Message Manual

by Lars Magne Ingebrigtsen

Short Contents

Table of Contents

Message

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

Message is distributed with Gnus. The Gnus distribution corresponding to this manual is Ma Gnus v0.12

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.

You can customize the Message Mode tool bar, see *M-x customize-apropos RET message-tool-bar*. This feature is only available in Emacs.

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 32), 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 list. In this case, it returns `((To . "Whom"))` 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.

Each list element should be a cons, where the CAR should be the name of a header (e.g., `Cc`) and the CDR should be the header value (e.g., '`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 2).

Addresses that match the `message-dont-reply-to-names` regular expression (or list of regular expressions) will be removed from the `Cc` header. A value of `nil` means exclude your name only.

`message-prune-recipient-rules` is used to prune the addresses used when doing a wide reply. It's meant to be used to remove duplicate addresses and the like. It's a list of lists, where the first element is a regexp to match the address to trigger the rule, and the second is a regexp that will be expanded based on the first, to match addresses to be pruned.

It's complicated to explain, but it's easy to use.

For instance, if you get an email from '`foo@example.org`', but '`foo@zot.example.org`' is also in the `Cc` list, then your wide reply will go out to both these addresses, since they are unique.

To avoid this, do something like the following:

```
(setq message-prune-recipient-rules
      '(("^\\([^@]+\\)@\\(.*\\)" "\\1@.*[.]\\2")))
```

If, for instance, you want all wide replies that involve messages from '`cvs@example.org`' to go to that address, and nowhere else (i.e., remove all other recipients if '`cvs@example.org`' is in the recipient list:

```
(setq message-prune-recipient-rules
      '(("cvs@example.org" ".")))
```

If `message-wide-reply-confirm-recipients` is non-`nil` you will be asked to confirm that you want to reply to multiple recipients. The default is `nil`.

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 2).

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.

The value of `message-cancel-message` is inserted in the body of the cancel message. The default is '`I am canceling my own article.`'.

When Message posts news messages, it inserts `Cancel-Lock` headers by default. This is a cryptographic header that ensures that only you can cancel your own messages, which is nice. The downside is that if you lose your '`.emacs`' file (which is where Gnus stores the secret cancel lock password (which is generated automatically the first time you use this feature)), you won't be able to cancel your message. If you want to manage a password yourself, you can put something like the following in your '`~/.gnus.el`' file:

```
(setq canlock-password "geheimnis"
      canlock-password-for-verify canlock-password)
```

Whether to insert the header or not is controlled by the `message-insert-canlock` variable.

Not many news servers respect the `Cancel-Lock` header yet, but this is expected to change in the future.

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:\\|^X-Trace:\\|^X-Complaints-To:\\|`
`Return-Path:\\|^Supersedes:\\|^NNTP-Posting-Date:\\|^X-Trace:\\|`
`^X-Complaints-To:\\|^Cancel-Lock:\\|^Cancel-Key:\\|^X-Hashcash:\\|`
`^X-Payment:\\|^Approved:`'.

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`
: In non-`nil`, all headers that match this regexp will be deleted when forwarding a message.

`message-forward-included-headers`
: In non-`nil`, only headers that match this regexp will be kept 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.

`message-forward-before-signature`
: If non-`nil`, put forwarded message before signature, else after.

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.

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\\|Delivered-To\\):`'.

1.11 Mailing Lists

Sometimes while posting to mailing lists, the poster needs to direct followups to the post to specific places. The Mail-Followup-To (MFT) was created to enable just this. Three example scenarios where this is useful:

- A mailing list poster can use MFT to express that responses should be sent to just the list, and not the poster as well. This will happen if the poster is already subscribed to the list.
- A mailing list poster can use MFT to express that responses should be sent to the list and the poster as well. This will happen if the poster is not subscribed to the list.
- If a message is posted to several mailing lists, MFT may also be used to direct the following discussion to one list only, because discussions that are spread over several lists tend to be fragmented and very difficult to follow.

Gnus honors the MFT header in other's messages (i.e., while following up to someone else's post) and also provides support for generating sensible MFT headers for outgoing messages as well.

1.11.1 Composing a correct MFT header automagically

The first step in getting Gnus to automagically generate a MFT header in posts you make is to give Gnus a list of the mailing lists addresses you are subscribed to. You can do this in more than one way. The following variables would come in handy.

`message-subscribed-addresses`

This should be a list of addresses the user is subscribed to. Its default value is `nil`. Example:

```
(setq message-subscribed-addresses
      '("ding@gnus.org" "bing@noose.org"))
```

`message-subscribed-regexps`

This should be a list of regexps denoting the addresses of mailing lists subscribed to. Default value is `nil`. Example: If you want to achieve the same result as above:

```
(setq message-subscribed-regexps
      '("\\(ding@gnus\\)\\|\\(bing@noose\\)\\.org")
```

`message-subscribed-address-functions`

This can be a list of functions to be called (one at a time!!) to determine the value of MFT headers. It is advisable that these functions not take any arguments. Default value is `nil`.

There is a pre-defined function in Gnus that is a good candidate for this variable. `gnus-find-subscribed-addresses` is a function that returns a list of addresses corresponding to the groups that have the `subscribed` (see Section "Group Parameters" in *The Gnus Manual*) group parameter set to a non-`nil` value. This is how you would do it.

```
(setq message-subscribed-address-functions
      '(gnus-find-subscribed-addresses))
```

`message-subscribed-address-file`

You might be one organized human freak and have a list of addresses of all subscribed mailing lists in a separate file! Then you can just set this variable to the name of the file and life would be good.

You can use one or more of the above variables. All their values are "added" in some way that works :-)

Now you are all set. Just start composing a message as you normally do. And just send it; as always. Just before the message is sent out, Gnus' MFT generation thingy kicks in and checks if the message already has a MFT field. If there is one, it is left alone. (Except if it's empty; in that case, the field is removed and is not replaced with an automatically generated one. This lets you disable MFT generation on a per-message basis.) If there is none, then the list of recipient addresses (in the To: and Cc: headers) is checked to see if one of them is a list address you are subscribed to. If none of them is a list address, then no MFT is generated; otherwise, a MFT is added to the other headers and set to the value of all addresses in To: and Cc:

Hm. "So", you ask, "what if I send an email to a list I am not subscribed to? I want my MFT to say that I want an extra copy." (This is supposed to be interpreted by others the same way as if there were no MFT, but you can use an explicit MFT to override someone else's to-address group parameter.) The function `message-generate-unsubscribed-mail-followup-to` might come in handy. It is bound to *C-c C-f C-a* by default. In any case, you can insert a MFT of your own choice; *C-c C-f C-m* (`message-goto-mail-followup-to`) will help you get started.

1.11.2 Honoring an MFT post

When you followup to a post on a mailing list, and the post has a MFT header, Gnus' action will depend on the value of the variable `message-use-mail-followup-to`. This variable can be one of:

`use` Always honor MFTs. The To: and Cc: headers in your followup will be derived from the MFT header of the original post. This is the default.

`nil` Always dishonor MFTs (just ignore the darned thing)

`ask` Gnus will prompt you for an action.

It is considered good netiquette to honor MFT, as it is assumed the fellow who posted a message knows where the followups need to go better than you do.

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

2.2.1 Commands for moving to headers

These following 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-o
: Go to the `From` header (`message-goto-from`). (The "o" in the key binding is for Originator.)

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`).

C-c C-f C-i
: This inserts the '`Importance:`' header with a value of '`high`'. This header is used to signal the importance of the message to the receiver. If the header is already present in the buffer, it cycles between the three valid values according to RFC 1376: '`low`', '`normal`' and '`high`'.

C-c C-f C-a
: Insert a reasonable '`Mail-Followup-To:`' header (see Section 1.11 [Mailing Lists], page 5) in a post to an unsubscribed list. When making original posts to a mailing list you are not subscribed to, you have to type in a '`Mail-Followup-To:`' header by hand. The contents, usually, are the addresses of the list and your own address. This function inserts such a header automatically. It fetches the contents of the '`To:`' header in the current mail buffer, and appends the current `user-mail-address`.

  If the optional argument `include-cc` is non-`nil`, the addresses in the '`Cc:`' header are also put into the '`Mail-Followup-To:`' header.

2.2.2 Commands to change headers

C-c C-o
: Sort headers according to `message-header-format-alist` (`message-sort-headers`).

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 C-l
: Send a message to the list only. Remove all addresses but the list address from `To:` and `Cc:` headers.

C-c M-n
: Insert a request for a disposition notification. (`message-insert-disposition-notification-to`). This means that if the recipient supports RFC 2298 she might send you a notification that she received the message.

M-x message-insert-importance-high
: Insert an '`Importance`' header with a value of '`high`', deleting headers if necessary.

M-x message-insert-importance-low
: Insert an '`Importance`' header with a value of '`low`', deleting headers if necessary.

C-c C-f s
: Change the current '`Subject`' header. Ask for new '`Subject`' header and append '`(was: <Old Subject>)`'. The old subject can be stripped on replying, see `message-subject-trailing-was-query` (see Section 3.1 [Message Headers], page 17).

C-c C-f x
: Set up the '`FollowUp-To`' header with a target newsgroup for a cross-post, add that target newsgroup to the '`Newsgroups`' header if it is not a mem-

ber of '`Newsgroups`', and insert a note in the body. If `message-cross-post-default` is `nil` or if this command is called with a prefix-argument, only the '`FollowUp-To`' header will be set but the target newsgroup will not be added to the '`Newsgroups`' header. The function to insert a note is controlled by the `message-cross-post-note-function` variable.

C-c C-f t Replace contents of '`To`' header with contents of '`Cc`' header (or the '`Bcc`' header, if there is no '`Cc`' header).

C-c C-f w Insert '`To`' and '`Cc`' headers as if you were doing a wide reply even if the message was not made for a wide reply first.

C-c C-f a Insert '`X-No-Archive: Yes`' in the header and a note in the body. The header and the note can be customized using `message-archive-header` and `message-archive-note`. When called with a prefix argument, ask for a text to insert. If you don't want the note in the body, set `message-archive-note` to `nil`.

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`).

C-a If at beginning of header value, go to beginning of line, else go to beginning of header value. (The header value comes after the header name and the colon.) This behavior can be disabled by toggling the variable `message-beginning-of-line`.

2.4 Insertion

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

C-c C-M-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`).

C-c M-m Mark some region in the current article with enclosing tags. See `message-mark-insert-begin` and `message-mark-insert-end`. When called with a prefix argument, use slrn style verbatim marks ('`#v+`' and '`#v-`').

C-c M-f Insert a file in the current article with enclosing tags. See `message-mark-insert-begin` and `message-mark-insert-end`. When called with a prefix argument, use slrn style verbatim marks ('`#v+`' and '`#v-`').

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 (*M-x mml-attach-file*), which will prompt for a file name and a MIME type.

If your Emacs supports drag and drop, you can also drop the file in the Message buffer. The variable `mml-dnd-protocol-alist` specifies what kind of action is done when you drop a file into the Message buffer. The variable `mml-dnd-attach-options` controls which MIME options you want to specify when dropping a file. If it is a list, valid members are `type`, `description` and `disposition`. `disposition` implies `type`. If it is `nil`, don't ask for options. If it is `t`, ask the user whether or not to specify options.

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

2.6 IDNA

IDNA is a standard way to encode non-ASCII domain names into a readable ASCII string. The details can be found in RFC 3490.

Message is a IDNA-compliant posting agent. The user generally doesn't have to do anything to make the IDNA happen—Message will encode non-ASCII domain names in `From`, `To`, and `Cc` headers automatically.

Until IDNA becomes more well known, Message queries you whether IDNA encoding of the domain name really should occur. Some users might not be aware that domain names can contain non-ASCII now, so this gives them a safety net if they accidentally typed a non-ASCII domain name.

The `message-use-idna` variable control whether IDNA is used. If the variable is `nil` no IDNA encoding will ever happen, if it is set to the symbol `ask` the user will be queried, and if set to `t` (which is the default if IDNA is fully available) IDNA encoding happens automatically.

If you want to experiment with the IDNA encoding, you can invoke *M-x message-idna-to-ascii-rhs RET* in the message buffer to have the non-ASCII domain names encoded while you edit the message.

Note that you must have GNU Libidn (`http://www.gnu.org/software/libidn/`) installed in order to use this functionality.

2.7 Security

Using the MML language, Message is able to create digitally signed and digitally encrypted messages. Message (or rather MML) currently support PGP (RFC 1991), PGP/MIME (RFC 2015/3156) and S/MIME.

2.7.1 Signing and encrypting commands

Instructing MML to perform security operations on a MIME part is done using the *C-c C-m s* key map for signing and the *C-c C-m c* key map for encryption, as follows.

C-c C-m s s
: Digitally sign current message using S/MIME.

C-c C-m s o
: Digitally sign current message using PGP.

C-c C-m s p
: Digitally sign current message using PGP/MIME.

C-c C-m c s
: Digitally encrypt current message using S/MIME.

C-c C-m c o
: Digitally encrypt current message using PGP.

C-c C-m c p
: Digitally encrypt current message using PGP/MIME.

C-c C-m C-n
: Remove security related MML tags from message.

These commands do not immediately sign or encrypt the message, they merely insert the proper MML secure tag to instruct the MML engine to perform that operation when the message is actually sent. They may perform other operations too, such as locating and retrieving a S/MIME certificate of the person you wish to send encrypted mail to. When the mml parsing engine converts your MML into a properly encoded MIME message, the secure tag will be replaced with either a part or a multipart tag. If your message contains other mml parts, a multipart tag will be used; if no other parts are present in your message a single part tag will be used. This way, message mode will do the Right Thing (TM) with signed/encrypted multipart messages.

Since signing and especially encryption often is used when sensitive information is sent, you may want to have some way to ensure that your mail is actually signed or encrypted. After invoking the above sign/encrypt commands, it is possible to preview the raw article by using *C-u C-c RET P* (`mml-preview`). Then you can verify that your long rant about what your ex-significant other or whomever actually did with that funny looking person at that strange party the other night, actually will be sent encrypted.

Note! Neither PGP/MIME nor S/MIME encrypt/signs RFC822 headers. They only operate on the MIME object. Keep this in mind before sending mail with a sensitive Subject line.

By default, when encrypting a message, Gnus will use the “signencrypt” mode, which means the message is both signed and encrypted. If you would like to disable this for a particular message, give the `mml-secure-message-encrypt-*` command a prefix argument, e.g., *C-u C-c C-m c p*.

Actually using the security commands above is not very difficult. At least not compared with making sure all involved programs talk with each other properly. Thus, we now describe what external libraries or programs are required to make things work, and some small general hints.

2.7.2 Using S/MIME

Note! This section assume you have a basic familiarity with modern cryptography, S/MIME, various PKCS standards, OpenSSL and so on.

The S/MIME support in Message (and MML) require OpenSSL. OpenSSL performs the actual S/MIME sign/encrypt operations. OpenSSL can be found at `http://www.openssl.org/`. OpenSSL 0.9.6 and later should work. Version 0.9.5a cannot extract mail addresses from certificates, and it insert a spurious CR character into MIME separators so you may wish to avoid it if you would like to avoid being regarded as someone who send strange mail. (Although by sending S/MIME messages you've probably already lost that contest.)

To be able to send encrypted mail, a personal certificate is not required. Message (MML) need a certificate for the person to whom you wish to communicate with though. You're asked for this when you type *C-c C-m c s*. Currently there are two ways to retrieve this certificate, from a local file or from DNS. If you chose a local file, it need to contain a X.509 certificate in PEM format. If you chose DNS, you're asked for the domain name where the certificate is stored, the default is a good guess. To my belief, Message (MML) is the first mail agent in the world to support retrieving S/MIME certificates from DNS, so you're not likely to find very many certificates out there. At least there should be one, stored at the domain `simon.josefsson.org`. LDAP is a more popular method of distributing certificates, support for it is planned. (Meanwhile, you can use `ldapsearch` from the command line to retrieve a certificate into a file and use it.)

As for signing messages, OpenSSL can't perform signing operations without some kind of configuration. Especially, you need to tell it where your private key and your certificate is stored. MML uses an Emacs interface to OpenSSL, aptly named `smime.el`, and it contain a `custom` group used for this configuration. So, try *M-x customize-group RET smime RET* and look around.

Currently there is no support for talking to a CA (or RA) to create your own certificate. None is planned either. You need to do this manually with OpenSSL or using some other program. I used Netscape and got a free S/MIME certificate from one of the big CA's on the net. Netscape is able to export your private key and certificate in PKCS #12 format. Use OpenSSL to convert this into a plain X.509 certificate in PEM format as follows.

```
$ openssl pkcs12 -in ns.p12 -clcerts -nodes > key+cert.pem
```

The '`key+cert.pem`' file should be pointed to from the `smime-keys` variable. You should now be able to send signed mail.

Note! Your private key is now stored unencrypted in the file, so take care in handling it. Storing encrypted keys on the disk are supported, and Gnus will ask you for a passphrase before invoking OpenSSL. Read the OpenSSL documentation for how to achieve this. If you use unencrypted keys (e.g., if they are on a secure storage, or if you are on a secure single user machine) simply press `RET` at the passphrase prompt.

2.7.3 Using PGP/MIME

PGP/MIME requires an external OpenPGP implementation, such as GNU Privacy Guard (`http://www.gnupg.org/`). Pre-OpenPGP implementations such as PGP 2.x and PGP 5.x are also supported. The default Emacs interface to the PGP implementation is EasyPG (see Section "EasyPG Assistant User's Manual" in *EasyPG Assistant User's Manual*), but PGG (see Section "PGG" in *PGG Manual*) and Mailcrypt are also supported. See Section 2.7.4 [PGP Compatibility], page 14.

Message internally calls GnuPG (the `gpg` command) to perform data encryption, and in certain cases (decrypting or signing for example), `gpg` requires user's passphrase. Currently the recommended way to supply your passphrase to `gpg` is to use the `gpg-agent` program.

To use `gpg-agent` in Emacs, you need to run the following command from the shell before starting Emacs.

```
eval `gpg-agent --daemon`
```

This will invoke `gpg-agent` and set the environment variable `GPG_AGENT_INFO` to allow `gpg` to communicate with it. It might be good idea to put this command in your '`.xsession`' or '`.bash_profile`'. See Section "Invoking GPG-AGENT" in *Using the GNU Privacy Guard*.

Once your `gpg-agent` is set up, it will ask you for a passphrase as needed for `gpg`. Under the X Window System, you will see a new passphrase input dialog appear. The dialog is provided by PIN Entry (the `pinentry` command), and as of version 0.7.2, `pinentry` cannot cooperate with Emacs on a single tty. So, if you are using a text console, you may need to put a passphrase into gpg-agent's cache beforehand. The following command does the trick.

```
gpg --use-agent --sign < /dev/null > /dev/null
```

2.7.4 Compatibility with older implementations

Note, if you are using the `gpg.el` you must make sure that the directory specified by `gpg-temp-directory` have permissions 0700.

Creating your own key is described in detail in the documentation of your PGP implementation, so we refer to it.

If you have imported your old PGP 2.x key into GnuPG, and want to send signed and encrypted messages to your fellow PGP 2.x users, you'll discover that the receiver cannot understand what you send. One solution is to use PGP 2.x instead (e.g., if you use `pgg`, set `pgg-default-scheme` to `pgp`). You could also convince your fellow PGP 2.x users to convert to GnuPG. As a final workaround, you can make the sign and encryption work in two steps; separately sign, then encrypt a message. If you would like to change this behavior you can customize the `mml-signencrypt-style-alist` variable. For example:

```
(setq mml-signencrypt-style-alist '(("smime" separate)
                                    ("pgp" separate)
                                    ("pgpauto" separate)
                                    ("pgpmime" separate)))
```

This causes to sign and encrypt in two passes, thus generating a message that can be understood by PGP version 2.

(Refer to `http://www.gnupg.org/gph/en/pgp2x.html` for more information about the problem.)

2.8 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 (‘`[...]`’).

This is a format-spec string, and you can use ‘`%l`’ to say how many lines were removed, and ‘`%c`’ to say how many characters were removed.

C-c M-k Kill the address under point.

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

TAB If `message-tab-body-function` is non-`nil`, execute the function it specifies. Otherwise use the function bound to *TAB* in `text-mode-map` or `global-map`.

2.9 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`).

2.10 Mail Aliases

The `message-mail-alias-type` variable controls what type of mail alias expansion to use. Currently two forms are supported: `mailabbrev` and `ecomplete`. 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.

If you're using `ecomplete`, all addresses from `To` and `Cc` headers will automatically be put into the '`~/.ecompleterc`' file. When you enter text in the `To` and `Cc` headers, `ecomplete` will check out the values stored there and "electrically" say what completions are possible. To choose one of these completions, use the *M-n* command to move down to the list. Use *M-n* and *M-p* to move down and up the list, and *RET* to choose a completion.

2.11 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 `t`, generate all required headers before starting to compose the message. This can also be a list of headers to generate:

```
(setq message-generate-headers-first
      '(References))
```

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

Note that some headers will be removed and re-generated before posting, because of the variable `message-deletable-headers` (see below).

`message-draft-headers`
: When running Message from Gnus, the message buffers are associated with a draft group. `message-draft-headers` says which headers should be generated when a draft is written to the draft group.

`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`
: Header lines to be inserted in outgoing messages before you edit the message, so you can edit or delete their lines. If set to a string, it is directly inserted. If set to a function, it is called and its result is inserted.

`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 abominations 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.

Here's an example of a value to deal with these headers when responding to a message:

```
(setq message-subject-re-regexp
      (concat
       "^[ \t]*"
         "\\("
           "\\("
             "[Aa][Nn][Tt][Ww]\\.?\\|"     ; antw
             "[Aa][Ww]\\|"                 ; aw
             "[Ff][Ww][Dd]?\\|"            ; fwd
             "[Oo][Dd][Pp]\\|"             ; odp
             "[Rr][Ee]\\|"                 ; re
             "[Rr][\311\351][Ff]\\.?\\|"   ; ref
             "[Ss][Vv]"                    ; sv
           "\\)"
           "\\(\\[[0-9]*\\]\\)"
           "*:[ \t]*"
         "\\)"
       "*[ \t]*"
       ))
```

`message-subject-trailing-was-query`

Controls what to do with trailing '`(was: <old subject>)`' in subject lines. If `nil`, leave the subject unchanged. If it is the symbol `ask`, query the user what to do. In this case, the subject is matched against `message-subject-trailing-was-ask-regexp`. If `message-subject-trailing-was-query` is `t`, always strip the trailing old subject. In this case, `message-subject-trailing-was-regexp` is used.

`message-alternative-emails`

Regexp matching alternative email addresses. The first address in the To, Cc or From headers of the original article matching this variable is used as the From field of outgoing messages, replacing the default From value.

For example, if you have two secondary email addresses john@home.net and john.doe@work.com and want to use them in the From field when composing a reply to a message addressed to one of them, you could set this variable like this:

```
(setq message-alternative-emails
```

```
(regexp-opt '("john@home.net" "john.doe@work.com")))
```

This variable has precedence over posting styles and anything that runs off `message-setup-hook`.

`message-allow-no-recipients`
: Specifies what to do when there are no recipients other than `Gcc` or `Fcc`. If it is `always`, the posting is allowed. If it is `never`, the posting is not allowed. If it is `ask` (the default), you are prompted.

`message-hidden-headers`
: A regexp, a list of regexps, or a list where the first element is `not` and the rest are regexps. It says which headers to keep hidden when composing a message.

```
(setq message-hidden-headers
      '(not "From" "Subject" "To" "Cc" "Newsgroups"))
```

Headers are hidden using narrowing, you can use *`M-x widen`* to expose them in the buffer.

`message-header-synonyms`
: A list of lists of header synonyms. E.g., if this list contains a member list with elements `Cc` and `To`, then `message-carefully-insert-headers` will not insert a `To` header when the message is already `Cc`ed to the recipient.

3.2 Mail Headers

`message-required-mail-headers`
: See Section 3.4 [News Headers], page 21, for the syntax of this variable. It is `(From Subject Date (optional . In-Reply-To) Message-ID (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:\\|^X-Draft-From:\\|^X-Gnus-Agent-Meta-Information:`’.

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

`message-generate-hashcash`
: Variable that indicates whether ‘`X-Hashcash`’ headers should be computed for the message. See Section “Hashcash” in *The Gnus Manual*. If `opportunistic`, only generate the headers when it doesn’t lead to the user having to wait.

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`, or `smtpmail-send-it` according to the system. Other valid values include `message-send-mail-with-mailclient`,

`message-send-mail-with-mh`, `message-send-mail-with-qmail`, `message-smtpmail-send-it` and `feedmail-send-it`.

The function `message-send-mail-with-sendmail` pipes your article to the `sendmail` binary for further queuing and sending. When your local system is not configured for sending mail using `sendmail`, and you have access to a remote SMTP server, you can set `message-send-mail-function` to `smtpmail-send-it` and make sure to setup the `smtpmail` package correctly. An example:

```
(setq message-send-mail-function 'smtpmail-send-it
      smtpmail-default-smtp-server "YOUR SMTP HOST")
```

To the thing similar to this, there is `message-smtpmail-send-it`. It is useful if your ISP requires the POP-before-SMTP authentication. See Section "POP before SMTP" in *The Gnus Manual*.

If you have a complex SMTP setup, and want some messages to go via one mail server, and other messages to go through another, you can use the '`X-Message-SMTP-Method`' header. These are the supported values:

'`smtpmail`'

```
X-Message-SMTP-Method: smtp smtp.fsf.org 587
```

This will send the message via '`smtp.fsf.org`', using port 587.

```
X-Message-SMTP-Method: smtp smtp.fsf.org 587 other-user
```

This is the same as the above, but uses '`other-user`' as the user name when authenticating. This is handy if you have several SMTP accounts on the same server.

'`sendmail`'

```
X-Message-SMTP-Method: sendmail
```

This will send the message via the locally installed sendmail/exim/etc installation.

`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-qmail-inject-program`

Location of the qmail-inject program.

`message-qmail-inject-args`

Arguments passed to qmail-inject programs. This should be a list of strings, one string for each argument. It may also be a function.

E.g., if you wish to set the envelope sender address so that bounces go to the right place or to deal with listserv's usage of that address, you might set this variable to `'("-f" "you@some.where")`.

`message-sendmail-f-is-evil`

Non-`nil` means don't add '`-f username`' to the sendmail command line. Doing so would be even more evil than leaving it out.

`message-sendmail-envelope-from`
: When `message-sendmail-f-is-evil` is `nil`, this specifies the address to use in the SMTP envelope. If it is `nil`, use `user-mail-address`. If it is the symbol `header`, use the '`From`' header of the message.

`message-mailer-swallows-blank-line`
: Set this to non-`nil` if the system's mailer runs the header and body together. (This problem exists on SunOS 4 when sendmail is run in remote mode.) The value should be an expression to test whether the problem will actually occur.

`message-send-mail-partially-limit`
: The limitation of messages sent as message/partial. The lower bound of message size in characters, beyond which the message should be sent in several parts. If it is `nil` (which is the default), 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 (for the local part) and the domain part. For the domain part, message will look (in this order) at `message-user-fqdn`, `system-name`, `mail-host-address` and `message-user-mail-address` (i.e., `user-mail-address`) until a probably valid fully qualified domain name (FQDN) was found.

`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 take no arguments, and 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 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`.

If you want to delete an entry from this list, the following Lisp snippet might be useful. Adjust accordingly if you want to remove another element.

```
(setq message-required-news-headers
      (delq 'Message-ID message-required-news-headers))
```

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:

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

`continuation-headers`
: Check whether there are continuation header lines that don't begin with whitespace.

`control-chars`
: Check for invalid characters.

`empty`
: Check whether the article is empty.

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

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

`illegible-text`
: Check whether there is any non-printable character in the body.

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

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

`long-lines`
: Check for too long lines in the body.

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

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

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

`newsgroups`
: Check whether the `Newsgroups` header exists and is not empty.

`quoting-style`
: Check whether text follows last quoted portion.

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

`reply-to`
: Check whether the `Reply-To` header looks ok.

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

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

`shoot`
: Check whether the domain part of the `Message-ID` header looks ok.

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

`signature`
: Check the length of the signature.

`size`
: Check for excessive size.

`subject`
: Check whether the `Subject` header exists and is not empty.

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

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

All these conditions are checked by default, except for `sender` for which the check is disabled by default if `message-insert-canlock` is non-`nil` (see Section 1.6 [Canceling News], page 4).

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

`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 Insertion Variables

`message-cite-style`
: The overall style to be used when replying to messages. This controls things like where the reply should be put relative to the original, how the citation is formatted, where the signature goes, etc.

Value is either `nil` (no variable overrides) or a let-style list of pairs `(VARIABLE VALUE)` to override default values.

See `gnus-posting-styles` to set this variable for specific groups. Presets to impersonate popular mail agents are available in the `message-cite-style-*` variables.

`message-cite-reply-position`
: Where the reply should be positioned. Available styles are `traditional` to reply inline, `above` for top-posting, and `below` for bottom-posting

`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-cite-prefix-regexp`
: Regexp matching the longest possible citation prefix on a line.

`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.

Note that Gnus provides a feature where clicking on 'writes:' hides the cited text. If you change the citation line too much, readers of your messages will have to adjust their Gnus, too. See the variable `gnus-cite-attribution-suffix`. See Section "Article Highlighting" in *The Gnus Manual*, for details.

`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 line you yank will have `message-yank-prefix` prepended to it (except for quoted lines which use `message-yank-cited-prefix` and empty lines which use `message-yank-empty-prefix`). The default is '`> `'.

`message-yank-cited-prefix`
: When yanking text from an article which contains already cited text, each line will be prefixed with the contents of this variable. The default is '`>`'. See also `message-yank-prefix`.

`message-yank-empty-prefix`
: When yanking text from an article, each empty line will be prefixed with the contents of this variable. The default is '`>`'. You can set this variable to an empty string to split the cited text into paragraphs automatically. See also `message-yank-prefix`.

`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.

`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-mark-insert-begin`
: String to mark the beginning of some inserted text.

`message-mark-insert-end`
: String to mark the end of some inserted text.

`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`
File containing the signature to be inserted at the end of the buffer. If a path is specified, the value of `message-signature-directory` is ignored, even if set. The default is '`~/.signature`'.

`message-signature-directory`
Name of directory containing signature files. Comes in handy if you have many such files, handled via Gnus posting styles for instance. If `nil` (the default), `message-signature-file` is expected to specify the directory if needed.

`message-signature-insert-empty-line`
If `t` (the default value) an empty line is inserted before the signature separator.

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.

3.7 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 `iso-8859-1` on non-MULE Emacsen; otherwise `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-fill-column`
Local value for the column beyond which automatic line-wrapping should happen for message buffers. If non-`nil` (the default), also turn on auto-fill in message buffers.

`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 mailish things. The default is '`~/Mail/`'. All other mail file variables are derived from `message-directory`.

`message-auto-save-directory`
Directory where Message auto-saves buffers if Gnus isn't running. If `nil`, Message won't auto-save. The default is '`~/Mail/drafts/`'.

`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. This hook is run very late: just before the message is actually sent as mail.

`message-send-news-hook`
: Hook run before sending news messages. This hook is run very late: just before the message is actually sent as news.

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

`message-cancel-hook`
: Hook run when canceling news articles.

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

`message-cite-articles-with-x-no-archive`
: If non-`nil`, don't strip quoted text from articles that have '`X-No-Archive`' set. Even if this variable isn't set, you can undo the stripping by hitting the `undo` keystroke.

`message-strip-special-text-properties`
: Emacs has a number of special text properties which can break message composing in various ways. If this option is set, message will strip these properties from the message composition buffer. However, some packages requires these properties to be present in order to work. If you use one of these packages, turn this option off, and hope the message composition doesn't break too bad.

`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*. The function will be called in the buffer where the message is.

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

    The default is:

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

The `message-news-p` function returns non-`nil` if the message looks like news, and the `message-send-via-news` function sends the message according to the `message-send-news-function` variable (see Section 3.5 [News Variables], page 24). The `message-mail-p` function returns non-`nil` if the message looks like mail, and the `message-send-via-mail` function sends the message according to the `message-send-mail-function` variable (see Section 3.3 [Mail Variables], page 19).

All the elements in this alist will be tried in order, so a message containing both a valid '`Newsgroups`' header and a valid '`To`' header, for example, will be sent as news, and then as mail.

3.8 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 Unix mailbox 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"’.

`message-fcc-externalize-attachments`
If `nil`, attach files as normal parts in Fcc copies; if it is non-`nil`, attach local files as external parts.

`message-interactive`
If non-`nil` wait for and display errors when sending a message; if `nil` let the mailer mail back a message to report errors.

`message-confirm-send`
When non-`nil`, Gnus will ask for confirmation when sending a message.

3.9 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`
Controls whether to create a new message buffer to compose a message. Valid values include:

`nil` Generate the buffer name in the Message way (e.g., *mail*, *news*, *mail to whom*, *news on group*, etc.) and continue editing in the existing buffer of that name. If there is no such buffer, it will be newly created.

`unique`

`t` Create the new buffer with the name generated in the Message way.

`unsent` Similar to `unique` but the buffer name begins with "*unsent ".

`standard` Similar to `nil` but the buffer name is simpler like *mail message*.

function 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.

The default value is `unsent`.

`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.

3.10 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 responding 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 GNU Free Documentation License

Version 1.2, November 2002

0. PREAMBLE

   The purpose of this License is to make a manual, textbook, or other functional and useful document "free" in the sense of freedom: to assure everyone the effective freedom to copy and redistribute it, with or without modifying it, either commercially or noncommercially. Secondarily, this License preserves for the author and publisher a way to get credit for their work, while not being considered responsible for modifications made by others.

   This License is a kind of "copyleft," which means that derivative works of the document must themselves be free in the same sense. It complements the GNU General Public License, which is a copyleft license designed for free software.

   We have designed this License in order to use it for manuals for free software, because free software needs free documentation: a free program should come with manuals providing the same freedoms that the software does. But this License is not limited to software manuals; it can be used for any textual work, regardless of subject matter or whether it is published as a printed book. We recommend this License principally for works whose purpose is instruction or reference.

1. APPLICABILITY AND DEFINITIONS

   This License applies to any manual or other work, in any medium, that contains a notice placed by the copyright holder saying it can be distributed under the terms of this License. Such a notice grants a world-wide, royalty-free license, unlimited in duration, to use that work under the conditions stated herein. The "Document," below, refers to any such manual or work. Any member of the public is a licensee, and is addressed as "you." You accept the license if you copy, modify or distribute the work in a way requiring permission under copyright law.

   A "Modified Version" of the Document means any work containing the Document or a portion of it, either copied verbatim, or with modifications and/or translated into another language.

   A "Secondary Section" is a named appendix or a front-matter section of the Document that deals exclusively with the relationship of the publishers or authors of the Document to the Document's overall subject (or to related matters) and contains nothing that could fall directly within that overall subject. (Thus, if the Document is in part a textbook of mathematics, a Secondary Section may not explain any mathematics.) The relationship could be a matter of historical connection with the subject or with related matters, or of legal, commercial, philosophical, ethical or political position regarding them.

The “Invariant Sections” are certain Secondary Sections whose titles are designated, as being those of Invariant Sections, in the notice that says that the Document is released under this License. If a section does not fit the above definition of Secondary then it is not allowed to be designated as Invariant. The Document may contain zero Invariant Sections. If the Document does not identify any Invariant Sections then there are none.

The “Cover Texts” are certain short passages of text that are listed, as Front-Cover Texts or Back-Cover Texts, in the notice that says that the Document is released under this License. A Front-Cover Text may be at most 5 words, and a Back-Cover Text may be at most 25 words.

A “Transparent” copy of the Document means a machine-readable copy, represented in a format whose specification is available to the general public, that is suitable for revising the document straightforwardly with generic text editors or (for images composed of pixels) generic paint programs or (for drawings) some widely available drawing editor, and that is suitable for input to text formatters or for automatic translation to a variety of formats suitable for input to text formatters. A copy made in an otherwise Transparent file format whose markup, or absence of markup, has been arranged to thwart or discourage subsequent modification by readers is not Transparent. An image format is not Transparent if used for any substantial amount of text. A copy that is not “Transparent” is called “Opaque.”

Examples of suitable formats for Transparent copies include plain ASCII without markup, Texinfo input format, LaTeX input format, SGML or XML using a publicly available DTD, and standard-conforming simple HTML, PostScript or PDF designed for human modification. Examples of transparent image formats include PNG, XCF and JPG. Opaque formats include proprietary formats that can be read and edited only by proprietary word processors, SGML or XML for which the DTD and/or processing tools are not generally available, and the machine-generated HTML, PostScript or PDF produced by some word processors for output purposes only.

The “Title Page” means, for a printed book, the title page itself, plus such following pages as are needed to hold, legibly, the material this License requires to appear in the title page. For works in formats which do not have any title page as such, “Title Page” means the text near the most prominent appearance of the work’s title, preceding the beginning of the body of the text.

A section “Entitled XYZ” means a named subunit of the Document whose title either is precisely XYZ or contains XYZ in parentheses following text that translates XYZ in another language. (Here XYZ stands for a specific section name mentioned below, such as “Acknowledgements,” “Dedications,” “Endorsements,” or “History.”) To “Preserve the Title” of such a section when you modify the Document means that it remains a section “Entitled XYZ” according to this definition.

The Document may include Warranty Disclaimers next to the notice which states that this License applies to the Document. These Warranty Disclaimers are considered to be included by reference in this License, but only as regards disclaiming warranties: any other implication that these Warranty Disclaimers may have is void and has no effect on the meaning of this License.

2. VERBATIM COPYING

You may copy and distribute the Document in any medium, either commercially or noncommercially, provided that this License, the copyright notices, and the license notice saying this License applies to the Document are reproduced in all copies, and that you add no other conditions whatsoever to those of this License. You may not use technical measures to obstruct or control the reading or further copying of the copies you make or distribute. However, you may accept compensation in exchange for copies. If you distribute a large enough number of copies you must also follow the conditions in section 3.

You may also lend copies, under the same conditions stated above, and you may publicly display copies.

3. COPYING IN QUANTITY

If you publish printed copies (or copies in media that commonly have printed covers) of the Document, numbering more than 100, and the Document's license notice requires Cover Texts, you must enclose the copies in covers that carry, clearly and legibly, all these Cover Texts: Front-Cover Texts on the front cover, and Back-Cover Texts on the back cover. Both covers must also clearly and legibly identify you as the publisher of these copies. The front cover must present the full title with all words of the title equally prominent and visible. You may add other material on the covers in addition. Copying with changes limited to the covers, as long as they preserve the title of the Document and satisfy these conditions, can be treated as verbatim copying in other respects.

If the required texts for either cover are too voluminous to fit legibly, you should put the first ones listed (as many as fit reasonably) on the actual cover, and continue the rest onto adjacent pages.

If you publish or distribute Opaque copies of the Document numbering more than 100, you must either include a machine-readable Transparent copy along with each Opaque copy, or state in or with each Opaque copy a computer-network location from which the general network-using public has access to download using public-standard network protocols a complete Transparent copy of the Document, free of added material. If you use the latter option, you must take reasonably prudent steps, when you begin distribution of Opaque copies in quantity, to ensure that this Transparent copy will remain thus accessible at the stated location until at least one year after the last time you distribute an Opaque copy (directly or through your agents or retailers) of that edition to the public.

It is requested, but not required, that you contact the authors of the Document well before redistributing any large number of copies, to give them a chance to provide you with an updated version of the Document.

4. MODIFICATIONS

You may copy and distribute a Modified Version of the Document under the conditions of sections 2 and 3 above, provided that you release the Modified Version under precisely this License, with the Modified Version filling the role of the Document, thus licensing distribution and modification of the Modified Version to whoever possesses a copy of it. In addition, you must do these things in the Modified Version:

A. Use in the Title Page (and on the covers, if any) a title distinct from that of the Document, and from those of previous versions (which should, if there were any, be listed in the History section of the Document). You may use the same title as a previous version if the original publisher of that version gives permission.
B. List on the Title Page, as authors, one or more persons or entities responsible for authorship of the modifications in the Modified Version, together with at least five of the principal authors of the Document (all of its principal authors, if it has fewer than five), unless they release you from this requirement.
C. State on the Title page the name of the publisher of the Modified Version, as the publisher.
D. Preserve all the copyright notices of the Document.
E. Add an appropriate copyright notice for your modifications adjacent to the other copyright notices.
F. Include, immediately after the copyright notices, a license notice giving the public permission to use the Modified Version under the terms of this License, in the form shown in the Addendum below.
G. Preserve in that license notice the full lists of Invariant Sections and required Cover Texts given in the Document's license notice.
H. Include an unaltered copy of this License.
I. Preserve the section Entitled "History," Preserve its Title, and add to it an item stating at least the title, year, new authors, and publisher of the Modified Version as given on the Title Page. If there is no section Entitled "History" in the Document, create one stating the title, year, authors, and publisher of the Document as given on its Title Page, then add an item describing the Modified Version as stated in the previous sentence.
J. Preserve the network location, if any, given in the Document for public access to a Transparent copy of the Document, and likewise the network locations given in the Document for previous versions it was based on. These may be placed in the "History" section. You may omit a network location for a work that was published at least four years before the Document itself, or if the original publisher of the version it refers to gives permission.
K. For any section Entitled "Acknowledgements" or "Dedications," Preserve the Title of the section, and preserve in the section all the substance and tone of each of the contributor acknowledgements and/or dedications given therein.
L. Preserve all the Invariant Sections of the Document, unaltered in their text and in their titles. Section numbers or the equivalent are not considered part of the section titles.
M. Delete any section Entitled "Endorsements." Such a section may not be included in the Modified Version.
N. Do not retitle any existing section to be Entitled "Endorsements" or to conflict in title with any Invariant Section.
O. Preserve any Warranty Disclaimers.

If the Modified Version includes new front-matter sections or appendices that qualify as Secondary Sections and contain no material copied from the Document, you may at your option designate some or all of these sections as invariant. To do this, add their titles to the list of Invariant Sections in the Modified Version's license notice. These titles must be distinct from any other section titles.

You may add a section Entitled "Endorsements," provided it contains nothing but endorsements of your Modified Version by various parties–for example, statements of peer review or that the text has been approved by an organization as the authoritative definition of a standard.

You may add a passage of up to five words as a Front-Cover Text, and a passage of up to 25 words as a Back-Cover Text, to the end of the list of Cover Texts in the Modified Version. Only one passage of Front-Cover Text and one of Back-Cover Text may be added by (or through arrangements made by) any one entity. If the Document already includes a cover text for the same cover, previously added by you or by arrangement made by the same entity you are acting on behalf of, you may not add another; but you may replace the old one, on explicit permission from the previous publisher that added the old one.

The author(s) and publisher(s) of the Document do not by this License give permission to use their names for publicity for or to assert or imply endorsement of any Modified Version.

5. COMBINING DOCUMENTS

   You may combine the Document with other documents released under this License, under the terms defined in section 4 above for modified versions, provided that you include in the combination all of the Invariant Sections of all of the original documents, unmodified, and list them all as Invariant Sections of your combined work in its license notice, and that you preserve all their Warranty Disclaimers.

   The combined work need only contain one copy of this License, and multiple identical Invariant Sections may be replaced with a single copy. If there are multiple Invariant Sections with the same name but different contents, make the title of each such section unique by adding at the end of it, in parentheses, the name of the original author or publisher of that section if known, or else a unique number. Make the same adjustment to the section titles in the list of Invariant Sections in the license notice of the combined work.

   In the combination, you must combine any sections Entitled "History" in the various original documents, forming one section Entitled "History"; likewise combine any sections Entitled "Acknowledgements," and any sections Entitled "Dedications." You must delete all sections Entitled "Endorsements."

6. COLLECTIONS OF DOCUMENTS

   You may make a collection consisting of the Document and other documents released under this License, and replace the individual copies of this License in the various documents with a single copy that is included in the collection, provided that you follow the rules of this License for verbatim copying of each of the documents in all other respects.

   You may extract a single document from such a collection, and distribute it individually under this License, provided you insert a copy of this License into the extracted document, and follow this License in all other respects regarding verbatim copying of that document.

7. AGGREGATION WITH INDEPENDENT WORKS

   A compilation of the Document or its derivatives with other separate and independent documents or works, in or on a volume of a storage or distribution medium, is called an "aggregate" if the copyright resulting from the compilation is not used to limit the legal rights of the compilation's users beyond what the individual works permit. When the Document is included in an aggregate, this License does not apply to the other works in the aggregate which are not themselves derivative works of the Document.

   If the Cover Text requirement of section 3 is applicable to these copies of the Document, then if the Document is less than one half of the entire aggregate, the Document's Cover Texts may be placed on covers that bracket the Document within the aggregate, or the electronic equivalent of covers if the Document is in electronic form. Otherwise they must appear on printed covers that bracket the whole aggregate.

8. TRANSLATION

   Translation is considered a kind of modification, so you may distribute translations of the Document under the terms of section 4. Replacing Invariant Sections with translations requires special permission from their copyright holders, but you may include translations of some or all Invariant Sections in addition to the original versions of these Invariant Sections. You may include a translation of this License, and all the license notices in the Document, and any Warranty Disclaimers, provided that you also include the original English version of this License and the original versions of those notices and disclaimers. In case of a disagreement between the translation and the original version of this License or a notice or disclaimer, the original version will prevail.

   If a section in the Document is Entitled "Acknowledgements," "Dedications," or "History," the requirement (section 4) to Preserve its Title (section 1) will typically require changing the actual title.

9. TERMINATION

   You may not copy, modify, sublicense, or distribute the Document except as expressly provided for under this License. Any other attempt to copy, modify, sublicense or distribute the Document is void, and will automatically terminate your rights under this License. However, parties who have received copies, or rights, from you under this License will not have their licenses terminated so long as such parties remain in full compliance.

10. FUTURE REVISIONS OF THIS LICENSE

    The Free Software Foundation may publish new, revised versions of the GNU Free Documentation License from time to time. Such new versions will be similar in spirit to the present version, but may differ in detail to address new problems or concerns. See http://www.gnu.org/copyleft/.

Each version of the License is given a distinguishing version number. If the Document specifies that a particular numbered version of this License "or any later version" applies to it, you have the option of following the terms and conditions either of that specified version or of any later version that has been published (not as a draft) by the Free Software Foundation. If the Document does not specify a version number of this License, you may choose any version ever published (not as a draft) by the Free Software Foundation.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.2
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts.
A copy of the license is included in the section entitled ``GNU
Free Documentation License.''
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the "with...Texts." line with this:

```
with the Invariant Sections being list their titles, with the
Front-Cover Texts being list, and with the Back-Cover Texts being
list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

7 Index

8 Key Index

C

M

T