NSDistributedLock

Inherits From:	NSObject
Conforms To:	NSObject (NSObject)
Declared In:	Foundation/NSDistributedLock.h

Class Description

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 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** method to attempt to acquire a lock. You should generally use the **unlock** method to release the lock rather than **breakLock**.

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** method and let you determine the polling frequency that makes sense for your application.

Method Types

Creating an NSDistributedLock	+ lockWithPath: – initWithPath:
Acquiring a lock	– tryLock
Relinquishing a lock	– breakLock – unlock
Getting lock information	– lockDate

Class Methods

lockWithPath:

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

Returns an NSDistributedLock object initialized to use the file system entry specified by *aPath* as the locking object. For applications to use the lock, this location in the file system must be accessible—and writable—to all hosts on which the applications might be running.

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

See also: – **initWithPath:**

Instance Methods

breakLock

– (void)**breakLock**

Forces the lock to be relinquished. 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** rather than **breakLock** to relinquish a lock.

Warning: Since **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** is invoked.

Raises NSGenericException if the lock could not be removed.

See also: – **unlock**

initWithPath:

– (NSDistributedLock *)**initWithPath:**(NSString *)*aPath*

Initializes a newly allocated NSDistributedLock object to use the file system entry specified by *aPath* as the lock. For applications to use the lock, this location in the file system must be accessible—and writable—to all hosts on which the applications might be running.

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

See also: – **lockWithPath:**

lockDate

– (NSDate *)**lockDate**

Returns the time that the lock occurred.

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. Returns **nil** if the lock doesn't exist.

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.

tryLock

– (BOOL)**tryLock**

Attempts to acquire the lock. Returns immediately with a value of YES if successful and NO otherwise.

Raises NSGenericException if a file system error occurs.

See also: – **unlock**

unlock

– (void)**unlock**

Relinquishes the lock. You should generally use the **unlock** method rather than **breakLock** to release a lock.

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

See also: – **breakLock**