
NSOperationQueue Class Reference

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



2008-11-19



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.

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

NSOperationQueue Class Reference 5

Overview	5
KVO-Compliant Properties	6
Threading Considerations	6
Tasks	6
Managing Operations in the Queue	6
Managing the Number of Running Operations	6
Suspending Operations	6
Instance Methods	7
addOperation:	7
cancelAllOperations	7
isSuspended	8
maxConcurrentOperationCount	8
operations	8
setMaxConcurrentOperationCount:	9
setSuspended:	9
waitUntilAllOperationsAreFinished	10
Constants	10
Concurrent Operation Constants	10

Document Revision History 11

Index 13

NSOperationQueue Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.5 and later.
Companion guide	Threading Programming Guide
Declared in	NSOperation.h
Related sample code	NSOperationSample

Overview

The `NSOperationQueue` class manages a set of `NSOperation` objects in a priority queue and regulates their execution. Operations remain in the queue until they are explicitly cancelled or finish executing. An application may create multiple operation queues, with each queue running up to its designated maximum number of operations.

A specific `NSOperation` object can be in only one operation queue at a time. Operations within a single queue coordinate their execution order using both priority levels and inter-operation object dependencies. Operation objects in different queues can coordinate their execution order using dependencies, which are not queue-specific.

Inter-operation dependencies provide an absolute execution order for operations. An operation object is not considered ready to execute until all of its dependent operations have finished executing. For operations that are ready to execute, the operation queue always executes the one with the highest priority relative to the other ready operations. For details on how to set priority levels and dependencies, see *NSOperation Class Reference*.

You should never manually start an operation while it is sitting in an operation queue. Once added, an operation stays in its queue until it finishes executing or is cancelled.

If the `isConcurrent` method of an operation returns `NO`, the operation queue automatically creates a new thread for that operation before running it. If the `isConcurrent` method returns `YES`, the operation object must create its own thread or otherwise configure its own runtime environment as part of its execution phase.

KVO-Compliant Properties

The `NSOperationQueue` class is key-value coding (KVC) and key-value observing (KVO) compliant. You can observe these properties as desired to control other parts of your application. The properties you can observe include the following:

- `operations` - read-only property
- `maxConcurrentOperationCount` - readable and writable property

For more information about key-value observing and how to attach observers to an object, see *Key-Value Observing Programming Guide*.

Threading Considerations

It is safe to use a single `NSOperationQueue` object from multiple threads without creating additional locks to synchronize access to that object.

Tasks

Managing Operations in the Queue

- [addOperation:](#) (page 7)
Adds the specified operation object to the receiver.
- [operations](#) (page 8)
Returns a new array containing the operations currently in the queue.
- [cancelAllOperations](#) (page 7)
Cancels all queued and executing operations.
- [waitUntilAllOperationsAreFinished](#) (page 10)
Blocks the current thread until all of the receiver's queued and executing operations finish executing.

Managing the Number of Running Operations

- [maxConcurrentOperationCount](#) (page 8)
Returns the maximum number of concurrent operations that the receiver can execute.
- [setMaxConcurrentOperationCount:](#) (page 9)
Sets the maximum number of concurrent operations that the receiver can execute.

Suspending Operations

- [setSuspended:](#) (page 9)
Modifies the execution of pending operations

- [isSuspended](#) (page 8)

Returns a Boolean value indicating whether the receiver is scheduling queued operations for execution.

Instance Methods

addOperation:

Adds the specified operation object to the receiver.

- (void)addOperation:(NSOperation *)*operation*

Parameters

operation

The operation object to be added to the queue. In memory-managed applications, this object is retained by the operation queue.

Discussion

An operation object can be in at most one operation queue at a time and cannot be added if it is currently executing or finished. This method throws an `NSInvalidArgumentException` exception if any of these conditions is true.

Once added, the specified *operation* remains in the queue until it is executed or cancelled.

Availability

Available in Mac OS X v10.5 and later.

See Also

- `cancel` (NSOperation)
- `isExecuting` (NSOperation)

Declared In

NSOperation.h

cancelAllOperations

Cancels all queued and executing operations.

- (void)cancelAllOperations

Discussion

This method sends a `cancel` message to all operations currently in the queue or executing. Queued operations are cancelled before they begin executing. If an operation is already executing, it is up to that operation to recognize the cancellation and stop what it is doing.

Availability

Available in Mac OS X v10.5 and later.

See Also

`cancel` (NSOperation)

Declared In

NSOperation.h

isSuspended

Returns a Boolean value indicating whether the receiver is scheduling queued operations for execution.

- (BOOL)isSuspended

Return Value

NO if operations are being scheduled for execution; otherwise, YES.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [setSuspended:](#) (page 9)

Declared In

NSOperation.h

maxConcurrentOperationCount

Returns the maximum number of concurrent operations that the receiver can execute.

- (NSInteger)maxConcurrentOperationCount

Return Value

The maximum number of concurrent operations set explicitly on the receiver using the `setMaxConcurrentOperationCount:` method.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [setMaxConcurrentOperationCount:](#) (page 9)

Declared In

NSOperation.h

operations

Returns a new array containing the operations currently in the queue.

- (NSArray *)operations

Return Value

A new array object containing the `NSOperation` objects in the order in which they were added to the queue.

Availability

Available in Mac OS X v10.5 and later.

Declared In

NSOperation.h

setMaxConcurrentOperationCount:

Sets the maximum number of concurrent operations that the receiver can execute.

- (void)setMaxConcurrentOperationCount:(NSInteger)count

Parameters

count

The maximum number of concurrent operations. Specify the value `NSOperationQueueDefaultMaxConcurrentOperationCount` if you want the receiver to choose an appropriate value based on the number of available processors and other relevant factors.

Discussion

The specified value affects only the receiver and the operations in its queue. Other operation queue objects can also execute their maximum number of operations in parallel.

Reducing the number of concurrent operations does not affect any operations that are currently executing. If you specify the value `NSOperationQueueDefaultMaxConcurrentOperationCount` (which is recommended), the maximum number of operations can change dynamically based on system conditions.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [maxConcurrentOperationCount](#) (page 8)

Declared In

NSOperation.h

setSuspended:

Modifies the execution of pending operations

- (void)setSuspended:(BOOL)suspend

Parameters

suspend

If YES, the queue stops scheduling queued operations for execution. If NO, the queue begins scheduling operations again.

Discussion

This method suspends or restarts the execution of queued operations only. It does not have any impact on the state of currently running operations. Running operations continue to run until their natural termination or until they are explicitly cancelled.

Availability

Available in Mac OS X v10.5 and later.

See Also

- [isSuspended](#) (page 8)

Declared In

NSOperation.h

waitUntilAllOperationsAreFinished

Blocks the current thread until all of the receiver's queued and executing operations finish executing.

```
- (void)waitUntilAllOperationsAreFinished
```

Discussion

When called, this method blocks the current thread and waits for the receiver's current and pending operations to finish executing. While the thread is blocked, the receiver continues to launch already queued operations and monitor those that are executing. During this time, the current thread cannot add operations to the queue, but other threads may. Once all of the pending operations are finished, this method returns.

Availability

Available in Mac OS X v10.5 and later.

Declared In

NSOperation.h

Constants

Concurrent Operation Constants

Indicates the number of supported concurrent operations.

```
enum {
    NSOperationQueueDefaultMaxConcurrentOperationCount = -1
};
```

Constants

NSOperationQueueDefaultMaxConcurrentOperationCount

The default maximum number of operations is determined dynamically by the `NSOperationQueue` object based on current system conditions.

Available in Mac OS X v10.5 and later.

Declared in `NSOperation.h`.

Declared In

NSOperation.h

Document Revision History

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

Date	Notes
2008-11-19	Updated the guidance related to KVO-compliant properties.
2008-10-15	Clarified ownership of operation objects when added to a queue.
2007-04-30	New document describing the methods for managing operation objects.

REVISION HISTORY

Document Revision History

Index

A

`addOperation:` instance method [7](#)

C

`cancelAllOperations` instance method [7](#)
`Concurrent Operation Constants` [10](#)

I

`isSuspended` instance method [8](#)

M

`maxConcurrentOperationCount` instance method [8](#)

N

`NSOperationQueueDefaultMaxConcurrentOperationCount`
constant [10](#)

O

`operations` instance method [8](#)

S

`setMaxConcurrentOperationCount:` instance method
[9](#)
`setSuspended:` instance method [9](#)

W

`waitUntilAllOperationsAreFinished` instance
method [10](#)