NAME

Mail::Box::File − handle file−based folders

INHERITANCE

```
 Mail::Box::File
   is a Mail::Box
   is a Mail::Reporter

 Mail::Box::File is extended by
   Mail::Box::Dbx
   Mail::Box::Mbox
```

SYNOPSIS

DESCRIPTION

`Mail::Box::File` is the base-class for all file-based folders: folders which bundle multiple messages into one single file. Usually, these messages are separated by a special line which indicates the start of the next one.

Extends “DESCRIPTION” in Mail::Box.

OVERLOADED

Extends “OVERLOADED” in Mail::Box.

overload: **“”**
Inherited, see “OVERLOADED” in Mail::Box

overload: **@{}**
Inherited, see “OVERLOADED” in Mail::Box

overload: **cmp**
Inherited, see “OVERLOADED” in Mail::Box

METHODS

Extends “METHODS” in Mail::Box.

Constructors

Extends “Constructors” in Mail::Box.

Mail::Box::File−>**new**(%options)

```
 −Option            −−Defined in     −−Default
  access              Mail::Box        'r'
  body_delayed_type   Mail::Box        Mail::Message::Body::Delayed
  body_type                            <see description>
  coerce_options      Mail::Box        []
  create              Mail::Box        <false>
  extract             Mail::Box        10240
  field_type          Mail::Box        undef
  fix_headers         Mail::Box        <false>
  folder              Mail::Box        $ENV{MAIL}
  folderdir           Mail::Box        $ENV{HOME}.'/Mail'
  head_delayed_type   Mail::Box        Mail::Message::Head::Delayed
  head_type           Mail::Box        Mail::Message::Head::Complete
  keep_dups           Mail::Box        <false>
  lock_extension                       '.lock'
  lock_file           Mail::Box        <foldername><lock−extension>
  lock_timeout        Mail::Box        1 hour
  lock_type           Mail::Box        Mail::Box::Locker::DotLock
  lock_wait           Mail::Box        10 seconds
  locker              Mail::Box        undef
  log                 Mail::Reporter   'WARNINGS'
  manager            Mail::Box        undef
  message_type       Mail::Box        Mail::Box::File::Message
  multipart_type     Mail::Box        Mail::Message::Body::Multipart
  remove_when_empty  Mail::Box        <true>
  save_on_exit       Mail::Box        <true>
  trace              Mail::Reporter   'WARNINGS'
  trusted            Mail::Box        <depends on folder location>
  write_policy                        undef
```

access => MODE
body_delayed_type => CLASS
body_type => CLASS|CODE

The default `body_type` option for `File` folders, which will cause messages larger than 10kB to be stored in files and smaller files in memory, is implemented like this:

```
 sub determine_body_type($$)
 {   my $head = shift;
     my $size = shift || 0;
     'Mail::Message::Body::'
        . ($size > 10000 ? 'File' : 'Lines');
 }
```

coerce_options => ARRAY
create => BOOLEAN
extract => INTEGER | CODE | METHOD | ’LAZY’|’ALWAYS’
field_type => CLASS
fix_headers => BOOLEAN
folder => FOLDERNAME
folderdir => DIRECTORY
head_delayed_type => CLASS
head_type => CLASS
keep_dups => BOOLEAN
lock_extension => FILENAME|STRING

When the dotlock locking mechanism is used, the lock is created with a hardlink to the folder file. For `Mail::Box::File` type of folders, this file is by default named as the folder-file itself followed by `.lock`. For example: the `Mail/inbox` folder file will have a hardlink made as `Mail/inbox.lock`.

You may specify an absolute filename, a relative (to the folder’s directory) filename, or an extension (preceded by a dot). So valid examples are:

```
 .lock        # appended to the folder's filename
 my_own_lockfile.test   # full filename, same dir
 /etc/passwd            # somewhere else
```

When the program runs with less privileges (as normal user), often the default inbox folder can not be locked with the lockfile name which is produced by default.

lock_file => FILENAME
lock_timeout => SECONDS
lock_type => CLASS|STRING|ARRAY
lock_wait => SECONDS
locker => OBJECT
log => LEVEL
manager => MANAGER
message_type => CLASS

multipart_type => CLASS
remove_when_empty => BOOLEAN
save_on_exit => BOOLEAN
trace => LEVEL
trusted => BOOLEAN
write_policy => 'REPLACE'|'INPLACE'|undef
Sets the default write policy, as default for a later call to write(policy). With `undef`, the best policy is autodetected.

The folder

Extends ''The folder'' in Mail::Box.

`$obj->`**addMessage**($message, `%options`)
Inherited, see ''The folder'' in Mail::Box

`$obj->`**addMessages**(@messages)
Inherited, see ''The folder'' in Mail::Box

Mail::Box::File−>**appendMessages**(%options)
Appending messages to a file based folder which is not opened is a little risky. In practice, this is often done without locking the folder. So, another application may write to the folder at the same time... :(Hopefully, all goes fast enough that the chance on collision is small.

All `%options` of **Mail::Box::Mbox::new()** can be supplied.

```
 -Option   --Defined in     --Default
  folder     Mail::Box        <required>
  lock_type                   NONE
  message    Mail::Box        undef
  messages   Mail::Box        undef
  share      Mail::Box        <false>
```

folder => FOLDERNAME
lock_type => ...
See Mail::Box::new(lock_type) for possible values.

message => MESSAGE
messages => ARRAY-OF-MESSAGES
share => BOOLEAN

`$obj->`**close**(%options)
Inherited, see ''The folder'' in Mail::Box

`$obj->`**copyTo**($folder, `%options`)
Inherited, see ''The folder'' in Mail::Box

`$obj->`**delete**(%options)
Inherited, see ''The folder'' in Mail::Box

`$obj->`**filename**()
Returns the filename for this folder, which may be an absolute or relative path to the file.

example:

```
 print $folder->filename;
```

`$obj->`**folderdir**([$directory])
Inherited, see ''The folder'' in Mail::Box

`$obj->`**name**()
Inherited, see ''The folder'' in Mail::Box

`$obj->`**organization**()
Inherited, see ''The folder'' in Mail::Box

`$obj−>`**size**()
Inherited, see ‘‘The folder’’ in Mail::Box

`$obj−>`**type**()
Inherited, see ‘‘The folder’’ in Mail::Box

`$obj−>`**update**(%options)
Inherited, see ‘‘The folder’’ in Mail::Box

`$obj−>`**url**()
Inherited, see ‘‘The folder’’ in Mail::Box

Folder flags

Extends ‘‘Folder flags’’ in Mail::Box.

`$obj−>`**access**()
Inherited, see ‘‘Folder flags’’ in Mail::Box

`$obj−>`**isModified**()
Inherited, see ‘‘Folder flags’’ in Mail::Box

`$obj−>`**modified**([BOOLEAN])
Inherited, see ‘‘Folder flags’’ in Mail::Box

`$obj−>`**writable**()
Inherited, see ‘‘Folder flags’’ in Mail::Box

The messages

Extends ‘‘The messages’’ in Mail::Box.

`$obj−>`**current**([$number|$message|$message_id])
Inherited, see ‘‘The messages’’ in Mail::Box

`$obj−>`**find**($message_id)
Inherited, see ‘‘The messages’’ in Mail::Box

`$obj−>`**findFirstLabeled**(`$label`, [BOOLEAN, [$msgs]])
Inherited, see ‘‘The messages’’ in Mail::Box

`$obj−>`**message**(`$index`, [$message])
Inherited, see ‘‘The messages’’ in Mail::Box

`$obj−>`**messageId**(`$message_id`, [$message])
Inherited, see ‘‘The messages’’ in Mail::Box

`$obj−>`**messageIds**()
Inherited, see ‘‘The messages’’ in Mail::Box

`$obj−>`**messages**(<’ALL’|$range|’ACTIVE’|’DELETED’|$label| !$label|$filter>)
Inherited, see ‘‘The messages’’ in Mail::Box

`$obj−>`**nrMessages**(%options)
Inherited, see ‘‘The messages’’ in Mail::Box

`$obj−>`**scanForMessages**($message, `$message_ids`, `$timespan`, `$window`)
Inherited, see ‘‘The messages’’ in Mail::Box

Sub-folders

Extends ‘‘Sub-folders’’ in Mail::Box.

`$obj−>`**listSubFolders**(%options)
Mail::Box::File−>**listSubFolders**(%options)
Inherited, see ‘‘Sub-folders’’ in Mail::Box

`$obj−>`**nameOfSubFolder**(`$subname`, [$parentname])

Mail::Box::File−>**nameOfSubFolder**(`$subname`, [$parentname])
: Inherited, see ‘‘Sub-folders’’ in Mail::Box

`$obj`−>**openRelatedFolder**(%options)
: Inherited, see ‘‘Sub-folders’’ in Mail::Box

`$obj`−>**openSubFolder**($subname, `%options`)
: Inherited, see ‘‘Sub-folders’’ in Mail::Box

`$obj`−>**topFolderWithMessages**()
Mail::Box::File−>**topFolderWithMessages**()
: Inherited, see ‘‘Sub-folders’’ in Mail::Box

Internals

Extends ‘‘Internals’’ in Mail::Box.

`$obj`−>**coerce**($message, `%options`)
: Inherited, see ‘‘Internals’’ in Mail::Box

`$obj`−>**create**($foldername, `%options`)
Mail::Box::File−>**create**($foldername, `%options`)

```
 −Option   −−Defined in−−Default
  folderdir  Mail::Box   undef
```

folderdir => DIRECTORY

`$obj`−>**determineBodyType**($message, `$head`)
: Inherited, see ‘‘Internals’’ in Mail::Box

`$obj`−>**folderToFilename**(`$foldername`, `$folderdir`, [$subext])
Mail::Box::File−>**folderToFilename**(`$foldername`, `$folderdir`, [$subext])
: Translate a folder name into a filename, using the `$folderdir` value to replace a leading `=`. `$subext` is only used for MBOX folders.

Mail::Box::File−>**foundIn**([$foldername], `%options`)
: Inherited, see ‘‘Internals’’ in Mail::Box

`$obj`−>**lineSeparator**([<STRING|’CR’|’LF’|’CRLF’>])
: Inherited, see ‘‘Internals’’ in Mail::Box

`$obj`−>**locker**()
: Inherited, see ‘‘Internals’’ in Mail::Box

`$obj`−>**messageCreateOptions**([$type, `$config`])
: Returns a key-value list of options to be used each time a new message is read from a file. The list is preceded by the `$type` of message which has to be created.

  This data is used by **readMessages()** and **updateMessages()**. With `$type` and `$config`, a new configuration is set.

`$obj`−>**moveAwaySubFolder**($directory, `$extension`)
: The `$directory` is renamed by appending the `$extension`, which defaults to `".d"`, to make place for a folder file on that specific location. `false` is returned if this failed.

`$obj`−>**parser**()
: Create a parser for this mailbox. The parser stays alive as long as the folder is open.

`$obj`−>**read**(%options)
: Inherited, see ‘‘Internals’’ in Mail::Box

`$obj`−>**readMessages**(%options)
: Inherited, see ‘‘Internals’’ in Mail::Box

`$obj`−>**storeMessage**($message)
: Inherited, see ‘‘Internals’’ in Mail::Box

`$obj->`**toBeThreaded**($messages)
Inherited, see "Internals" in Mail::Box

`$obj->`**toBeUnthreaded**($messages)
Inherited, see "Internals" in Mail::Box

`$obj->`**updateMessages**(%options)
For file based folders, the file handle stays open until the folder is closed. Update is therefore rather simple: move to the end of the last known message, and continue reading...

`$obj->`**write**(%options)

```
 -Option      --Defined in     --Default
  force         Mail::Box        <false>
  policy                         undef
  save_deleted  Mail::Box        <false>
```

force => BOOLEAN
policy => 'REPLACE'|'INPLACE'|undef
In what way will the mail folder be updated. If not specified during the write, the value of the new(write_policy) at folder creation is taken.

Valid values:

- `REPLACE`

  First a new folder is written in the same directory as the folder which has to be updated, and then a call to move will throw away the old immediately replacing it by the new.

  Writing in `REPLACE` module is slightly optimized: messages which are not modified are copied from file to file, byte by byte. This is much faster than printing the data which is will be done for modified messages.

- `INPLACE`

  The original folder file will be opened read/write. All message which where not changed will be left untouched, until the first deleted or modified message is detected. All further messages are printed again.

- `undef`

  As default, or when `undef` is explicitly specified, first `REPLACE` mode is tried. Only when that fails, an `INPLACE` update is performed.

`INPLACE` will be much faster than `REPLACE` when applied on large folders, however requires the `truncate` function to be implemented on your operating system (at least available for recent versions of Linux, Solaris, Tru64, HPUX). It is also dangerous: when the program is interrupted during the update process, the folder is corrupted. Data may be lost.

However, in some cases it is not possible to write the folder with `REPLACE`. For instance, the usual incoming mail folder on UNIX is stored in a directory where a user can not write. Of course, the `root` and `mail` users can, but if you want to use this Perl module with permission of a normal user, you can only get it to work in `INPLACE` mode. Be warned that in this case folder locking via a lockfile is not possible as well.

save_deleted => BOOLEAN

`$obj->`**writeMessages**(%options)
Inherited, see "Internals" in Mail::Box

Other methods

Extends "Other methods" in Mail::Box.

`$obj->`**timespan2seconds**($time)

Mail::Box::File−>**timespan2seconds**($time)
Inherited, see ''Other methods'' in Mail::Box

Error handling

Extends ''Error handling'' in Mail::Box.

$obj−>**AUTOLOAD**()
Inherited, see ''Error handling'' in Mail::Reporter

$obj−>**addReport**($object)
Inherited, see ''Error handling'' in Mail::Reporter

$obj−>**defaultTrace**([$level]|[$loglevel, $tracelevel]|[$level, $callback])
Mail::Box::File−>**defaultTrace**([$level]|[$loglevel, $tracelevel]|[$level, $callback])
Inherited, see ''Error handling'' in Mail::Reporter

$obj−>**errors**()
Inherited, see ''Error handling'' in Mail::Reporter

$obj−>**log**([$level, [$strings]])
Mail::Box::File−>**log**([$level, [$strings]])
Inherited, see ''Error handling'' in Mail::Reporter

$obj−>**logPriority**($level)
Mail::Box::File−>**logPriority**($level)
Inherited, see ''Error handling'' in Mail::Reporter

$obj−>**logSettings**()
Inherited, see ''Error handling'' in Mail::Reporter

$obj−>**notImplemented**()
Inherited, see ''Error handling'' in Mail::Reporter

$obj−>**report**([$level])
Inherited, see ''Error handling'' in Mail::Reporter

$obj−>**reportAll**([$level])
Inherited, see ''Error handling'' in Mail::Reporter

$obj−>**trace**([$level])
Inherited, see ''Error handling'' in Mail::Reporter

$obj−>**warnings**()
Inherited, see ''Error handling'' in Mail::Reporter

Cleanup

Extends ''Cleanup'' in Mail::Box.

$obj−>**DESTROY**()
Inherited, see ''Cleanup'' in Mail::Box

DETAILS

File based folders

File based folders maintain a folder (a set of messages) in one single file. The advantage is that your folder has only one single name, which speeds-up access to all messages at once.

The disadvantage over directory based folder (see Mail::Box::Dir) is that you have to construct some means to keep all message apart, for instance by adding a message separator, and this will cause problems. Where access to all messages at once is faster in file based folders, access to a single message is (much) slower, because the whole folder must be read.

DETAILS

Extends ''DETAILS'' in Mail::Box.

DIAGNOSTICS

Error: Cannot append messages to folder file `$filename: $!`

Appending messages to a not-opened file-organized folder may fail when the operating system does not allow write access to the file at hand.

Error: Cannot create directory `$dir` for folder `$name`.

While creating a file-organized folder, at most one level of directories is created above it. Apparently, more levels of directories are needed, or the operating system does not allow you to create the directory.

Error: Cannot create folder file `$name: $!`

The file-organized folder file cannot be created for the indicated reason. In common cases, the operating system does not grant you write access to the directory where the folder file should be stored.

Error: Cannot get a lock on `$type` folder `$self`.

A lock is required to get access to the folder. If no locking is needed, specify the NONE lock type.

Error: Cannot move away sub-folder `$dir`
Warning: Cannot remove folder `$name` file `$filename: $!`

Writing an empty folder will usually cause that folder to be removed, which fails for the indicated reason. new(remove_when_empty)

Warning: Cannot remove folder `$name` file `$filename: $!`

Writing an empty folder will usually cause that folder to be removed, which fails for the indicated reason. new(remove_when_empty) controls whether the empty folder will removed; setting it to false (`0`) may be needed to avoid this message.

Error: Cannot replace `$filename` by `$tempname`, to update folder `$name: $!`

The replace policy wrote a new folder file to update the existing, but was unable to give the final touch: replacing the old version of the folder file for the indicated reason.

Warning: Changes not written to read-only folder `$self`.

You have opened the folder read-only −−which is the default set by new(access)−−, made modifications, and now want to close it. Set close(force) if you want to overrule the access mode, or close the folder with close(write) set to `NEVER`.

Error: Copying failed for one message.

For some reason, for instance disc full, removed by external process, or read-protection, it is impossible to copy one of the messages. Copying will proceed for the other messages.

Error: Destination folder `$name` is not writable.

The folder where the messages are copied to is not opened with write access (see new(access)). This has no relation with write permission to the folder which is controlled by your operating system.

Warning: Different messages with id `$msgid`

The message id is discovered more than once within the same folder, but the content of the message seems to be different. This should not be possible: each message must be unique.

Error: File too short to get write message `$nr` ($size, `$need`)

Mail::Box is lazy: it tries to leave messages in the folders until they are used, which saves time and memory usage. When this message appears, something is terribly wrong: some lazy message are needed for updating the folder, but they cannot be retrieved from the original file anymore. In this case, messages can be lost.

This message does appear regularly on Windows systems when using the 'replace' write policy. Please help to find the cause, probably something to do with Windows incorrectly handling multiple filehandles open in the same file.

Warning: Folder `$name` file `$filename` is write-protected.

The folder is opened writable or for appending via new(access), but the operating system does not permit writing to the file. The folder will be opened read-only.

Error: Folder `$name` not deleted: not writable.
: The folder must be opened with write access via new(access), otherwise removing it will be refused. So, you may have write-access according to the operating system, but that will not automatically mean that this `delete` method permits you to. The reverse remark is valid as well.

Error: Invalid timespan '$timespan' specified.
: The string does not follow the strict rules of the time span syntax which is permitted as parameter.

Warning: Message-id '$msgid' does not contain a domain.
: According to the RFCs, message-ids need to contain a unique random part, then an `@`, and then a domain name. This is made to avoid the creation of two messages with the same id. The warning emerges when the `@` is missing from the string.

Error: Package `$package` does not implement `$method`.
: Fatal error: the specific package (or one of its superclasses) does not implement this method where it should. This message means that some other related classes do implement this method however the class at hand does not. Probably you should investigate this and probably inform the author of the package.

Error: Unable to create subfolder `$name` of `$folder`.
: The copy includes the subfolders, but for some reason it was not possible to copy one of these. Copying will proceed for all other sub-folders.

Error: Unable to update folder `$self`.
: When a folder is to be written, both replace and inplace write policies are tried, If both fail, the whole update fails. You may see other, related, error messages to indicate the real problem.

SEE ALSO

This module is part of Mail-Box distribution version 3.009, built on August 18, 2020. Website: *http://perl.overmeer.net/CPAN/*

LICENSE