NAME

Mail::Box::Locker − manage the locking of mail folders

INHERITANCE

```
 Mail::Box::Locker
   is a Mail::Reporter

 Mail::Box::Locker is extended by
   Mail::Box::Locker::DotLock
   Mail::Box::Locker::FcntlLock
   Mail::Box::Locker::Flock
   Mail::Box::Locker::Multi
   Mail::Box::Locker::Mutt
   Mail::Box::Locker::NFS
   Mail::Box::Locker::POSIX
```

SYNOPSIS

```
 use Mail::Box::Locker;
 my $locker = new Mail::Box::Locker(folder => $folder);

 $locker−>lock;
 $locker−>isLocked;
 $locker−>hasLock;
 $locker−>unlock;

 use Mail::Box;
 my $folder = Mail::Box−>new(lock_method => 'DOTLOCK');
 print $folder−>locker−>type;
```

DESCRIPTION

Each Mail::Box will create its own `Mail::Box::Locker` object which will handle the locking for it. You can access of the object directly from the folder, as shown in the examples below.

Extends ‘‘DESCRIPTION’’ in Mail::Reporter.

METHODS

Extends ‘‘METHODS’’ in Mail::Reporter.

Constructors

Extends ‘‘Constructors’’ in Mail::Reporter.

Mail::Box::Locker−>**new**(%options)

Create a new lock. You may do this directly. However, in most cases the lock will not be separately instantiated but will be the second class in a multiple inheritance construction with a Mail::Box.

Generally the client program specifies the locking behavior through options given to the folder class.

```
 −Option −−Defined in     −−Default
  expires                   1 hour
  file                      undef
  folder                    <undef>
  log      Mail::Reporter   'WARNINGS'
  method                    'DOTLOCK'
  timeout                   10
  trace    Mail::Reporter   'WARNINGS'
```

expires => SECONDS

How long can a lock exist? If a different e−mail program leaves a stale lock, then this lock will be removed automatically after the specified number of seconds.

file => FILENAME
: Name of the file to lock. By default, the name of the folder is taken.

folder => FOLDER
: Which FOLDER is to be locked, a Mail::Box object.

log => LEVEL

method => STRING|CLASS|ARRAY
: Which kind of locking, specified as one of the following names as STRING. You may also specify a CLASS name, or an ARRAY of names. In case of an ARRAY, a 'multi' locker is started with all thee full CLASS name.

  Supported locking names are

  'DOTLOCK' | 'dotlock'
  : The folder handler creates a file which signals that it is in use. This is a bit problematic, because not all mail-handling software agree on the name of the file to be created.

    On various folder types, the lockfile differs. See the documentation for each folder, which describes the locking strategy as well as special options to change the default behavior.

  'FLOCK' | 'flock'
  : For some folder handlers, locking is based on a file locking mechanism provided by the operating system. However, this does not work on all systems, such as network filesystems, and such. This also doesn't work on folders based on directories (Mail::Box::Dir and derived).

  'FCNTLLOCK' | 'fcntllock'
  : POSIX locking via File::FcntlLock, which works on more platforms. However, that module requires a C compiler to install.

  'POSIX' | 'posix'
  : Use the POSIX standard fcntl locking.

  'MULTI' | 'multi'
  : Use ALL available locking methods at the same time, to have a bigger chance that the folder will not be modified by some other application which uses an unspecified locking method. When one of the locking methods disallows access, the locking fails.

  'MUTT'| 'mutt'
  : Use the external program 'mutt_dotlock' to lock and unlock.

  'NFS' | 'nfs'
  : A kind of `dotlock` file-locking mechanism, but adapted to work over NFS. Extra precaution is needed because an `open O_EXCL` on NFS is not an atomic action.

  'NONE' | 'none'
  : Do not use locking.

  The other option is to produce your own `Mail::Box::Locker` derived class, which implements the desired locking method. (Please consider offering it for inclusion in the public Mail::Box module!) Create an instance of that class with this parameter:

```
my $locker = Mail::Box::Locker::MyOwn->new;
$folder->open(locker => $locker);
```

timeout => SECONDS|'NOTIMEOUT'
: How long to wait while trying to acquire the lock. The lock request will fail when the specified number of seconds is reached. If `'NOTIMEOUT'` is specified, the module will wait until the lock can be taken.

  Whether it is possible to limit the wait time is platform− and locking-method-specific. For instance, the 'dotlock' method on Windows will always wait until the lock has been received.

trace => LEVEL

Attributes

`$obj`−>**expires**([SECONDS])
Get/Set the expiration time. Not available for all lockers.

`$obj`−>**timeout**([SECONDS])
Get/Set the timeout. Not available for all lockers.

The Locker

`$obj`−>**filename**([$filename])
Returns the filename which is used to lock the folder, optionally after setting it to the specified `$filename`.

example:

```
 print $locker->filename;
```

`$obj`−>**folder**([$folder])
Returns the folder object which is locker.

`$obj`−>**name**()
Returns the method used to lock the folder. See the new(method) for details on how to specify the lock method. The name of the method is returned in upper-case.

example:

```
 if($locker->name eq 'FLOCK') ...
```

Locking

`$obj`−>**hasLock**()
Check whether the folder has the lock.

example:

```
 if($locker->hasLock) {...}
 if($folder->locker->hasLock) {...}
```

`$obj`−>**isLocked**()
Test if the folder is locked by this or a different application.

example:

```
 if($locker->isLocked) {...}
 if($folder->locker->isLocked) {...}
```

`$obj`−>**lock**($folder)
Get a lock on a folder. This will return false if the lock fails.

example:

```
 die unless $locker->lock;
 if($folder->locker->lock) {...}
```

`$obj`−>**unlock**()
Undo the lock on a folder.

example:

```
 $locker->unlock;
 $folder->locker->unlock;
```

Error handling

Extends ‘‘Error handling’’ in Mail::Reporter.

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

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

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

$obj−>**logPriority**($level)
Mail::Box::Locker−>**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::Reporter.

$obj−>**DESTROY**()
When the locker is destroyed, for instance when the folder is closed or the program ends, the lock will be automatically removed.

DIAGNOSTICS

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.

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