Messaging in the Emacs World

Mew

1 Read me first

Mew is an interface to integrate

- Email
- NetNews
- MIME(Multipurpose Internet Mail Extensions)
- PGP(Pretty Good Privacy)

and to make it easy to view and compose them. With Mew you can send a picture of a birthday cake with the song "Happy Birthday to you" to your friend, which is encrypted by strong cryptograph. NetNews is supposed to be integrated in version 2.xx or later.

Mew is an acronym of "Messaging in the Emacs World". You should spell it with the first letter capitalized and pronounce it as it is(i.e. meow of cats). When the author started programming it, he chose a cute word from his English dictionary. So, Mew.

The features of Mew version 1.9x are as follows:

- You can easily display a very complicated structured message. What you should do to view messages is just type '`SPC`'.
- If you know file operations such as copy, you can compose a very complicated message without any troubles.
- You can start to read messages before the termination of listing of messages.
- Since Mew preserves lists of messages in Summary mode, you can list up the gap between the last and current incrementally when you move into.
- Mew neatly guesses a default folder for refiling(Those who receives many messages cannot live without this feature).
- In Draft mode, you can complete field names, Email addresses, receiver's names, domain names, and folder names.
- You can easily pick up messages which you want by specifying conditions such as Subject: and Date:.
- Useful marks are provided. You can handle "encoded with uuencode then split" messages with one operation.
- Mew automatically decodes a message encrypted with PGP. It also automatically verifies a signed message.
- You can easily encrypt or sign a message with PGP.
- It takes a time to analyze MIME syntax or to verify a PGP signature. While a user read a message, Mew processes the next message so as to display the next message faster. Analyzed messages are cached for a while.
- You can give a single view for multiple folders.
- If you use Mew on XEmacs, you can enjoy icon-based interface which is equivalent to key-based interface.

Please use Emacs 20.7 or later, or XEmacs 21.1.11 or later. Mew doesn't support earlier versions.

Mew may support beta versions of Emacs (e.g. Emacs 21) but Mew conforms the spec of official release when available.

Throughout this manual, "Emacs" means all supported platforms. "Mule" indicates multilingual platforms such as Mule 2, Emacs 20, and XEmacs complied with the –with-mule option while "Bilingual Emacs" means English-and-Latin1 platforms including Emacs 19, Emacs 20 executed with the –unibyte option, XEmacs complied without the –with-mule option. "XEmacs" indicates graphical platforms such as Emacs 21 and XEmacs whereas text-only platforms are called "Text Emacs".

2 Let's get started

Mew has the following six modes:

– Summary mode :: A mode to list up messages and to select one.
– Virtual mode :: A mode to list up messages selected with a specific condition from folders. It's similar to Summary mode.
– Message mode :: A mode to display a content of text.
– Draft mode :: A mode to prepare a message to be sent, answered, and forwarded.
– Header mode :: A mode to edit the header of a message and to send it. forwarded.
– Addrbook mode :: A mode to register an entry to Addrbook.

To start Mew, you can choose one from the followings:

1. '`M-x mew`' :: Execute Mew. If '`mew-auto-get`' is '`t`', messages stored in your mailbox are fetched to the +inbox folder and messages in the +inbox folder are listed up in Summary mode. If '`mew-auto-get`' is '`nil`', simply list up messages in the inbox folder.
2. '`C-uM-x mew`' :: Perform '`M-x mew`' thinking that '`mew-auto-get`' is reversed.
3. '`M-x mew-send`' :: Enter Draft mode for message composing.
4. '`C-xm`' :: Enter Draft mode for message composing if '`mail-user-agent`' is configured.

When Mew is executed on Text Emacs, a shape of "/\\ - \\/", which stands for Mew, spins. Two cute cats appear on the cover page on XEmacs.

When fetching messages stored in your mailbox, you may be required to input your password. Before you type your password, carefully see if either the following conditions is satisfied:

– Emacs is running on a local computer
– Though Emacs is running on a remote computer, an encryption mechanism is used for the communication.

If both conditions are not satisfied, don't input password. Otherwise, your password would be wire-tapped.

If Mew is not executed, see whether or not Mew is installed and/or whether or not the following configurations are put into a site configuration file or your ".emacs".

```
(autoload 'mew "mew" nil t)
(autoload 'mew-send "mew" nil t)
;; mew-mail-domain-list is automatically set if you configure
;; 'mew-config-alist'.
(setq mew-mail-domain-list '("your mail domain"))
(setq mew-pop-server "your pop server")    ;; if not localhost
(setq mew-smtp-server "your smtp server")  ;; if not localhost
(setq mew-icon-directory "icon directory") ;; if using XEmacs
;; See also mew-config-alist for advanced use
(autoload 'mew-user-agent-compose "mew" nil t)
(if (boundp 'mail-user-agent)
    (setq mail-user-agent 'mew-user-agent))
(if (fboundp 'define-mail-user-agent)
    (define-mail-user-agent
      'mew-user-agent
      'mew-user-agent-compose
      'mew-draft-send-message
      'mew-draft-kill
      'mew-send-hook))
```

The configuration above assumes that you are using POP to retrieve messages. If you want to use a local spool, you also need, for instance, the following configuration:

```
(setq mew-mailbox-type 'mbox)
;; MH's "inc", see contrib/00readme for more information.
(setq mew-mbox-command "inc")
(setq mew-mbox-command-arg "-truncate -file /var/mail/user")
```

3 Viewing messages

If you input '`M-x mew`', Mew moves messages from your mailbox to the +inbox folder and displays as follows:

```
1  07/17 Itojun        v6: items to be no in6_pcbnotify() doesn't
2  07/18 Utashiro      Re: behavior after I'm afraid that mark-ring
3  07/19 Nom-sun       refile info.       Sorry for my late respon
```

We call this Summary mode. This section mainly explains how to read messages in Summary mode.

3.1 Reading Basis

To read messages in arrival order, type '`SPC`' to display them. That's it. It's easy, isn't it?

Yet this might not be comprehensive, so we list up basic commands for page process as follows:

'`SPC`'	Read through messages. That is, display a message, scroll it, and move-then-display another message. Refer to See Section 9.1 [level-one], page 38 to know which direction the cursor moves.
'`DEL`'	Back-scroll this message. Unnecessary header fields are hidden over the window. Type '`DEL`' to see them when a message is displayed.
'`.`'	Remove the cache of this message or part and analyze the message, then display this message or part again. If the size of the current message exceeds '`mew-file-max-size`', MIME analysis is skipped then the beginning of the raw message is displayed. In this situation, this command analyzes the current message without the limitation then displays it. If the length of a header exceeds '`mew-header-max-length`', a broken message is displayed. In this situation, this command analyzes the current message without the limitation then displays it. If called with '`C-u`', analyze the message with '`mew-decode-broken`' reversed(see See Section 3.8 [illegal], page 9).
'`,`'	Display this message in the raw format(i.e. without MIME analysis). The beginning part of the message, whose size specified by '`mew-file-max-size`', is displayed. If called with '`C-u`', the entire message is displayed in the raw format.
'`RET`'	Make this message scroll up with one line.
'`M-RET`'	
'`-`'	Make this message scroll down with one line.
'`C-n`'	Go to the next line.
'`C-p`'	Go to the previous line.
'`n`'	Move to below then display. Targets includes parts, messages marked with '`*`', and non-marked messages. When called with '`C-u`', parts are skipped.
'`p`'	Move to above then display. Targets includes parts, messages marked with '`*`', and non-marked messages. When called with '`C-u`', parts are skipped.
'`j`'	Jump to a message according to the number which you input.

3.2 Displaying MIME

It's not difficult to read a multipart message. As usual, just type '`SPC`'.

A multipart message is marked with "M" on the left side of date as follows:

```
     4  07/19 Shigeya-san    Re: imget very fir OK, how about this?
     5 M07/20 Itojun         MagicPoint         I made the material of
     6  07/21 Motonori-san   Re: imget very fir Preserving messages on P
```

When you type '`SPC`' on "5", its header is displayed in Message mode then its multipart structure is displayed in Summary mode as follows:

```
     4  07/19 Shigeya-san    Re: imget very fir OK, how about this?
     5 M07/20 Itojun         MagicPoint         I made the material of
B     2  Image/Gif                    MagicPoint logo            mgp.gif
Q     3  Application/Postscript       Presentation Material      ohp.ps
     6  07/21 Motonori-san   Re: imget very fir Preserving messages on P
```

If the first part is Text/Plain, the first part is not visualized in Summary mode but the first part is displayed with its header in Message mode.

Each line of multipart consists of

- marks (Content-Transfer-Encoding:)
- part number
- data type (Content-Type:)
- description (Content-Description:)
- file name (Content-Disposition:).

Content-Description can be considered Subject: for each part. This format is very similar to that of attachment region in Draft mode. For more information of each column, please refer to See Section 4.6 [mime-comp], page 17.

If you type '`SPC`' or '`n`', the cursor moves onto part 1 and the content is displayed according to its data type. For instance, Text/Plain is showed in Message mode and PostScript is visualized with ghostview

Please note that '`n`' and '`p`' moves lines including multipart. To display the message below skipping multipart, type '`C-u n`'. Likewise, to display the message above (not this message), input '`C-u p`'.

Mew processes MIME recursively. The following example is a forwarded multipart message.

```
   501 M02/22 Itojun         Fw: MagicPoint           Itojun send me this
      2  Message/Rfc822               MagicPoint
B     2.2  Image/Gif                    MagicPoint logo          mgp.gif
Q     2.3  Application/Postscript       Presentation Material    ohp.ps
```

(Memo) I strongly discourage you to embed an object other than text as a top level single part to a message directly. Rather, I do recommend to make multipart whose part 1 is description text for part 2 and part 2 is an object other than text.

Mew displays a message which directory contains a single part other than text as a multipart.

Since the syntax of MIME messages can be complex, it sometime takes much time to analyze the syntax. However, Mew guesses the message to be read next and analyzes it beforehand while the user are reading the current message. Analyzed messages are cached for a while.

To notice the end of a message explicitly, Mew displays the string "[End of message]" in the end of the message. In the end of each part, Mew displays the string "[Message is continued]". These strings are customized by '`mew-end-of-message-string`' and '`mew-end-of-part-string`', respectively.

3.3 Visualizing PGP/MIME

Basic commands such as '`SPC`' visualize messages signed or encrypted with PGP as well. Let's start with a simple example:

```
8 S07/22 Sakai-san      Re: home was full  A bug of MsgStore.pm
9 E07/23 Neat Sumikawa  Wine               From good morning to
```

Massage 8 and 9 is marked with "S" and "E", respectively. This means that the *body* is signed and encrypted, respectively.

PGP/MIME also allows to sign and/or encrypt some parts of a message. In this case, these marks don't appear. It is a message whose *body* is singed or encrypted that the "S" mark or the "E" mark appears on.

"Sign" and "encrypt" used above means the last procedure applied the body is "sign" and "encrypted", respectively. Examples above may have been produced with more complex process. As far as the firmer message, for instance, the body may have been signed after encrypting the body. It is likely that some part of the latter message was first signed then the body was encrypted.

If some parts or the body is encrypted, Mew asks you to input your pass-phrase to get plain text. Please refer to See Chapter 2 [Start], page 2 to know what you should take care when inputing pass-phrase. The pass-phrase is used to decrypt your secret key. The secret key is then used to decrypt cipher text.

To visualize PGP/MIME, you need to input your pass-phrase every time you encounter cipher text. This is because Mew does not cache pass-phrases anywhere for security reasons. If you feel this is inconvenient, set the following configuration so that your pass-phrases are cached for a while. Please refer to See Section 3.4 [folder], page 7 to know what you should take care to use this feature.

```
(setq mew-use-cached-passwd t)
```

Since decrypted messages are cached for a while, you perhaps need not to input your pass-phrase at the next time when the message will be displayed, even if you do not use the pass-phrase cache.

To verify signatures, senders' public keys are used. So, you are not asked to type your pass-phrase.

Since Mew automatically verifies signatures and/or decrypts cipher text with inputed pass-phrase, it is likely that users don't notice that the original message has signatures and/or which parts were encrypted.

To tell users the results of verification of signatures or which parts were decrypted, Mew inserts the X-Mew: field in the header as follows:

```
X-Mew: <body> PGP decrypted.
        Good PGP sign "kazu@mew.org" COMPLETE
```

The number in "<>" indicates which part was protected with PGP. "body" means the body was protected. This example tells us that the body was singed by kazu then encrypted for the reader. Mew first decrypted it then verified its signature of the decrypted message. The signature is good. So, nobody has modified the content since it was signed by the secret key whose ID is kazu@mew.org. The validity of the public key used to verify the signature is "complete". Thus, the receiver believes that the public key actually belongs to the user whom the ID tells. That is, this message was verified by the trusted public key AND its results was good, so no alternation was found.

In the following example, the signature of the body, which is multipart, was first verified then part 2, which is a message, was decrypted. That is, the composing process was that part 2 was first encrypted then the entire body was signed.

```
X-Mew: <body multi> Good PGP sign "kazu@mew.org" COMPLETE
X-Mew: <2 message> PGP decrypted.
```

Smart users may wonder what if a bad guy or gal sends you a message with an illegal X-Mew: field. Take it easy. First Mew carefully removes the X-Mew: field then inserted a valid X-Mew: field to the header.

X-Mew: tells you many other types of result. For example, no public key is available, the decryption failed, etc. The following example indicates that the public key whose key ID is 0x1B8BF431 is missing.

```
X-Mew: <body multi> No his/her public key. ID = 0x1B8BF431
```

In this case, if you type '`C-cC-f`', Mew tries fetching this public key using URL specified in '`mew-pgp-keyserver-url-template`'. If the X-Mew: field does not exist, '`C-cC-f`' takes the From: field as ID. Also, '`C-uC-cC-f`' extracts key IDs from fields specified in '`mew-x-pgp-key-list`' in addition to the X-Mew: field then tries fetching them.

Mew supports PGPv2, PGPv5, and GNUPG. You can select one of those by '`C-cC-v`' in Summary mode. If you want to use those PGPs, you should set the command name of PGPv2, PGPv5, and GNUPG to '`mew-prog-pgp2`', '`mew-prog-pgp5`', and '`mew-prog-gpg`', respectively. Also, set the default PGP name to '`mew-prog-pgp`'. Note that pass-phrases are cached independently for each PGP.

3.4 Updating and visiting folder

To fetch arrived messages, store them to the +inbox folder and list up them, use '`i`'. The list is appended to the bottom of Summary mode for the +inbox folder.

At that time, you may be required to input your password. Please refer to See Chapter 2 [Start], page 2 to know what you should take care when inputing password. If you get sick of inputing your password time to time, please make use of the password-cache mechanism with following configuration.

```
(setq mew-use-cached-passwd t)
```

With this configuration, any passwords including POP and PGP are cached. While a password is cached, you can omit to input the password. The password will be expired after a certain period (20 minutes by default). However, its timer is cleared (reset to 20 minutes) if the password is internally used.

Password is stored in Emacs with RAW format. So, you have to take care so that other people don't touch your Emacs. If you leave your desk and an Emacs expert touches your Emacs, your passwords would be stolen.

To list up messages in Summary mode or flush it, use '`s`'. This command asks you to input range. Important ranges for Mew are as follows:

'`update`' From "the last message in the Summary mode + 1" to "the last message in the corresponding folder". That is, the gap between the Summary mode and the existing folder

'`all`' All messages in the folder. When Summary mode becomes inconsistent, use this range to flush the list.

The default range is usually '`update`'. So, just type '`RET`' after '`s`' to get the up-to-date list of the current folder.

You can specify the following ranges though they are not important for Mew.

'`<num1>-<num2>`'
From <num1> to <num2>

'`<num1>-`' From <num1> to the last

'`-<num2>`' From the first to <num2>

To go to another folder, type '`g`'. You can make use of folder name completion with '`TAB`'. When you move a folder and if Mew considered that its list of messages are old, Mew automatically displays the differences.

Here is a summary for commands up above.

`i`	Fetch arrived messages, store them to the +inbox folder and list up them
`s`	List up messages in Summary mode or flush it.
`g`	Go to another folder.

3.5 Write, answer, and forward

The following commands are prepared for writing, replying, and forwarding a message.

`w`	Write a message. A new draft is prepared in Draft mode.
`a`	Answer to this message. A new draft is prepared in Draft mode. Mew automatically decides To: and Cc:.
`A`	Answer to this message. A new draft is prepared in Draft mode. Mew automatically decides To: and Cc: and cites the body.
`f`	Forward this message to a third person. A new draft is prepared in Draft mode and this message is automatically attached.
`F`	Forward messages marked with '`@`' to a third person. A new draft is prepared in Draft mode and this message is automatically attached. For more information, refer to See Section 5.3 [multi mark], page 27.

If an error message returns, let's give one more try with the following commands.

`E`	Edit this message again to retry sending. Or edit this rfc822 part typically included MIME-encapsulated error message. In the +draft folder, it just edits the message. Otherwise, copy the message to the +draft folder, then edit.
`M-e`	Edit an old fashioned error message in which the original message is encapsulated after "—— Original message follows ——".

3.6 Useful features

Mew provides you with convenient commands in Summary mode as follows:

`v`	Toggle "Summary mode only" and "Summary & Message mode". If you choose "Summary mode only", you can quickly put the '`D`' mark since the next message is not displayed.
`M-l`	Make the current line to the center of Summary mode.
`C-cC-s`	Incremental search forward in Message mode.
`C-cC-r`	Incremental search backward in Message mode.
`y`	Save this message or this part into the file whose name is specified by you. If executed with '`C-u`' on Mule, you can specify coding-system for text.
`#`	Print this message or this part.
`|`	Send this message or this part via pipe.
`O`	Pack messages and list them up again.
`B`	De-capsulate embedded messages in MIME format.
`D`	Delete all messages in the +trash folder(See Section 5.1 [delete mark], page 26).
`Z`	Read Addrbook(See Section 4.3 [addrbook], page 14) and update its information. If you type '`C-u Z`', information of folders is also updated in addition to that of Addrbook. If '`mew-use-folders-file-p`' is '`t`', the list of folders is stored in "~/Mail/.folders". The default value is '`t`'.

'C-cC-a'	Register the information on the current message to Addrbook(See Section 4.3 [addrbook], page 14).
'C-cC-v'	Select PGP version(See Section 3.3 [pgp-viewing], page 6).
'C-cC-z'	Let PGP decrypt and/or verify good-old-PGP messages.

3.7 Sorting messages

To sort messages in the current folder, use '`S`'. Then you will be asked to input the field name as follows:

```
Sort by? (default date):
```

Strings stored in the specified field should not be compared as text in some cases. For example, while Subject: can be considered as text, Date: and X-Mail-Count: should be treated as date and number, respectively. In this way, we call how to treat stored strings "mode". Default modes for typical field names are configured in '`mew-sort-key-alist`'.

When you want to specify the mode of sorting, the mode is followed by '`:`'. For example, to sort with X-Mail-Count field as arithmetic value (not text), input like this:

```
x-mail-count:num
```

You can complete field names and modes with '`TAB`'.

Mew provides four modes for sorting:

'text'	String with preceding "Re: " and or "Fw: " removed.
'ml'	The same as text but preceding mailing-list-string removed.
'num'	Number.
'date'	Date.

The default field name, when you asked "Sort by?", can be specified to '`mew-sort-default-key`'. The following is an example to change the default value from "date" to "x-ml-count".

```
(setq mew-sort-default-key "x-ml-count")
```

You can also set a default field name for each folder by '`mew-sort-default-key-alist`'. For folders not explicitly specified here, '`mew-sort-default-key`' is used for their default field name. The following is an example that specifies "subject" for the +inbox folder and "x-mail-count" for the +mew-dist folder.

```
(setq mew-sort-default-key-alist
      '(("+inbox" . "subject")
        ("+mew-dist" . "x-mail-count")))
```

3.8 Illegal message

The following message contains Japanese text in its body. The charset parameter is not specified in the Content-Type: field. So, the body should be treated as US-ASCII.

```
To: piglet
Subject: illegal message
From: pooh
MIME-Version: 1.0
Content-Type: Text/Plain

Japanese comes here.
```

The following message is illegal as well.

```
From: "=?iso-2022-jp?B?GyRCOzNLXE9CSScbKEI=?=" <kazu@iijlab.net>
```

The string surrounded by "=?" and "?=" in the example above was originally Japanese. The spec of mail defines that only ASCII characters can be contained in a header. So, if a string, whose character set is other than ASCII, to be stored in a header, the string must be encoded into ASCII strings according to the defined rule. But it is certainly illegal to embed the ASCII strings with '"'. Strings surrounded by '"' is treated as is. Therefore, the string between "=?" and "?=" in the example should not be decoded into Japanese.

Several mailers are careless about the spec and made mistakes of this kind. The right way to do is ask the programmers of such mailers to make the programs conformant to the spec. However, since there are so many mailers of this kind around the world, Mew tries to decode messages as much as possible. Then, Mew displays warnings like this:

```
X-Mew: Charset for body is not specified.
To: has encoded-words in quoted text.
```

If you want to decode messages strictly, set '`mew-decode-broken`' to nil. This value can be toggled dynamically by '`.`'(see See Section 3.1 [singlepart], page 4).

4 Composing messages

This section shows you how to compose a message. With Mew, you can create only MIME messages(messages without MIME-Version: cannot be composed).

The followings are methods to enter Draft mode to write a new message.

1. Type '`M-x mew-send`'.
2. Type '`C-xm`' if '`mail-user-agent`' is configured.
3. Press '`w`' in Summary mode.

Then a buffer like the following is prepared.

```
To:
Subject:
X-Mailer:Mew version 1.95 on Emacs 20.7
----
```

We call this "Draft mode". In Draft mode, we call the region above "—-" header. Also the region below "—-" body.

Also, replying a message ('`a`' or '`A`') and/or forwarding messages ('`f`' or '`F`') lead you to Draft mode from Summary mode.

A draft is temporary stored under the +draft folder. You can write multiple messages at the same time.

Now let's see how to use Draft mode.

4.1 Completions in a header

In header, completions each field are assigned to '`TAB`' as follows:

- Field completion
- Address completion and expansion (To:, Cc:, etc)
- Folder completion (Fcc:)

<Field completion>

If the cursor is on the beginning of a line and the previous line does not end with ",", you can complete field defined in the '`mew-fields`' variable with '`TAB`'.

```
To: kazu@mew.org
R'TAB'
```

At the point above, if you type '`TAB`', you get:

```
To: kazu@mew.org
Reply-To:
```

<Address completion and expansion>

You can define an easy-to-remember short name for a long or hard-to-remember address with Mew's Addrbook feature. For example, consider the following configuration:

```
pooh:           winnie-the-pooh@100acre.woodwest.uk
```

This means to replace the string "pooh" with "winnie-the-pooh@100acre.woodwest.uk". Note that short names should usually be configured in "~/.im/Addrbook". For more information about Addrbook, please refer to See Section 4.3 [addrbook], page 14.

In a header in Draft mode and on the field supposed to write addresses and one or more characters precede, you can complete an short name for address with '`TAB`'.

Let's look at the following example.

```
To: piglet@beech.tree.uk,
        po`TAB'
```

If you input `TAB' up above, "pooh" is completed (unless other candidates exist).

```
To: piglet@beech.tree.uk,
        pooh`TAB'
```

One more type of `TAB' expands it to "winnie-the-pooh@100acre.woodwest.uk".

```
To: piglet@beech.tree.uk,
        winnie-the-pooh@100acre.woodwest.uk
```

If you type `TAB' at improper point for address completion, `TAB' is inserted. Consider the following example:

```
To: pooh,`TAB'
```

In this case, `TAB' is just inserted.

A string ended with "@" is explicitly expanded. Consider the following case where similar short names are defined.

```
pooh:           winnie-the-pooh@100acre.woodwest.uk
pooh-pooh:      pooh-pooh@somewhere.jp
```

To expand "pooh" to "winnie-the-pooh@100acre.woodwest.uk", take this way.

```
To: pooh@`TAB'
```

<Folder completion>

At a point, such as Fcc:, supposed to complete a folder, you can complete a folder with `TAB'. Let's look at an example.

```
Fcc: `TAB'
```

Here, "+" is completed.

```
Fcc: +`TAB'
```

One more type of `TAB' shows candidates. Please input appropriate characters then type `TAB'.

```
Fcc: +B`TAB'
```

If a candidate can be solely decided, you get a completion.

```
Fcc: +Backup
```

<Hints of customization>

You can define which field allows address and folder completion in `mew-field-completion-switch'. The following declaration is used by default.

```
'(("To:"          . mew-complete-address)
  ("Cc:"          . mew-complete-address)
  ("Dcc:"         . mew-complete-address)
  ("Bcc:"         . mew-complete-address)
  ("Reply-To:"    . mew-complete-address)
  ("Fcc:"         . mew-complete-folder)
  ("Resent-To:"   . mew-complete-address)
  ("Resent-Cc:"   . mew-complete-address)
  ("Resent-Dcc:"  . mew-complete-address)
  ("Resent-Bcc:"  . mew-complete-address)
  ("Newsgroups:" . mew-complete-newsgroups))
```

4.2 Circular completions in a header

In a header, circular completions are assigned to '`C-cTAB`'. Circular completion means that a value of alist is replaced by the next value of the list. The end of the list is considered continuous to the top of the list. Circular completions in a header are different for each field as follows:

– Circular completion of domain name (To:, Cc:, etc)
– Circular completion of From: (From:)

<Circular completion of domain name>

On a field where addresses are written, use '`C-cTAB`' for domain completion. Candidates are selected from '`mew-mail-domain-list`'.

```
To: kazu@'C-cTAB'
```

If you try to complete just after `"@"` as up above, the first domain of '`mew-mail-domain-list`' is inserted.

```
To: kazu@mew.org'C-cTAB'
```

After completion, one more type of '`C-cTAB`' inserts the next domain of '`mew-mail-domain-list`'. This completion is looped.

```
To: kazu@wide.ad.jp
```

If a candidate can be solely decided, it is inserted.

```
To: kazu@w'C-cTAB'
```

The example up above gets:

```
To: kazu@wide.ad.jp
```

<Circular completion of From:>

On the From: field, '`C-cTAB`' circularly completes its value from '`mew-from-list`'. The first value of the list (aka '`mew-from`') may have already inserted as follow:

```
From: Kazu Yamamoto <Kazu@Mew.org>
```

Typing '`C-cTAB`' anywhere on the value replaces the value with the next value of '`mew-from-list`'. For example,

```
From: Kazu Yamamoto <Kazu@Mew.org>'C-cTAB'
```

becomes as follows:

```
From: Kazuhiko Yamamoto <kazu@wide.ad.jp>
```

You can define association of field key and circular completion function in '`mew-field-circular-completion-switch`'. The following declaration is used by default.

```
'(("To:"           . mew-circular-complete-domain)
  ("Cc:"           . mew-circular-complete-domain)
  ("Dcc:"          . mew-circular-complete-domain)
  ("Bcc:"          . mew-circular-complete-domain)
  ("Reply-To:"     . mew-circular-complete-domain)
  ("Resent-To:"    . mew-circular-complete-domain)
  ("Resent-Cc:"    . mew-circular-complete-domain)
  ("Resent-Dcc:"   . mew-circular-complete-domain)
  ("Resent-Bcc:"   . mew-circular-complete-domain)
  ("From:"         . mew-circular-complete-from)
  ("Resent-From:"  . mew-circular-complete-from))
```

4.3 Address Book

Mew 1.95 provides an address book("~/Mail/Addrbook). The address book provides 2 formats. One is to specify expansion rules, the other is to define personal information.

First, let's look at the format to specify expansion rules.

```
<shortname>: <address1>[, <address2>, <address3>,...]
```

In this way, you should specify a short name and a full address separating by '`:`'. If you want to expand the short name to multiple addresses, specify them separating by '`,`'. (This is exactly same as addresses separated by '`,`' in the To: field, for instance.) SPC is allowed after '`,`'. The following is an example:

```
pooh:           winnie-the-pooh@100acre.woodwest.uk
piglet:         piglet@beech.tree.uk
friends:        pooh, piglet
```

Multi-level expansion is possible. For example, let's expand "friends" as follows:

```
To: friends'TAB'
```

"friends" is expanded to "pooh" and "piglet" internally, then each word is also expand resulting as follows:

```
To: winnie-the-pooh@100acre.woodwest.uk, piglet@beech.tree.uk
```

Next, the format to define personal information is shown below:

```
<shortname> <address1>[, <address2>, <address3>,...] <nickname> <fullname>
```

In this way, four elements are separated by SPC. <shortname> is a short name. <nickname> and <fullname> are his/her nickname and his/her full name, respectively. The second element is his/her addresses. If he/she has multiple addresses, enumerate them separating by '`,`'. SPC is allowed after '`,`'. So, this SPC is not the separator of the elements. SPC surrounded by '`"`' is not the separator, neither. Let's see an example:

```
kazu kazu@mew.org, kazu@iijlab.net Kazu-kun  "Kazuhiko Yamamoto"
```

Unlike the format of expansion rules, the format of personal information means that each address will be replaced one by one. Consider the following example:

```
To: kazu'TAB'
```

Typing '`TAB`' after "kazu" leads to "kazu@mew.org".

```
To: kazu@mew.org'TAB'
```

Typing '`TAB`' after "kazu@mew.org" makes "kazu@iijlab.net" appeared.

```
To: kazu@iijlab.net'TAB'
```

Then "kazu@mew.org" will appear again if you type '`TAB`' after "kazu@iijlab.net". In this way, each address is replaced one by one. After deciding an address, you can add its full name.

```
To: kazu@mew.org'M-TAB'
```

Like this, typing '`M-TAB`' replace the address with the following format.

```
To: Kazuhiko Yamamoto <kazu@mew.org>
```

In the format of personal information, you can omit each element. When you want to omit intermediate element, specify '`*`'. The following is an example to define nicknames for addresses.

```
* kazu@mew.org, kazu@iijlab.net Kazu-kun
```

The nickname is used to replace addresses in Summary mode and to replace the citation prefix(See Section 4.5 [cite], page 16) in Draft mode.

The comment letters are '`;`' and '`#`'. '`;`' is valid only when it appears in the beginning of lines. The entire line is ignored. '`#`' is valid everywhere. The strings between '`#`' and the end of the line is ignored.

As a matter of fact, there are short names to be defined automatically. When you send a message, addresses on the To: and Cc: field are automatically registered with their user names as short names. Consider the following:

```
To: kazu@mew.org
```

When this message is sent, a short name "kazu" is automatically registered for the address "kazu@mew.org". If there is already a short name of "kazu", the next action is decided according to '`mew-addrbook-override-by-newone`'. If '`nil`', the old entry remains. Otherwise, the new entry overrides the old one. When expanded, the address book is prior to the automatic short name. So, only automatic short names which do not exist in the address book are valid. The limit number of automatic short names is 1000('`mew-lisp-max-length`'). If the number is over 1000, the oldest entry is removed. This information is automatically saved to "~/Mail/.mew-alias".

Summary mode provide the feature to register the information of the current message into Addrbook. To register a expansion rule, type '`C-cC-a`'. To register personal information, type '`C-uC-cC-a`'.

```
#If you want to register this entry, type C-c C-c.
#If you want to NOT register this entry, type C-c C-q.
Shortname: kazu
Addresses: kazu@mew.org
Nickname:
Name: Kazuhiko Yamamoto
Comments:
```

Add or modify the information if necessary. To register this information, type '`C-cC-c`'. To quit the registration, type '`C-cC-q`'

4.4 Sending a message

When you are ready to send a message after writing up, choose one of the following two commands.

'`C-cC-m`' Compose a message, put it into +queue, and let it be waiting to be sent.

'`C-cC-c`' Compose a message and send it. You are asked, "Really send this message? (y or n) ". Type '`n`' to send it.

Let's look at how a message is modified when sent. Let's consider the following example:

```
To: pooh
Subject: the next Sunday
From: Piglet <piglet@beech.tree.uk>
X-Mailer:Mew version 1.95 on Emacs 20.7
----
Would you like to play with me in the next Sunday?

// Piglet
```

This message is modified as follows, for example, if it is queued by '`C-cC-m`'.

```
Date: Mon, 13 Mar 2000 19:49:50 +0900 (JST)
Message-Id: <20000313.194950.59499544.piglet@beech.tree.uk>
To: winnie-the-pooh@100acre.woodwest.uk
Subject: the next Sunday
From: Piglet <piglet@beech.tree.uk>
X-Mailer:Mew version 1.95 on Emacs 20.7
Mime-Version: 1.0
Content-Type: Text/Plain; charset=us-ascii
```

```
Content-Transfer-Encoding: 7bit

Would you like to play with me in the next Sunday?

// Piglet
```

As you can see, Data: and Message-Id: are added. You should note that both its data type and its character set are guessed correctly and added.

There are two methods to send messages waiting in +queue. Please note that both of them are commands in Summary mode.

'i' If '`mew-auto-flush-queue`' is '`t`', flush +queue after receiving messages. In dial-up environments, this is a good idea in a sence of saving connection fee and of authentication for sending. '`mew-auto-flush-queue`' is default to '`t`'.

'C-cC-c' Flush +queue. This would be convenient if you like to preview composed messages first in +queue and send them next. If '`mew-ask-flush-queue`' is '`t`', you are asked, Flush queue? (y or n) ". '`mew-ask-flush-queue`' is default to '`nil`'.

4.5 Signature and citation

Here are explained commands to process text in a body.

The first one is signature. To insert "~/.signature" on the cursor point, type '`C-cTAB`'. You can define your own signature file to '`mew-signature-file`'. Setting '`mew-signature-as-lastpart`' and '`mew-signature-insert-last`', you can customize the action of '`C-cTAB`'.

'C-cTAB' Insert "~/.signature" on the cursor point.

The next one is citation. If you use '`a`' or '`A`' in Summary mode, a draft for reply is prepared and Emacs is split into three windows. The top is Summary mode, the middle is Message mode, and the bottom is Draft mode.

Here are commands to cite text from Message mode to Draft mode.

'C-cC-y' Copy and paste a part of message from Message mode WITH citation prefix and label.

1. Roughly speaking, it copies the body in Message mode. For example, if Text/Plain is displayed, the entire Message mode is copied. If Message/Rfc822 is displayed, the body without the header is copied.
2. If called with '`C-u`', the header is also copied if exists.
3. If an Emacs mark exists, the target is the region between the mark and the cursor.

'C-cC-t' Copy and paste a part of message from Message mode WITHOUT citation prefix and label.

The default label and prefix is as follows:

```
From: SUMIKAWA Munechika <sumikawa@ebina.hitachi.co.jp>
Subject: Wine
Date: Wed, 23 Jul 1997 11:40:50 +0900

> Hi, it's Sumikawa, the neat from good morning to good night.
>
> Talking the party of wonderful wine, I would propose Cabernet
> Sauvignon, Bordeaux, '90. It would be great if Pinot Noir
> is blended a bit.
```

In Draft mode, you can cite any text displayed in Message mode("*mew message*" buffer). So, you can cite text from multiple messages easily. Select a message in Summary mode and display it in Message mode, then cite it in Draft mode. Please repeat this procedure as you like. Triple windows are prepared for this purpose.

You can use "supercite" with Mew but before you start using it, configure as follows.

```
(setq mew-cite-prefix-function 'mew-cite-prefix-username)
```

With this configuration, the citation prefix is preceded by a user name.

```
From: SUMIKAWA Munechika <sumikawa@ebina.hitachi.co.jp>
Subject: Wine
Date: Wed, 23 Jul 1997 11:40:50 +0900

sumikawa> Hi, it's Sumikawa, the neat from good morning to good night.
sumikawa>
sumikawa> Talking the party of wonderful wine, I would propose Cabernet
sumikawa> Sauvignon, Bordeaux, '89. It would be great if Pinot Noir
sumikawa> is blended a bit.
```

In addition to the configuration above, add the following.

```
(setq mew-addrbook-for-cite-label 'nickname)
(setq mew-addrbook-for-cite-prefix 'nickname)
```

With the first line, an address in the citation label is replaced with its nickname(See Section 4.3 [addrbook], page 14). The second line indicates that a user name in the prefix is replaced with its nickname.

```
From: sumitch
Subject: Wine
Date: Wed, 23 Jul 1997 11:40:50 +0900

sumitch> Hi, it's Sumikawa, the neat from good morning to good night.
sumitch>
sumitch> Talking the party of wonderful wine, I would propose Cabernet
sumitch> Sauvignon, Bordeaux, '89. It would be great if Pinot Noir
sumitch> is blended a bit.
```

If you get the following citation style instead of up above, '`mail-citation-hook`' might be defined.

```
In article .....
```

To use Mew original citation style, put the following into your ".emacs".

```
(setq mail-citation-hook nil)
```

4.6 Composing multipart

OK. Let's see how to create multipart.

When you are writing a message in +draft/1 and type '`C-cC-a`', the following lines are inserted at the bottom of the draft.

```
----------------------------- attachments -----------------------------
      Multipart/Mixed                                        1/
     1  Text/Plain(guess)                                      CoverPage*
     2                                                         .
--------0-1-2-3-4-5-6-7-8-9-----------------------------------------------
```

"1/" is a temporary directory to create multipart and locates "~/Mail/draft/mime/1". The part 1, "Coverpage", refers to the body. Now the entire draft looks like:

```
To: mew-dist
Subject: This is header
X-Mailer:Mew version 1.95 on Emacs 20.7
----
This is body.

----------------------------- attachments -----------------------------
      Multipart/Mixed                                         1/
     1  Text/Plain(guess)                                       CoverPage*
     2                                                          .
--------0-1-2-3-4-5-6-7-8-9-------------------------------------------
```

Here we call three regions as follows:

- the region above "----" '`header`'
- the region from "----" to "attachments" '`body`'
- the region below "attachments" '`attachments`'

In Draft mode, key bindings are different on each region.

To '`TAB`', for instance, functions are assigned as follows:

header	Completions.
body	Insert TAB.
attachments	Do nothing.

To '`c`', functions are assigned as follows:

header	Insert c.
body	Insert c.
attachments	Copy a file.

The following is a summary of commands in attachments.

'`C-p`'	Go to the previous file in the current directory.
'`C-n`'	Go to the next file in the current directory.
'`C-f`'	Go to the first subdirectory.
'`C-b`'	Go to the parent directory.
'`c`'	Copy a file (via networks) on ".". To copy a remote file, use the "/[user@]hostname:/filepath" syntax.
'`l`'	Link a file with a symbolic link on ".". If you want to edit the attached file, you should '`c`' instead of '`l`' so that you don't edit the original file.
'`d`'	Delete this file or this directory.
'`m`'	Create a subdirectory(i.e. multipart) on ".".
'`f`'	Open this file into a buffer.
'`F`'	Open a new file into a buffer on ".".
'`y`'	Link the message which is displayed in Message mode on ".".
'`e`'	Input external-body on ".".
'`a`'	Sampling voice and insert as audio file on ".".

'`p`'	Extract the PGP key for the inputed user on "`.`".
'`D`'	Input a description(Content-Description:).
'`T`'	Change the data type(Content-Type:).
'`C`'	Specify charset for a Text/* object.
'`P`'	Specify a file name(Content-Disposition:) to save this part in the receiver side. If you type just '`RET`' without any string, its value is cleared. Then the file name in the sender side is displayed with '`*`'.

In attachments, data types are guessed by suffix. The current supported suffixes are as follows:

```
.txt       Text/Plain
.html      Text/Html
.rfc822    Message/Rfc822
[0-9]+     Message/Rfc822
.ext       Message/External-body
.ps        Application/PostScript
.tar       Application/Octet-stream ;; dummy
.gif       Image/Gif
.jpg       Image/Jpeg
.jpeg      Image/Jpeg
.png       Image/Png
.xwd       Image/X-xwd
.xbm       Image/X-xbm
.bmp       Image/X-bmp
.au        Audio/Basic
.mpg       Video/Mpeg
.mpeg      Video/Mpeg
.pgp       Application/Octet-Stream
.pka       Application/Pgp-keys
.*         Text/Plain
```

For instance, if you copy files with '`c`', the part becomes as follows:(Please choose an appropriate suffix for the file name so that Mew can guesses its data type.)

```
---------------------------- attachments -----------------------------
     Multipart/Mixed                                      1/
     1  Text/Plain(guess)                                   CoverPage*
B    2  Image/Gif                  MagicPoint logo          mgp.gif
Q    3  Application/Postscript     Presentation Material    ohp.ps
     4                                                      .
--------0-1-2-3-4-5-6-7-8-9-------------------------------------------
```

Each line of multipart consists of

- marks (Content-Transfer-Encoding:)
- part number
- data type (Content-Type:)
- description (Content-Description:)
- file name (Content-Disposition:).

Please refer to See Section 4.12 [mark-b-comp], page 24 to know how to change mark (Content-Transfer-Encoding:). You can change data types(Content-Type:) by '`T`' at any time. You can also insert descriptions(Content-Description:) by '`D`'. This description column is overwritten when encrypted as described in See Section 4.12 [mark-b-comp], page 24.

Strictly speaking, the fifth column is the copied file name or the value of Content-Disposition:, namely the file name to which the receiver saves the part. If Content-Disposition: exists, Mew displays it. Otherwise, Mew displays the copied file name with '`*`' appended. When you copy a file, the file name is specified as Content-Disposition:. But this is not true for both Message/* and Multipart/*. To specify Content-Disposition:, use '`P`'.

Files mean singlepart while directories are regarding with multipart. So, you can create very complex multipart MIME as if you created file system. Very easy, isn't it?

The default data type for directories is Multipart/Mixed. Of course, you can change it by '`T`'.

If you become ready to send a multipart message, type '`C-cC-m`' or '`C-cC-c`' to send it as described the previous subsection.

Let's consider Message/External-body by '`e`'. If access-type is "ftp" or "anon-ftp", you can enjoy completion for a remote file name thanks to ange-ftp. If access-type is "local-file", of course, file completion is available.

If you want to quit creating multipart and to get back to singlepart, type '`d`' in the top level multipart.

4.7 Defining charset

Mew has a mechanism to decide charset of the transfer form for both singlepart and multipart.

<Singlepart>

When you type '`C-cC-m`' or '`C-cC-c`' to compose a message on Draft mode, Mew decides charset of the transfer from from the internal representation of its body. On Bilingual Emacs, US-ASCII is chosen for 7bit charset while ISO-8859-1 is selected for 8bit charset. On Mule, charset of the transfer form is chosen based on the rule that Mew defines.

<Multipart>

Since data to be attached as a part of multipart is a file, it is stored on disk. So, to decide its charset of the transfer form, it is necessary to load the file into an Emacs buffer converting it into internal representation. After that, Mew decides charset of the transfer form for the file by the same way of singlepart.

On Bilingual Emacs, Mew reads a file as it is. So, if the file is 7bit, US-ASCII is chosen. Otherwise ISO-8859-1 is selected.

On Mule, Mew reads a file according to the local convention(i.e. auto conversion). Functions to decide local convention are called set[up]-<language>-environment. For more information about local convention, read their descriptions.

For example, in Japan, ISO-2022-JP, EUC-JP, and Shift_JIS is neatly guessed and stored in buffer as internal representation for Japanese. Mew chooses ISO-2022-JP as charset of the transfer form from the internal representation. That is, even if the character set of the file is EUC-JP or Shift_JIS, it is automatically converted into ISO-2022-JP, which is the transfer form of Japanese. So, you can attach a file without taking care of its character set.

If you want to specify character set of a file to be attached, type '`I`'. Let's call the character set "input character set". Also, if you want to specify charset of the transfer form, use '`C`'.

Information of character set is displayed in parentheses. If charset of the transfer form is specified explicitly, it is displayed. Otherwise, if input character set is specified, it is displayed with "*". Otherwise, "guess" is displayed.

Let's look at the following example. Since part 1 is a body, it stored in an Emacs buffer. Because "guess" is displayed, it is Mew that decides charset of the transfer form according to the rules that Mew defines.

Since iso-8859-1 is specified as input character set, Mew loads the file considering that its character set is iso-8859-1, then converts it into internal representation. Charset of the transfer form is decided according to the rules that Mew defines.

The input character set is unknown just from this example. (But a user certainly knows what it is since he actually specified.) Anyway, a file will be loaded and be converted into internal representation, then be converted into EUC-JP which is specified as charset of the transfer form.

```
---------------------------- attachments ----------------------------
      Multipart/Mixed                                        1/
     1  Text/Plain(guess)                                      CoverPage*
     2  Text/Plain(*iso-8859-1)                                text1
B    3  Text/Plain(euc-jp)                                     text2
     4                                                         .
--------0-1-2-3-4-5-6-7-8-9-----------------------------------------
```

Note that both 'C' and 'I' are not available on Bilingual Emacs.

4.8 Replying to a message and deciding recipients

While you specify addresses of the To: and Cc: field for a new message by yourself, addresses are automatically prepared for a reply message.

For a reply message, Mew prepares addresses for the To: and Cc: fields according to the following rules:

If From: of the message to be replied is not from me:
- Reply-To: doesn't exist in the message to be replied
  - Copy From: of the message to be replied to To: (1)
  - Copy To: and Cc: of the message to be replied to Cc: (2)
- Reply-To: exists in the message to be replied
  - Copy From: and Reply-To: of the message to be replied to To: (3)
  - Copy To: and Cc: of the message to be replied to Cc: (4)

If From: of a message to be replied is from me:
- Copy To: of the message to be replied to To: (5)
- Copy Cc: of the message to be replied to Cc: (6)

If there are multiple entries for a certain address, they are uniquefied. Addresses ended with ":;", which stands for anonymous recipients, are automatically removed.

Your addresses are automatically removed. To define your multiple addresses, please use '`mew-mail-address-list`'. An example is as follows:

```
(setq mew-mail-address-list
      '("pooh@[a-z]*.aist-nara.ac.jp"
        "pooh@mew.org"
        "winnie@iijlab.net"))
```

You can customize which fields are copied in the case (1)-(6) with the following variables:

(1) '`mew-noreplyto-to-list`'
(2) '`mew-noreplyto-cc-list`'
(3) '`mew-replyto-to-list`'
(4) '`mew-replyto-cc-list`'
(5) '`mew-fromme-to-list`'
(6) '`mew-fromme-cc-list`'

If you want to reply only to the address specified by Reply-To:, configure as follows:

```
(setq mew-replyto-to-list '("Reply-To:"))
(setq mew-replyto-cc-list nil)
```

If '`a`' or '`A`' is executed with '`C-u`', From: of the message to be replied is copied to To:, and Cc: becomes empty. You can use this to reply the sender only.

4.9 Forwarding messages

To forward messages, type '`f`' or '`F`' in Summary mode. Then, Draft mode appears and the messages are already attached to the attachments region.

Also, you can prepare the attachments region by yourself, then copy('`c`') the messages or make links('`l`') to the messages. If the file name of the messages is numeric([0-9]+), they are automatically considered as messages. '`y`' is very convenient because it make a link to the message displayed in Message mode.

By default, the entire message is forwarded. If you want to remove some parts of its header, define '`mew-field-delete-for-forwarding`'. The following is an example to remove "Received:" and "Return-Path:" when forwarded.

```
(setq mew-field-delete-for-forwarding '("Received:" "Return-Path:"))
```

4.10 Re-sending messages

You occasionally wish to send messages modifying the header of a message.

For instance, you may want to send a message with the same body to multiple receivers independently. Please imagine the case where you put a created message destined to pooh to +queue and then you wish to send an another message to piglet by copying it and modifying its header. Let's call this recycle sending.

Also, you occasionally wish to send a message adding the Resent-To: field to a target message. This is a kind of forwarding. This is called header conversion because a message header is modified. The forwarding described the previous subsection is called encapsulation because a message is embedded into another new message.

Header mode exists for this reason, modifying a part of the header, sending/queuing the created message. You can think this is a kind of Draft mode which prohibits modifications of its body.

The following commands are provided in Summary mode to enter Header mode.

'`W`' Recycle sending. Enter Header mode in order to modify To:, Cc:, From: of a target message. Typical usage is fo messages in +queue.

'`r`' Re-sending. Enter Header mode in order to add Resent-To:, Resent-Cc:, Resent-From: of a target message. Re-sending may confuse receivers, so you should think carefully before using it.

In Header mode, you can make use of completion and circular completion like in Draft mode. When you are finished inputing header, send the message using one of the following commands. You may wonder because body is not displayed. However, the body and a part of the target message is certainly used.

'`C-cC-m`' Compose a message, put it into +queue, and let it be waiting to be sent.

'`C-cC-c`' Compose a message and send it. You are asked, "Really send this message? (y or n) ". Type '`n`' to send it.

4.11 Using PGP

This section describes to sign or encrypt "text only" message with PGP. The following commands are explained.

'`C-cC-s`' Sign the entire draft with PGP. Input your passphrase.

'`C-cC-e`' Encrypt the entire draft with PGP.

'`C-cC-b`' Sign then encrypt the entire draft with PGP. Input your passphrase.

'`C-cC-r`' Encrypt then sign the entire draft with PGP. Input your passphrase.

To encrypt a message, receivers' public keys are used. In the contrary, your secret key is used to sign a message. So, you need to input your pass-phrase when sign up. Note that if you use the pass-phrase cache and if pass-phrases are cached, you do not have to type your pass-phrase(See Section 3.3 [pgp-viewing], page 6).

They are shortcut methods of mark based composing described in the next subsection.

If you use PGP with Mew, you have to select Email address for your PGP userid(e.g. "Kazuhiko Yamamoto <kazu@mew.org>").

To sign a message, type '`C-cC-s`'. When signing, because your secret key is needed to be decrypted, you may ask your passphrase, if not cached. Created messages are stored in +queue.

Your secret key is identified by the address of From:. If From: does not exist, PGP automatically selects your default secret key. When you want to specify a secret key which is not identified the address of From:, type '`C-uC-cC-s`'.

To encrypt a message, type '`C-cC-e`'. A message is encrypted with public keys identified the addresses on To: and Cc:. Since this is encryption only, you are not asked your passphrase. Created cipher messages are stored in +queue.

This message is encrypted with your public key in addition to the receivers. So you can decrypt created messages. For example, you can go to +queue by '`g`' and preview a created message just in case.

To sign a draft and then encrypt it, type '`C-cC-b`'. To encrypt a draft and then sign it, type '`C-cC-r`'. In both cases, created messages are stored in +queue.

It is very likely that you forget to sign and/or encrypt a draft even if you want to do so. To resolve this, Mew provides automatic PGP mechanism for the massage creation function, '`C-cC-m`' or '`C-cC-c`'.

If you want to protect privacy of all drafts, set '`mew-protect-privacy-always`' to '`t`' and set '`mew-protect-privacy-always-type`' to one of PGP services.

If you want to protect privacy of drafts replying encrypted messages, set '`mew-protect-privacy-encrypted`' to '`t`' and set '`mew-protect-privacy-encrypted-type`' to one of PGP services. This configuration is preferred to the configuration for all drafts described above in the case of replying encrypted messages.

The following services are available. The strings in brace are symbol of each service. The service applied when typing '`C-cC-m`' or '`C-cC-c`' is displayed in the mode line.

pgp-signature (PS)
: Sign

pgp-encryption (PE)
: Encrypt

pgp-signature-encryption (PSPE)
: Sign then encrypt

pgp-encryption-signature (PEPS)
: Encrypt then sign

The following example is to sign all drafts.

```
(setq mew-protect-privacy-always t)
(setq mew-protect-privacy-always-type 'pgp-signature)
```

The following example is to encrypt drafts replying encrypted messages.

```
(setq mew-protect-privacy-encrypted t)
(setq mew-protect-privacy-encrypted-type 'pgp-encryption)
```

In Draft mode, 'C-cC-pC-a' toggles 'mew-protect-privacy-always' and 'C-cC-pC-e' toggles 'mew-protect-privacy-encrypted'.

You can specify privacy services for the current draft in Draft mode beforehand so that you don't forget to apply the privacy services when sending. To set privacy services to the current draft evaluated when typing 'C-cC-m' or 'C-cC-c', type 'C-cC-pC-d' then input one of privacy services above.

4.12 Mark based composer

To support PGP/MIME, mark based composing is provided. Remember the previous example.

```
---------------------------- attachments -----------------------------
      Multipart/Mixed                                          1/
     1  Text/Plain(guess)                                        CoverPage*
B    2  Image/Gif                    MagicPoint logo             mgp.gif
Q    3  Application/Postscript       Presentation Material       ohp.ps
     4                                                           .
--------0-1-2-3-4-5-6-7-8-9-------------------------------------------
```

You can find the 'B' mark and the 'Q' mark at the beginning of line. Mew provides you with a new concept of "encoding". Encoding includes Base64, Quoted-Printable, Gzip64(Gzip + Base64), sign with PGP, encrypt with PGP.

Currently 6 marks are prepared.

'" "'	No encoding. But 8bit text would be encoded.
'B'	Base64
'Q'	Quoted-Printable
'G'	Gzip64(compressed with gzip then encoded with Base64. This is experimental. Don't use this if receivers don't use Mew.)
'PS'	Sign with PGP
'PE'	Encrypt with PGP

Additional key binding for marks in attachments is as follows:

'B'	Put the 'B' mark to encode with Base64.
'Q'	Put the 'Q' mark to encode with Quoted-Printable.
'G'	Put the 'G' mark to encode with Gzip64. This is applicable only to Text/Plain and Application/Postscript since compression is not effective other objects. For example, JPEG is already compressed.
'S'	Put the 'PS' mark to sign with PGP.
'E'	Put the 'PE' mark to encrypt with PGP. Input decryptors' addresses.
'U'	Unmark. The original mark appears.

Consider the following example. The second part will be signed with PGP then encrypted with PGP for "kazu". Take it easy! It's description is overwritten but saved. The third part will be encoded with Gzip64.

```
---------------------------- attachments -----------------------------
      Multipart/Mixed                                          1/
     1  Text/Plain(guess)                                        CoverPage*
PSPE 2  Image/Gif                    kazu@mew.org                mgp.gif
G    3  Application/Postscript       Presentation Material       ohp.ps
```

```
     4                                                           .
--------0-1-2-3-4-5-6-7-8-9-------------------------------------------
```

After putting marks, type '`C-cC-m`' or '`C-cC-c`' to create PGP/MIME messages.

4.13 PGP key distribution

To distribute a PGP public key, please use '`p`' on attachments in Draft mode. It asks whose public key you want to distribute. Just type '`RET`' if it is yours. If you want to distribute another person's, input his Email address with completion. The PGP public key will distributed as Content-Type: Application/Pgp-keys.

If Mew finds that the part is Application/Pgp-keys, it tries to add the PGP public key onto your PGP public keyring. Remember that Mew is careless about both TRUST and VALIDITY. It is YOU who set these values. Please use "pgp -ke" and "pgp -ks" to change them. If you don't know what TRUST and VALIDITY is, you should learn the web of trust system provided by PGP BEFORE using PGP to protect your privacy.

4.14 Sending messages with anonymous receivers

You occasionally wish to send a message to receivers with anonymous destinations.

For instance, please imagine that pooh is trying to invite multiple friends to party. Piglet should reply only to pooh. However, piglet would make a mistake of replying nother people if friends are enumerated. Moreover, pooh want not to disclose whom he invited until the day of the party.

To achieve this, Mew makes use of ":;" for anonymous destinations. Please look at the following example.

```
To: party:piglet@beech.tree.uk,roo@beech.tree.uk;
From: Pooh <winnie-the-pooh@100acre.woodwest.uk>
```

There is an explanation string "party" after ":". Then some addresses are enumerated separated by "," and are terminated by ";". If you write addresses in this format, Mew send the message to the addresses with them removed form the header. With this example, piglet and roo will receive the follogin message.

```
To: party:;
From: Pooh <winnie-the-pooh@100acre.woodwest.uk>
```

A receiver can understand that the sender is pooh, but can't understand who else received this. Also, the string "party:;" is not address, so the receiver can't reply to it.

You should understand that enumerating many addresses on To: or Cc: is not good essentially, anyway. If you have many chances to send messages to certain multiple people, you should create a mailing-list.

5 Funny marks

Here are described marks in Summary mode. Mark is displayed right side of message number as follows:

```
1D 07/17 Itojun        v6: items to be no in6_pcbnotify() doesn't
2o 07/18 Utashiro      Re: behavior after I'm afraid that mark-ring
38 07/19 Nom-sun       refile info.       Sorry for my late respon
```

There are four marks at present.

'D'

'X' The mark to delete.

'o' The mark to refile, that is, to move a message to another folder.

'@' The mark to process messages at the same time.

'*' The mark to review.

They are explained step by step.

5.1 Delete 'D', 'X'

If you want to delete a message, type 'd' in Summary mode to put the 'D' mark on it. Take it easy. Since putting the 'D' mark causes nothing, mistakes are not fatal. By default, typing 'x' moves messages marked with 'D' to the +trash folder.

To delete messages in the +trash folder really, there are two methods by default.

1. Type 'D' in Summary mode.
2. Put the 'D' mark in the +trash folder then type 'x'.

So far, the word "by default" was repeated. This means that 'mew-msg-rm-policy' is set to ''trashonly'. You can set one of the following values to 'mew-msg-rm-policy'. Each explanation is about action when 'x' is pressed.

''totrash'
: Refile to the +trash folder if not in the +trash folder. Just umark the 'D' mark if in the +trash folder.

''always' Really remove messages marked with 'D' always anyway.

''trashonly'
: Really remove messages marked with 'D' if in the +trash folder. In other fonders, refile to the +trash folder.

''uselist'
: Really remove messages marked with 'D' if in a folder found in 'mew-msg-rm-folder-list'. In other folders, refile to the +trash folder.

'other values'
: Considered as 'trashonly.

Customize the action of 'x' as you like.

It is convenient if you can change the '*' mark to the 'D' mark since you can put many the 'D' marks at once. To achieve this, use 'md'.

There is also another mark 'X' which is like the mark 'X'. Messages marked with 'X' are deleted when 'x' is typed regardless the value of 'mew-msg-rm-policy'. The 'X' mark can be put by 'M-d'.

The following is a summary regarding with the 'D' mark and the 'X' mark.

'`d`'	Put the '`D`' mark.
'`M-d`'	Put the '`X`' mark.
'`x`'	Process messages marked with '`D`' according to '`mew-msg-rm-policy`'. Also, delete messages marked with '`X`'.
'`mxd`'	Process messages marked with '`D`' only according to '`mew-msg-rm-policy`'.
'`mxM-d`'	Delete messages marked with '`X`' only.
'`md`'	Change all '`*`' mark to the '`D`' mark.
'`mM-d`'	Change all '`*`' mark to the '`X`' mark.

5.2 Refile '`o`'

To refile a message, type '`o`' and input a folder name, then '`o`' is put. A folder is guessed neatly, so most time what you should to is just type '`RET`'. For more information, refer to See Chapter 6 [Refile], page 30.

If you type '`o`' on a message marked with '`o`', the refile folder is shown. When typing '`x`', messages marked with '`o`' are actually refiled.

A summary about the '`o`' mark is here.

'`o`'	Put the '`o`' mark.
'`x`'	Refile messages marked with '`o`'.
'`mxo`'	Refile messages marked with '`o`' only. This command doesn't process other marks.
'`mo`'	Change all '`*`' marks to the '`o`' mark to be refiled.

5.3 Multiple '`@`'

To process multiple messages, put the '`@`' mark. Here is a summary concerned with the '`@`' mark.

'`@`'	Put the '`@`' mark.
'`F`'	Prepare a draft to forward multiple messages marked with '`@`' in MIME format.
'`M-s`'	Apply unshar on messages marked with '`@`'.
'`M-t`'	Apply `"uudecode"` on messages marked with '`@`'.
'`M-b`'	De-capsulate messages embedded in the messages marked with '`@`'.
'`J`'	A large message is occasionally fragmented into multiple messages whose Content-Type: is Message/Partial. This command produces the original message from Message/Partial messages marked with '`@`'.

For '`M-s`' and '`M-t`', messages marked with '`@`' are supposed to be in order(numbers can be discrete). If out of order, sorting with '`S`' would help.

5.4 Review '*'

Please put the '*' mark onto messages that you want to review later. '?' also put the '*' mark onto matched messages(for more information, refer to See Chapter 7 [Pick], page 35). Use 'N' and 'P' to walk around messages marked with '*'.

Here is a summary for '*' commands.

'*' Put the '*' mark.

'N' Jump to the message marked with '*' below and display it.

'P' Jump to the message marked with '*' above and display it.

'M-n' Display a message marked with '*' and find a keyword and highlight it in the forward direction. The keyword is stored in a buffer local variable in Summary mode and is set when you use 'C-u?' or 'C-u/' (see See Chapter 7 [Pick], page 35). If no key word is set to the variable, this command first asks you a keyword. If you want to change the stored keyword, execute this command with 'C-u'.

'M-p' A reverse version of 'M-n'.

'ma' Put the '*' mark to all unmarked messages.

'mr' Put the '*' mark to all matching messages with inputed regular expression.

'md' Change '*' marks to 'D' mark. It is useful to delete messages selected by '?'.

'mo' Change the '*' marks to the 'o' mark. It is useful to refile messages selected by '?'.

5.5 Deleting marks

Refiling and deleting is not processed unless you type 'x'. So, if you press 'u' to unmark before you type 'x', messages do not disappear accidentally.

Here is a summary for unmark commands.

'u' Cancel the mark on this message.

'U' Cancel all marks according to what you input.

5.6 Mark strength

Marks are classified into two categories, "strong mark" and "weak mark". The same level mark can be overwritten. A strong mark can overwrite a weak mark.

When you put a mark, the following action is taken.

Strong marks :: 'o' and 'D'
If marks an unmarked message, display the next message. If overwrites, stay the current line.

Weak marks :: '*' and '@'
Stay the current line always.

Refer to See Section 9.1 [level-one], page 38 to know which direction the cursor moves after putting a strong mark.

You can exchange marks as follows:

'm@' '*' -> '@' :: It is useful when you pick messages by '?' then pass them to "uumerge" with 'M-t'.

'm*' '@' -> '*'

‘`ms`’	‘`@`’ `<->` ‘`*`’
‘`md`’	‘`*`’ `->` ‘`D`’ :: It is useful when you put ‘`D`’ marks to messages picked by ‘`?`’.
‘`mo`’	‘`*`’ `->` ‘`o`’ :: It is useful when you put ‘`o`’ marks to messages picked by ‘`?`’.

6 Happy refiling

When you come to receive hundreds of messages in a day(don't you believe it?), refiling messages becomes a very tough job. Mew neatly guesses default folders where the message is supposed to be refiled when you type '`o`'. You can see an example below.

```
Folder name (+work/mew-dist): +
```

If the default value in () is proper, just type '`RET`'. The messages will be marked with '`o`' if its refiling folders are decided.

As you know, the more excellent refiling guess algorithms become, the less user's job troublesome. Mew provides you with the following rules.

6.1 Guess by mailing-list folders

Many users tend to refile messages destined to a mailing-list to a folder whose name is the same as the mailing-list. Mew provides a mechanism to guess a mailing-list folder for messages destined to mailing-lists.

Suppose that you have a folder named +misc/pooh-lovers. The following message is probably to be refiled to this folder.

```
To: pooh-lovers@mew.org
```

Likewise, Mew searches a matching folder forward with addresses on To: and Cc:. There are many people who don't use recursive folders. With Mew, however, you would not be smart if you don't use it.

Smart users may wonder that they get a trouble in the following situation where private addresses are on To: or Cc:.

```
To: piglet@mew.org
Cc: pooh-lovers@mew.org
```

Since pooh is a member of pooh-lovers, he receives this message. But he has a folder for his friend, pooh. So, +from/piglet may be chosen.

To avoid this, Mew allows you to specify which folders are to be ignored. The default is +from. So, please take a convention to refile personal messages under +from.

When Mew guesses a candidate by the folders, it asks you:

```
Folder name (+misc/pooh-lovers): +
```

Just type '`RET`' if the default is exactly what you want.

If you specify a new folder with '`o`', the folder is created and added to the folder list to be used for guess. Convenient, isn't it?

The function name to provide this feature is '`mew-refile-guess-by-folder`'.

6.2 Guess by user defined rules

There are some cases where the refile guess mechanism by folders doesn't work as you wish. For example, for both a message whose To: is staff@mew.org and another message whose To: is staff@iijlab.net, the same folder would be selected with guess by folders(e.g. "+net/staff"). So, Mew allows you to define your own rules explicitly.

Let's look at an example.

```
(setq  mew-refile-guess-alist
  '(("To:"
      ("staff@mew.org" . "+net/mew/staff")
      ("staff@iijlab.net" . "+net/iijlab/staff")
```

```
        )))
```

This means that if To: contains staff@mew.org +net/mew/staff is selected and if To: has staff@iijlab.net +net/iijlab/staff is chosen.

The format of this rule is as follow:

```
rule ::= '((<key> <alist>) (<key> <alist>) (<key> <alist>) ...)
```

The whole is a list of (<key> <alist>). A field name is specified for <key>. The format for <alist> is as follows:

```
<alist> ::= (<value> . <folder>|<rule>) (<value> . <folder>|<rule>) ...
```

<value> is a field value for <key>. <folder> means a folder to be chosen if matched. Please note that <value> and <folder> is separated with '`.`'.

There are two special <key>s: '`nil`' and '`t`'. '`nil`' is used to specify <value> to be returned when nothing is guessed. '`t`' is for <value> to be returned in addition to guessed values.

If you know regular expression, a more advanced rule can be defined like this.

```
(setq mew-refile-guess-alist
  '(("Newsgroups:"
     ("^nifty\\.\\([^ ]+\\)" . "+Nifty/\\1")
     (".*"          . "+rec/news"))
    ("To:"
     ("\\(inet\\|wide\\)@wnoc-fuk" . "+wide/\\1-wnoc-fuk"))
    ("From:"
     ("uucp@"       . "+adm/uucp")
     ("ftpsync@"    . "+adm/ftpsync"))
    (nil . "+unknown")))
```

The function name to provide this feature is '`mew-refile-guess-by-alist`'.

6.3 Guess by thread

Mew provides a mechanism to guess a folder where the parent message of a current message was refiled before.

For example, pooh, piglet, and roo had a chat to go and get honey. So, pooh made +project/honey then refiled the message to it. The further messages, if they are properly replied, they are supposed to be refiled to +project/honey.

Information that which folder was chosen for messages is stored to "~/Mail/.mew-refile-msgid-alist". '`mew-lisp-max-length`' controls the amount of this information. The default value is 1000 messages. If you want limits it to 2000 messages, put the following to "~/.emacs".

```
(setq mew-lisp-max-length 2000)
```

The function name to provide this feature is '`mew-refile-guess-by-message-id`'.

6.4 Guess by private folders

In addition to the mechanism to select a mailing-list folder described in See Section 6.1 [by-folder], page 30, Mew provides a mechanism to choose a private folder. Since private folders locate under +from, we can say that this mechanism select a folder from the folders under +from. Let's see the following example:

```
To: pooh@mew.org
From: piglet@mew.org
```

pooh received a message from piglet. If pooh uses this mechanism, +from/piglet will be chosen according to From:. (Folders under +from can be recurse. And you can select the entire address for a folder name instead of the user part.)

The function to provide this feature is '`mew-refile-guess-by-from-folder`'.

Next, let's consider a care where pooh replied to piglet. Since pooh Cc:ed the message to himself, the message was also delivered to him.

```
To: piglet@mew.org
Cc: pooh@mew.org
From: pooh@mew.org
```

How do you feel if you are pooh? You may want to refile this to +from/pooh. Also, you may want to move this to +from/piglet. So, it can be customized.

If '`mew-refile-guess-from-me-is-special`' is '`t`' and if an address in From: is yourself, '`mew-refile-guess-by-from-folder`' select a folder under +from according to To: and/or Cc:.

6.5 Guess by From:

Mew also provides a mechanism to guess a folder by the place where a message that has the same From: field is refiled.

Suppose that piglet has two addresses, piglet@beech.tree.uk and p-p-p@mew.org. pooh wants to refile messages from piglet to +from/piglet no matter what his From: is. This policy can, of course, be implemented if pooh specifies rules explicitly as follows:

```
(setq  mew-refile-guess-alist
  '(("From:"
      ("piglet@beech.tree.uk" . "+from/piglet")
      ("p-p-p@mew.org"        . "+from/piglet"))))
```

But such a work may bother you. So, first refile a message whose From: is piglet@beech.tree.uk to +from/piglet. At this time, +from/piglet is created. Next, refile a message whose From: is p-p-p@mew.org to +from/piglet. Here Mews learns that p-p-p@mew.org was refiled to +from/piglet. After this, when messages whose From: is p-p-p@mew.org are refiled, +from/piglet is chosen.

For another example, you can refile messages from machinery to +adm/misc without defining an explicit rule.

Information for relationship between From: and folder is stored to "~/Mail/.mew-refile-from-alist". '`mew-lisp-max-length`' controls the amount of this information as the same as See Section 6.3 [by-thread], page 31.

The function name to provide this feature is '`mew-refile-guess-by-from`'.

If the value of '`mew-refile-guess-from-me-is-special`' is '`t`', '`mew-refile-guess-by-from`' acts as '`mew-refile-guess-by-from-folder`'(See Section 6.4 [by-from-folder], page 31).

6.6 Guess by Newsgroups:

For those who read NetNews received by Email with Mew, Mew provides a mechanism to guess a folder by Newsgroups:. It will be also useful when Mew integrates NetNews in the future. The function name to provide this feature is '`mew-refile-guess-by-newsgroups`'.

6.7 Guess by default rule

The default rule is extract an address from From: and chooses '`+from/user@domain`'. But if '`mew-refile-guess-strip-domainpart`' is '`t`', it extracts the user part. So, '`+from/user`' is chosen.

The function name is '`mew-refile-guess-by-default`'.

6.8 Controlling rules

Mew controls guess rules by two variables, '`mew-refile-guess-control`' and '`mew-refile-ctrl-multi`'. If you want multiple candidates, set '`mew-refile-ctrl-multi`' '`t`'. Otherwise, set it '`nil`'.

By default, '`mew-refile-guess-control`' is declared as follows(since it is a declaration, '`defvar`' is used):

```
(defvar mew-refile-guess-control
  '(mew-refile-guess-by-alist
    mew-refile-ctrl-throw
    mew-refile-guess-by-newsgroups
    mew-refile-guess-by-folder
    mew-refile-ctrl-throw
    mew-refile-ctrl-auto-boundary
    mew-refile-guess-by-thread
    mew-refile-ctrl-throw
    mew-refile-guess-by-from-folder
    mew-refile-ctrl-throw
    mew-refile-guess-by-from
    mew-refile-ctrl-throw
    mew-refile-guess-by-default))
```

Mew executes every function defined in '`mew-refile-guess-control`' in order. Each function may guess multiple candidates.

Let's see the following example of '`mew-refile-guess-control`' action.

'`mew-refile-guess-by-alist`'
guessed +aaa, +bbb.

'`mew-refile-guess-by-folder`'
guessed +ccc, +ddd.

'`mew-refile-guess-by-default`'
guessed +eee.

If you want to provide all candidates, +aaa - +eee, set '`mew-refile-ctrl-multi`' '`t`'. If you want to provide +aaa only, set it '`nil`'.

If you want +aaa - +ddd but don't want left candidates, in other words, you want +eee only when no candidate is guessed by functions executed before, set '`mew-refile-ctrl-multi`' '`t`' and insert '`mew-refile-ctrl-throw`' between '`mew-refile-guess-by-folder`' and '`mew-refile-guess-by-default`'.

'`C-uo`' displays the flow of guess rules in Message buffer.

6.9 Auto refile

Those who receives many messages everyday is prone to store thousands of messages in the +inbox folder. In such a case, they may want to speak out like this, "Hey messages, get out of the +inbox folder to somewhere". Mew provides a feature to satisfy such laziness. :) '`M-o`' is the spell.

When you execute this function, it marks specific messages with '`o`'. The specific messages mean messages which are not marked with '`o`' nor '`D`' if '`mew-refile-auto-refile-skip-any-mark`' is '`nil`'. If '`mew-refile-auto-refile-skip-any-mark`' is '`t`', they mean non-marked messages. The default value of '`mew-refile-auto-refile-skip-any-mark`' is '`nil`'. If executed with '`C-u`', the targets mean messages marked with '`*`' regardless the value of '`mew-refile-auto-refile-skip-any-mark`'.

Refile rule is the same described in the previous section. Please note that what this function does is just mark messages with '`o`'. Messages are not refiled until you will press '`x`'.

Mew's refile mechanism is so smart that it would be harmful for this function. That is, most users would not understand where messages have been refiled if Mew made most use of its guess mechanism. :) For this reason, break is provided to limit usage of guess functions. Recall the declaration up above.

```
(defvar mew-refile-guess-control
  '(mew-refile-guess-by-alist
    mew-refile-ctrl-throw
    mew-refile-guess-by-newsgroups
    mew-refile-guess-by-folder
    mew-refile-ctrl-throw
    mew-refile-ctrl-auto-boundary
    mew-refile-guess-by-thread
    mew-refile-ctrl-throw
    mew-refile-guess-by-from-folder
    mew-refile-ctrl-throw
    mew-refile-guess-by-from
    mew-refile-ctrl-throw
    mew-refile-guess-by-default))
```

You can find the '`mew-refile-ctrl-auto-boundary`' function in '`mew-refile-guess-control`'. Only when auto refile is used, Mew ignores guess functions below this function. If guess functions above '`mew-refile-ctrl-auto-boundary`' didn't guess any folder for a message, the message is not marked with '`o`'. Insert '`mew-refile-ctrl-auto-boundary`' before you ruin.

7 How to select message which you want

You may want to pick up messages whose Subject: contains a string "party" and whose From: is kazu@mew.org. Mew provides three methods to accomplish this.

‘`?`’
: Pick messages according to a pick pattern which you input, then put the ‘`*`’ mark onto them.

‘`/`’

‘`V`’
: Go to Virtual mode which gives a single view to picked messages from multiple folders. Enter a virtual folder name, comma-separated folders, and pick pattern.

These commands select messages by using "mewls". "mewls" targets headers only. If you want to select messages whose body contain a certain keyword, use another external command ("grep" by default). If you call these commands with ‘`C-u`’, they executes the external command instead of "mewls".

Following sections describes how to input conditions and Virtual mode in detail.

7.1 How to input conditions

When Mew asks a user to input pick pattern, the following message is displayed.

```
Pick pattern:
```

Input pick pattern combining the following expressions (which are listed in the strong order):

‘`field=string`’
: Match if the "field" field contains the "string" string (case-insensitive). If you specify "head", it means the entire header.

‘`field==string`’
: Match if the "field" field contains the "string" string (case-sensitive). If you specify "head", it means the entire header.

‘`field!=string`’
: Match if the "field" field does not the "string" string (case-insensitive). If you specify "head", it means the entire header.

‘`field!==string`’
: Match if the "field" field does not the "string" string (case-sensitive). If you specify "head", it means the entire header.

‘`( <pattern> )`’
: Evaluate <pattern> first.

‘`! <pattern>`’
: Match if not <pattern>.

‘`<pattern1> & <pattern2>`’
: Match if <pattern1> AND <pattern2>.

‘`<pattern1> | <pattern2>`’
: Match if <pattern1> OR <pattern2>.

Some examples are shown below.

(a) Messages whose From: contains "kazu".

```
from=kazu
```

(b) Messages whose To: contains "mew" OR Cc: contains "mew".

```
to=mew | cc=mew
```

(c) Messages whose To: contains "mew" OR Cc: contains "mew" AND From: contains "kazu".

```
(to=mew | cc=mew) & from=kazu
```

Now you can guess how to input more complex patterns.

If you want to select messages which contains a certain keyword in its header or its body, type '`?`' or '`/`' with '`C-u`' to call an external command. In this case, you are asked as follows:

```
Grep pattern:
```

Please input a keyword solely. (Don't input expressions up above.) By the way, if you want to display a portion which contains selected keyword, use '`M-n`' or '`M-p`'(see See Section 5.4 [review mark], page 28).

7.2 Virtual mode

Virtual mode gives you a single view for matching messages from multiple folders. Press '`/`' (or '`V`') in Summary mode to enter Virtual mode.

First you are asked a name of virtual folder

```
Virtual folder name (virtual) :
```

Input an appropriate string. If you type just '`RET`', "++virtual" is selected. Then you are asked to input a single or multiple folder name. If you want to specify multiple folders, please separate them with ",". Of course, you can complete folder names with '`TAB`'.

```
Folder name (+inbox) : +inbox, +mew
```

Now input conditions.

```
Pick pattern:
```

OK. You get a Virtual mode. In Virtual mode, you can use same commands that you can find in Summary mode except refile, delete, pick and etc. You should take note that Virtual folder is really virtual and it does not exist in file system. If you exit Mew, it then disappears.

8 Give me a break

Here are how to quit and suspend Mew and how to erase a mode(buffer).

<Summary mode and Virtual mode>

‘`q`’	Suspend Mew then switch to another buffer. All buffers of Mew remain, so you can resume with buffer operations.
‘`Q`’	Quit Mew. All buffers of Mew are erased.
‘`C-cC-q`’	Erase the current mode(buffer).

<Draft mode and Header mode>

‘`C-cC-q`’	Erase the draft.

<Addrbook mode>

‘`C-cC-q`’	Erase the buffer.

9 Customizing Mew

Here are explained how to change default setting of Mew and make Mew your favorite. Edit "~/.emacs" to do it!

9.1 Beginner course

Here are described the following variables.

– mew-draft-mode-hook
– mew-from
– mew-fcc
– mew-cc
– mew-dcc
– mew-window-use-full
– mew-summary-show-direction
– mew-summary-mark-direction

Draft mode evaluates '`text-mode-hook`' and '`mew-draft-mode-hook`' in order. If you don't set '`auto-fill-mode`' to '`text-mode-hook`', It would be useful to define '`mew-draft-mode-hook`' as follows:

```
(setq mew-draft-mode-hook (function (lambda () (auto-fill-mode 1))))
```

You may want to use another address than which your Email manager defined. An example is the case that a host name appears in your Email address since the configuration by your Email manager is imperfect. (In this case, ask him to resolve this problem first.) If From: is specified on a draft, Mew treats it as it is. You can specify From: with completion. If you want to prepare From: in drafts, set '`mew-from`' as follows:

```
(setq mew-from "Kazu Yamamoto <Kazu@Mew.org>")
```

This feature means that you cannot trust From: in all cases. I would sincerely ask all Mew users NOT to cheat others with this feature. And please note that a bad guy can set himself up as another guy very easily. For important messages, use PGP/MIME.

If you want to back up your messages every time when you write them with Fcc:, put the following line into ".emacs".

```
(setq mew-fcc "+Backup")
```

Define '`mew-cc`'('`mew-dcc`') to use Cc:(Dcc:).

If you want to use Mew with full Emacs frame, set like this.

```
(setq mew-window-use-full t)
```

'`SPC`' in Summary mode decide the next action to view a message according to '`mew-summary-show-direction`'. Likewise, you can specify the cursor direction after putting a strong mark by '`mew-summary-mark-direction`'. You can select one from the following candidates.

'`up`'	Display the message above.
'`down`'	Display the message below.
'`next`'	Display the next message in the direction.
'`stop`'	Do not display the next message.

The default value for both is '`next`'. If you read messages from the bottom, set as follows:

```
(setq mew-summary-show-direction 'up)
```

9.2 Junior course

Here are described the following variables.

- mew-use-highlight-cursor-line
- mew-use-highlight-mouse-line
- mew-use-highlight-x-face

If '`mew-use-highlight-cursor-line`' is '`t`', underline is put on the cursor line in Summary mode. The default is '`t`'.

If '`mew-use-highlight-mouse-line`' is '`t`' on XEmacs, the mouse line is painted in Summary mode. This is very convenient to read messages clicking the middle button of the mouse. The default value is '`t`'.

If '`mew-use-highlight-x-face`' is '`t`' on XEmacs, X-Face: in a header is iconified in Message mode. The default value on XEmacs is '`t`'.

9.3 Senior course

Here are described the following variables.

- mew-header-alist
- mew-cite-fields
- mew-cite-format
- mew-cite-prefix

Set header fields that you want to insert every time when you write messages to '`mew-header-alist`' as an associate list. The following is an example.

```
(setq mew-header-alist
      '(("X-fingerprint:" . "6B 63 38 88 67 5E 96 8E  CE A4 62 73 3F 11 64 94")
        ("X-URL:" . "http://www.mew.org/~kazu/")))
```

To customize citation label, define field to cite in '`mew-cite-fields`' and format in '`mew-cite-format`'. Define citation symbol to '`mew-cite-prefix`'. The default declaration is as follows:

```
(defvar mew-cite-fields '("From:" "Subject:" "Date:"))
(defvar mew-cite-format "From: %s\nSubject: %s\nDate: %s\n\n")
(defvar mew-cite-prefix "> ")
```

To add Message-ID: to citation label and to change the prefix with a user name, take this way.

```
(setq mew-cite-fields '("From:" "Subject:" "Date:" "Message-ID:"))
(setq mew-cite-format "From: %s\nSubject: %s\nDate: %s\nMessage-ID: %s\n\n")
(setq mew-cite-prefix-function 'mew-cite-prefix-username)
```

9.4 Hooks

Here is a summary of hooks used in Mew.

'`mew-env-hook`'
: Hook called at initialize time before setting environment.

'`mew-init-hook`'
: Hook called at initialize time.

'`mew-summary-mode-hook`'
: Hook called in Summary mode.

'`mew-virtual-mode-hook`'
: Hook called in Virtual mode.

'`mew-thread-display-hook`'
: Hook called after new threads are displayed.

'`mew-header-mode-hook`'
: Hook called in Header mode.

'`mew-draft-mode-hook`'
: Hook called in Draft mode.

'`mew-draft-mode-newdraft-hook`'
: Hook called in Draft mode only when new draft is prepared.

'`mew-draft-mode-reedit-hook`'
: Hook called in Draft mode when a message not in +draft is re-edited.

'`mew-draft-mode-reedit-draft-hook`'
: Hook called in Draft mode when a message in +draft is re-edited.

'`mew-draft-mode-edit-again-hook`'
: Hook called in Draft mode when a message returned with the old sytale is edited again.

'`mew-message-mode-hook`'
: Hook called in Message mode.

'`mew-message-hook`'
: Hook called whenever message displayed.

'`mew-make-message-hook`'
: Hook called before making a message in Draft mode. A good example is as follows: (add-hook 'mew-make-message-hook 'ispell-message)

'`mew-send-hook`'
: Hook called before sending a message in Draft mode.

'`mew-real-send-hook`'
: Hook called before really sending/queuing a message in Draft mode

'`mew-smtp-sentinel-hook`'
: Hook called when a SMTP process finished.

'`mew-smtp-flush-hook`'
: Hook called after the queue is flushed.

'`mew-suspend-hook`'
: Hook called on suspend.

'`mew-quit-hook`'
: Hook called on quit.

'`mew-pop-sentinel-hook`'
: Hook called when a POP process finished.

'`mew-pop-sentinel-non-biff-hook`'
: Hook called when a non-Biff POP process finished.

'`mew-scan-sentinel-hook`'
: Hook called when scan finished.

'`mew-summary-ls-no-scan-hook`'
: Hook called when mew-summary-ls doesn't scan a folder.

'`mew-summary-exec-hook`'
: Hook called when mew-summary-exec finished.

'`mew-summary-toggle-disp-msg-hook`'
: Hook called when mew-summary-toggle-disp-msg finished.

'`mew-syntax-format-hook`'
: Hook called when mew-syntax-format is called.

'`mew-addrbook-mode-hook`'
: Hook called in Addrbook mode.

'`mew-cite-hook`'
: Hook for an external cite mechanism. If you want to use super-cite, (setq mew-cite-hook 'sc-cite-original).

'`mew-before-cite-hook`'
: Called in mew-summary-reply-with-citation before citation.

9.5 Receiving messages

Mew uses POP to retrieve messages. This section describes following variables to control POP.

- mew-pop-delete
- mew-pop-size
- mew-pop-body-lines
- mew-pop-ssh-server
- mew-pop-user
- mew-pop-auth
- mew-pop-auth-list
- mew-use-biff
- mew-use-biff-bell
- mew-pop-biff-interval

xxx to be written

9.6 Sending messages

Mew uses SMTP to send messages. This section describes following variables to control SMTP.

- mew-smtp-ssh-server
- mew-smtp-auth
- mew-smtp-auth-list
- mew-smtp-user
- mew-smtp-helo-domain
- mew-smtp-mail-from
- mew-smtp-msgid-user
- mew-smtp-msgid-domain
- mew-use-8bit

```
message-id = *random*.user@smtp-msgid-domain
```

xxx to be written

9.7 Changing receiving/sending behavior

With '`mew-config-alist`', you can differ actions of sending/receiving messages. Let's look at the following example:

```
(setq mew-config-alist
      '(("home"
         ("inbox-folder"   . "+inbox-home")
         ("pop-server"     . "pop.iij4u.or.jp"))
        ("ext"
         ("pop-ssh-server" . "ssh.mew.org"))
        ("default"
         ("pop-server"     . "pop.mew.org"))))
```

This example defines three cases: "home", "ext", "default". For each case, some pair of key and value are defined.

We explain the semantics of '`mew-config-alist`' with this example. If the case is "home", looking up "pop-server" results in "pop.iij4u.or.jp". If the case is "ext", looking up "pop-server" results in "pop.mew.org" since there is no specified key for the case and the "default" is used. If the case is "default", looking up "pop-server" results in "pop.mew.org".

If the case is "home", looking up "inbox-folder" results in "+inbox-home". If the case is "ext", looking up "inbox-folder" results in the value of '`mew-inbox-folder`' since there is no specified key for both "ext" and "default". and the "default" is used. If the case is "default", looking up "inbox-folder" also results in the value of '`mew-inbox-folder`'.

Each key which can be specified in '`mew-config-alist`' corresponds to the variable '`mew-"key"`'. The following list enumerates such keys.

```
"smtp-msgid-domain", "smtp-msgid-user", "cc", "fcc", "dcc", "reply-to",
"organization", "header-alist", "smtp-server", "smtp-port",
"smtp-ssh-server", "smtp-helo-domain", "smtp-mail-from", "smtp-user",
"smtp-auth", "smtp-auth-list", "pop-server", "pop-port",
"pop-ssh-server", "pop-user", "pop-auth", "pop-size", "pop-body-lines",
"pop-delete", "mailbox-type", "mbox-command", "mbox-command-arg",
"inbox-folder"
```

To understand what these variables mean, consult with each documentation. Also, there are special keys which don't correspond with variables. Look at the followings:

```
"user", "name", "mail-domain"
```

These keys plays a role of '`mew-"key"`' with the rule below.

```
mail-address = user@mail-domain
from = name <mail-address>
```

Cases configured in '`mew-config-alist`' can be specified to the receiving case and the sending case. When Mew boots up, both the receiving case and the sending case are "default".

If you want to change both the receiving case and the sending case, type '`C`'. You can input a case with completion.

If you set '`mew-config-synchronize`' to '`nil`', you can specify the receiving case and the sending case independently. Withi this configuration, '`C`' is for the receiving case and '`C-uC`' is for the sending case.

If one of the cases or both are not "default", they are displayed in the mode line of Summary mode. The following is an example where the receiving case is "home" and the sending case is "ext".

```
(Summary home:ext)
```

In Draft mode, if the sending case is not "default", the value is displayed in mode line. The following is an example that the sending case is "home".

```
    (Draft home)
```

To change the sending case of a draft in Draft mode, use ‘C-cC-o’. Its header is dynamically modified according to the value of the "mail-domain" key and the "header-alist" key. Note that the sending case is a local variable of the Draft mode.

The following is a summary of commands concerned with case.

‘C’	In Summary mode, set the receiving case or both cases. If ‘mew-config-synchronize’ is ‘t’ (this is default), set both cases. If ‘nil’, set the receiving case only.
‘C-uC’	In Summary mode, set the sending case or both cases. If ‘mew-config-synchronize’ is ‘t’, set both cases. If ‘nil’, set the sending case only.
‘C-cC-o’	In Draft mode, set the sending case of a draft.

10 Life with icons

If you use Mew on XEmacs, you can read and/or write messages with icon-based interface. The icon-based interface was designed totally equivalent to the original key-based interface.

So, how to use the icon-based interface? I believe that it is intuitive enough that no future explanation is necessary. Nonetheless, some basic rules are shown below.

To execute a function bounded to each BASIC icon in Summary, Virtual, and Draft mode toolbar, click the left button of your mouse.

For multipart messages, multipart icons appears in toolbars. Clicking the left button on a multipart icon visualizes the part. When you press the multipart icon, a popup menu appears. Thanks to this menu, many operations can be applied to the part.

By default, multipart icons are displayed at the right size of the basic icons in toolbars. If you like the left size, configure as follows:

```
(setq mew-multipart-icon-position 'left)
```

11 Email convention

When you exchange messages with other people, you should obey minimum manner. It gives a bad impression to the people if you violate the manner since such messages are hard to read. We always should try to write concise yet comprehensive explanations and make an effort to make receivers well-understood.

We should take good care of the following items.

Fill To: and Cc: fields precisely
: To: is for target receivers and Cc: is for those who are received for their information. If his address is not on To:, he might skip to read the message. We should take care not to deliver to wrong people by mistaking the addresses.

The number of addresses To: and Cc: should be small
: It is discouraged to specify a lot of mail addresses on To: and Cc:. You should create a mailing-list instead.

Write a short and clear summary of body in Subject:
: There are people who decide to read contents from their Subject:. So, they might skip messages with improper Subject:. We should not write a long subject since it is hard to read.

Carriage return means end of line and null line expresses end of paragraph. One line should be limited to 70 characters or so.
: It is very hard to read if you write a message on every other line or in a long line without folding. Especially long lines are trouble to cite. Some people write a message with indent but it is meaningless. Text is displayed differently on each machines, so even if the layout is excellent on your machine, it might not be so on other machines.

Cite only necessary sentences
: You should not bother to remove unnecessary sentences. With Mew, citation must be a piece of cake.

Make your signature simple
: A long signature is just self-satisfaction.

Do not send prank messages
: I don't want to warn this kind of stuff. Nonetheless, someone sends Happy or Unhappy messages to others. You should understand that people doubt your character.

Attach data files that the receivers can read
: It is text only that you can send without any agreement with the receivers. If you want to send data files other than text, you should make an agreement with the receivers. To mailing-list, you should send text only.

If you wish to learn manner on the Internet more comprehensively, please refer to RFC1855(See Chapter 22 [Bib], page 62).

12 What's MIME?

Messages so far, more exactly RFC822 messages, cannot contain objects other than text. MIME is multi-purpose message to extend RFC822.

MIME has

```
MIME-Version: 1.0
```

field in its header. Without this field, it is an RFC822 message. In MIME, Content-Type: to indicate data type and Content-Transfer-Encoding to specify encoding are important fields. The following sections describe these fields and feature of MIME.

12.1 Labeling data type

With MIME, data type can be specified in Content-Type:(CT:) field. The following is an example message whose body is US-ASCII text.

```
MIME-Version: 1.0
Content-Type: Text/Plain; charset=us-ascii
Content-Transfer-Encoding: 7bit
Subject: hello
From: Kazu

Hi all,
```

If CT: is omitted, the content is treated as "Text/Plain; charset=us-ascii". And if CT: is "Text/Plain" and charset is not specified, its charset is considered as US-ASCII.

Likewise, if CT: is text, charset can be specified in the context of MIME. For Japanese, ISO-2022-JP is used.

MIME can embed multiple objects in its body, so called multipart. Each part in multipart consists of content-header and content-body. CT: appears in content-header as well as header. In the contrary, you can take header as a special type of content-header.

For more information, please refer to See Section 12.3 [mime-multi], page 47.

Important CT: is listed below.

'Text/Plain'	Text
'Message/Rfc822'	Message including MIME which has a header and a body
'Multipart/Mixed'	Multipart
'Application/Postscript'	PostScript
'Application/Octet-stream'	Binary stream. Can be considered as a binary file
'Image/Gif'	GIF
'Image/Jpeg'	JPEG
'Audio/Basic'	Audio file with AU format

'`Video/Mpeg`'
MPEG

'`Message/External-body`'
An phantom object whose real object exists outside of the message

12.2 Encoding for transport-safe

"uuencode" has been used for a long time to transport binary. It encodes three 8-bit characters into four 6-bit characters, however, the result contains many kinds of symbols. Some of them have special meanings in header so they cannot be used to extent header functionality.

Space character bothers the transport system. Space character cannot exist in end of line of the file system of BITNET. Suppose that an encoded object with uuencode contains space character in end of line. When a message gateway BITNET received this kind of message, it removes the space character, of course. In the result, receivers cannot decode and extract the original object.

MIME specified 2 encoding methods for body.

Base64 encoding
Encode three 8-bit characters into four 6-bit characters with 64 letters, "0-9A-Za-z/+". PEM originates it.

Quoted-Printable encoding
Represent non-printable characters in hexagonal preceded by "=".

Encoding is specified by Content-Transfer-Encoding:(CTE:) in content-header. The candidate values are as follows:

7bit	No encoding is applied. The content consists of 7 bit lines.
8bit	No encoding is applied. The content consists of 8 bit lines.
binary	No encoding is applied. The content is 8 bit stream.
base64	Encoded with Base64. The content consists of 7 bit lines.
quoted-printable	Encoded with Quoted-Printable. The content consists of 7 bit lines.

If CTE: is omitted, it is treated as '`7bit`'.

Since ISO-2022-JP is 7bit character set, CTE: is 7bit. That it, CTE: can be omitted. You may encode it with base64 or quoted-printable, of course. However, you cannot read messages in folder directly with such a encoding, I don't recommend.

12.3 Multipart structure

If CT: is multipart, its content-body has multiple objects. They are separated with a string specified in the "boundary" parameter. Let's look at an example.

```
Date: Mon, 27 Mar 2000 10:02:35 +0900 (JST)
Message-Id: <20000327.100235.104031489.kazu@iijlab.net>
To: sumikawa@ebina.hitachi.co.jp
Subject: image of cats
From: Kazu Yamamoto (=?iso-2022-jp?B?GyRCOzNLXE9CSScbKEI=?=)
 <kazu@iijlab.net>
X-Mailer:Mew version 1.95 on Emacs 20.7
Mime-Version: 1.0
Content-Type: Multipart/Mixed; boundary="simple"
Content-Transfer-Encoding: 7bit
```

```
--simple
Content-Type: Text/Plain; charset=us-ascii
Content-Transfer-Encoding: 7bit

Here is image of cats. Enjoy!

--Kazu

--simple
Content-Type: Image/Png
Content-Transfer-Encoding: base64
Content-Description: "pretty cats"
Content-Disposition: attachment; filename="cats.png"

iVBORw0KGgoAAAANSUhEUgAAAksAAAHPCAMAAAB0qm5jAAAABGdBTUEAALGPC/xhBQAAAv1Q
KigtpJpW6tByWWZLzbBhgVg+/eyJtbOVyIhIo3hAxtSO5KlWd4FwR0gxvIRwgaGPmnBoWD8v
87pY15xK6cadkmlAxKuPyF5SV4Re2Oe5VZCMsIBK9spitKhjTVs93sFyoKyX3KZs5vLTfpJk
xsKlS2hlkWJcflRNz5dqNTwzx9HJ67h2aE9B1Luf896Sr35g7/nluJpeMktQZnJ39cWB5K9y
pJCA1+HZ1JBQ8dd3o3FapMmm5dStrKqCRF5jhJB9NDpBpnx0sVdLyMmPa1xajntbzZWBgm9P
aE5QzLOX8I5Z7dq6iWBG67Be6OvjiFxU7olk9cJd+9Jx/9aPeGVF8Ou2+dWv/q1zv3dbrJeJ

--simple--
```

In this case, a string "simple" is used. A string specified in the "boundary" parameter is preceded with "–". The last one is also followed by "–".

Each part consists of a content-header and a content-body. They are separated with a null line as header and body. Changing a point of view, header and body is special content-header and special content-body respectively.

When you send objects other than text, you should use multipart. Of course, it is not illegal to contain, for example, Audio/Basic in body but the receivers would be really confused. You are kind if you enclose describing text in the first part and embed Audio/Basic in the second part.

Multipart can take recursive. So, you can enjoy multipart of multipart.

By the way, preceding CRLF is included in a boundary. For example up above, the boundary is "CRLF–simpleCRLF".

12.4 Header extensions

Header contains information used for transportation, so it should be strongly prohibited to insert improper characters that make transport agents misoperated. With MIME, non-ASCII characters are encoded into transport-safe characters then stored as a field value with the following format.

```
=?<charset>?<encoding>?<encoded-string>?=
```

<charset> is identical to the charset parameter of CT: Text/Plain. For <encoding>, '`B`' or '`Q`' is used. The former is exactly base64 and the latter is a kind of Quoted-Printable.

For ISO-2022-JP, '`B`' is encouraged. '`Q`' is also acceptable, however, few message interfaces support it(of course, Mew does).

For instance, the author's Japanese name in Subject: is encoded as follows:

```
Subject: =?ISO-2022-JP?B?GyRCOzNLXE9CSScbKEI=?=
```

It is not parameter values but field values that this format can handle. One of the reasons why this format must not be applied to parameter values is that the "=" keyword conflicts the separator between parameter names and parameter values. To encode non-ASCII characters in a parameter value, another format should be used. Please see the following example to understand the differences:

```
Content-Disposition: attachment;
 filename*=iso-2022-jp''%1B%24BF%7CK%5C8l%24N%25U%25%21%25%24%25k%1B%28B
```

13 Ahhh, Kanji code

"Kanji" is main characters for Japanese which typically have meanings and two sounds. The total number of Kanji usually used is over 3,000. Kanji was originated from Chinese characters and had been modified and simplified in Japan for a long time.

There are also about 80 characters, so called "Hiragana", each expresses just one sound and has a soft shape. Kanji is mainly used for nouns and beginning portion of verbs while Hiragana is used for other parts including last portion of verbs. Japanese sentences typically consist of Kanji in 30% and Hiragana in 70%. There is one more character set, called "Katakana", which is another notation of Hiragana. Katakana has exactly same sound of Hiragana and a little hard shape and is used to express exported words from other countries based on their sounds.

I describe an example of struggle history for non-alphabetical character set in messaging system.

13.1 Email and localization

A spec of Email, RFC822, was defined with a hope to ensure interoperability in 1982. Since Email was grown in America, its header and body could not contains other character sets than US-ASCII.

It is, however, very inconvenient for people whose language is not English. So, despite of extension of header, many people from various countries extended RFC822 messages to contain non-English characters from their native language.

In Europe, Latin 1 started to be used that presents umlaut(accent) characters by 8 bit word. Latin 1 is sometime called ISO-8859-1.

In Japan, there are three major codes, (1) JIS code which is 7-bit 2 characters, (2) EUC code which is 8-bit 2 characters and is used in UNIX, (3) Shift_JIS which is 8-bit 2 characters and is used in PCs. Pioneers of JUNET which is the antecedent of Japanese Internet chose a switch mechanism of ASCII and JIS with ESC sequence, so called JUNET code, for transportation.

JUNET code is sometime called ISO-2022-JP. With JUNET code, we can tell what their character sets are in addition to switch them.

The extension such as Latin 1 and JUNET code is an agreement within the region. You are compelled to use English to send a message across regions in the context of RFC822.

RFC822 is so ambiguous that we misunderstand that JUNET code can be used for header and body since it is 7 bit. Probably this is a good explanation to blow away your misunderstanding. "RFC822 defines that the syntax of header and body is 7bit and the semantics of header and body is US-ASCII". JUNET code is syntactically legal but its semantics is illegal.

13.2 The appearance of MIME

To satisfy users' desire such as transportation of picture and audio and to bridge localized RFC822, MIME was defined in 1992. With MIME, the character parameter can be specified. Since JUNET code is called ISO-2022-JP, Japanese message looks as follows:

```
Content-Type: Text/Plain; charset=iso-2022-jp
Content-Transfer-Encoding: 7bit

Japanese text.
```

Is this charset useful? Absolutely! The charset parameter can tell user interface an exact character set. Suppose that a Norway guy send a message with ISO-8859-1 to Japanese. If his interface supports ISO-8859-1 then it is no problem to display the body. Otherwise the interface can safely ignore the body. Mew makes use of the charset parameter to convert messages to Mule's internal representation.

Some people say like this; "If we use ISO-2022-JP-2 which is upper compatible to ISO-2022-JP and can handle numerous character set, the charset parameter is not necessary since ISO-2022-JP-2 itself contains information about character set." Maybe, just maybe, such people don't understand MIME.

Ideally speaking, this assessment is correct. But MIME takes a practical stance. MIME does not suppose that all people in the world will start using ISO-2022-JP-2 tomorrow. Moreover, MIME is designed to be robust against unstable transfer programs. All transfer program in the world are not well implemented. And all site cannot use rich resources. If you ask to use UNICODE as of today, how do you feel?

MIME provides the charset parameter to bridge between numerous localized regions. The additional procedure under MIME is to label the charset parameter and we can use ISO-2022-JP as we used to. If you wish to make ISO-2022-JP-2 an Internet standard, you should make an effort to spread region where ISO-2022-JP-2 is used by default. Likewise, ISO-2022-JP-2 and MIME is not inconsistent. Rather, ISO-2022-JP-2 can make most use of MIME to make itself widely spread. Of course, the name of "charset" is not proper for character switching mechanisms such as ISO-2022-xx.

With MIME, you can encode non-ASCII character set and insert it into header. This scheme prevents errors of Email transfer programs and makes it possible to convey non-ASCII strings in header. We don't have to say "Do not use Japanese on Subject:" anymore!

MIME is not a spec to prohibit localized RFC822. So, MIME interfaces are supposed to act as follows:

Viewing

1. Allow user to choose a default charset.
2. If MIME-Version: doesn't exist, the body is treated as the default charset.
3. If MIME-Version: exists and Content-Type: is not provided, the body is treated as US-ASCII.
4. If both MIME-Version: and Content-Type: exist, specified charset is used.

Composing

1. Specify MIME-Version:, Content-Type:, and its charset.
2. Choose minimum character set for charset. For instance, US-ASCII for English. If the rule is violated, it is likely that a message to be read cannot be read. For instance, consider a message labeled ISO-2022-JP whose body is US-ASCII in fact. Mailers which support US-ASCII only could not handle such a message.

If you store messages in a spool or folders after conversion from ISO-2022-JP to EUC-JP, please don't be blind. You should check charset out, and convert only ISO-2022-JP messages to EUC-JP.

Insertion of non-ASCII in header is one of MIME features but in fact MIME-Version: is not necessary.

13.3 The concept of canonicalization

Unfortunately, each computer in the world represents data with its own format. The followings are end of lines used in major OSes.

– UNIX :: LF(0x0a)
– MS-DOS :: CRLF(0x0d0a)
– MacOS :: CR(0x0d)

As you know, if there is no agreement for end of line, text is not transfered between these OSes safely. RFC822 defines to transform end of line into CRLF. This kind of format conversion is called canonicalization. Converting Shift_JIS and EUC-JP to JUNET code is a kind of canonicalization.

OK, let's think about encryption and signature with PGP. Suppose that a Mac user signed text whose line breaks are CR then sent it to a UNIX user. If the UNIX user transforms line break to

LF then verifies the signature, it is obvious that the verification fails. You thus understand that canonicalization is necessary.

When you encrypt or sign text with PGP, first convert it to ISO-2022-JP then transform its end of lines into CRLF.

14 Mew's policy

The following words mention Mew's spirit:

> Mew wants something simple or nothing at all.

That is, complicated mechanism is not necessary. What Mew provides is simple yet comprehensive feature.

Many people tell me like this: "A mailer that I used before has this kind of feature and I think it's very convenient. So, please implement it in Mew". Sorry, such a explanation might not convince me. Mew is trying to carry out stuff that other programs have not ever done so. The word "good-old" seems to me not enough.

If you believe in your opinion, please pursue the author patiently. Since I'm hell-of-a busy person I'm prone to forget what you explained before. And because I'm just a man I regret to say I'm subjective. It is quite possible for me not to understand your novel ideas. So, please don't give up!

15 Where did Mew come from?

It might not be useful but let's open the book about Mew's history...

15.1 Departure from mh-e

I took part in the FJPEM research project of WIDE Project in the fall of 1993. At that time, Mine had implemented mhpem to make it convenient to use FJPEM with mh-e. Inspired by mhpem, I spend many time to enhance mhpem as escape from my master thesis in winter 1994. :)

mhpem was a lovely program to decrypt cipher messages automatically, however, we met some problems. The biggest one was that mh-e was not flexible enough to enhance some features. I really wanted to cache decrypted message to display it quickly for the second time but it was quite hard to enhance mh-e to do so. When a new version of mh-e was released and it appeared that mhpem didn't work with it, I decided to kick it out.

I want to cache decrypted PEM. Also MIME. Why isn't there a good interface to handle MIME with easy operations? Tell me the reason not to be able to cite multiple messages to one replying message. It's ridiculous that we cannot enjoy cheerful marks. I'd hate to refile messages with difficult operations. All in all, it appeared impossible for me to implement what I want by modifying mh-e.

Auto decryption in mhpem, asynchronous scan in mhasync, dynamic window setup in GNUS, cheerful mark system for multiple messages in gnus-mark, refile feature in VM, message cache mechanism, and beautiful and flexible programming style.... A pieces of puzzle was gradually getting together in my mind. It was early in the spring of 1994.

15.2 Birth of Mew

In April 1994, I started to program Mew. Since there were references, inc and scan not to wait the exit, dynamic window configuration, marks, and message cache were implemented in early time.

To tell the truth, Mew displayed MIME messages in MIME mode. When you type '`SPC`' on a multipart MIME message, Mew moved from Summary mode to MIME mode. Utashiro said to me, however, "Why MIME mode? Summary mode is enough, isn't it?". It's a really breakthrough.

At this time, big problems were how to refile messages and to compose complex MIME.

It is certain that if users are always forced to write Lisp, message refile becomes easy. But I really hate to require users to setup Lisp by themselves. "It is ridiculous to try to choose default refile-folder from message header since there are thousands of candidates. Rather, it is quite reasonable to select the default from existing folder". When this idea hit my head, I could not wait for the next morning. After that, refile feature was enhanced by Nomura.

How to compose complicated MIME message with easy operations? It is a tragedy for users to be compelled to learn a composition grammar. I really wanted to provide simple yet comprehensive method to users. It was Youki that gave me the answer. Yes, MIME is file structure! We can consider that a singlepart is a file while a multipart is a directory. Users can create a file tree without any troubles, of course, and it is a Mew's job to convert the file structure to MIME format.

15.3 Meet to PGP

I couldn't swipe out reservations to the implementation of FJPEM for a long time, then PGP finally appeared in front of me in the early summer of 1994. At that time, MIT tried to find a solution to make non-commercial PGP legal against the RSA patent, so called PGP 2.5. The release of PGP 2.6 settled down the relationship between Phil and RSADSI, yet PGP 2.6 based on RSAREF was under US export control.

Meet to 2.6ui based on 2.3a finally leaded me to be a PGP user. I cannot forget my first impression to PGP. It's a really well-designed program. In the summer of 1995, I took part in

NDSS, so called ISOC Security Symposium in San Diego. In addition of the proceedings, PGP by Simson published from O'Reilly was delivered to early registered participants. It's a masterpiece of masterpieces. For these reasons, I have spent time to integrate PGP and MIME rather than to integrate PEM and MIME.

The idea that marks can represent encryption and signature was given by Mine when we wrote a paper for IPSJ together. It took a time a bit to understand its merit that marks could be canceled at any time.

15.4 Independence from MH

"Get together in Fukuoka for good sushi!" Under these words, excellent programmers came to Momochi, Fukuoka, where Fukuoka dome locates, in April, 1997. Using Institute of Systems & Information Technologies, Kyushu as a hacking room and Hyatt Regency as sleeping rooms, MH independence project for 3 nights started.

Eating anyway, hacking as we like, and discussing comprehensively. We repeated these steps over and over. As we left Fukuoka, Mew became somehow independent on MH by using IM instead. (Engel's coefficient of this week is, of course, very high...)

Afterwords, we discussed on mailing list, tried to achieve consensus, and then hacked again and again. Finally master Utashiro casted a spell to make IM much faster then IM beta version was released in early July.

15.5 Integration of NetNews

In the fall of 1994, I held Mew BOF(Birds Of a Feather) at WIDE camp. Mr. Sano left unforgettable words to me: "Mailing-list is some between Email and NetNews". It's like I saw a light in darkness. Now that Perl 5 become stable, the integration of NetNews is coming soon.

16 Where will Mew go?

Here are items which is planed in Mew 2.

IMAP
: We are planning to support IMAP.

Database for full body search
: Mew 2 will support database for full body search (WAIS for English?).

Database for message relations
: Mew 2 will support database for message relations. Thanks to this, thread can be implemented. Also, it will be possible to manage unread messages. Possibly refiling messages will be unnecessary.

NetNews
: Mew 2 will support NetNews.

17 Availability and mailing-list

This chapter describes how to get Mew and related mailing-lists.

17.1 How to get Mew?

The latest Mew is available from the following repository.

```
ftp://ftp.Mew.org/pub/Mew/mew-current.tar.gz
```

Samples are given time to time under the following file name:

```
ftp://ftp.Mew.org/pub/Mew/samples.tar.gz
```

17.2 Mailing list

A new version is announced to

```
mew-release@Mew.org
```

in English. If you wish to join, please send a message to

```
mew-release-ctl@Mew.org
```

whose the first line of its body is "#help" (without quotes). This list is under access control so that nobody but the author can post.

18 Acknowledgement

First and foremost, I would like to thank Mine Sakurai. If she had not have implemented mhpem and inspired me, Mew would not be realistic. She also implemented most part of mew-url and mew-pem(that are obsolete, though). It is she that gives me an idea of mark based composing. To her goes my deepest gratitude.

The author of Mew would sincerely like to acknowledge Kazumasa Utashiro for telling me simplicity of Summary mode, Youki Kadobayashi for noticing me a scheme to map file structures to MIME, Motonori Nakamura for his precious suggestions regarding with Email, and Susumu Sano for giving me a hint to integrate Email and NetNews.

Kiyotaka Sakai always contributes neat codes. Yoshinari Nomura enhanced functionality of message refile. Mikitomo Ushijima always proofreads my poor English. Keiichi Shima encouraged me to write Info. I very much appreciate.

I would like to thank David Worenklein for contributing many codes. I'm grateful to Atsushi Shionozaki, Darren Stalder, and David Worenklein for proofreading my paper. I'd like to acknowledge to Scandinavia guys for their good suggestions.

My deep gratitude goes to Jun'ichiro Ito, Yoshiaki Kasahara, Motonori Nakamura, Yoshinari Nomura, Kazunori Nishi, and Hidenori Ohta for their help to create IM. My thank also goes to Kazumasa Utashiro for making IM much faster. I would acknowledge to Fujikazu Okunishi for his contribution on OS/2 area and his comprehensive tests. Significant works to run Mew on Win95/WNT were mostly accomplished by Shuichi Kitaguchi and Shuhei Yamaguchi. Yuuichi Teranishi made Mew funny on XEmacs.

I would like to express my thanks to devoted users: Hidetaka Koie, Hideyuki Shirai, and Shun-ichi Goto.

Mew is supported by contributions from many users. I'd like to express hearty thanks to those who are not listed here. The author is hell-of-a busy guy, so he cannot afford to respond all Emails. I'm so sorry if you have not received my reply.

19 Copyright

Mew conforms the following copyright.

If you wish to distribute Mew in CD or something, please let me know. I do not go mad even if you don't tell me but I'm very grad if you do so. I have not declined requirements.

The copyright of this Info belongs to the author. It is granted to copy, modify, redistribute this Info but no warranty.

20 About the author

Kazuhiko YAMAMOTO // Kazu

In 1970, Kazu was born at Hikari city, Yamaguchi prefecture, Japan, where is surrounded by a beach with "Rainbow pine forest" in the south, by an abundant river in the east, by gentle mountains in the north and the west. When he was a junior high-school student, he watched the movie "War Game", which made him interested in computer security. He had been grown with beautiful nature at Hikari city until high school.

He was moved to Fukuoka city to take a course of electronic engineering at Kyushu university in 1988. Despite of pain from the urban city, the dirty ocean, and the language barrier, he somehow graduated from the university. Email appeared to him in the third year and the Internet fascinated him through the fourth. The trial to be a employee of Sun Microsystems to visit the US failed because US' visa policy became severe at that time.

He took a course of Computer Science and Communication Engineering at the graduate school of Kyushu university in 1992. The specialty was Internet routing. He started to make security knowledge and technology widely spread to the Japan portion of the Internet community. He finally released "Happy Networking", one of the most popular beginner's guide of the Internet, in the spring of 1993.

He moved to Nara Institute of Science and Technology as a research associate in 1994. During the four years of faculty life, he developed Mew. Since he reached a conclusion that he should concentrate on programming in young days and it's possible to educate students when he will became older, he moved to IIJ Research Laboratory in 1998.

Research areas: Messaging system, IP version 6
Favorite words: "Challenges start everyday."
Favorite words: "Living is learning. Giving is taking."

Email: Kazu@Mew.org
URL: http://www.mew.org/~kazu/
PGP fingerprint: 6B 63 38 88 67 5E 96 8E CE A4 62 73 3F 11 64 94

21 Terminology

Here is terminology for Mew.

‘`folder`’	Directories to save received messages.
‘`Summary mode`’	A mode to display a list of messages.
‘`Message mode`’	A mode to display a content of a text message.
‘`Draft mode`’	A mode to write or compose a message.
‘`MIME`’	A format to contain objects other than text in a body and to embed non-alphabetical words in a header. With MIME you can enclose text files, picture files and audio files at the same time, and insert your first language such as Japanese into Subject:. It is an acronym of "Multipurpose Internet Mail Extensions". For more information, please refer to See Chapter 12 [MIME], page 46.
‘`PGP`’	A program to accomplish encrypted message and digital signature created by Phil Zimmermann. An acronym for "Pretty Good Privacy".
‘`composer`’	An interface to write or compose a message. Regarding with Mew, it is Draft mode.
‘`viewer`’	An interface to read or view messages. As far as Mew is concerned, it is Summary mode and Message mode.
‘`Email`’	A short word for "electronic mail".
‘`NetNews`’	It was also known as "USENET news".
‘`message`’	An integrated concept of Email, NetNews, MIME, etc. A short word of "Internet message".

22 Bibliography

This section shows you bibliography. RFC(Request For Comments) is a series of documents to share our knowledge about the Internet Protocol suite. It is available, for instance, from the following repository.

`ftp://ftp.isi.edu/in-notes/`

Please read the following RFC to learn manner on the Internet.

- S. Hambridge, "Netiquette Guidelines", RFC 1855, 1995

To understand the format of good old text Email and the delivery system, please refer to:

- D. Crocker, "Standard for the format of ARPA Internet text messages", RFC 822, 1982
- J. Postel, "Simple Mail Transfer Protocol", RFC 821, 1982

MIME is specified in the following RFCs.

- N. Freed and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies", RFC 2045, 1996.
- N. Freed and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types" RFC 2046, 1996.
- K. Moore, "MIME (Multipurpose Internet Mail Extensions) Part Three: Message Header Extensions for Non-ASCII Text", RFC 2047, 1996.
- N. Freed, J. Klensin and J. Postel, "Multipurpose Internet Mail Extensions (MIME) Part Four: Registration Procedures", RFC 2048, 1996.
- N. Freed and N. Borenstein, "Multipurpose Internet Mail Extensions (MIME) Part Five: Conformance Criteria and Examples", RFC 2049, 1996.
- R. Troost, S. Dorner, K. Moore, "Communicating Presentation Information in Internet Messages: The Content-Disposition Header Field", RFC 2183, 1997.
- N. Free and K. Moore, "MIME Parameter Value and Encoded Word Extensions: Character Sets, Languages, and Continuations", RFC 2231, 1997.

You can find the format and distribution protocol of NetNews in the following RFCs.

- M. Horton and R. Adams, "Standard for interchange of USENET messages", RFC 1036, 1987.
- B. Kantor and P. Lapsley, "Network News Transfer Protocol: A Proposed Standard for the Stream-Based Transmission of News", RFC 977, 1986.

The following document contains concise yet comprehensive explanations about PGP.

- PGP: Petty Good Privacy, Simson Garfinkel, O'Reilly & Associates, Inc, 1995.

For details of an integration of PGP and MIME, please refer to:

- J. Galvin, S. Murphy, S. Crocker and N. Freed, "Security Multiparts for MIME: Multipart/Signed and Multipart/Encrypted", RFC 1847, 1995.
- M. Elkins, "MIME Security with Pretty Good Privacy (PGP)", RFC 2015, 1996.

23 Variable Index

Short Contents

Table of Contents