

# libSntpThread Reference Manual

## 1.0

Generated by Doxygen 1.5.1

Sat Oct 20 15:48:46 2007



# Contents

|          |  |          |
|----------|--|----------|
| <b>1</b> | <b>libSmtplibThread Hierarchical Index</b>       | <b>1</b> |
| 1.1      | libSmtplibThread Class Hierarchy . . . . .       | 1        |
| <b>2</b> | <b>libSmtplibThread Class Index</b>              | <b>3</b> |
| 2.1      | libSmtplibThread Class List . . . . .            | 3        |
| <b>3</b> | <b>libSmtplibThread Class Documentation</b>      | <b>5</b> |
| 3.1      | SMTPConnection Class Reference . . . . .         | 5        |
| 3.2      | SMTPConnectionPool Class Reference . . . . .     | 8        |
| 3.3      | Thread Class Reference . . . . .                 | 13       |
| 3.4      | ThreadedSMTPConnection Class Reference . . . . . | 19       |



# Chapter 1

## libSmtplib Thread Hierarchical Index

### 1.1 libSmtplib Thread Class Hierarchy

This inheritance list is sorted roughly, but not completely, alphabetically:

|                                  |    |
|----------------------------------|----|
| SMTPConnection . . . . .         | 5  |
| SMTPConnectionPool . . . . .     | 8  |
| Thread . . . . .                 | 13 |
| ThreadedSMTPConnection . . . . . | 19 |



## Chapter 2

# libSmtplibThread Class Index

### 2.1 libSmtplibThread Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

|   |    |
|---|----|
| <b>SMTPConnection</b> . . . . .         | 5  |
| <b>SMTPConnectionPool</b> . . . . .     | 8  |
| <b>Thread</b> . . . . .                 | 13 |
| <b>ThreadedSMTPConnection</b> . . . . . | 19 |





# Chapter 3

## libSmtplib Thread Class Documentation

### 3.1 SMTPConnection Class Reference

```
#include <SMTPConnection.h>
```

#### Public Types

- enum **SMTPState**

#### Public Member Functions

- **SMTPConnection** ()
- **~SMTPConnection** ()
- **SMTPState** **getSMTPState** ()
- void **setSMTPHost** (string host)
- void **setSMTPPort** (int port)
- void **setMyHostname** (string myname)
- int **sendMail** (string &from, string &to, string &msg)
- void **disconnect** ()

#### 3.1.1 Detailed Description

SMTPConnection-Class.

This class is inherited from the Thread-class. This means, you can initialize a SMTPConnection-object, setup your **SMTPConnection** (p. 5), start threading and push a message to the **SMTPConnection** (p. 5). While the **SMTPConnection** (p. 5) send the message in its thread, you can continue doing other stuff.

**SMTPConnection** (p. 5) speaks plain SMTP, no ESMTP.

Definition at line 46 of file SMTPConnection.h.

### 3.1.2 Member Enumeration Documentation

#### 3.1.2.1 enum SMTPConnection::SMTPState

Definition of the possible states of an SMTP-connection.

Definition at line 61 of file SMTPConnection.h.

### 3.1.3 Constructor & Destructor Documentation

#### 3.1.3.1 SMTPConnection::SMTPConnection ()

Constructor.

Definition at line 28 of file SMTPConnection.cc.

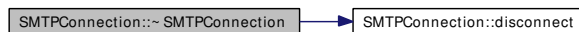
#### 3.1.3.2 SMTPConnection::~~SMTPConnection ()

Destructor.

Definition at line 62 of file SMTPConnection.cc.

References disconnect().

Here is the call graph for this function:



### 3.1.4 Member Function Documentation

#### 3.1.4.1 SMTPConnection::SMTPState SMTPConnection::getSMTPState ()

Gets the current state of the connection. Since this is an unthreaded class, this will either be "SMTPState::NOT\_CONNECTED" or "SMTPState::CONNECTED".

**Returns:**

**SMTPConnection::SMTPState** (p. 6)

Definition at line 80 of file SMTPConnection.cc.

Referenced by ThreadedSMTPConnection::getSMTPState().

#### 3.1.4.2 void SMTPConnection::setSMTPHost (string *host*)

Sets the IP of the SMTP-host to connect to.

**Parameters:**

***string*** host IP-address of SMTP-host.

Definition at line 485 of file SMTPConnection.cc.

Referenced by ThreadedSMTPConnection::setSMTPHost().

#### 3.1.4.3 void SMTPConnection::setSMTPPort (int *port*)

Sets the port-number of the SMTP-host to connect to.

**Parameters:**

*int* port Port-number on SMTP-host.

Definition at line 496 of file SMTPConnection.cc.

Referenced by ThreadedSMTPConnection::setSMTPPort().

#### 3.1.4.4 void SMTPConnection::setMyHostname (string *myname*)

Sets the hostname to send in the HELO-command of the SMTP-dialog.

**Parameters:**

*string* myname HELO-hostname.

Definition at line 817 of file SMTPConnection.cc.

Referenced by ThreadedSMTPConnection::setMyHostname().

#### 3.1.4.5 int SMTPConnection::sendMail (string & *from*, string & *to*, string & *msg*)

Sends a mail via SMTP.

**Parameters:**

*string* from FROM-email

*string* to TO-email

*string* msg Message-body of email. Should include additional headers like "Subject" etc.

**Returns:**

int status: 0 = success, -1 = error

Definition at line 804 of file SMTPConnection.cc.

Referenced by ThreadedSMTPConnection::Execute().

#### 3.1.4.6 void SMTPConnection::disconnect ()

When connected, disconnects from the SMTP-Server.

Definition at line 833 of file SMTPConnection.cc.

Referenced by ThreadedSMTPConnection::Execute(), and ~SMTPConnection().

The documentation for this class was generated from the following files:

- smtpthread/SMTPConnection.h
- smtpthread/SMTPConnection.cc

## 3.2 SMTPConnectionPool Class Reference

```
#include <SMTPConnectionPool.h>
```

### Public Member Functions

- **SMTPConnectionPool** ()
- **~SMTPConnectionPool** ()
- int **sendMail** (string &from, string &to, string &msg)
- void **setPoolSize** (unsigned long)
- unsigned long **getPoolSize** ()
- void **setSMTPHost** (string host)
- void **setSMTPPort** (int port)
- void **setMyHostname** (string myname)
- void **setOptimizePool** (bool)
- void **setMinPoolSize** (long)
- void **setMaxPoolSize** (long)

### 3.2.1 Detailed Description

SMTPConnectionPool-Class.

The **SMTPConnectionPool** (p. 8) sets up an amount of SMTPConnections. These connections are threaded objects and run each in a separate tread. To send a message, just utilize the sendMail-method. The **SMTPConnectionPool** (p. 8) then tries to find a ready connection and delegates the message to this connection.

Definition at line 24 of file SMTPConnectionPool.h.

### 3.2.2 Constructor & Destructor Documentation

#### 3.2.2.1 SMTPConnectionPool::SMTPConnectionPool ()

Default Constructor.

Definition at line 20 of file SMTPConnectionPool.cc.

#### 3.2.2.2 SMTPConnectionPool::~~SMTPConnectionPool ()

Default Destructor.

Definition at line 30 of file SMTPConnectionPool.cc.

### 3.2.3 Member Function Documentation

#### 3.2.3.1 int SMTPConnectionPool::sendMail (string & *from*, string & *to*, string & *msg*)

Send an email to the connection-pool.

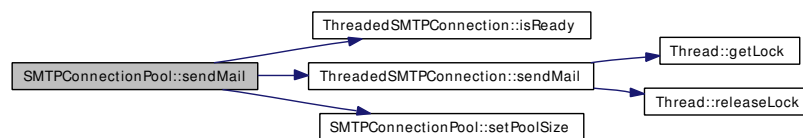
**Parameters:**

- string* from sender-email, used in the SMTP-dialog.
- string* to receiver-email, used in the SMTP-dialog.
- string* msg email-body to send, including additional headers.

Definition at line 93 of file SMTPConnectionPool.cc.

References ThreadedSMTPConnection::isReady(), ThreadedSMTPConnection::sendMail(), and setPoolSize().

Here is the call graph for this function:

**3.2.3.2 void SMTPConnectionPool::setPoolSize (unsigned long)**

Sets the pool-size. The pool-size is the number of SMTP-connectons, which are connected to the SMTP-Server.

**Parameters:**

- unsigned* long poolSize number of SMTP-connections.

Definition at line 253 of file SMTPConnectionPool.cc.

Referenced by sendMail(), setMaxPoolSize(), and setMinPoolSize().

**3.2.3.3 unsigned long SMTPConnectionPool::getPoolSize ()**

Returns the pool-size.

**Returns:**

- unsigned long pool-size.

Definition at line 313 of file SMTPConnectionPool.cc.

**3.2.3.4 void SMTPConnectionPool::setSMTPHost (string host)**

Sets the IP-address of the SMTP-host.

NOTE: Must be an IP-address and no domain-name.

**Parameters:**

- string* host IP-address of the SMTP-server.

Definition at line 321 of file SMTPConnectionPool.cc.

References ThreadedSMTPConnection::setSMTPHost().

Here is the call graph for this function:



### 3.2.3.5 void SMTPConnectionPool::setSMTPPort (int port)

Sets the port-number to connect to. Defaults to 25.

#### Parameters:

*int* port port-number of the remote SMTP-server.

Definition at line 335 of file SMTPConnectionPool.cc.

References `ThreadedSMTPConnection::setSMTPPort()`.

Here is the call graph for this function:



### 3.2.3.6 void SMTPConnectionPool::setMyHostname (string myname)

Sets the hostname, which is send in the HELO-command.

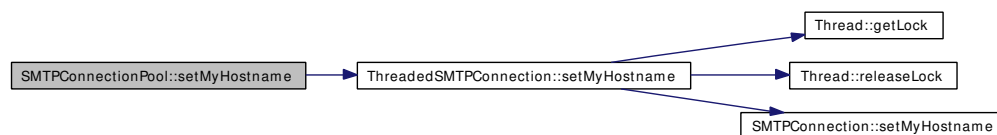
#### Parameters:

*string* myname hostname of the mail-sending-host.

Definition at line 349 of file SMTPConnectionPool.cc.

References `ThreadedSMTPConnection::setMyHostname()`.

Here is the call graph for this function:



### 3.2.3.7 void SMTPConnectionPool::setOptimizePool (bool)

Tells the pool whether or not to optimize the pool-size.

Optimizing tries to find a pool-size, which has enough connections so that a new sendmail-command does not have to wait for previous connection to finish.

The optimal pool-size is calculated and set, if within the upper and lower limits set with "setMinPoolSize" and "setMaxPoolSize".

Please remember, that changing the pool-size also affects allocating or deallocating of new/existing objects. This results in small performanc-impacts. The bast way is to already know the optimal pool-size, initialize the pool with that size and disable optimizing the pool.

NOTE: Optimizing the pool-size is not a trivial task. The algorithm used in this implementation, is a rather simple approach. Works for me, but might not be smart enough for you.

#### Parameters:

*bool* opt true / false = optimize / do not optimize the pool-size.

#### See also:

**SMTPConnectionPool::setMinPoolSize** (p. 11)

**SMTPConnectionPool::setMaxPoolSize** (p. 11)

Definition at line 363 of file SMTPConnectionPool.cc.

#### 3.2.3.8 void SMTPConnectionPool::setMinPoolSize (long)

Sets the minimum size of the connection-pool. Takes effect only, when pool-optimization is active.

#### Parameters:

*long* minimum minimal pool-size.

Definition at line 370 of file SMTPConnectionPool.cc.

References `setPoolSize()`.

Here is the call graph for this function:



#### 3.2.3.9 void SMTPConnectionPool::setMaxPoolSize (long)

Sets the maximum size of the connection-pool. Takes effect only, when pool-optimization is active.

#### Parameters:

*long* maximum maximal pool-size.

Definition at line 383 of file SMTPConnectionPool.cc.

References `setPoolSize()`.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- smtpthread/SMTPConnectionPool.h
- smtpthread/SMTPConnectionPool.cc

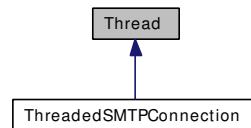


## 3.3 Thread Class Reference

```
#include <Thread.h>
```

Inherited by **ThreadedSMTPConnection**.

Inheritance diagram for Thread:



### Public Member Functions

- **Thread** ()
- **~Thread** ()
- int **Start** (void \*arg)
- pthread\_t **getThreadId** ()
- bool **isRunning** ()
- void **Stop** ()

### Protected Member Functions

- int **Run** (void \*arg)
- virtual void **Setup** ()
- virtual void **Execute** (void \*)
- void \* **Arg** () const
- void **Arg** (void \*a)
- void **getLock** ()
- void **releaseLock** ()

### Static Protected Member Functions

- static void \* **EntryPoint** (void \*)

### Protected Attributes

- bool **\_isRunning**
- pthread\_mutex\_t **\_mutex**

#### 3.3.1 Detailed Description

Thread-Class.

This abstract class wraps a class around a threading function.

Threading is started through calling 'Start', which sets the protected member-variable '\_isRunning'. The state of the thread can be checked with the member-function 'isRunning'.

You must overwrite the method 'Execute' to do something usefull within the thread.

'Execute' should be constructed as an endless loop like:

```
void InheritedClassname::Execute( void * arg ) { while( _isRunning ) { // do
something...  } }
```

When 'Start' is called, it starts the thread and executes 'Setup' and 'Execute'.

To stop threading, simply call 'Stop'.

Definition at line 42 of file Thread.h.

### 3.3.2 Constructor & Destructor Documentation

#### 3.3.2.1 Thread::Thread ()

Constructor.

Definition at line 15 of file Thread.cc.

References `_isRunning`, and `_mutex`.

#### 3.3.2.2 Thread::~~Thread ()

Destructor.

NOTE: Under normal conditions I would use a purely virtual destructor. But unfortunately most people (me too) often forget to call the 'Stop' member-function in the destructor of the derived classes.

So calling delete on an object, which still has a running thread, may not behave well, especially when the destructor removes members of the object, while the thread is trying to access them.

So, I implemented a non-virtual destructor, which stops threading, when the thread is still running. This will cause most compilers to issue a warning, but it seems to be the easiest way, to get your code working... ;)

Definition at line 26 of file Thread.cc.

References `_isRunning`, and `Stop()`.

Here is the call graph for this function:



### 3.3.3 Member Function Documentation

#### 3.3.3.1 int Thread::Start (void \* arg)

Start threading.

**Parameters:**

*void\** arg typeless argument.

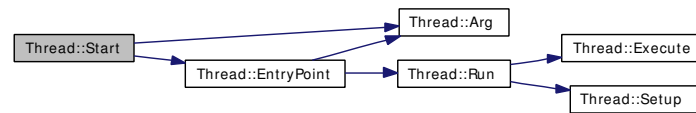
**Returns:**

int exit-code of 'pthread\_create'

Definition at line 40 of file Thread.cc.

References Arg(), and EntryPoint().

Here is the call graph for this function:



### 3.3.3.2 pthread\_t Thread::getThreadId ()

Returns the thread-id.

**Returns:**

pthread\_t thread-id

Definition at line 130 of file Thread.cc.

### 3.3.3.3 bool Thread::isRunning ()

Checks, whether or not the thread is running.

**Returns:**

bool true / false = running / not running

Definition at line 141 of file Thread.cc.

References `_isRunning`.

### 3.3.3.4 void Thread::Stop ()

Stop threading.

Definition at line 151 of file Thread.cc.

References `_isRunning`.

Referenced by `~Thread()`.

### 3.3.3.5 int Thread::Run (void \* arg) [protected]

Gets called immediatly after "EntryPoint". Calls "Setup" and "Execute".

**Parameters:**

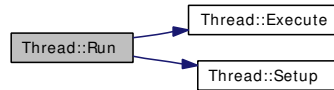
*void\** arg typeless argument.

Definition at line 66 of file Thread.cc.

References `Execute()`, and `Setup()`.

Referenced by `EntryPoint()`.

Here is the call graph for this function:



### 3.3.3.6 `void * Thread::EntryPoint (void *)` [static, protected]

This is the entrypoint for threading. This function must be declared as "static", because the pthread-library expects a normal C-type-function as an entrypoint.

Parameters are passed as pointer to void and are the same as passed to "Start".

#### Parameters:

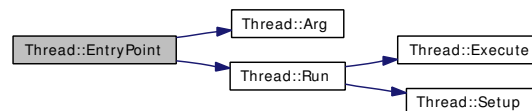
*void\** arg typeless argument.

Definition at line 80 of file Thread.cc.

References `Arg()`, and `Run()`.

Referenced by `Start()`.

Here is the call graph for this function:



### 3.3.3.7 `void Thread::Setup ()` [protected, virtual]

Initializes some thread-variables and sets up a mutex. The mutex can be accessed through the member-variable "`_mutex`".

Definition at line 104 of file Thread.cc.

References `_isRunning`, and `_mutex`.

Referenced by `Run()`.

### 3.3.3.8 `void Thread::Execute (void *)` [protected, virtual]

The main threading-functionality goes here. You should create an endless loop like:

```
while( _isRunning) { // do something usefull... }
```

Parameters are passed as pointer to void and are the same as passed to "Start".

#### Parameters:

*void\** arg typeless argument.

Reimplemented in **ThreadedSMTPConnection** (p. 20).

Definition at line 119 of file Thread.cc.

Referenced by Run().

#### 3.3.3.9 void\* Thread::Arg () const [inline, protected]

Helper to initialize the parameters provided to "Start".

Definition at line 146 of file Thread.h.

Referenced by EntryPoint(), and Start().

#### 3.3.3.10 void Thread::Arg (void \* a) [inline, protected]

Initializes the parameters passed to "Start".

#### Parameters:

*void\** a typeless argument.

Definition at line 156 of file Thread.h.

#### 3.3.3.11 void Thread::getLock () [protected]

Locks the mutex "\_mutex".

NOTE: Use with care, cause it might get you a deadlock!

#### See also:

**Thread::releaseLock** (p. 17)

Definition at line 167 of file Thread.cc.

References \_mutex.

Referenced by ThreadedSMTPConnection::Execute(), ThreadedSMTPConnection::sendMail(), ThreadedSMTPConnection::setMyHostname(), ThreadedSMTPConnection::setSMTPHost(), and ThreadedSMTPConnection::setSMTPPort().

#### 3.3.3.12 void Thread::releaseLock () [protected]

Unlocks the mutex "\_mutex".

#### See also:

**Thread::getLock** (p. 17)

Definition at line 178 of file Thread.cc.

References \_mutex.

Referenced by ThreadedSMTPConnection::Execute(), ThreadedSMTPConnection::sendMail(), ThreadedSMTPConnection::setMyHostname(), ThreadedSMTPConnection::setSMTPHost(), and ThreadedSMTPConnection::setSMTPPort().

### 3.3.4 Member Data Documentation

#### 3.3.4.1 `bool Thread::_isRunning` [protected]

Whether or not the thread is running.

Definition at line 162 of file Thread.h.

Referenced by ThreadedSMTPConnection::Execute(), isRunning(), Setup(), Stop(), Thread(), and ~Thread().

#### 3.3.4.2 `pthread_mutex_t Thread::_mutex` [protected]

Mutex.

Definition at line 165 of file Thread.h.

Referenced by getLock(), releaseLock(), Setup(), and Thread().

The documentation for this class was generated from the following files:

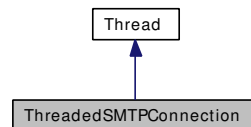
- smtpthread/Thread.h
- smtpthread/Thread.cc

## 3.4 ThreadedSMTPConnection Class Reference

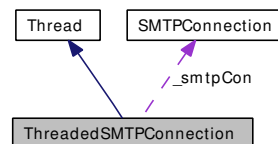
```
#include <ThreadedSMTPConnection.h>
```

Inherits **Thread**.

Inheritance diagram for ThreadedSMTPConnection:



Collaboration diagram for ThreadedSMTPConnection:



### Public Member Functions

- **ThreadedSMTPConnection** ()
- void **Execute** (void \*arg)
- bool **isReady** ()
- **SMTPConnection::SMTPState** **getSMTPState** ()
- void **setSMTPHost** (string host)
- void **setSMTPPort** (int port)
- void **setMyHostname** (string myname)
- void **sendMail** (string &from, string &to, string &msg)

#### 3.4.1 Detailed Description

ThreadedSMTPConnection-Class.

This class is inherited from the Thread-class. This means, you can initialize a ThreadedSMTPConnection-object, setup your **ThreadedSMTPConnection** (p.19), start threading and push a message to the **ThreadedSMTPConnection** (p.19). While the **ThreadedSMTPConnection** (p.19) sends the message in its thread, you can continue doing other stuff.

**ThreadedSMTPConnection** (p.19) speaks plain SMTP, no ESMTP.

Definition at line 47 of file ThreadedSMTPConnection.h.

#### 3.4.2 Constructor & Destructor Documentation

##### 3.4.2.1 ThreadedSMTPConnection::ThreadedSMTPConnection ()

Default constructor.

Definition at line 34 of file ThreadedSMTPConnection.cc.

### 3.4.3 Member Function Documentation

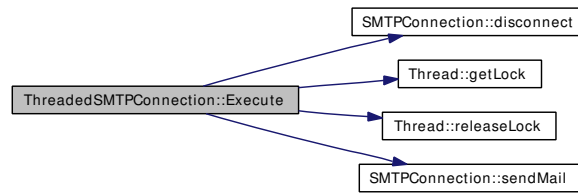
#### 3.4.3.1 void ThreadedSMTPConnection::Execute (void \* *arg*) [virtual]

The main thread-loop. Checks whether or not a mail is provided to the object and sends the mail. Reimplemented from **Thread** (p. 16).

Definition at line 49 of file ThreadedSMTPConnection.cc.

References Thread::\_isRunning, SMTPConnection::disconnect(), Thread::getLock(), Thread::releaseLock(), and SMTPConnection::sendMail().

Here is the call graph for this function:



#### 3.4.3.2 bool ThreadedSMTPConnection::isReady ()

Returns the current status of the thread.

**Returns:**

bool true / false = thread is ready / is not ready to accept messages.

Definition at line 133 of file ThreadedSMTPConnection.cc.

Referenced by SMTPConnectionPool::sendMail().

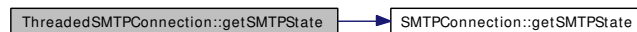
#### 3.4.3.3 SMTPConnection::SMTPState ThreadedSMTPConnection::getSMTPState ()

Returns the current state of the SMTP-dialog.

Definition at line 149 of file ThreadedSMTPConnection.cc.

References SMTPConnection::getSMTPState().

Here is the call graph for this function:



#### 3.4.3.4 void ThreadedSMTPConnection::setSMTPHost (string *host*)

Sets the IP-address of the SMTP-server.

NOTE: Must be an IP-address and not a domain-name!



**Parameters:**

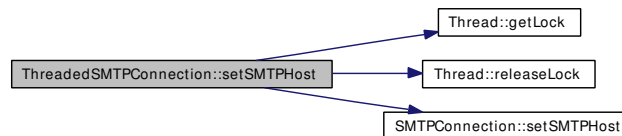
*string* host IP-adress of SMTP-server.

Definition at line 192 of file ThreadedSMTPConnection.cc.

References Thread::getLock(), Thread::releaseLock(), and SMTPConnection::setSMTPHost().

Referenced by SMTPConnectionPool::setSMTPHost().

Here is the call graph for this function:

**3.4.3.5 void ThreadedSMTPConnection::setSMTPPort (int port)**

Sets the port-number to connect to.

**Parameters:**

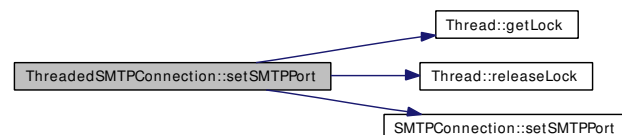
*int* port Port-number of the SMTP-server.

Definition at line 216 of file ThreadedSMTPConnection.cc.

References Thread::getLock(), Thread::releaseLock(), and SMTPConnection::setSMTPPort().

Referenced by SMTPConnectionPool::setSMTPPort().

Here is the call graph for this function:

**3.4.3.6 void ThreadedSMTPConnection::setMyHostname (string myname)**

Sets the hostname to send within the HELO-command.

**Parameters:**

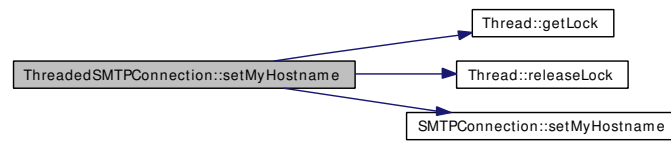
*string* myname Hostname of the sending host.

Definition at line 236 of file ThreadedSMTPConnection.cc.

References Thread::getLock(), Thread::releaseLock(), and SMTPConnection::setMyHostname().

Referenced by SMTPConnectionPool::setMyHostname().

Here is the call graph for this function:



#### 3.4.3.7 void ThreadedSMTPConnection::sendMail (string & *from*, string & *to*, string & *msg*)

Send an email to the SMTP-server.

##### Parameters:

- string* *from* email-address of the sender.
- string* *to* email-address of the receiver.
- string* *msg* email-body, including all additional headers.

Definition at line 166 of file ThreadedSMTPConnection.cc.

References `Thread::getLock()`, and `Thread::releaseLock()`.

Referenced by `SMTPConnectionPool::sendMail()`.

Here is the call graph for this function:



The documentation for this class was generated from the following files:

- smtpthread/ThreadedSMTPConnection.h
- smtpthread/ThreadedSMTPConnection.cc

# Index

- \_isRunning
    - Thread, 18
  - \_mutex
    - Thread, 18
  - ~SMTPConnection
    - SMTPConnection, 6
  - ~SMTPConnectionPool
    - SMTPConnectionPool, 8
  - ~Thread
    - Thread, 14
- Arg
  - Thread, 17
- disconnect
  - SMTPConnection, 7
- EntryPoint
  - Thread, 16
- Execute
  - Thread, 16
  - ThreadedSMTPConnection, 20
- getLock
  - Thread, 17
- getPoolSize
  - SMTPConnectionPool, 9
- getSMTPState
  - SMTPConnection, 6
  - ThreadedSMTPConnection, 20
- getThreadId
  - Thread, 15
- isReady
  - ThreadedSMTPConnection, 20
- isRunning
  - Thread, 15
- releaseLock
  - Thread, 17
- Run
  - Thread, 15
- sendMail
  - SMTPConnection, 7
  - SMTPConnectionPool, 8
  - ThreadedSMTPConnection, 22
- setMaxPoolSize
  - SMTPConnectionPool, 11
- setMinPoolSize
  - SMTPConnectionPool, 11
- setMyHostname
  - SMTPConnection, 7
  - SMTPConnectionPool, 10
  - ThreadedSMTPConnection, 21
- setOptimizePool
  - SMTPConnectionPool, 10
- setPoolSize
  - SMTPConnectionPool, 9
- setSMTPHost
  - SMTPConnection, 6
  - SMTPConnectionPool, 9
  - ThreadedSMTPConnection, 20
- setSMTPPort
  - SMTPConnection, 6
  - SMTPConnectionPool, 10
  - ThreadedSMTPConnection, 21
- Setup
  - Thread, 16
- SMTPConnection, 5
  - ~SMTPConnection, 6
  - disconnect, 7
  - getSMTPState, 6
  - sendMail, 7
  - setMyHostname, 7
  - setSMTPHost, 6
  - setSMTPPort, 6
  - SMTPConnection, 6
  - SMTPState, 6
- SMTPConnectionPool, 8
  - SMTPConnectionPool, 8
- SMTPConnectionPool
  - ~SMTPConnectionPool, 8
  - getPoolSize, 9
  - sendMail, 8
  - setMaxPoolSize, 11
  - setMinPoolSize, 11
  - setMyHostname, 10
  - setOptimizePool, 10
  - setPoolSize, 9
  - setSMTPHost, 9

- setSMTPPort, 10
- SMTPConnectionPool, 8
- SMTPState
  - SMTPConnection, 6
- Start
  - Thread, 14
- Stop
  - Thread, 15
- Thread, 13
  - \_isRunning, 18
  - \_mutex, 18
  - ~Thread, 14
  - Arg, 17
  - EntryPoint, 16
  - Execute, 16
  - getLock, 17
  - getThreadId, 15
  - isRunning, 15
  - releaseLock, 17
  - Run, 15
  - Setup, 16
  - Start, 14
  - Stop, 15
  - Thread, 14
- ThreadedSMTPConnection, 19
  - ThreadedSMTPConnection, 19
- ThreadedSMTPConnection
  - Execute, 20
  - getSMTPState, 20
  - isReady, 20
  - sendMail, 22
  - setMyHostname, 21
  - setSMTPHost, 20
  - setSMTPPort, 21
  - ThreadedSMTPConnection, 19