Emacs MIME Manual

by Lars Magne Ingebrigtsen

This file documents the Emacs MIME interface functionality.

Table of Contents

Emacs MIME

This manual documents the libraries used to compose and display MIME messages.

This manual is directed at users who want to modify the behavior of the MIME encoding/decoding process or want a more detailed picture of how the Emacs MIME library works, and people who want to write functions and commands that manipulate MIME elements.

MIME is short for *Multipurpose Internet Mail Extensions*. This standard is documented in a number of RFCs; mainly RFC2045 (Format of Internet Message Bodies), RFC2046 (Media Types), RFC2047 (Message Header Extensions for Non-ASCII Text), RFC2048 (Registration Procedures), RFC2049 (Conformance Criteria and Examples). It is highly recommended that anyone who intends writing MIME-compliant software read at least RFC2045 and RFC2047.

1 Decoding and Viewing

This chapter deals with decoding and viewing MIME messages on a higher level.

The main idea is to first analyze a MIME article, and then allow other programs to do things based on the list of *handles* that are returned as a result of this analysis.

1.1 Dissection

The `mm-dissect-buffer` is the function responsible for dissecting a MIME article. If given a multipart message, it will recursively descend the message, following the structure, and return a tree of MIME handles that describes the structure of the message.

1.2 Non-MIME

Gnus also understands some non-MIME attachments, such as postscript, uuencode, binhex, yenc, shar, forward, gnatsweb, pgp, diff. Each of these features can be disabled by add an item into `mm-uu-configure-list`. For example,

```
(require 'mm-uu)
(add-to-list 'mm-uu-configure-list '(pgp-signed . disabled))
```

`postscript`
: PostScript file.

`uu`
: Uuencoded file.

`binhex`
: Binhex encoded file.

`yenc`
: Yenc encoded file.

`shar`
: Shar archive file.

`forward`
: Non-MIME forwarded message.

`gnatsweb`
: Gnatsweb attachment.

`pgp-signed`
: PGP signed clear text.

`pgp-encrypted`
: PGP encrypted clear text.

`pgp-key`
: PGP public keys.

`emacs-sources`
: Emacs source code. This item works only in the groups matching `mm-uu-emacs-sources-regexp`.

`diff`
: Patches. This is intended for groups where diffs of committed files are automatically sent to. It only works in groups matching `mm-uu-diff-groups-regexp`.

`verbatim-marks`
: Slrn-style verbatim marks.

`LaTeX`
: LaTeX documents. It only works in groups matching `mm-uu-tex-groups-regexp`.

Some inlined non-MIME attachments are displayed using the face `mm-uu-extract`. By default, no MIME button for these parts is displayed. You can force displaying a button using *K b* (`gnus-summary-display-buttonized`) or add `text/x-verbatim` to `gnus-buttonized-mime-types`, See section "MIME Commands" in *Gnus Manual*.

1.3 Handles

A MIME handle is a list that fully describes a MIME component.

The following macros can be used to access elements in a handle:

`mm-handle-buffer`
: Return the buffer that holds the contents of the undecoded MIME part.

`mm-handle-type`
: Return the parsed `Content-Type` of the part.

`mm-handle-encoding`
: Return the `Content-Transfer-Encoding` of the part.

`mm-handle-undisplayer`
: Return the object that can be used to remove the displayed part (if it has been displayed).

`mm-handle-set-undisplayer`
: Set the undisplayer object.

`mm-handle-disposition`
: Return the parsed `Content-Disposition` of the part.

`mm-get-content-id`
: Returns the handle(s) referred to by `Content-ID`.

1.4 Display

Functions for displaying, removing and saving.

`mm-display-part`
: Display the part.

`mm-remove-part`
: Remove the part (if it has been displayed).

`mm-inlinable-p`
: Say whether a MIME type can be displayed inline.

`mm-automatic-display-p`
: Say whether a MIME type should be displayed automatically.

`mm-destroy-part`
: Free all resources occupied by a part.

`mm-save-part`
: Offer to save the part in a file.

`mm-pipe-part`
: Offer to pipe the part to some process.

`mm-interactively-view-part`
: Prompt for a mailcap method to use to view the part.

1.5 Display Customization

`mm-inline-media-tests`
: This is an alist where the key is a MIME type, the second element is a function to display the part *inline* (i.e., inside Emacs), and the third element is a form to be `eval`ed to say whether the part can be displayed inline.

  This variable specifies whether a part *can* be displayed inline, and, if so, how to do it. It does not say whether parts are *actually* displayed inline.

`mm-inlined-types`
: This, on the other hand, says what types are to be displayed inline, if they satisfy the conditions set by the variable above. It's a list of MIME media types.

`mm-automatic-display`
: This is a list of types that are to be displayed "automatically", but only if the above variable allows it. That is, only inlinable parts can be displayed automatically.

`mm-automatic-external-display`
: This is a list of types that will be displayed automatically in an external viewer.

`mm-keep-viewer-alive-types`
: This is a list of media types for which the external viewer will not be killed when selecting a different article.

`mm-attachment-override-types`
: Some MIME agents create parts that have a content-disposition of '`attachment`'. This variable allows overriding that disposition and displaying the part inline. (Note that the disposition is only overridden if we are able to, and want to, display the part inline.)

`mm-discouraged-alternatives`
: List of MIME types that are discouraged when viewing '`multipart/alternative`'. Viewing agents are supposed to view the last possible part of a message, as that is supposed to be the richest. However, users may prefer other types instead, and this list says what types are most unwanted. If, for instance, '`text/html`' parts are very unwanted, and '`text/richtext`' parts are somewhat unwanted, you could say something like:

```
(setq mm-discouraged-alternatives
      '("text/html" "text/richtext")
      mm-automatic-display
      (remove "text/html" mm-automatic-display))
```

  Adding `"image/.*"` might also be useful. Spammers use images as the preferred part of '`multipart/alternative`' messages, so you might not notice there are other parts. See also `gnus-buttonized-mime-types`, section "MIME Commands" in *Gnus Manual*. After adding `"multipart/alternative"` to `gnus-`

`buttonized-mime-types` you can choose manually which alternative you'd like to view. For example, you can set those variables like:

```
(setq gnus-buttonized-mime-types
      '("multipart/alternative" "multipart/signed")
      mm-discouraged-alternatives
      '("text/html" "image/.*"))
```

In this case, Gnus will display radio buttons for such a kind of spam message as follows:

```
1.  (*) multipart/alternative  ( ) image/gif

2.  (*) text/plain             ( ) text/html
```

`mm-inline-large-images`
: When displaying inline images that are larger than the window, Emacs does not enable scrolling, which means that you cannot see the whole image. To prevent this, the library tries to determine the image size before displaying it inline, and if it doesn't fit the window, the library will display it externally (e.g. with '`ImageMagick`' or '`xv`'). Setting this variable to `t` disables this check and makes the library display all inline images as inline, regardless of their size.

`mm-inline-override-types`
: `mm-inlined-types` may include regular expressions, for example to specify that all '`text/.*`' parts be displayed inline. If a user prefers to have a type that matches such a regular expression be treated as an attachment, that can be accomplished by setting this variable to a list containing that type. For example assuming `mm-inlined-types` includes '`text/.*`', then including '`text/html`' in this variable will cause '`text/html`' parts to be treated as attachments.

`mm-text-html-renderer`
: This selects the function used to render HTML. The predefined renderers are selected by the symbols `gnus-article-html`, `w3`, `w3m`[1], `links`, `lynx`, `w3m-standalone` or `html2text`. If `nil` use an external viewer. You can also specify a function, which will be called with a MIME handle as the argument.

`mm-inline-text-html-with-images`
: Some HTML mails might have the trick of spammers using '`<img>`' tags. It is likely to be intended to verify whether you have read the mail. You can prevent your personal informations from leaking by setting this option to `nil` (which is the default). It is currently ignored by Emacs/w3. For emacs-w3m, you may use the command `t` on the image anchor to show an image even if it is `nil`.[2]

`mm-w3m-safe-url-regexp`
: A regular expression that matches safe URL names, i.e. URLs that are unlikely to leak personal information when rendering HTML email (the default value is '`\\`cid:`'). If `nil` consider all URLs safe. In Gnus, this will be overridden

[1] See http://emacs-w3m.namazu.org/ for more information about emacs-w3m

[2] The command `T` will load all images. If you have set the option `w3m-key-binding` to `info`, use `i` or `I` instead.

according to the value of the variable `gnus-safe-html-newsgroups`, See section "Various Various" in *Gnus Manual*.

`mm-inline-text-html-with-w3m-keymap`
: You can use emacs-w3m command keys in the inlined text/html part by setting this option to non-`nil`. The default value is `t`.

`mm-external-terminal-program`
: The program used to start an external terminal.

`mm-enable-external`
: Indicate whether external MIME handlers should be used.

  If `t`, all defined external MIME handlers are used. If `nil`, files are saved to disk (`mailcap-save-binary-file`). If it is the symbol `ask`, you are prompted before the external MIME handler is invoked.

  When you launch an attachment through mailcap (see Section 4.12 [mailcap], page 27) an attempt is made to use a safe viewer with the safest options—this isn't the case if you save it to disk and launch it in a different way (command line or double-clicking). Anyhow, if you want to be sure not to launch any external programs, set this variable to `nil` or `ask`.

1.6 Files and Directories

`mm-default-directory`
: The default directory for saving attachments. If `nil` use `default-directory`.

`mm-tmp-directory`
: Directory for storing temporary files.

`mm-file-name-rewrite-functions`
: A list of functions used for rewriting file names of MIME parts. Each function is applied successively to the file name. Ready-made functions include

  `mm-file-name-delete-control`
  : Delete all control characters.

  `mm-file-name-delete-gotchas`
  : Delete characters that could have unintended consequences when used with flawed shell scripts, i.e. '`|`', '`>`' and '`<`'; and '`-`', '`.`' as the first character.

  `mm-file-name-delete-whitespace`
  : Remove all whitespace.

  `mm-file-name-trim-whitespace`
  : Remove leading and trailing whitespace.

  `mm-file-name-collapse-whitespace`
  : Collapse multiple whitespace characters.

  `mm-file-name-replace-whitespace`
  : Replace whitespace with underscores. Set the variable `mm-file-name-replace-whitespace` to any other string if you do not like underscores.

The standard Emacs functions `capitalize`, `downcase`, `upcase` and `upcase-initials` might also prove useful.

`mm-path-name-rewrite-functions`
: List of functions used for rewriting the full file names of MIME parts. This is used when viewing parts externally, and is meant for transforming the absolute name so that non-compliant programs can find the file where it's saved.

1.7 New Viewers

Here's an example viewer for displaying `text/enriched` inline:

```
(defun mm-display-enriched-inline (handle)
  (let (text)
    (with-temp-buffer
      (mm-insert-part handle)
      (save-window-excursion
        (enriched-decode (point-min) (point-max))
        (setq text (buffer-string))))
    (mm-insert-inline handle text)))
```

We see that the function takes a MIME handle as its parameter. It then goes to a temporary buffer, inserts the text of the part, does some work on the text, stores the result, goes back to the buffer it was called from and inserts the result.

The two important helper functions here are `mm-insert-part` and `mm-insert-inline`. The first function inserts the text of the handle in the current buffer. It handles charset and/or content transfer decoding. The second function just inserts whatever text you tell it to insert, but it also sets things up so that the text can be "undisplayed" in a convenient manner.

2 Composing

Creating a MIME message is boring and non-trivial. Therefore, a library called `mml` has been defined that parses a language called MML (MIME Meta Language) and generates MIME messages.

The main interface function is `mml-generate-mime`. It will examine the contents of the current (narrowed-to) buffer and return a string containing the MIME message.

2.1 Simple MML Example

Here's a simple '`multipart/alternative`':

```
<#multipart type=alternative>
This is a plain text part.
<#part type=text/enriched>
<center>This is a centered enriched part</center>
<#/multipart>
```

After running this through `mml-generate-mime`, we get this:

```
Content-Type: multipart/alternative; boundary="=-=-="


--=-=-=


This is a plain text part.

--=-=-=
Content-Type: text/enriched


<center>This is a centered enriched part</center>

--=-=-=--
```

2.2 MML Definition

The MML language is very simple. It looks a bit like an SGML application, but it's not.

The main concept of MML is the *part*. Each part can be of a different type or use a different charset. The way to delineate a part is with a '`<#part ...>`' tag. Multipart parts can be introduced with the '`<#multipart ...>`' tag. Parts are ended by the '`<#/part>`' or '`<#/multipart>`' tags. Parts started with the '`<#part ...>`' tags are also closed by the next open tag.

There's also the '`<#external ...>`' tag. These introduce '`external/message-body`' parts.

Each tag can contain zero or more parameters on the form '`parameter=value`'. The values may be enclosed in quotation marks, but that's not necessary unless the value contains white space. So '`filename=/home/user/#hello$^yes`' is perfectly valid.

The following parameters have meaning in MML; parameters that have no meaning are ignored. The MML parameter names are the same as the MIME parameter names; the things in the parentheses say which header it will be used in.

‘`type`’ The MIME type of the part (`Content-Type`).

‘`filename`’
Use the contents of the file in the body of the part (`Content-Disposition`).

‘`charset`’ The contents of the body of the part are to be encoded in the character set specified (`Content-Type`). See Section 2.5 [Charset Translation], page 14.

‘`name`’ Might be used to suggest a file name if the part is to be saved to a file (`Content-Type`).

‘`disposition`’
Valid values are ‘`inline`’ and ‘`attachment`’ (`Content-Disposition`).

‘`encoding`’
Valid values are ‘`7bit`’, ‘`8bit`’, ‘`quoted-printable`’ and ‘`base64`’ (`Content-Transfer-Encoding`). See Section 2.5 [Charset Translation], page 14.

‘`description`’
A description of the part (`Content-Description`).

‘`creation-date`’
RFC822 date when the part was created (`Content-Disposition`).

‘`modification-date`’
RFC822 date when the part was modified (`Content-Disposition`).

‘`read-date`’
RFC822 date when the part was read (`Content-Disposition`).

‘`recipients`’
Who to encrypt/sign the part to. This field is used to override any auto-detection based on the To/CC headers.

‘`sender`’ Identity used to sign the part. This field is used to override the default key used.

‘`size`’ The size (in octets) of the part (`Content-Disposition`).

‘`sign`’ What technology to sign this MML part with (`smime`, `pgp` or `pgpmime`)

‘`encrypt`’ What technology to encrypt this MML part with (`smime`, `pgp` or `pgpmime`)

Parameters for ‘`text/plain`’:

‘`format`’ Formatting parameter for the text, valid values include ‘`fixed`’ (the default) and ‘`flowed`’. Normally you do not specify this manually, since it requires the textual body to be formatted in a special way described in RFC 2646. See Section 2.7 [Flowed text], page 15.

Parameters for ‘`application/octet-stream`’:

‘`type`’ Type of the part; informal—meant for human readers (`Content-Type`).

Parameters for ‘message/external-body’:

‘access-type’
: A word indicating the supported access mechanism by which the file may be obtained. Values include ‘ftp’, ‘anon-ftp’, ‘tftp’, ‘localfile’, and ‘mailserver’. (Content-Type.)

‘expiration’
: The RFC822 date after which the file may no longer be fetched. (Content-Type.)

‘size’
: The size (in octets) of the file. (Content-Type.)

‘permission’
: Valid values are ‘read’ and ‘read-write’ (Content-Type).

Parameters for ‘sign=smime’:

‘keyfile’
: File containing key and certificate for signer.

Parameters for ‘encrypt=smime’:

‘certfile’
: File containing certificate for recipient.

2.3 Advanced MML Example

Here’s a complex multipart message. It’s a ‘multipart/mixed’ that contains many parts, one of which is a ‘multipart/alternative’.

```
<#multipart type=mixed>
<#part type=image/jpeg filename=~/rms.jpg disposition=inline>
<#multipart type=alternative>
This is a plain text part.
<#part type=text/enriched name=enriched.txt>
<center>This is a centered enriched part</center>
<#/multipart>
This is a new plain text part.
<#part disposition=attachment>
This plain text part is an attachment.
<#/multipart>
```

And this is the resulting MIME message:

```
Content-Type: multipart/mixed; boundary="=-=-="


--=-=-=



--=-=-=
Content-Type: image/jpeg;
 filename="~/rms.jpg"
```

```
Content-Disposition: inline;
 filename="~/rms.jpg"
Content-Transfer-Encoding: base64
```


```
/9j/4AAQSkZJRgABAQAAAQABAAD/2wBDAAgGBgcGBQgHBwcJCQgKDBQNDAsLDBkSEw8UHRof
Hh0aHBwgJC4nICIsIxwcKDcpLDAxNDQ0Hyc5PTgyPC4zNDL/wAALCAAwADABAREA/8QAHwAA
AQUBAQEBAQEAAAAAAAAAAAECAwQFBgcICQoL/8QAtRAAAgEDAwIEAwUFBAQAAAF9AQIDAAQR
BRIhMUEGE1FhByJxFDKBkaEII0KxwRVS0fAkM2JyggkKFhcYGRolJicoKSo0NTY3ODk6Q0RF
RkdISUpTVFVWV1hZWmNkZWZnaGlqc3R1dnd4eXqDhIWGh4iJipKTlJWWl5iZmqKjpKWmp6ip
qrKztLW2t7i5usLDxMXGx8jJytLT1NXW19jZ2uHi4+Tl5ufo6erx8vP09fb3+Pn6/9oACAEB
AAA/AO/rifFHjldNuGsrDa0qcSSHkA+gHrXKw+LtWLrMb+RgTyhbr+HSug07xNqV9fQtZrNI
AyiaE/NuBPOOOP0rvRNE880KOC8TbXXGCv1FPqjrF4LDR7u5L7SkTFT/ALWOP1xXgTuXfc7E
sx6nua6rwp4IvvEM8chCxWx0dzn7wz6V9AaB4SO7w9p5itow0rDLSY5Pt9K43x066P4xs71m
2QXiGCbA4yOVJ9+1aYORkdK434lyNH4ahCnG66VT9Nj15JFbPdX0MS43M4VQf5/yr2vSpLnw
5ZW8dlCZ8KFXjOPX0/mK6rSPEGt3Angu44fNEReHYNvIH3TzXDeKNO8RX+kSX2ouZkicTIOc
L+g7E81OulFjpVtv3bwgB3HJyK5L4quY/C9sVxk3ij/xx6850u7t1mtp/wDlpEw3An3Jr3Dw
34gsbWza4nBlhC5LDsaW6+IFgupQyCF3iHH7gA7c9R9ay7zx6t7aX9jHC4smhfBkGCvHGfrm
tLQ7hbnRrV1GPkAP1x1/Hr+Ncr8Vzjwrb/Ak6v/AKA9eQRyYlQk8Yx9K6XTNbkgia2ciSIn
7p5Ga9AtteOLTLKO6it4i7dVRFJDcZ4PvXN+JvEMF9bILVGXJLSZ4zkjivRPDaeX4b08HOTC
pOffmua+KkbS+GLVUGT9tT/0B68eeIpIFYjB70+OOVXyoOM9+M1eaWeCLzHPyGO/NVWvJJm
jQ8KGH1NfQWhXSXh2c8eArRLwO3HSv/2Q==
```


```
--=-=-=
Content-Type: multipart/alternative; boundary="==-=-="


--==-=-=


This is a plain text part.

--==-=-=
Content-Type: text/enriched;
 name="enriched.txt"


<center>This is a centered enriched part</center>

--==-=-=--

--=-=-=

This is a new plain text part.

--=-=-=
Content-Disposition: attachment
```

```
This plain text part is an attachment.

--=-=-=--
```

2.4 Encoding Customization

`mm-body-charset-encoding-alist`

Mapping from MIME charset to encoding to use. This variable is usually used except, e.g., when other requirements force a specific encoding (digitally signed messages require 7bit encodings). The default is

```
((iso-2022-jp . 7bit)
 (iso-2022-jp-2 . 7bit)
 (utf-16 . base64)
 (utf-16be . base64)
 (utf-16le . base64))
```

As an example, if you do not want to have ISO-8859-1 characters quoted-printable encoded, you may add `(iso-8859-1 . 8bit)` to this variable. You can override this setting on a per-message basis by using the `encoding` MML tag (see Section 2.2 [MML Definition], page 9).

`mm-coding-system-priorities`

Prioritize coding systems to use for outgoing messages. The default is `nil`, which means to use the defaults in Emacs, but is `(iso-8859-1 iso-2022-jp iso-2022-jp-2 shift_jis utf-8)` when running Emacs in the Japanese language environment. It is a list of coding system symbols (aliases of coding systems are also allowed, use *`M-x describe-coding-system`* to make sure you are specifying correct coding system names). For example, if you have configured Emacs to prefer UTF-8, but wish that outgoing messages should be sent in ISO-8859-1 if possible, you can set this variable to `(iso-8859-1)`. You can override this setting on a per-message basis by using the `charset` MML tag (see Section 2.2 [MML Definition], page 9).

As different hierarchies prefer different charsets, you may want to set `mm-coding-system-priorities` according to the hierarchy in Gnus. Here's an example:

```
(add-to-list 'gnus-newsgroup-variables 'mm-coding-system-priorities)
(setq gnus-parameters
      (nconc
       ;; Some charsets are just examples!
       '(("^cn\\." ;; Chinese
          (mm-coding-system-priorities
           '(iso-8859-1 cn-big5 chinese-iso-7bit utf-8)))
         ("^cz\\.\\|^pl\\." ;; Central and Eastern European
          (mm-coding-system-priorities '(iso-8859-2 utf-8)))
         ("^de\\." ;; German language
          (mm-coding-system-priorities '(iso-8859-1 iso-8859-15 utf-8)))
         ("^fr\\." ;; French
```

```
                        (mm-coding-system-priorities '(iso-8859-15 iso-8859-1 utf-8)))
                       ("^fj\\." ;; Japanese
                        (mm-coding-system-priorities
                         '(iso-8859-1 iso-2022-jp iso-2022-jp-2 shift_jis utf-8)))
                       ("^ru\\." ;; Cyrillic
                        (mm-coding-system-priorities
                         '(koi8-r iso-8859-5 iso-8859-1 utf-8))))
                     gnus-parameters))
```

`mm-content-transfer-encoding-defaults`

Mapping from MIME types to encoding to use. This variable is usually used except, e.g., when other requirements force a safer encoding (digitally signed messages require 7bit encoding). Besides the normal MIME encodings, `qp-or-base64` may be used to indicate that for each case the most efficient of quoted-printable and base64 should be used.

`qp-or-base64` has another effect. It will fold long lines so that MIME parts may not be broken by MTA. So do `quoted-printable` and `base64`.

Note that it affects body encoding only when a part is a raw forwarded message (which will be made by `gnus-summary-mail-forward` with the arg 2 for example) or is neither the '`text/*`' type nor the '`message/*`' type. Even though in those cases, you can override this setting on a per-message basis by using the `encoding` MML tag (see Section 2.2 [MML Definition], page 9).

`mm-use-ultra-safe-encoding`

When this is non-`nil`, it means that textual parts are encoded as quoted-printable if they contain lines longer than 76 characters or starting with "From " in the body. Non-7bit encodings (8bit, binary) are generally disallowed. This reduce the probability that a non-8bit clean MTA or MDA changes the message. This should never be set directly, but bound by other functions when necessary (e.g., when encoding messages that are to be digitally signed).

2.5 Charset Translation

During translation from MML to MIME, for each MIME part which has been composed inside Emacs, an appropriate charset has to be chosen.

If you are running a non-MULE Emacs, this process is simple: If the part contains any non-ASCII (8-bit) characters, the MIME charset given by `mail-parse-charset` (a symbol) is used. (Never set this variable directly, though. If you want to change the default charset, please consult the documentation of the package which you use to process MIME messages. See section "Various Message Variables" in *Message Manual*, for example.) If there are only ASCII characters, the MIME charset US-ASCII is used, of course.

Things are slightly more complicated when running Emacs with MULE support. In this case, a list of the MULE charsets used in the part is obtained, and the MULE charsets are translated to MIME charsets by consulting the table provided by Emacs itself or the variable `mm-mime-mule-charset-alist` for XEmacs. If this results in a single MIME charset, this is used to encode the part. But if the resulting list of MIME charsets contains more than one element, two things can happen: If it is possible to encode the part via UTF-8, this charset is used. (For this, Emacs must support the `utf-8` coding system, and the part must

consist entirely of characters which have Unicode counterparts.) If UTF-8 is not available for some reason, the part is split into several ones, so that each one can be encoded with a single MIME charset. The part can only be split at line boundaries, though—if more than one MIME charset is required to encode a single line, it is not possible to encode the part.

When running Emacs with MULE support, the preferences for which coding system to use is inherited from Emacs itself. This means that if Emacs is set up to prefer UTF-8, it will be used when encoding messages. You can modify this by altering the `mm-coding-system-priorities` variable though (see Section 2.4 [Encoding Customization], page 13).

The charset to be used can be overridden by setting the `charset` MML tag (see Section 2.2 [MML Definition], page 9) when composing the message.

The encoding of characters (quoted-printable, 8bit etc) is orthogonal to the discussion here, and is controlled by the variables `mm-body-charset-encoding-alist` and `mm-content-transfer-encoding-defaults` (see Section 2.4 [Encoding Customization], page 13).

2.6 Conversion

A (multipart) MIME message can be converted to MML with the `mime-to-mml` function. It works on the message in the current buffer, and substitutes MML markup for MIME boundaries. Non-textual parts do not have their contents in the buffer, but instead have the contents in separate buffers that are referred to from the MML tags.

An MML message can be converted back to MIME by the `mml-to-mime` function.

These functions are in certain senses "lossy"—you will not get back an identical message if you run `mime-to-mml` and then `mml-to-mime`. Not only will trivial things like the order of the headers differ, but the contents of the headers may also be different. For instance, the original message may use base64 encoding on text, while `mml-to-mime` may decide to use quoted-printable encoding, and so on.

In essence, however, these two functions should be the inverse of each other. The resulting contents of the message should remain equivalent, if not identical.

2.7 Flowed text

The Emacs MIME library will respect the `use-hard-newlines` variable (see section "Hard and Soft Newlines" in *Emacs Manual*) when encoding a message, and the "format=flowed" Content-Type parameter when decoding a message.

On encoding text, regardless of `use-hard-newlines`, lines terminated by soft newline characters are filled together and wrapped after the column decided by `fill-flowed-encode-column`. Quotation marks (matching '`^>* ?`') are respected. The variable controls how the text will look in a client that does not support flowed text, the default is to wrap after 66 characters. If hard newline characters are not present in the buffer, no flow encoding occurs.

On decoding flowed text, lines with soft newline characters are filled together and wrapped after the column decided by `fill-flowed-display-column`. The default is to wrap after `fill-column`.

`mm-fill-flowed`
: If non-`nil` a format=flowed article will be displayed flowed.

3 Interface Functions

The `mail-parse` library is an abstraction over the actual low-level libraries that are described in the next chapter.

Standards change, and so programs have to change to fit in the new mold. For instance, RFC2045 describes a syntax for the `Content-Type` header that only allows ASCII characters in the parameter list. RFC2231 expands on RFC2045 syntax to provide a scheme for continuation headers and non-ASCII characters.

The traditional way to deal with this is just to update the library functions to parse the new syntax. However, this is sometimes the wrong thing to do. In some instances it may be vital to be able to understand both the old syntax as well as the new syntax, and if there is only one library, one must choose between the old version of the library and the new version of the library.

The Emacs MIME library takes a different tack. It defines a series of low-level libraries (‘`rfc2047.el`’, ‘`rfc2231.el`’ and so on) that parses strictly according to the corresponding standard. However, normal programs would not use the functions provided by these libraries directly, but instead use the functions provided by the `mail-parse` library. The functions in this library are just aliases to the corresponding functions in the latest low-level libraries. Using this scheme, programs get a consistent interface they can use, and library developers are free to create write code that handles new standards.

The following functions are defined by this library:

`mail-header-parse-content-type`
: Parse a `Content-Type` header and return a list on the following format:

```
("type/subtype"
 (attribute1 . value1)
 (attribute2 . value2)
 ...)
```

Here’s an example:

```
(mail-header-parse-content-type
 "image/gif; name=\"b980912.gif\"")
⇒ ("image/gif" (name . "b980912.gif"))
```

`mail-header-parse-content-disposition`
: Parse a `Content-Disposition` header and return a list on the same format as the function above.

`mail-content-type-get`
: Takes two parameters—a list on the format above, and an attribute. Returns the value of the attribute.

```
(mail-content-type-get
 '("image/gif" (name . "b980912.gif")) 'name)
⇒ "b980912.gif"
```

`mail-header-encode-parameter`
: Takes a parameter string and returns an encoded version of the string. This is used for parameters in headers like `Content-Type` and `Content-Disposition`.

`mail-header-remove-comments`
: Return a comment-free version of a header.

```
(mail-header-remove-comments
 "Gnus/5.070027 (Pterodactyl Gnus v0.27) (Finnish Landrace)")
⇒ "Gnus/5.070027  "
```

`mail-header-remove-whitespace`
: Remove linear white space from a header. Space inside quoted strings and comments is preserved.

```
(mail-header-remove-whitespace
 "image/gif; name=\"Name with spaces\"")
⇒ "image/gif;name=\"Name with spaces\""
```

`mail-header-get-comment`
: Return the last comment in a header.

```
(mail-header-get-comment
 "Gnus/5.070027 (Pterodactyl Gnus v0.27) (Finnish Landrace)")
⇒ "Finnish Landrace"
```

`mail-header-parse-address`
: Parse an address and return a list containing the mailbox and the plaintext name.

```
(mail-header-parse-address
 "Hrvoje Niksic <hniksic@srce.hr>")
⇒ ("hniksic@srce.hr" . "Hrvoje Niksic")
```

`mail-header-parse-addresses`
: Parse a string with list of addresses and return a list of elements like the one described above.

```
(mail-header-parse-addresses
 "Hrvoje Niksic <hniksic@srce.hr>, Steinar Bang <sb@metis.no>")
⇒ (("hniksic@srce.hr" . "Hrvoje Niksic")
     ("sb@metis.no" . "Steinar Bang"))
```

`mail-header-parse-date`
: Parse a date string and return an Emacs time structure.

`mail-narrow-to-head`
: Narrow the buffer to the header section of the buffer. Point is placed at the beginning of the narrowed buffer.

`mail-header-narrow-to-field`
: Narrow the buffer to the header under point. Understands continuation headers.

`mail-header-fold-field`
: Fold the header under point.

`mail-header-unfold-field`
: Unfold the header under point.

`mail-header-field-value`
: Return the value of the field under point.

`mail-encode-encoded-word-region`
: Encode the non-ASCII words in the region. For instance, '`Naïve`' is encoded as '`=?iso-8859-1?q?Na=EFve?=`'.

`mail-encode-encoded-word-buffer`
: Encode the non-ASCII words in the current buffer. This function is meant to be called narrowed to the headers of a message.

`mail-encode-encoded-word-string`
: Encode the words that need encoding in a string, and return the result.

```
(mail-encode-encoded-word-string
 "This is naïve, baby")
⇒ "This is =?iso-8859-1?q?na=EFve,?= baby"
```

`mail-decode-encoded-word-region`
: Decode the encoded words in the region.

`mail-decode-encoded-word-string`
: Decode the encoded words in the string and return the result.

```
(mail-decode-encoded-word-string
 "This is =?iso-8859-1?q?na=EFve,?= baby")
⇒ "This is naïve, baby"
```

Currently, `mail-parse` is an abstraction over `ietf-drums`, `rfc2047`, `rfc2045` and `rfc2231`. These are documented in the subsequent sections.

4 Basic Functions

This chapter describes the basic, ground-level functions for parsing and handling. Covered here is parsing `From` lines, removing comments from header lines, decoding encoded words, parsing date headers and so on. High-level functionality is dealt with in the first chapter (see Chapter 1 [Decoding and Viewing], page 3).

4.1 rfc2045

RFC2045 is the "main" MIME document, and as such, one would imagine that there would be a lot to implement. But there isn't, since most of the implementation details are delegated to the subsequent RFCs.

So '`rfc2045.el`' has only a single function:

`rfc2045-encode-string`
: Takes a parameter and a value and returns a '`PARAM=VALUE`' string. *value* will be quoted if there are non-safe characters in it.

4.2 rfc2231

RFC2231 defines a syntax for the `Content-Type` and `Content-Disposition` headers. Its snappy name is *MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations*.

In short, these headers look something like this:

```
Content-Type: application/x-stuff;
 title*0*=us-ascii'en'This%20is%20even%20more%20;
 title*1*=%2A%2A%2Afun%2A%2A%2A%20;
 title*2="isn't it!"
```

They usually aren't this bad, though.

The following functions are defined by this library:

`rfc2231-parse-string`
: Parse a `Content-Type` header and return a list describing its elements.

```
(rfc2231-parse-string
 "application/x-stuff;
 title*0*=us-ascii'en'This%20is%20even%20more%20;
 title*1*=%2A%2A%2Afun%2A%2A%2A%20;
 title*2=\"isn't it!\"")
⇒ ("application/x-stuff"
    (title . "This is even more ***fun*** isn't it!"))
```

`rfc2231-get-value`
: Takes one of the lists on the format above and returns the value of the specified attribute.

`rfc2231-encode-string`
: Encode a parameter in headers likes `Content-Type` and `Content-Disposition`.

4.3 ietf-drums

drums is an IETF working group that is working on the replacement for RFC822.

The functions provided by this library include:

`ietf-drums-remove-comments`
: Remove the comments from the argument and return the results.

`ietf-drums-remove-whitespace`
: Remove linear white space from the string and return the results. Spaces inside quoted strings and comments are left untouched.

`ietf-drums-get-comment`
: Return the last most comment from the string.

`ietf-drums-parse-address`
: Parse an address string and return a list that contains the mailbox and the plain text name.

`ietf-drums-parse-addresses`
: Parse a string that contains any number of comma-separated addresses and return a list that contains mailbox/plain text pairs.

`ietf-drums-parse-date`
: Parse a date string and return an Emacs time structure.

`ietf-drums-narrow-to-header`
: Narrow the buffer to the header section of the current buffer.

4.4 rfc2047

RFC2047 (Message Header Extensions for Non-ASCII Text) specifies how non-ASCII text in headers are to be encoded. This is actually rather complicated, so a number of variables are necessary to tweak what this library does.

The following variables are tweakable:

`rfc2047-header-encoding-alist`
: This is an alist of header / encoding-type pairs. Its main purpose is to prevent encoding of certain headers.

  The keys can either be header regexps, or `t`.

  The values can be `nil`, in which case the header(s) in question won't be encoded, `mime`, which means that they will be encoded, or `address-mime`, which means the header(s) will be encoded carefully assuming they contain addresses.

`rfc2047-charset-encoding-alist`
: RFC2047 specifies two forms of encoding—`Q` (a Quoted-Printable-like encoding) and `B` (base64). This alist specifies which charset should use which encoding.

`rfc2047-encode-function-alist`
: This is an alist of encoding / function pairs. The encodings are `Q`, `B` and `nil`.

`rfc2047-encoded-word-regexp`
: When decoding words, this library looks for matches to this regexp.

`rfc2047-encoded-word-regexp-loose`
: This is a version from which the regexp for the Q encoding pattern of `rfc2047-encoded-word-regexp` is made loose.

`rfc2047-encode-encoded-words`
: The boolean variable specifies whether encoded words (e.g. '`=?us-ascii?q?hello?=`') should be encoded again. `rfc2047-encoded-word-regexp` is used to look for such words.

`rfc2047-allow-irregular-q-encoded-words`
: The boolean variable specifies whether irregular Q encoded words (e.g. '`=?us-ascii?q?hello??=`') should be decoded. If it is non-`nil`, `rfc2047-encoded-word-regexp-loose` is used instead of `rfc2047-encoded-word-regexp` to look for encoded words.

Those were the variables, and these are this functions:

`rfc2047-narrow-to-field`
: Narrow the buffer to the header on the current line.

`rfc2047-encode-message-header`
: Should be called narrowed to the header of a message. Encodes according to `rfc2047-header-encoding-alist`.

`rfc2047-encode-region`
: Encodes all encodable words in the region specified.

`rfc2047-encode-string`
: Encode a string and return the results.

`rfc2047-decode-region`
: Decode the encoded words in the region.

`rfc2047-decode-string`
: Decode a string and return the results.

`rfc2047-encode-parameter`
: Encode a parameter in the RFC2047-like style. This is a replacement for the `rfc2231-encode-string` function. See Section 4.2 [rfc2231], page 21.

  When attaching files as MIME parts, we should use the RFC2231 encoding to specify the file names containing non-ASCII characters. However, many mail softwares don't support it in practice and recipients won't be able to extract files with correct names. Instead, the RFC2047-like encoding is acceptable generally. This function provides the very RFC2047-like encoding, resigning to such a regrettable trend. To use it, put the following line in your '`~/.gnus.el`' file:

```
(defalias 'mail-header-encode-parameter 'rfc2047-encode-parameter)
```

4.5 time-date

While not really a part of the MIME library, it is convenient to document this library here. It deals with parsing `Date` headers and manipulating time. (Not by using tesseracts, though, I'm sorry to say.)

These functions convert between five formats: A date string, an Emacs time structure, a decoded time list, a second number, and a day number.

Here's a bunch of time/date/second/day examples:

```
(parse-time-string "Sat Sep 12 12:21:54 1998 +0200")
⇒ (54 21 12 12 9 1998 6 nil 7200)

(date-to-time "Sat Sep 12 12:21:54 1998 +0200")
⇒ (13818 19266)

(time-to-seconds '(13818 19266))
⇒ 905595714.0

(seconds-to-time 905595714.0)
⇒ (13818 19266 0)

(time-to-days '(13818 19266))
⇒ 729644

(days-to-time 729644)
⇒ (961933 65536)

(time-since '(13818 19266))
⇒ (0 430)

(time-less-p '(13818 19266) '(13818 19145))
⇒ nil

(subtract-time '(13818 19266) '(13818 19145))
⇒ (0 121)

(days-between "Sat Sep 12 12:21:54 1998 +0200"
              "Sat Sep 07 12:21:54 1998 +0200")
⇒ 5

(date-leap-year-p 2000)
⇒ t

(time-to-day-in-year '(13818 19266))
⇒ 255

(time-to-number-of-days
 (time-since
  (date-to-time "Mon, 01 Jan 2001 02:22:26 GMT")))
⇒ 4.146122685185185
```

And finally, we have `safe-date-to-time`, which does the same as `date-to-time`, but returns a zero time if the date is syntactically malformed.

The five data representations used are the following:

date An RFC822 (or similar) date string. For instance: `"Sat Sep 12 12:21:54 1998 +0200"`.

time An internal Emacs time. For instance: `(13818 26466)`.

seconds A floating point representation of the internal Emacs time. For instance: `905595714.0`.

days An integer number representing the number of days since 00000101. For instance: `729644`.

decoded time
A list of decoded time. For instance: `(54 21 12 12 9 1998 6 t 7200)`.

All the examples above represent the same moment.

These are the functions available:

`date-to-time`
Take a date and return a time.

`time-to-seconds`
Take a time and return seconds. Note that Emacs has a built-in function, `float-time`, that does this.

`seconds-to-time`
Take seconds and return a time.

`time-to-days`
Take a time and return days.

`days-to-time`
Take days and return a time.

`date-to-day`
Take a date and return days.

`time-to-number-of-days`
Take a time and return the number of days that represents.

`safe-date-to-time`
Take a date and return a time. If the date is not syntactically valid, return a "zero" time.

`time-less-p`
Take two times and say whether the first time is less (i. e., earlier) than the second time.

`time-since`
Take a time and return a time saying how long it was since that time.

`subtract-time`
Take two times and subtract the second from the first. I. e., return the time between the two times.

`days-between`
: Take two days and return the number of days between those two days.

`date-leap-year-p`
: Take a year number and say whether it's a leap year.

`time-to-day-in-year`
: Take a time and return the day number within the year that the time is in.

4.6 qp

This library deals with decoding and encoding Quoted-Printable text.

Very briefly explained, qp encoding means translating all 8-bit characters (and lots of control characters) into things that look like '`=EF`'; that is, an equal sign followed by the byte encoded as a hex string.

The following functions are defined by the library:

`quoted-printable-decode-region`
: QP-decode all the encoded text in the specified region.

`quoted-printable-decode-string`
: Decode the QP-encoded text in a string and return the results.

`quoted-printable-encode-region`
: QP-encode all the encodable characters in the specified region. The third optional parameter *fold* specifies whether to fold long lines. (Long here means 72.)

`quoted-printable-encode-string`
: QP-encode all the encodable characters in a string and return the results.

4.7 base64

Base64 is an encoding that encodes three bytes into four characters, thereby increasing the size by about 33%. The alphabet used for encoding is very resistant to mangling during transit.

The following functions are defined by this library:

`base64-encode-region`
: base64 encode the selected region. Return the length of the encoded text. Optional third argument *no-line-break* means do not break long lines into shorter lines.

`base64-encode-string`
: base64 encode a string and return the result.

`base64-decode-region`
: base64 decode the selected region. Return the length of the decoded text. If the region can't be decoded, return `nil` and don't modify the buffer.

`base64-decode-string`
: base64 decode a string and return the result. If the string can't be decoded, `nil` is returned.

4.8 binhex

`binhex` is an encoding that originated in Macintosh environments. The following function is supplied to deal with these:

`binhex-decode-region`
: Decode the encoded text in the region. If given a third parameter, only decode the `binhex` header and return the filename.

4.9 uudecode

`uuencode` is probably still the most popular encoding of binaries used on Usenet, although `base64` rules the mail world.

The following function is supplied by this package:

`uudecode-decode-region`
: Decode the text in the region.

4.10 yenc

`yenc` is used for encoding binaries on Usenet. The following function is supplied by this package:

`yenc-decode-region`
: Decode the encoded text in the region.

4.11 rfc1843

RFC1843 deals with mixing Chinese and ASCII characters in messages. In essence, RFC1843 switches between ASCII and Chinese by doing this:

```
This sentence is in ASCII.
The next sentence is in GB.~{<:Ky2;S{#,NpJ)l6HK!#~}Bye.
```

Simple enough, and widely used in China.

The following functions are available to handle this encoding:

`rfc1843-decode-region`
: Decode HZ-encoded text in the region.

`rfc1843-decode-string`
: Decode a HZ-encoded string and return the result.

4.12 mailcap

The '`~/.mailcap`' file is parsed by most MIME-aware message handlers and describes how elements are supposed to be displayed. Here's an example file:

```
image/*; gimp -8 %s
audio/wav; wavplayer %s
application/msword; catdoc %s ; copiousoutput ; nametemplate=%s.doc
```

This says that all image files should be displayed with `gimp`, that WAVE audio files should be played by `wavplayer`, and that MS-WORD files should be inlined by `catdoc`.

The `mailcap` library parses this file, and provides functions for matching types.

`mailcap-mime-data`
: This variable is an alist of alists containing backup viewing rules.

Interface functions:

`mailcap-parse-mailcaps`
: Parse the '`~/.mailcap`' file.

`mailcap-mime-info`
: Takes a MIME type as its argument and returns the matching viewer.

5 Standards

The Emacs MIME library implements handling of various elements according to a (somewhat) large number of RFCs, drafts and standards documents. This chapter lists the relevant ones. They can all be fetched from http://quimby.gnus.org/notes/.

RFC822
STD11 Standard for the Format of ARPA Internet Text Messages.

RFC1036 Standard for Interchange of USENET Messages

RFC2045 Format of Internet Message Bodies

RFC2046 Media Types

RFC2047 Message Header Extensions for Non-ASCII Text

RFC2048 Registration Procedures

RFC2049 Conformance Criteria and Examples

RFC2231 MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations

RFC1843 HZ - A Data Format for Exchanging Files of Arbitrarily Mixed Chinese and ASCII characters

draft-ietf-drums-msg-fmt-05.txt
Draft for the successor of RFC822

RFC2112 The MIME Multipart/Related Content-type

RFC1892 The Multipart/Report Content Type for the Reporting of Mail System Administrative Messages

RFC2183 Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field

RFC2646 Documentation of the text/plain format parameter for flowed text.

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 non-commercially. 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

P

Q

R

S

T

U

V

Y