
NSError Class Reference





Apple Inc.
© 2009 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

NSError Class Reference 5

Overview	5
Adopted Protocols	5
Tasks	6
Creating and Raising an NSError Object	6
Querying an NSError Object	6
Getting Exception Stack Frames	6
Class Methods	6
exceptionWithName:reason:userInfo:	6
raise:format:	7
raise:format:arguments:	8
Instance Methods	8
callStackReturnAddresses	8
callStackSymbols	9
initWithName:reason:userInfo:	9
name	10
raise	10
reason	11
userInfo	11

Document Revision History 13

Index 15

NSError Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.0 and later.
Companion guide	Exception Programming Topics for Cocoa
Declared in	NSError.h
Related sample code	CoreRecipes EnhancedAudioBurn QuickLookSketch Sketch+Accessibility Sketch-112

Overview

`NSError` is used to implement exception handling and contains information about an exception. An exception is a special condition that interrupts the normal flow of program execution. Each application can interrupt the program for different reasons. For example, one application might interpret saving a file in a directory that is write-protected as an exception. In this sense, the exception is equivalent to an error. Another application might interpret the user's key-press (for example, Control-C) as an exception: an indication that a long-running process should be aborted.

Note: The exception handling mechanism uses `longjmp` to control the flow of execution. Any code written for an application that uses exception handling is therefore subject to the restrictions associated with this functionality. See your compiler documentation for more information on the `longjmp` function.

Adopted Protocols

- NSCoding
- `encodeWithCoder:`
 - `initWithCoder:`

NSCopying

- `copyWithZone:`

Tasks

Creating and Raising an NSException Object

- + `exceptionWithName:reason:userInfo:` (page 6)
Creates and returns an exception object .
- + `raise:format:` (page 7)
A convenience method that creates and raises an exception.
- + `raise:format:arguments:` (page 8)
Creates and raises an exception with the specified name, reason, and arguments.
- `initWithName:reason:userInfo:` (page 9)
Initializes and returns a newly allocated exception object.
- `raise` (page 10)
Raises the receiver, causing program flow to jump to the local exception handler.

Querying an NSException Object

- `name` (page 10)
Returns an `NSString` object used to uniquely identify the receiver.
- `reason` (page 11)
Returns an `NSString` object containing a “human-readable” reason for the receiver.
- `userInfo` (page 11)
Returns an `NSDictionary` object containing application-specific data pertaining to the receiver.

Getting Exception Stack Frames

- `callStackReturnAddresses` (page 8)
Returns the call return addresses related to a raised exception.
- `callStackSymbols` (page 9)
Returns an array containing the current call symbols.

Class Methods

`exceptionWithName:reason:userInfo:`

Creates and returns an exception object .

```
+ (NSException *)exceptionWithName:(NSString *)name reason:(NSString *)reason
    userInfo:(NSDictionary *)userInfo
```

Parameters

name

The name of the exception.

reason

A human-readable message string summarizing the reason for the exception.

userInfo

A dictionary containing user-defined information relating to the exception

Return Value

The created `NSException` object or `nil` if the object couldn't be created.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [initWithName:reason:userInfo:](#) (page 9)
- [name](#) (page 10)
- [reason](#) (page 11)
- [userInfo](#) (page 11)

Related Sample Code

- ClipboardViewer
- Core Data HTML Store
- CoreRecipes

Declared In

`NSException.h`

raise:format:

A convenience method that creates and raises an exception.

```
+ (void)raise:(NSString *)name format:(NSString *)format, ...
```

Parameters

name

The name of the exception.

format,

A human-readable message string (that is, the exception reason) with conversion specifications for the variable arguments that follow.

...

Variable information to be inserted into the formatted exception reason (in the manner of `printf`).

Discussion

The user-defined information is `nil` for the generated exception object.

Availability

Available in Mac OS X v10.0 and later.

See Also+ [raise:format:arguments:](#) (page 8)- [raise](#) (page 10)**Related Sample Code**

EnhancedAudioBurn

QuickLookSketch

Sketch+Accessibility

Sketch-112

TemperatureTester

Declared In

NSException.h

raise:format:arguments:

Creates and raises an exception with the specified name, reason, and arguments.

+ (void)raise:(NSString *)*name* format:(NSString *)*format* arguments:(va_list)*argList***Parameters***name*

The name of the exception.

*format*A human-readable message string (that is, the exception reason) with conversion specifications for the variable arguments in *argList*.*argList*Variable information to be inserted into the formatted exception reason (in the manner of `vprintf`).**Discussion**The user-defined dictionary of the generated object is `nil`.**Availability**

Available in Mac OS X v10.0 and later.

See Also+ [raise:format:](#) (page 7)- [raise](#) (page 10)**Declared In**

NSException.h

Instance Methods

callStackReturnAddresses

Returns the call return addresses related to a raised exception.

- (NSArray *)callStackReturnAddresses

Return Value

An array of `NSNumber` objects encapsulating `NSUInteger` values. Each value is a call frame return address. The array of stack frames starts at the point at which the exception was first raised, with the first items being the most recent stack frames.

Discussion

`NSException` subclasses posing as the `NSException` class or subclasses or other API elements that interfere with the exception-raising mechanism may not get this information.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`NSException.h`

callStackSymbols

Returns an array containing the current call symbols.

```
- (NSArray *)callStackSymbols
```

Return Value

An array containing the current call stack symbols.

Discussion

This method returns an array of strings describing the call stack backtrace at the moment the exception was first raised. The format of each string is non-negotiable and is determined by the `backtrace_symbols()` API

Availability

Available in Mac OS X v10.6 and later.

Declared In

`NSException.h`

initWithName:reason:userInfo:

Initializes and returns a newly allocated exception object.

```
- (id)initWithName:(NSString *)name reason:(NSString *)reason userInfo:(NSDictionary *)userInfo
```

Parameters

name

The name of the exception.

reason

A human-readable message string summarizing the reason for the exception.

userInfo

A dictionary containing user-defined information relating to the exception

Return Value

The created `NSException` object or `nil` if the object couldn't be created.

Discussion

This is the designated initializer.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [exceptionWithName:reason:userInfo:](#) (page 6)

Declared In

NSException.h

name

Returns an `NSString` object used to uniquely identify the receiver.

- (NSString *)name

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [exceptionWithName:reason:userInfo:](#) (page 6)

- [initWithName:reason:userInfo:](#) (page 9)

Declared In

NSException.h

raise

Raises the receiver, causing program flow to jump to the local exception handler.

- (void)raise

Discussion

All other methods that raise an exception invoke this method, so set a breakpoint here if you are debugging exceptions. When there are no exception handlers in the exception handler stack, unless the exception is raised during the posting of a notification, this method calls the uncaught exception handler, in which last-minute logging can be performed. The program then terminates, regardless of the actions taken by the uncaught exception handler.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [raise:format:](#) (page 7)

+ [raise:format:arguments:](#) (page 8)

Related Sample Code

Core Data HTML Store

Declared In

NSException.h

reason

Returns an `NSString` object containing a “human-readable” reason for the receiver.

- (`NSString *`)reason

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [exceptionWithName:reason:userInfo:](#) (page 6)

- [initWithName:reason:userInfo:](#) (page 9)

Related Sample Code

CoreRecipes

Declared In

`NSException.h`

userInfo

Returns an `NSDictionary` object containing application-specific data pertaining to the receiver.

- (`NSDictionary *`)userInfo

Discussion

Returns `nil` if no application-specific data exists. As an example, if a method’s return value caused the exception to be raised, the return value might be available to the exception handler through this method.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [exceptionWithName:reason:userInfo:](#) (page 6)

- [initWithName:reason:userInfo:](#) (page 9)

Declared In

`NSException.h`

Document Revision History

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

Date	Notes
2009-08-28	Updated for Mac OS X v 10.6. Added callStackSymbols method.
2007-01-30	Updated for Mac OS X version 10.5
2006-05-23	First publication of this content as a separate document.

REVISION HISTORY

Document Revision History

Index

C

callStackReturnAddresses **instance method** [8](#)
callStackSymbols **instance method** [9](#)

E

exceptionWithName:reason:userInfo: **class method**
[6](#)

I

initWithName:reason:userInfo: **instance method**
[9](#)

N

name **instance method** [10](#)

R

raise **instance method** [10](#)
raise:format: **class method** [7](#)
raise:format:arguments: **class method** [8](#)
reason **instance method** [11](#)

U

userInfo **instance method** [11](#)