
Exception Handling Framework Reference

Data Management: Event Handling



2006-10-03



Apple Inc.
© 2004, 2006 Apple Computer, 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, Mac OS, and Objective-C 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

Introduction **Introduction** 5

Part I **Classes** 7

NExceptionHandler Class Reference 9

Overview 9

Tasks 10

Class Methods 10

Instance Methods 11

Delegate Methods 13

Constants 14

Document Revision History 19

Index 21

Introduction

Framework	/System/Library/Frameworks/ExceptionHandling.framework
Header file directories	/System/Library/Frameworks/ExceptionHandling.framework/Headers
Declared in	NSEExceptionHandler.h

This collection of documents provides the API reference for the Exception Handling framework. This framework provides facilities for monitoring and debugging exceptional conditions in Objective-C code.

Currently only one class reference is part of this collection: the reference for the `NSEExceptionHandler` class, which is defined in `NSEExceptionHandler.h`.

Classes

NSEExceptionHandler Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/ExceptionHandling.framework
Declared in	ExceptionHandling/NSEExceptionHandler.h
Availability	Mac OS X v10.0
Companion guide	Exception Programming Topics for Cocoa

Overview

The `NSEExceptionHandler` class provides facilities for monitoring and debugging exceptional conditions in Objective-C programs. It works by installing a special uncaught exception handler via the `NSSetUncaughtExceptionHandler` function. Consequently, to use the services of `NSEExceptionHandler`, you must not install your own custom uncaught exception handler.

To use these services, you set a bit mask in the singleton `NSEExceptionHandler` instance and, optionally, a delegate. The constants comprising the bit mask indicate the type of exception to be monitored and the behavior of the `NSEExceptionHandler` object (or, simply, the exception handler). The delegate is asked to approve the logging and handling of each monitored `NSEExceptionHandler` object.

The constants for configuring exception handler behavior can be categorized in several ways:

- Uncaught exceptions versus caught exceptions—or, more accurately, exceptions that would be caught (for example, by the top-level handler)
- Exception type or cause: system exceptions (such as invalid memory accesses), Objective-C runtime errors (such as messages sent to freed objects), and other exceptions
- Exception handler behavior: logging the exception (including a stack trace) to the console, handling the exception, and suspending program execution so the debugger can be attached

The way the exception handler handles an exception depends on the type of exception; the exception handler converts system exceptions and runtime errors into `NSEException` objects with a stack trace embedded in their `userInfo` dictionary; for all other uncaught exceptions, it terminates the thread on which they occur. The constants used to configure an `NSEExceptionHandler` object are described in [Logging and Handling Constants](#) (page 14) and [System Hang Constants](#) (page 16).

The `defaults` command-line system also allows you to set values corresponding to the `enum` constants used to configure the exception handler; see “Controlling Application Response to Exceptions” for details.

Tasks

Getting the Default Exception Handler

- + [defaultExceptionHandler](#) (page 10)
Returns the singleton `NSExceptionHandler` instance.

Getting and Setting Exception Masks

- [exceptionHandlingMask](#) (page 11)
Returns a bit mask representing the types of exceptions monitored by the receiver and its handling and logging behavior.
- [exceptionHangingMask](#) (page 11)
Returns a bit mask representing the types of exceptions that will halt execution for debugging.
- [setExceptionHandlingMask:](#) (page 12)
Sets the bit mask of constants specifying the types of exceptions monitored by the receiver and its handling and logging behavior.
- [setExceptionHangingMask:](#) (page 12)
Sets the bit mask of constants specifying the types of exceptions that will halt execution for debugging.

Getting and Setting the Delegate

- [delegate](#) (page 11)
Returns the delegate of the `NSExceptionHandler` object.
- [setDelegate:](#) (page 12)
Sets the delegate of the `NSExceptionHandler` object.

Logging and handling exceptions

- [exceptionHandler:shouldHandleException:mask:](#) (page 13) *delegate method*
Implemented by the delegate to evaluate whether the delegating `NSExceptionHandler` instance should handle a given exception.
- [exceptionHandler:shouldLogException:mask:](#) (page 13) *delegate method*
Implemented by the delegate to evaluate whether the delegating `NSExceptionHandler` instance should log a given exception.

Class Methods

`defaultExceptionHandler`

Returns the singleton `NSExceptionHandler` instance.

+ (NSEExceptionHandler *)defaultExceptionHandler

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSEExceptionHandler.h

Instance Methods

delegate

Returns the delegate of the NSEExceptionHandler object.

- (id)delegate

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSEExceptionHandler.h

exceptionHandlingMask

Returns a bit mask representing the types of exceptions monitored by the receiver and its handling and logging behavior.

- (unsigned int)exceptionHandlingMask

Return Value

A bit mask composed of one or more constants specifying the types of exceptions monitored and whether they are handled or logged (or both). See [Logging and Handling Constants](#) (page 14) for information about the constants.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSEExceptionHandler.h

exceptionHangingMask

Returns a bit mask representing the types of exceptions that will halt execution for debugging.

- (unsigned int)exceptionHangingMask

Return Value

A bit mask composed of one or more constants specifying the types of exceptions that will halt execution for debugging. See [System Hang Constants](#) (page 16) for information about the constants.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSExceptionHandler.h

setDelegate:

Sets the delegate of the NSExceptionHandler object.

- (void)setDelegate:(id)anObject

Parameters

anObject

The object to receive the delegation messages described in [“Logging and handling exceptions”](#) (page 10)

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSExceptionHandler.h

setExceptionHandlerMask:

Sets the bit mask of constants specifying the types of exceptions monitored by the receiver and its handling and logging behavior.

- (void)setExceptionHandlerMask:(unsigned int)aMask

Parameters

aMask

A bit mask composed of one or more constants specifying the types of exceptions monitored and whether they are handled or logged (or both). You specify multiple constants by performing a bitwise-OR operation. See [Logging and Handling Constants](#) (page 14) for information about the constants.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSExceptionHandler.h

setExceptionHangingMask:

Sets the bit mask of constants specifying the types of exceptions that will halt execution for debugging.

- (void)setExceptionHangingMask:(unsigned int)aMask

Parameters

aMask

A bit mask composed of one or more constants specifying the types of exceptions that will halt execution for debugging. You specify multiple constants by performing a bitwise-OR operation. See [System Hang Constants](#) (page 16) for information about the constants.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSEExceptionHandler.h

Delegate Methods

exceptionHandler:shouldHandleException:mask:

Implemented by the delegate to evaluate whether the delegating NSEExceptionHandler instance should handle a given exception.

```
- (BOOL)exceptionHandler:(NSEExceptionHandler *)sender
    shouldHandleException:(NSEException *)exception mask:(unsigned int)aMask
```

Parameters

sender

The NSEExceptionHandler object sending the message.

exception

An NSEException object describing the exception to be evaluated.

aMask

The bit mask indicating the types of exceptions handled by the NSEExceptionHandler object. See [Logging and Handling Constants](#) (page 14) and [System Hang Constants](#) (page 16) for descriptions of the possible enum constants.

Return Value

YES to have the NSEExceptionHandler object handle the exception, NO otherwise.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSEExceptionHandler.h

exceptionHandler:shouldLogException:mask:

Implemented by the delegate to evaluate whether the delegating NSEExceptionHandler instance should log a given exception.

```
- (BOOL)exceptionHandler:(NSEExceptionHandler *)sender shouldLogException:(NSEException
    *)exception mask:(unsigned int)aMask
```

Parameters*sender*The `NSExceptionHandler` object sending the message.*exception*An `NSException` object describing the exception to be evaluated.*aMask*The bit mask indicating the types of exceptions logged by the `NSExceptionHandler` object. See [Logging and Handling Constants](#) (page 14) and [System Hang Constants](#) (page 16) for descriptions of the possible `enum` constants.**Return Value**YES to have the `NSExceptionHandler` object log the exception, NO otherwise.**Availability**

Available in Mac OS X v10.0 and later.

Declared In`NSExceptionHandler.h`

Constants

Logging and Handling Constants

Use one or more of the following constants in the parameter of [setExceptionHandlerMask:](#) (page 12) to specify the types of exceptions that the exception handler should monitor and whether it should handle or log them.

```
enum {
    NSLogUncaughtExceptionMask          = 1 << 0,
    NSHandleUncaughtExceptionMask       = 1 << 1,
    NSLogUncaughtSystemExceptionMask    = 1 << 2,
    NSHandleUncaughtSystemExceptionMask = 1 << 3,
    NSLogUncaughtRuntimeErrorMask       = 1 << 4,
    NSHandleUncaughtRuntimeErrorMask    = 1 << 5,
    NSLogTopLevelExceptionMask          = 1 << 6,
    NSHandleTopLevelExceptionMask       = 1 << 7,
    NSLogOtherExceptionMask             = 1 << 8,
    NSHandleOtherExceptionMask          = 1 << 9
};
```

Constants`NSLogUncaughtExceptionMask`

The exception handler logs uncaught exceptions.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.`NSHandleUncaughtExceptionMask`

The exception handler handles uncaught exceptions by terminating the thread in which they occur.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

`NSLogUncaughtSystemExceptionMask`

The exception handler logs uncaught system exceptions.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

`NSHandleUncaughtSystemExceptionMask`

The exception handler handles uncaught system exceptions by converting them to `NSException` objects containing a stack trace.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

`NSLogUncaughtRuntimeErrorMask`

The exception handler logs uncaught runtime errors.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

`NSHandleUncaughtRuntimeErrorMask`

The exception handler handles uncaught runtime errors by converting them to `NSException` objects containing a stack trace.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

`NSLogTopLevelExceptionMask`

The exception handler logs exceptions that would be caught by the top-level handler.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

`NSHandleTopLevelExceptionMask`

The exception handler handles exceptions caught by the top-level handler by converting them to `NSException` objects containing a stack trace.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

`NSLogOtherExceptionMask`

The exception handler logs exceptions caught by handlers lower than the top-level handler.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

`NSHandleOtherExceptionMask`

The exception handler handles exceptions caught by handlers lower than the top-level handler by converting them to `NSException` objects containing a stack trace.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

Discussion

When exception-handling domains are nested, [NSLogTopLevelExceptionMask](#) (page 15) and [NSHandleTopLevelExceptionMask](#) (page 15) deal with exceptions that would make it to the top two levels of exception handlers. In the main thread of a Cocoa application, the top-level handler is the global `NSApplication` instance.

Declared In

`ExceptionHandler/ExceptionHandler.h`

System Hang Constants

Use one or more of the following constants in the parameter of [setExceptionHandlerMask:](#) (page 12) to specify the types of exceptions that cause the exception to halt execution so a debugger can be attached.

```
enum {
    NSHangOnUncaughtExceptionMask      = 1 << 0,
    NSHangOnUncaughtSystemExceptionMask = 1 << 1,
    NSHangOnUncaughtRuntimeErrorMask   = 1 << 2,
    NSHangOnTopLevelExceptionMask      = 1 << 3,
    NSHangOnOtherExceptionMask         = 1 << 4
};
```

Constants

`NSHangOnUncaughtExceptionMask`

The exception handler suspends execution when it detects an uncaught exception (other than a system exception or runtime error).

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

`NSHangOnUncaughtSystemExceptionMask`

The exception handler suspends execution when it detects an uncaught system exception.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

`NSHangOnUncaughtRuntimeErrorMask`

The exception handler suspends execution when it detects an uncaught runtime error.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

`NSHangOnTopLevelExceptionMask`

The exception handler suspends execution when it detects an exception that would be handled by the top-level handler.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

`NSHangOnOtherExceptionMask`

The exception handler suspends execution when it detects an exception that would be handled by an object other than the top-level handler.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

Discussion

When exception-handling domains are nested, [NSHangOnTopLevelExceptionMask](#) (page 16) deals with exceptions that would make it to the top two levels of exception handlers. In the main thread of a Cocoa application, the top-level handler is the global `NSApplication` instance.

Declared In

`ExceptionHandler/ExceptionHandler.h`

Mask Definitions

The following `#define` constants are conveniences for specifying complete sets of exception-handling enum constants.

NSHangOnEveryExceptionMask
 NSLogAndHandleEveryExceptionMask

Constants

NSHangOnEveryExceptionMask

Combines via bitwise-OR all the constants listed in [System Hang Constants](#) (page 16).

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

NSLogAndHandleEveryExceptionMask

Combines via bitwise-OR all the constants listed in [Logging and Handling Constants](#) (page 14).

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

Declared In

`ExceptionHandler/ExceptionHandler.h`

Exception Global String Constants

Two of the following global string constants identify exceptions generated by the framework for Objective-C runtime errors and system exceptions such as invalid memory accesses. The other constant is used as a key to access the stack trace in the `userInfo` dictionary of an `NSException` object, when requested.

```
EXCEPTIONHANDLING_EXPORT NSString *NSUncaughtSystemExceptionException;
EXCEPTIONHANDLING_EXPORT NSString *NSUncaughtRuntimeErrorException;
EXCEPTIONHANDLING_EXPORT NSString *NSStackTraceKey;
```

Constants

NSUncaughtSystemExceptionException

Identifies an uncaught system exception.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

NSUncaughtRuntimeErrorException

Identifies an Objective-C runtime error.

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

NSStackTraceKey

The key for fetching the stack trace (an `NSString` object) in the `userInfo` dictionary of the `NSException` object passed into one of the delegate methods described in [“Logging and handling exceptions”](#) (page 10).

Available in Mac OS X v10.0 and later.

Declared in `NSExceptionHandler.h`.

Declared In

`ExceptionHandler/ExceptionHandler.h`

Document Revision History

This table describes the changes to *Exception Handling Framework Reference*.

Date	Notes
2006-10-03	New collection that describes the API used to monitor and debug exceptional conditions in Objective-C code.

REVISION HISTORY

Document Revision History

Index

D

defaultExceptionHandler **class method** 10
delegate **instance method** 11

E

Exception Global String Constants 17
exceptionHandler:shouldHandleException:mask:
 <NSObject> **delegate method** 13
exceptionHandler:shouldLogException:mask:
 <NSObject> **delegate method** 13
exceptionHandlingMask **instance method** 11
exceptionHangingMask **instance method** 11

L

Logging and Handling Constants 14

M

Mask Definitions 16

N

NSHandleOtherExceptionMask **constant** 15
NSHandleTopLevelExceptionMask **constant** 15
NSHandleUncaughtExceptionMask **constant** 14
NSHandleUncaughtRuntimeErrorMask **constant** 15
NSHandleUncaughtSystemExceptionMask **constant**
 15
NSHangOnEveryExceptionMask **constant** 17
NSHangOnOtherExceptionMask **constant** 16
NSHangOnTopLevelExceptionMask **constant** 16
NSHangOnUncaughtExceptionMask **constant** 16

NSHangOnUncaughtRuntimeErrorMask **constant** 16
NSHangOnUncaughtSystemExceptionMask **constant**
 16
NSLogAndHandleEveryExceptionMask **constant** 17
NSLogOtherExceptionMask **constant** 15
NSLogTopLevelExceptionMask **constant** 15
NSLogUncaughtExceptionMask **constant** 14
NSLogUncaughtRuntimeErrorMask **constant** 15
NSLogUncaughtSystemExceptionMask **constant** 15
NSStackTraceKey **constant** 17
NSUncaughtRuntimeErrorException **constant** 17
NSUncaughtSystemExceptionException **constant** 17

S

setDelegate: **instance method** 12
setExceptionHandlingMask: **instance method** 12
setExceptionHangingMask: **instance method** 12
System Hang Constants 16