

---

# NSLock Class Reference

[Cocoa > Process Management](#)



2008-02-08



Apple Inc.  
© 2008 Apple Inc.  
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

Apple Inc.  
1 Infinite Loop  
Cupertino, CA 95014  
408-996-1010

Apple, the Apple logo, Cocoa, Mac, and Mac OS are trademarks of Apple Inc., registered in the United States and other countries.

iPhone is a trademark of Apple Inc.

Simultaneously published in the United States and Canada.

**Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.**

**IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR**

**CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.**

**THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.**

**Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.**

# Contents

---

## **NSLock Class Reference 5**

---

- Overview 5
- Adopted Protocols 6
- Tasks 6
  - Acquiring a Lock 6
  - Naming the Lock 6
- Instance Methods 6
  - lockBeforeDate: 6
  - name 7
  - setName: 7
  - tryLock 7

---

## **Document Revision History 9**

---

---

## **Index 11**

---




# NSLock Class Reference

---

<b>Inherits from</b>	NSObject
<b>Conforms to</b>	NSLocking NSObject (NSObject)
<b>Framework</b>	/System/Library/Frameworks/Foundation.framework
<b>Availability</b>	Available in Mac OS X v10.0 and later.
<b>Companion guide</b>	Threading Programming Guide
<b>Declared in</b>	NSLock.h
<b>Related sample code</b>	Aperture Image Resizer ExtractMovieAudioToAIFF QTEExtractAndConvertToAIFF QTQuartzPlayer SimpleThreads

## Overview

An `NSLock` object is used to coordinate the operation of multiple threads of execution within the same application. An `NSLock` object can be used to mediate access to an application's global data or to protect a critical section of code, allowing it to run atomically.

 **Warning:** The `NSLock` class uses POSIX threads to implement its locking behavior. When sending an unlock message to an `NSLock` object, you must be sure that message is sent from the same thread that sent the initial lock message. Unlocking a lock from a different thread can result in undefined behavior.

You should not use this class to implement a recursive lock. Calling the `lock` method twice on the same thread will lock up your thread permanently. Use the `NSRecursiveLock` class to implement recursive locks instead.

Unlocking a lock that is not locked is considered a programmer error and should be fixed in your code. The `NSLock` class reports such errors by printing an error message to the console when they occur.

## Adopted Protocols

- NSLocking
- lock
  - unlock

## Tasks

### Acquiring a Lock

- [lockBeforeDate:](#) (page 6)  
Attempts to acquire a lock before a given time and returns a Boolean value indicating whether the attempt was successful.
- [tryLock](#) (page 7)  
Attempts to acquire a lock and immediately returns a Boolean value that indicates whether the attempt was successful.

### Naming the Lock

- [setName:](#) (page 7)  
Assigns a name to the receiver.
- [name](#) (page 7)  
Returns the name associated with the receiver.

## Instance Methods

### lockBeforeDate:

Attempts to acquire a lock before a given time and returns a Boolean value indicating whether the attempt was successful.

- (BOOL)lockBeforeDate:(NSDate \*)*limit*

#### Parameters

*limit*

The time limit for attempting to acquire a lock.

#### Return Value

YES if the lock is acquired before *limit*, otherwise NO.

#### Discussion

The thread is blocked until the receiver acquires the lock or *limit* is reached.

### Availability

Available in Mac OS X v10.0 and later.

### Declared In

NSLock.h

## name

Returns the name associated with the receiver.

- (NSString \*)name

### Return Value

The name of the receiver.

### Availability

Available in Mac OS X v10.5 and later.

### See Also

- [setName:](#) (page 7)

### Declared In

NSLock.h

## setName:

Assigns a name to the receiver.

- (void)setName:(NSString \*)newName

### Parameters

*newName*

The new name for the receiver. This method makes a copy of the specified string.

### Discussion

You can use a name string to identify a lock within your code. Cocoa also uses this name as part of any error descriptions involving the receiver.

### Availability

Available in Mac OS X v10.5 and later.

### See Also

- [name](#) (page 7)

### Declared In

NSLock.h

## tryLock

Attempts to acquire a lock and immediately returns a Boolean value that indicates whether the attempt was successful.

- (BOOL)tryLock

**Return Value**

YES if the lock was acquired, otherwise NO.

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

NSLock.h



# Document Revision History

---

This table describes the changes to *NSLock Class Reference*.

Date	Notes
2008-02-08	Added a warning describing what happens when you unlock a lock that is not currently locked.
2007-05-04	Updated for Mac OS X v10.5.
2006-05-23	First publication of this content as a separate document.

## REVISION HISTORY

### Document Revision History

# Index

---

## L

---

lockBeforeDate: [instance method 6](#)

## N

---

name [instance method 7](#)

## S

---

setName: [instance method 7](#)

## T

---

tryLock [instance method 7](#)