
NSDistributedLock Class Reference

[Cocoa](#) > [Process Management](#)



2007-01-22



Apple Inc.
© 2007 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.

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

NSDistributedLock Class Reference 5

- Overview 5
- Tasks 5
 - Creating an NSDistributedLock 5
 - Acquiring a Lock 6
 - Relinquishing a Lock 6
 - Getting Lock Information 6
- Class Methods 6
 - lockWithPath: 6
- Instance Methods 7
 - breakLock 7
 - initWithPath: 7
 - lockDate 8
 - tryLock 8
 - unlock 9

Document Revision History 11

Index 13

NSDistributedLock Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.0 and later.
Companion guide	Threading Programming Guide
Declared in	NSDistributedLock.h

Overview

The `NSDistributedLock` class defines an object that multiple applications on multiple hosts can use to restrict access to some shared resource, such as a file.

The lock is implemented by an entry (such as a file or directory) in the file system. For multiple applications to use an `NSDistributedLock` object to coordinate their activities, the lock must be writable on a file system accessible to all hosts on which the applications might be running.

Use the [tryLock](#) (page 8) method to attempt to acquire a lock. You should generally use the [unlock](#) (page 9) method to release the lock rather than [breakLock](#) (page 7).

`NSDistributedLock` doesn't conform to the `NSLocking` protocol, nor does it have a `lock` method. The protocol's `lock` method is intended to block the execution of the thread until successful. For an `NSDistributedLock` object, this could mean polling the file system at some predetermined rate. A better solution is to provide the [tryLock](#) (page 8) method and let you determine the polling frequency that makes sense for your application.

Tasks

Creating an `NSDistributedLock`

+ [lockWithPath:](#) (page 6)

Returns an `NSDistributedLock` object initialized to use as the locking object the file-system entry specified by a given path.

- [initWithPath:](#) (page 7)
Initializes an `NSDistributedLock` object to use as the lock the file-system entry specified by a given path.

Acquiring a Lock

- [tryLock](#) (page 8)
Attempts to acquire the receiver and immediately returns a Boolean value that indicates whether the attempt was successful.

Relinquishing a Lock

- [breakLock](#) (page 7)
Forces the lock to be relinquished.
- [unlock](#) (page 9)
Relinquishes the receiver.

Getting Lock Information

- [lockDate](#) (page 8)
Returns the time the receiver was acquired by any of the `NSDistributedLock` objects using the same path.

Class Methods

lockWithPath:

Returns an `NSDistributedLock` object initialized to use as the locking object the file-system entry specified by a given path.

```
+ (NSDistributedLock *)lockWithPath:(NSString *)aPath
```

Parameters

aPath

All of *aPath* up to the last component itself must exist. You can use `NSFileManager` to create (and set permissions) for any nonexistent intermediate directories.

Return Value

An `NSDistributedLock` object initialized to use as the locking object the file-system entry specified by *aPath*.

Discussion

For applications to use the lock, *aPath* must be accessible to—and writable by—all hosts on which the applications might be running.

Availability

Available in Mac OS X v10.0 and later.

See Also

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

Declared In

NSDistributedLock.h

Instance Methods

breakLock

Forces the lock to be relinquished.

- (void)breakLock

Discussion

This method always succeeds unless the lock has been damaged. If another process has already unlocked or broken the lock, this method has no effect. You should generally use [unlock](#) (page 9) rather than `breakLock` to relinquish a lock.



Warning: Because `breakLock` can release another process's lock, it should be used with great caution.

Even if you break a lock, there's no guarantee that you will then be able to acquire the lock—another process might get it before your [tryLock](#) (page 8) is invoked.

Raises an `NSGenericException` if the lock could not be removed.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [unlock](#) (page 9)

Declared In

NSDistributedLock.h

initWithPath:

Initializes an `NSDistributedLock` object to use as the lock the file-system entry specified by a given path.

- (id)initWithPath:(NSString *)aPath

Parameters

aPath

All of *aPath* up to the last component itself must exist. You can use `NSFileManager` to create (and set permissions) for any nonexistent intermediate directories.

Return Value

An `NSDistributedLock` object initialized to use as the locking object the file-system entry specified by *aPath*.

Discussion

For applications to use the lock, *aPath* must be accessible to—and writable by—all hosts on which the applications might be running.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [lockWithPath:](#) (page 6)

Declared In

`NSDistributedLock.h`

lockDate

Returns the time the receiver was acquired by any of the `NSDistributedLock` objects using the same path.

- (`NSDate *`)lockDate

Return Value

The time the receiver was acquired by any of the `NSDistributedLock` objects using the same path. Returns `nil` if the lock doesn't exist.

Discussion

This method is potentially useful to applications that want to use an age heuristic to decide if a lock is too old and should be broken.

If the creation date on the lock isn't the date on which you locked it, you've lost the lock: it's been broken since you last checked it.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSDistributedLock.h`

tryLock

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

- (`BOOL`)tryLock

Return Value

YES if the attempt to acquire the receiver was successful, otherwise NO.

Discussion

Raises `NSGenericException` if a file-system error occurs.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [unlock](#) (page 9)

Declared In

NSDistributedLock.h

unlock

Relinquishes the receiver.

- (void)unlock

Discussion

You should generally use the `unlock` method rather than [breakLock](#) (page 7) to release a lock.

An `NSGenericException` is raised if the receiver doesn't already exist.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [breakLock](#) (page 7)

Declared In

NSDistributedLock.h

Document Revision History

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

Date	Notes
2007-01-22	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

B

breakLock [instance method 7](#)

I

initWithPath: [instance method 7](#)

L

lockDate [instance method 8](#)

lockWithPath: [class method 6](#)

T

tryLock [instance method 8](#)

U

unlock [instance method 9](#)