c C-t Insert a `To` header that contains the `Reply-To` or `From` header of the message you're following up (`message-insert-to`).

C-c C-n Insert a `Newsgroups` header that reflects the `Followup-To` or `Newsgroups` header of the article you're replying to (`message-insert-newsgroups`).

C-c M-r Rename the buffer (`message-rename-buffer`). If given a prefix, prompt for a new buffer name.

2.7 Sending

C-c C-c Send the message and bury the current buffer (`message-send-and-exit`).

C-c C-s Send the message (`message-send`).

C-c C-d Bury the message buffer and exit (`message-dont-send`).

C-c C-k Kill the message buffer and exit (`message-kill-buffer`).

2.8 Mail Aliases

The `message-mail-alias-type` variable controls what type of mail alias expansion to use. Currently only one form is supported—Message uses `mailabbrev` to handle mail aliases. If this variable is `nil`, no mail alias expansion will be performed.

`mailabbrev` works by parsing the '`/etc/mailrc`' and '`~/.mailrc`' files. These files look like:

```
alias lmi "Lars Magne Ingebrigtsen <larsi@ifi.uio.no>"
alias ding "ding@ifi.uio.no (ding mailing list)"
```

After adding lines like this to your '`~/.mailrc`' file, you should be able to just write '`lmi`' in the `To` or `Cc` (and so on) headers and press *SPC* to expand the alias.

No expansion will be performed upon sending of the message—all expansions have to be done explicitly.

2.9 Spelling

There are two popular ways to have Emacs spell-check your messages: `ispell` and `flyspell`. `ispell` is the older and probably more popular package. You typically first write the message, and then run the entire thing through `ispell` and fix all the typos. To have this happen automatically when you send a message, put something like the following in your '`.emacs`' file:

```
(add-hook 'message-send-hook 'ispell-message)
```

If you're in the habit of writing in different languages, this can be controlled by the `ispell-message-dictionary-alist` variable:

```
(setq ispell-message-dictionary-alist
      '(("^Newsgroups:.*\\bde\\." . "deutsch8")
("." . "default")))
```

`ispell` depends on having the external '`ispell`' command installed.

The other popular method is using `flyspell`. This package checks your spelling while you're writing, and marks any mis-spelled words in various ways.

To use `flyspell`, put something like the following in your '`.emacs`' file:

```
(defun my-message-setup-routine ()
  (flyspell-mode 1))
(add-hook 'message-setup-hook 'my-message-setup-routine)
```

`flyspell` depends on having the external '`ispell`' command installed.

3 Variables

3.1 Message Headers

Message is quite aggressive on the message generation front. It has to be – it's a combined news and mail agent. To be able to send combined messages, it has to generate all headers itself (instead of letting the mail/news system do it) to ensure that mail and news copies of messages look sufficiently similar.

`message-generate-headers-first`
: If non-`nil`, generate all required headers before starting to compose the message.

  The variables `message-required-mail-headers` and `message-required-news-headers` specify which headers are required.

`message-from-style`
: Specifies how `From` headers should look. There are four valid values:

  `nil` Just the address – '`king@grassland.com`'.

  `parens` '`king@grassland.com (Elvis Parsley)`'.

  `angles` '`Elvis Parsley <king@grassland.com>`'.

  `default` Look like `angles` if that doesn't require quoting, and `parens` if it does. If even `parens` requires quoting, use `angles` anyway.

`message-deletable-headers`
: Headers in this list that were previously generated by Message will be deleted before posting. Let's say you post an article. Then you decide to post it again to some other group, you naughty boy, so you jump back to the `*post-buf*` buffer, edit the `Newsgroups` line, and ship it off again. By default, this variable makes sure that the old generated `Message-ID` is deleted, and a new one generated. If this isn't done, the entire empire would probably crumble, anarchy would prevail, and cats would start walking on two legs and rule the world. Allegedly.

`message-default-headers`
: This string is inserted at the end of the headers in all message buffers.

`message-subject-re-regexp`
: Responses to messages have subjects that start with '`Re: `'. This is *not* an abbreviation of the English word "response", but is Latin, and means "in response to". Some illiterate nincompoops have failed to grasp this fact, and have "internationalized" their software to use abonimations like '`Aw: `' ("antwort") or '`Sv: `' ("svar") instead, which is meaningless and evil. However, you may have to deal with users that use these evil tools, in which case you may set this variable to a regexp that matches these prefixes. Myself, I just throw away non-compliant mail.

`message-alternative-emails`
: A regexp to match the alternative email addresses. The first matched address (not primary one) is used in the `From` field.

3.2 Mail Headers

`message-required-mail-headers`
: See Section 3.4 [News Headers], page 14, for the syntax of this variable. It is `(From Date Subject (optional . In-Reply-To) Message-ID Lines (optional . User-Agent))` by default.

`message-ignored-mail-headers`
: Regexp of headers to be removed before mailing. The default is '`^[GF]cc:\|^Resent-Fcc:\|^Xref:`'.

`message-default-mail-headers`
: This string is inserted at the end of the headers in all message buffers that are initialized as mail.

3.3 Mail Variables

`message-send-mail-function`
: Function used to send the current buffer as mail. The default is `message-send-mail-with-sendmail`. If you prefer using MH instead, set this variable to `message-send-mail-with-mh`.

`message-mh-deletable-headers`
: Most versions of MH doesn't like being fed messages that contain the headers in this variable. If this variable is non-`nil` (which is the default), these headers will be removed before mailing when sending messages via MH. Set it to `nil` if your MH can handle these headers.

`message-send-mail-partially-limit`
: The limit on the size of messages sent as '`message/partial`'. This is the minimum message size in characters beyond which the message should be sent in several parts. If it is `nil`, the size is unlimited.

3.4 News Headers

`message-required-news-headers` a list of header symbols. These headers will either be automatically generated, or, if that's impossible, they will be prompted for. The following symbols are valid:

`From`
: This required header will be filled out with the result of the `message-make-from` function, which depends on the `message-from-style`, `user-full-name`, `user-mail-address` variables.

`Subject`
: This required header will be prompted for if not present already.

`Newsgroups`
: This required header says which newsgroups the article is to be posted to. If it isn't present already, it will be prompted for.

`Organization`
: This optional header will be filled out depending on the `message-user-organization` variable. `message-user-organization-file` will be used if

this variable is `t`. This variable can also be a string (in which case this string will be used), or it can be a function (which will be called with no parameters and should return a string to be used).

`Lines` This optional header will be computed by Message.

`Message-ID`
This required header will be generated by Message. A unique ID will be created based on the date, time, user name and system name. Message will use `system-name` to determine the name of the system. If this isn't a fully qualified domain name (FQDN), Message will use `mail-host-address` as the FQDN of the machine.

`User-Agent`
This optional header will be filled out according to the `message-newsreader` local variable.

`In-Reply-To`
This optional header is filled out using the `Date` and `From` header of the article being replied to.

`Expires` This extremely optional header will be inserted according to the `message-expires` variable. It is highly deprecated and shouldn't be used unless you know what you're doing.

`Distribution`
This optional header is filled out according to the `message-distribution-function` variable. It is a deprecated and much misunderstood header.

`Path` This extremely optional header should probably never be used. However, some *very* old servers require that this header is present. `message-user-path` further controls how this `Path` header is to look. If it is `nil`, use the server name as the leaf node. If it is a string, use the string. If it is neither a string nor `nil`, use the user name only. However, it is highly unlikely that you should need to fiddle with this variable at all.

In addition, you can enter conses into this list. The car of this cons should be a symbol. This symbol's name is the name of the header, and the cdr can either be a string to be entered verbatim as the value of this header, or it can be a function to be called. This function should return a string to be inserted. For instance, if you want to insert `Mime-Version: 1.0`, you should enter `(Mime-Version . "1.0")` into the list. If you want to insert a funny quote, you could enter something like `(X-Yow . yow)` into the list. The function `yow` will then be called without any arguments.

If the list contains a cons where the car of the cons is `optional`, the cdr of this cons will only be inserted if it is non-`nil`.

Other variables for customizing outgoing news articles:

`message-syntax-checks`
Controls what syntax checks should not be performed on outgoing posts. To disable checking of long signatures, for instance, add

```
(signature . disabled)
```

to this list.

Valid checks are:

`subject-cmsg`
: Check the subject for commands.

`sender`
: Insert a new `Sender` header if the `From` header looks odd.

`multiple-headers`
: Check for the existence of multiple equal headers.

`sendsys`
: Check for the existence of version and sendsys commands.

`message-id`
: Check whether the `Message-ID` looks ok.

`from`
: Check whether the `From` header seems nice.

`long-lines`
: Check for too long lines.

`control-chars`
: Check for invalid characters.

`size`
: Check for excessive size.

`new-text`
: Check whether there is any new text in the messages.

`signature`
: Check the length of the signature.

`approved`
: Check whether the article has an `Approved` header, which is something only moderators should include.

`empty`
: Check whether the article is empty.

`invisible-text`
: Check whether there is any invisible text in the buffer.

`empty-headers`
: Check whether any of the headers are empty.

`existing-newsgroups`
: Check whether the newsgroups mentioned in the `Newsgroups` and `Followup-To` headers exist.

`valid-newsgroups`
: Check whether the `Newsgroups` and `Followup-to` headers are valid syntactically.

`repeated-newsgroups`
: Check whether the `Newsgroups` and `Followup-to` headers contains repeated group names.

`shorten-followup-to`
: Check whether to add a `Followup-to` header to shorten the number of groups to post to.

All these conditions are checked by default.

`message-ignored-news-headers`
: Regexp of headers to be removed before posting. The default is '`^NNTP-Posting-Host:\\|^Xref:\\|^[BGF]cc:\\|^Resent-Fcc:`'.

`message-default-news-headers`
: This string is inserted at the end of the headers in all message buffers that are initialized as news.

3.5 News Variables

`message-send-news-function`
: Function used to send the current buffer as news. The default is `message-send-news`.

`message-post-method`
: Gnusish *select method* (see the Gnus manual for details) used for posting a prepared news message.

3.6 Various Message Variables

`message-default-charset`
: Symbol naming a MIME charset. Non-ASCII characters in messages are assumed to be encoded using this charset. The default is `nil`, which means ask the user. (This variable is used only on non-MULE Emacsen. See section "Charset Translation" in *Emacs MIME Manual*, for details on the MULE-to-MIME translation process.

`message-signature-separator`
: Regexp matching the signature separator. It is '`^-- *$`' by default.

`mail-header-separator`
: String used to separate the headers from the body. It is '`--text follows this line--`' by default.

`message-directory`
: Directory used by many mailey things. The default is '`~/Mail/`'.

`message-signature-setup-hook`
: Hook run when initializing the message buffer. It is run after the headers have been inserted but before the signature has been inserted.

`message-setup-hook`
: Hook run as the last thing when the message buffer has been initialized, but before yanked text is inserted.

`message-header-setup-hook`
: Hook called narrowed to the headers after initializing the headers.

  For instance, if you're running Gnus and wish to insert a '`Mail-Copies-To`' header in all your news articles and all messages you send to mailing lists, you could do something like the following:

```
(defun my-message-header-setup-hook ()
  (let ((group (or gnus-newsgroup-name "")))
    (when (or (message-fetch-field "newsgroups")
              (gnus-group-find-parameter group 'to-address)
              (gnus-group-find-parameter group 'to-list))
      (insert "Mail-Copies-To: never\n"))))

(add-hook 'message-header-setup-hook
          'my-message-header-setup-hook)
```

`message-send-hook`
: Hook run before sending messages.

  If you want to add certain headers before sending, you can use the `message-add-header` function in this hook. For instance:

```
(add-hook 'message-send-hook 'my-message-add-content)
(defun my-message-add-content ()
  (message-add-header "X-In-No-Sense: Nonsense")
  (message-add-header "X-Whatever: no"))
```

  This function won't add the header if the header is already present.

`message-send-mail-hook`
: Hook run before sending mail messages.

`message-send-news-hook`
: Hook run before sending news messages.

`message-sent-hook`
: Hook run after sending messages.

`message-mode-syntax-table`
: Syntax table used in message mode buffers.

`message-send-method-alist`
: Alist of ways to send outgoing messages. Each element has the form

```
(TYPE PREDICATE FUNCTION)
```

  type A symbol that names the method.

  predicate A function called without any parameters to determine whether the message is a message of type *type*.

  function A function to be called if *predicate* returns non-`nil`. *function* is called with one parameter – the prefix.

```
((news message-news-p message-send-via-news)
 (mail message-mail-p message-send-via-mail))
```

3.7 Sending Variables

`message-fcc-handler-function`
: A function called to save outgoing articles. This function will be called with the name of the file to store the article in. The default function is `message-output` which saves in inbox format.

`message-courtesy-message`
: When sending combined messages, this string is inserted at the start of the mailed copy. If the string contains the format spec '`%s`', the newsgroups the article has been posted to will be inserted there. If this variable is `nil`, no such courtesy message will be added. The default value is '`"The following message is a courtesy copy of an article\nthat has been posted to %s as well.\n\n"`'.

3.8 Message Buffers

Message will generate new buffers with unique buffer names when you request a message buffer. When you send the message, the buffer isn't normally killed off. Its name is changed and a certain number of old message buffers are kept alive.

`message-generate-new-buffers`
: If non-`nil`, generate new buffers. The default is `t`. If this is a function, call that function with three parameters: The type, the to address and the group name. (Any of these may be `nil`.) The function should return the new buffer name.

`message-max-buffers`
: This variable says how many old message buffers to keep. If there are more message buffers than this, the oldest buffer will be killed. The default is 10. If this variable is `nil`, no old message buffers will ever be killed.

`message-send-rename-function`
: After sending a message, the buffer is renamed from, for instance, '`*reply to Lars*`' to '`*sent reply to Lars*`'. If you don't like this, set this variable to a function that renames the buffer in a manner you like. If you don't want to rename the buffer at all, you can say:

```
(setq message-send-rename-function 'ignore)
```

`message-kill-buffer-on-exit`
: If non-`nil`, kill the buffer immediately on exit.

3.9 Message Actions

When Message is being used from a news/mail reader, the reader is likely to want to perform some task after the message has been sent. Perhaps return to the previous window configuration or mark an article as replied.

The user may exit from the message buffer in various ways. The most common is *C-c C-c*, which sends the message and exits. Other possibilities are *C-c C-s* which just sends the message, *C-c C-d* which postpones the message editing and buries the message buffer, and *C-c C-k* which kills the message buffer. Each of these actions have lists associated with them that contains actions to be executed: `message-send-actions`, `message-exit-actions`, `message-postpone-actions`, and `message-kill-actions`.

Message provides a function to interface with these lists: `message-add-action`. The first parameter is the action to be added, and the rest of the arguments are which lists to add this action to. Here's an example from Gnus:

```
(message-add-action
 `(set-window-configuration ,(current-window-configuration))
 'exit 'postpone 'kill)
```

This restores the Gnus window configuration when the message buffer is killed, postponed or exited.

An *action* can be either: a normal function, or a list where the `car` is a function and the `cdr` is the list of arguments, or a form to be `eval`ed.

4 Compatibility

Message uses virtually only its own variables—older `mail-` variables aren't consulted. To force Message to take those variables into account, you can put the following in your `.emacs` file:

```
(require 'messcompat)
```

This will initialize many Message variables from the values in the corresponding mail variables.

5 Appendices

5.1 Responses

To determine where a message is to go, the following algorithm is used by default.

reply A *reply* is when you want to respond *just* to the person who sent the message via mail. There will only be one recipient. To determine who the recipient will be, the following headers are consulted, in turn:

`Reply-To`

`From`

wide reply A *wide reply* is a mail response that includes *all* entities mentioned in the message you are responded to. All mailboxes from the following headers will be concatenated to form the outgoing `To`/`Cc` headers:

`From` (unless there's a `Reply-To`, in which case that is used instead).

`Cc`

`To`

If a `Mail-Copies-To` header is present, it will also be included in the list of mailboxes. If this header is '`never`', that means that the `From` (or `Reply-To`) mailbox will be suppressed.

followup A *followup* is a response sent via news. The following headers (listed in order of precedence) determine where the response is to be sent:

`Followup-To`

`Newsgroups`

If a `Mail-Copies-To` header is present, it will be used as the basis of the new `Cc` header, except if this header is '`never`'.

6 Index

N

O

P

Q

S

U

Y

7 Key Index

C

M

Short Contents

Table of Contents