
Foundation Functions Reference



2010-03-24



Apple Inc.
© 2010 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, Bonjour, Cocoa, iPhone, Keychain, Mac, Mac OS, Objective-C, and Pages are trademarks of Apple Inc., registered in the United States and other countries.

IOS is a trademark or registered trademark of Cisco in the U.S. and other countries and is used under license.

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

Foundation Functions Reference 9

Overview	9
Functions by Task	9
Assertions	9
Bundles	10
Byte Ordering	10
Decimals	12
Exception Handling	13
Java Setup	13
Hash Tables	14
HFS File Types	15
Managing Map Tables	15
Managing Object Allocation and Deallocation	16
Interacting with the Objective-C Runtime	16
Logging Output	17
Managing File Paths	17
Managing Points	17
Managing Ranges	17
Managing Rectangles	18
Managing Sizes	19
Uncaught Exception Handlers	19
Managing Memory	20
Managing Zones	20
Functions	21
NSAllHashTableObjects	21
NSAllMapTableKeys	21
NSAllMapTableValues	22
NSAllocateCollectable	22
NSAllocateMemoryPages	23
NSAllocateObject	23
NSAssert	24
NSAssert1	25
NSAssert2	26
NSAssert3	27
NSAssert4	28
NSAssert5	29
NSCAssert	30
NSCAssert1	31
NSCAssert2	31
NSCAssert3	32
NSCAssert4	33

NSCAssert5	34
NSClassFromString	34
NSCompareHashTables	35
NSCompareMapTables	35
NSContainsRect	36
NSConvertHostDoubleToSwapped	36
NSConvertHostFloatToSwapped	37
NSConvertSwappedDoubleToHost	37
NSConvertSwappedFloatToHost	37
NSCopyHashTableWithZone	38
NSCopyMapTableWithZone	38
NSCopyMemoryPages	39
NSCopyObject	39
NSCountHashTable	40
NSCountMapTable	41
NSCParameterAssert	41
NSCreateHashTable	42
NSCreateHashTableWithZone	42
NSCreateMapTable	43
NSCreateMapTableWithZone	43
NSCreateZone	44
NSDeallocateMemoryPages	45
NSDeallocateObject	45
NSDecimalAdd	45
NSDecimalCompact	46
NSDecimalCompare	47
NSDecimalCopy	47
NSDecimalDivide	47
NSDecimalIsNotANumber	48
NSDecimalMultiply	48
NSDecimalMultiplyByPowerOf10	49
NSDecimalNormalize	49
NSDecimalPower	50
NSDecimalRound	51
NSDecimalString	51
NSDecimalSubtract	52
NSDecrementExtraRefCountWasZero	52
NSDefaultMallocZone	53
NSDivideRect	53
NSEndHashTableEnumeration	54
NSEndMapTableEnumeration	54
NSEnumerateHashTable	54
NSEnumerateMapTable	55
NSEqualPoints	55
NSEqualRanges	56
NSEqualRects	56

NSEqualSizes	57
NSExtraRefCount	57
NSFileTypeForHFSTypeCode	58
NSFreeHashTable	58
NSFreeMapTable	59
NSFullUserName	59
NSGetSizeAndAlignment	59
NSGetUncaughtExceptionHandler	60
NSHashGet	60
NSHashInsert	61
NSHashInsertIfAbsent	61
NSHashInsertKnownAbsent	62
NSHashRemove	62
NSHeight	63
NSHFSTypeCodeFromFileType	63
NSHFSTypeOfFile	64
NSHomeDirectory	64
NSHomeDirectoryForUser	65
NSHostByteOrder	65
NSIncrementExtraRefCount	65
NSInsetRect	66
NSIntegralRect	67
NSIntersectionRange	67
NSIntersectionRect	68
NSIntersectsRect	68
NSIsEmptyRect	69
NSJavaBundleCleanup	69
NSJavaBundleSetup	70
NSJavaClassesForBundle	70
NSJavaClassesFromPath	70
NSJavaNeedsToLoadClasses	71
NSJavaNeedsVirtualMachine	71
NSJavaObjectNamedInPath	72
NSJavaProvidesClasses	72
NSJavaSetup	72
NSJavaSetupVirtualMachine	73
NSLocalizedString	73
NSLocalizedStringFromTable	74
NSLocalizedStringFromTableInBundle	75
NSLocalizedStringWithDefaultValue	75
NSLocationInRange	76
NSLog	76
NSLogPageSize	77
NSLogv	77
NSMakeCollectable	78
NSMakePoint	79

NSMakeRange	79
NSMakeRect	79
NSMakeSize	80
NSMapGet	80
NSMapInsert	81
NSMapInsertIfAbsent	82
NSMapInsertKnownAbsent	82
NSMapMember	83
NSMapRemove	83
NSMaxRange	84
NSMaxX	84
NSMaxY	85
NSMidX	85
NSMidY	86
NSMinX	86
NSMinY	87
NSMouseInRect	87
NSNextHashEnumeratorItem	88
NSNextMapEnumeratorPair	89
NSOffsetRect	89
NSOpenStepRootDirectory	90
NSPageSize	90
NSParameterAssert	91
NSPointFromCGPoint	91
NSPointFromString	92
NSPointInRect	93
NSPointToCGPoint	93
NSProtocolFromString	94
NSRangeFromString	94
NSReallocateCollectable	95
NSRealMemoryAvailable	95
NSRectFromCGRect	96
NSRectFromString	96
NSRectToCGRect	97
NSRecycleZone	97
NSResetHashTable	98
NSResetMapTable	98
NSRoundDownToMultipleOfPageSize	99
NSRoundUpToMultipleOfPageSize	99
NSSearchPathForDirectoriesInDomains	100
NSSelectorFromString	100
NSSetUncaughtExceptionHandler	101
NSSetZoneName	101
NSShouldRetainWithZone	102
NSSizeFromCGSize	102
NSSizeFromString	103

NSSizeToCGSize	103
NSStringFromClass	104
NSStringFromHashTable	104
NSStringFromMapTable	105
NSStringFromPoint	105
NSStringFromProtocol	106
NSStringFromRange	106
NSStringFromRect	107
NSStringFromSelector	107
NSStringFromSize	108
NSSwapBigDoubleToHost	108
NSSwapBigFloatToHost	109
NSSwapBigIntToHost	109
NSSwapBigLongLongToHost	110
NSSwapBigLongToHost	110
NSSwapBigShortToHost	110
NSSwapDouble	111
NSSwapFloat	111
NSSwapHostDoubleToBig	112
NSSwapHostDoubleToLittle	112
NSSwapHostFloatToBig	113
NSSwapHostFloatToLittle	113
NSSwapHostIntToBig	113
NSSwapHostIntToLittle	114
NSSwapHostLongLongToBig	114
NSSwapHostLongLongToLittle	115
NSSwapHostLongToBig	115
NSSwapHostLongToLittle	116
NSSwapHostShortToBig	116
NSSwapHostShortToLittle	117
NSSwapInt	117
NSSwapLittleDoubleToHost	117
NSSwapLittleFloatToHost	118
NSSwapLittleIntToHost	118
NSSwapLittleLongLongToHost	119
NSSwapLittleLongToHost	119
NSSwapLittleShortToHost	120
NSSwapLong	120
NSSwapLongLong	121
NSSwapShort	121
NSTemporaryDirectory	121
NSUnionRange	122
NSUnionRect	122
NSUserName	123
NSWidth	123
NSZoneCalloc	124

NSZoneFree 125
NSZoneFromPointer 125
NSZoneMalloc 126
NSZoneName 126
NSZoneRealloc 127
NS_DURING 127
NS_ENDHANDLER 128
NS_HANDLER 128
NS_VALUEReturn 128
NS_VOIDRETURN 129

Document Revision History 131

Foundation Functions Reference

Framework:	Foundation/Foundation.h
Declared in	NSBundle.h NSByteOrder.h NSDecimal.h NSException.h NSGeometry.h NSHFSFileTypes.h NSHashTable.h NSJavaSetup.h NSMapTable.h NSObjCRuntime.h NSObject.h NSPathUtilities.h NSRange.h NSZone.h

Overview

This chapter describes the functions and function-like macros defined in the Foundation Framework.

Functions by Task

Assertions

For additional information about Assertions, see *Assertions and Logging Programming Guide*.

[NSAssert](#) (page 24)

Generates an assertion if a given condition is false.

[NSAssert1](#) (page 25)

Generates an assertion if a given condition is false.

[NSAssert2](#) (page 26)

Generates an assertion if a given condition is false.

[NSAssert3](#) (page 27)

Generates an assertion if a given condition is false.

[NSAssert4](#) (page 28)

Generates an assertion if a given condition is false.

[NSAssert5](#) (page 29)

Generates an assertion if a given condition is false.

[NSCAssert](#) (page 30)

Generates an assertion if the given condition is false.

[NSCAssert1](#) (page 31)

`NSCAssert1` is one of a series of macros that generate assertions if the given condition is false.

[NSCAssert2](#) (page 31)

`NSCAssert2` is one of a series of macros that generate assertions if the given condition is false.

[NSCAssert3](#) (page 32)

`NSCAssert3` is one of a series of macros that generate assertions if the given condition is false.

[NSCAssert4](#) (page 33)

`NSCAssert4` is one of a series of macros that generate assertions if the given condition is false.

[NSCAssert5](#) (page 34)

`NSCAssert5` is one of a series of macros that generate assertions if the given condition is false.

[NSCParameterAssert](#) (page 41)

Evaluates the specified parameter.

[NSParameterAssert](#) (page 91)

Validates the specified parameter.

Bundles

For additional information on generating strings files see Strings Files in *Internationalization Programming Topics*.

[NSLocalizedString](#) (page 73)

Returns a localized version of a string.

[NSLocalizedStringFromTable](#) (page 74)

Returns a localized version of a string.

[NSLocalizedStringFromTableInBundle](#) (page 75)

Returns a localized version of a string.

[NSLocalizedStringWithDefaultValue](#) (page 75)

Returns a localized version of a string.

Byte Ordering

[NSConvertHostDoubleToSwapped](#) (page 36)

Performs a type conversion.

[NSConvertHostFloatToSwapped](#) (page 37)

Performs a type conversion.

[NSConvertSwappedDoubleToHost](#) (page 37)

Performs a type conversion.

[NSConvertSwappedFloatToHost](#) (page 37)

Performs a type conversion.

- [NSHostByteOrder](#) (page 65)
Returns the endian format.
- [NSSwapBigDoubleToHost](#) (page 108)
A utility for swapping the bytes of a number.
- [NSSwapBigFloatToHost](#) (page 109)
A utility for swapping the bytes of a number.
- [NSSwapBigIntToHost](#) (page 109)
A utility for swapping the bytes of a number.
- [NSSwapBigLongLongToHost](#) (page 110)
A utility for swapping the bytes of a number.
- [NSSwapBigLongToHost](#) (page 110)
A utility for swapping the bytes of a number.
- [NSSwapBigShortToHost](#) (page 110)
A utility for swapping the bytes of a number.
- [NSSwapDouble](#) (page 111)
A utility for swapping the bytes of a number.
- [NSSwapFloat](#) (page 111)
A utility for swapping the bytes of a number.
- [NSSwapHostDoubleToBig](#) (page 112)
A utility for swapping the bytes of a number.
- [NSSwapHostDoubleToLittle](#) (page 112)
A utility for swapping the bytes of a number.
- [NSSwapHostFloatToBig](#) (page 113)
A utility for swapping the bytes of a number.
- [NSSwapHostFloatToLittle](#) (page 113)
A utility for swapping the bytes of a number.
- [NSSwapHostIntToBig](#) (page 113)
A utility for swapping the bytes of a number.
- [NSSwapHostIntToLittle](#) (page 114)
A utility for swapping the bytes of a number.
- [NSSwapHostLongLongToBig](#) (page 114)
A utility for swapping the bytes of a number.
- [NSSwapHostLongLongToLittle](#) (page 115)
A utility for swapping the bytes of a number.
- [NSSwapHostLongToBig](#) (page 115)
A utility for swapping the bytes of a number.
- [NSSwapHostLongToLittle](#) (page 116)
A utility for swapping the bytes of a number.
- [NSSwapHostShortToBig](#) (page 116)
A utility for swapping the bytes of a number.
- [NSSwapHostShortToLittle](#) (page 117)
A utility for swapping the bytes of a number.
- [NSSwapInt](#) (page 117)
A utility for swapping the bytes of a number.

- [NSSwapLittleDoubleToHost](#) (page 117)
A utility for swapping the bytes of a number.
- [NSSwapLittleFloatToHost](#) (page 118)
A utility for swapping the bytes of a number.
- [NSSwapLittleIntToHost](#) (page 118)
A utility for swapping the bytes of a number.
- [NSSwapLittleLongLongToHost](#) (page 119)
A utility for swapping the bytes of a number.
- [NSSwapLittleLongToHost](#) (page 119)
A utility for swapping the bytes of a number.
- [NSSwapLittleShortToHost](#) (page 120)
A utility for swapping the bytes of a number.
- [NSSwapLong](#) (page 120)
A utility for swapping the bytes of a number.
- [NSSwapLongLong](#) (page 121)
A utility for swapping the bytes of a number.
- [NSSwapShort](#) (page 121)
A utility for swapping the bytes of a number.

Decimals

You can also use the class `NSDecimalNumber` for decimal arithmetic.

- [NSDecimalAdd](#) (page 45)
Adds two decimal values.
- [NSDecimalCompact](#) (page 46)
Compacts the decimal structure for efficiency.
- [NSDecimalCompare](#) (page 47)
Compares two decimal values.
- [NSDecimalCopy](#) (page 47)
Copies the value of a decimal number.
- [NSDecimalDivide](#) (page 47)
Divides one decimal value by another.
- [NSDecimalIsNotANumber](#) (page 48)
Returns a Boolean that indicates whether a given decimal contains a valid number.
- [NSDecimalMultiply](#) (page 48)
Multiplies two decimal numbers together.
- [NSDecimalMultiplyByPowerOf10](#) (page 49)
Multiplies a decimal by the specified power of 10.
- [NSDecimalNormalize](#) (page 49)
Normalizes the internal format of two decimal numbers to simplify later operations.
- [NSDecimalPower](#) (page 50)
Raises the decimal value to the specified power.
- [NSDecimalRound](#) (page 51)
Rounds off the decimal value.

[NSDecimalString](#) (page 51)

Returns a string representation of the decimal value.

[NSDecimalSubtract](#) (page 52)

Subtracts one decimal value from another.

Exception Handling

You can find the following macros implemented in `NSException.h`. *Exception Programming Topics* discusses these macros and gives examples of their usage. These macros are useful for code that needs to run on versions of the system prior to Mac OS X v10.3. For later versions of the operating system, you should use the Objective-C compiler directives `@try`, `@catch`, `@throw`, and `@finally`; for information about these directives, see Exception Handling in *The Objective-C Programming Language*.

[NS_DURING](#) (page 127)

Marks the start of the exception-handling domain.

[NS_ENDHANDLER](#) (page 128)

Marks the end of the local event handler.

[NS_HANDLER](#) (page 128)

Marks the end of the exception-handling domain and the start of the local exception handler.

[NS_VALUEReturn](#) (page 128)

Permits program control to exit from an exception-handling domain with a value of a specified type.

[NS_VOIDRETURN](#) (page 129)

Permits program control to exit from an exception-handling domain.

Java Setup

[NSJavaBundleCleanup](#) (page 69)

This function has been deprecated.

[NSJavaBundleSetup](#) (page 70)

This function has been deprecated.

[NSJavaClassesForBundle](#) (page 70)

Loads the Java classes located in the specified bundle.

[NSJavaClassesFromPath](#) (page 70)

Loads the Java classes located at the specified path.

[NSJavaNeedsToLoadClasses](#) (page 71)

Returns a Boolean value that indicates whether a virtual machine is needed or if Java classes are provided.

[NSJavaNeedsVirtualMachine](#) (page 71)

Returns a Boolean value that indicates whether a Java virtual machine is required.

[NSJavaObjectNamedInPath](#) (page 72)

Creates an instance of the named class using the class loader previously specified at the given path.

[NSJavaProvidesClasses](#) (page 72)

Returns a Boolean value that indicates whether Java classes are provided.

[NSJavaSetup](#) (page 72)

Loads the Java virtual machine with specified parameters.

[NSJavaSetupVirtualMachine](#) (page 73)

Sets up the Java virtual machine.

Hash Tables

[NSAllHashTableObjects](#) (page 21)

Returns all of the elements in the specified hash table.

[NSCompareHashTables](#) (page 35)

Returns a Boolean value that indicates whether the elements of two hash tables are equal.

[NSCopyHashTableWithZone](#) (page 38)

Performs a shallow copy of the specified hash table.

[NSCountHashTable](#) (page 40)

Returns the number of elements in a hash table.

[NSCreateHashTable](#) (page 42)

Creates and returns a new hash table.

[NSCreateHashTableWithZone](#) (page 42)

Creates a new hash table in a given zone.

[NSEndHashTableEnumeration](#) (page 54)

Used when finished with an enumerator.

[NSEnumerateHashTable](#) (page 54)

Creates an enumerator for the specified hash table.

[NSFreeHashTable](#) (page 58)

Deletes the specified hash table.

[NSHashGet](#) (page 60)

Returns an element of the hash table.

[NSHashInsert](#) (page 61)

Adds an element to the specified hash table.

[NSHashInsertIfAbsent](#) (page 61)

Adds an element to the specified hash table only if the table does not already contain the element.

[NSHashInsertKnownAbsent](#) (page 62)

Adds an element to the specified hash table.

[NSHashRemove](#) (page 62)

Removes an element from the specified hash table.

[NSNextHashEnumeratorItem](#) (page 88)

Returns the next hash-table element in the enumeration.

[NSResetHashTable](#) (page 98)

Deletes the elements of the specified hash table.

[NSStringFromHashTable](#) (page 104)

Returns a string describing the hash table's contents.

HFS File Types

- [NSFileTypeForHFSTypeCode](#) (page 58)
Returns a string encoding a file type code.
- [NSHFSTypeCodeFromFileType](#) (page 63)
Returns a file type code.
- [NSHFSTypeOfFile](#) (page 64)
Returns a string encoding a file type.

Managing Map Tables

- [NSAllMapTableKeys](#) (page 21)
Returns all of the keys in the specified map table.
- [NSAllMapTableValues](#) (page 22)
Returns all of the values in the specified table.
- [NSCompareMapTables](#) (page 35)
Compares the elements of two map tables for equality.
- [NSCopyMapTableWithZone](#) (page 38)
Performs a shallow copy of the specified map table.
- [NSCountMapTable](#) (page 41)
Returns the number of elements in a map table.
- [NSCreateMapTable](#) (page 43)
Creates a new map table in the default zone.
- [NSCreateMapTableWithZone](#) (page 43)
Creates a new map table in the specified zone.
- [NSEndMapTableEnumeration](#) (page 54)
Used when finished with an enumerator.
- [NSEnumerateMapTable](#) (page 55)
Creates an enumerator for the specified map table.
- [NSFreeMapTable](#) (page 59)
Deletes the specified map table.
- [NSMapGet](#) (page 80)
Returns a map table value for the specified key.
- [NSMapInsert](#) (page 81)
Inserts a key-value pair into the specified table.
- [NSMapInsertIfAbsent](#) (page 82)
Inserts a key-value pair into the specified table.
- [NSMapInsertKnownAbsent](#) (page 82)
Inserts a key-value pair into the specified table if the pair had not been previously added.
- [NSMapMember](#) (page 83)
Indicates whether a given table contains a given key.
- [NSMapRemove](#) (page 83)
Removes a key and corresponding value from the specified table.

[NSNextMapEnumeratorPair](#) (page 89)

Returns a Boolean value that indicates whether the next map-table pair in the enumeration are set.

[NSResetMapTable](#) (page 98)

Deletes the elements of the specified map table.

[NSStringFromMapTable](#) (page 105)

Returns a string describing the map table's contents.

Managing Object Allocation and Deallocation

[NSAllocateObject](#) (page 23)

Creates and returns a new instance of a given class.

[NSCopyObject](#) (page 39)

Creates an exact copy of an object.

[NSDeallocateObject](#) (page 45)

Destroys an existing object.

[NSDecrementExtraRefCountWasZero](#) (page 52)

Decrements the specified object's reference count.

[NSExtraRefCount](#) (page 57)

Returns the specified object's reference count.

[NSIncrementExtraRefCount](#) (page 65)

Increments the specified object's reference count.

[NSShouldRetainWithZone](#) (page 102)

Indicates whether an object should be retained.

Interacting with the Objective-C Runtime

[NSGetSizeAndAlignment](#) (page 59)

Obtains the actual size and the aligned size of an encoded type.

[NSClassFromString](#) (page 34)

Obtains a class by name.

[NSStringFromClass](#) (page 104)

Returns the name of a class as a string.

[NSSelectorFromString](#) (page 100)

Returns the selector with a given name.

[NSStringFromSelector](#) (page 107)

Returns a string representation of a given selector.

[NSStringFromProtocol](#) (page 106)

Returns the name of a protocol as a string.

[NSProtocolFromString](#) (page 94)

Returns a the protocol with a given name.

Logging Output

[NSLog](#) (page 76)

Logs an error message to the Apple System Log facility.

[NSLogv](#) (page 77)

Logs an error message to the Apple System Log facility.

Managing File Paths

[NSFullUserName](#) (page 59)

Returns a string containing the full name of the current user.

[NSHomeDirectory](#) (page 64)

Returns the path to the current user's home directory.

[NSHomeDirectoryForUser](#) (page 65)

Returns the path to a given user's home directory.

[NSOpenStepRootDirectory](#) (page 90)

Returns the root directory of the user's system.

[NSSearchPathForDirectoriesInDomains](#) (page 100)

Creates a list of directory search paths.

[NSTemporaryDirectory](#) (page 121)

Returns the path of the temporary directory for the current user.

[NSUserName](#) (page 123)

Returns the logon name of the current user.

Managing Points

[NSEqualPoints](#) (page 55)

Returns a Boolean value that indicates whether two points are equal.

[NSMakePoint](#) (page 79)

Creates a new `NSPoint` from the specified values.

[NSPointFromString](#) (page 92)

Returns a point from a text-based representation.

[NSStringFromPoint](#) (page 105)

Returns a string representation of a point.

[NSPointFromCGPoint](#) (page 91)

Returns an `NSPoint` typecast from a `CGPoint`.

[NSPointToCGPoint](#) (page 93)

Returns a `CGPoint` typecast from an `NSPoint`.

Managing Ranges

[NSEqualRanges](#) (page 56)

Returns a Boolean value that indicates whether two given ranges are equal.

[NSIntersectionRange](#) (page 67)

Returns the intersection of the specified ranges.

[NSLocationInRange](#) (page 76)

Returns a Boolean value that indicates whether a specified position is in a given range.

[NSMakeRange](#) (page 79)

Creates a new `NSRange` from the specified values.

[NSMaxRange](#) (page 84)

Returns the sum of the location and length of the range.

[NSRangeFromString](#) (page 94)

Returns a range from a textual representation.

[NSStringFromRange](#) (page 106)

Returns a string representation of a range.

[NSUnionRange](#) (page 122)

Returns the union of the specified ranges.

Managing Rectangles

[NSContainsRect](#) (page 36)

Returns a Boolean value that indicates whether one rectangle completely encloses another.

[NSDivideRect](#) (page 53)

Divides a rectangle into two new rectangles.

[NSEqualRects](#) (page 56)

Returns a Boolean value that indicates whether the two rectangles are equal.

[NSIsEmptyRect](#) (page 69)

Returns a Boolean value that indicates whether a given rectangle is empty.

[NSHeight](#) (page 63)

Returns the height of a given rectangle.

[NSInsetRect](#) (page 66)

Insets a rectangle by a specified amount.

[NSIntegralRect](#) (page 67)

Adjusts the sides of a rectangle to integer values.

[NSIntersectionRect](#) (page 68)

Calculates the intersection of two rectangles.

[NSIntersectsRect](#) (page 68)

Returns a Boolean value that indicates whether two rectangles intersect.

[NSMakeRect](#) (page 79)

Creates a new `NSRect` from the specified values.

[NSMaxX](#) (page 84)

Returns the largest x coordinate of a given rectangle.

[NSMaxY](#) (page 85)

Returns the largest y coordinate of a given rectangle.

[NSMidX](#) (page 85)

Returns the x coordinate of a given rectangle's midpoint.

[NSMidY](#) (page 86)

Returns the y coordinate of a given rectangle's midpoint.

[NSMinX](#) (page 86)

Returns the smallest x coordinate of a given rectangle.

[NSMinY](#) (page 87)

Returns the smallest y coordinate of a given rectangle.

[NSMouseInRect](#) (page 87)

Returns a Boolean value that indicates whether the point is in the specified rectangle.

[NSOffsetRect](#) (page 89)

Offsets the rectangle by the specified amount.

[NSPointInRect](#) (page 93)

Returns a Boolean value that indicates whether a given point is in a given rectangle.

[NSRectFromString](#) (page 96)

Returns a rectangle from a text-based representation.

[NSStringFromRect](#) (page 107)

Returns a string representation of a rectangle.

[NSRectFromCGRect](#) (page 96)

Returns an `NSRect` typecast from a `CGRect`.

[NSRectToCGRect](#) (page 97)

Returns a `CGRect` typecast from an `NSRect`.

[NSUnionRect](#) (page 122)

Calculates the union of two rectangles.

[NSWidth](#) (page 123)

Returns the width of the specified rectangle.

Managing Sizes

[NSEqualSizes](#) (page 57)

Returns a Boolean that indicates whether two size values are equal.

[NSMakeSize](#) (page 80)

Returns a new `NSSize` from the specified values.

[NSSizeFromString](#) (page 103)

Returns an `NSSize` from a text-based representation.

[NSStringFromSize](#) (page 108)

Returns a string representation of a size.

[NSSizeFromCGSize](#) (page 102)

Returns an `NSSize` typecast from a `CGSize`.

[NSSizeToCGSize](#) (page 103)

Returns a `CGSize` typecast from an `NSSize`.

Uncaught Exception Handlers

Whether there's an uncaught exception handler function, any uncaught exceptions cause the program to terminate, unless the exception is raised during the posting of a notification.

[NSGetUncaughtExceptionHandler](#) (page 60)

Returns the top-level error handler.

[NSSetUncaughtExceptionHandler](#) (page 101)

Changes the top-level error handler.

Managing Memory

[NSDefaultMallocZone](#) (page 53)

Returns the default zone.

[NSAllocateCollectable](#) (page 22)

Allocates collectable memory.

[NSReallocateCollectable](#) (page 95)

Reallocates collectable memory.

[NSMakeCollectable](#) (page 78)

Makes a newly allocated Core Foundation object eligible for collection.

[NSAllocateMemoryPages](#) (page 23)

Allocates a new block of memory.

[NSCopyMemoryPages](#) (page 39)

Copies a block of memory.

[NSDeallocateMemoryPages](#) (page 45)

Deallocates the specified block of memory.

[NSLogPageSize](#) (page 77)

Returns the binary log of the page size.

[NSPageSize](#) (page 90)

Returns the number of bytes in a page.

[NSRealMemoryAvailable](#) (page 95)

Returns information about the user's system.

[NSRoundDownToMultipleOfPageSize](#) (page 99)

Returns the specified number of bytes rounded down to a multiple of the page size.

[NSRoundUpToMultipleOfPageSize](#) (page 99)

Returns the specified number of bytes rounded up to a multiple of the page size.

Managing Zones

[NSCreateZone](#) (page 44)

Creates a new zone.

[NSRecycleZone](#) (page 97)

Frees memory in a zone.

[NSSetZoneName](#) (page 101)

Sets the name of the specified zone.

[NSZoneCalloc](#) (page 124)

Allocates memory in a zone.

[NSZoneFree](#) (page 125)

Deallocates a block of memory in the specified zone.

[NSZoneFromPointer](#) (page 125)

Gets the zone for a given block of memory.

[NSZoneMalloc](#) (page 126)

Allocates memory in a zone.

[NSZoneName](#) (page 126)

Returns the name of the specified zone.

[NSZoneRealloc](#) (page 127)

Allocates memory in a zone.

Functions

NSAllHashTableObjects

Returns all of the elements in the specified hash table.

```
NSArray * NSAllHashTableObjects (
    NSHashTable *table
);
```

Return Value

An array object containing all the elements of *table*.

Discussion

This function should be called only when the table elements are objects, not when they're any other data type.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSCreateHashTable](#) (page 42)

[NSFreeHashTable](#) (page 58)

Declared In

NSHashTable.h

NSAllMapTableKeys

Returns all of the keys in the specified map table.

```
NSArray * NSAllMapTableKeys (
    NSMapTable *table
);
```

Return Value

An array object containing all the keys in *table*. This function should be called only when *table* keys are objects, not when they're any other type of pointer.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSMapMember](#) (page 83)

[NSMapGet](#) (page 80)

[NSEnumerateMapTable](#) (page 55)

[NSNextMapEnumeratorPair](#) (page 89)

[NSAllMapTableValues](#) (page 22)

Declared In

NSMapTable.h

NSAllMapTableValues

Returns all of the values in the specified table.

```
NSArray * NSAllMapTableValues (
    NSMapTable *table
);
```

Return Value

An array object containing all the values in *table*. This function should be called only when *table* values are objects, not when they're any other type of pointer.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSMapMember](#) (page 83)

[NSMapGet](#) (page 80)

[NSEnumerateMapTable](#) (page 55)

[NSNextMapEnumeratorPair](#) (page 89)

[NSAllMapTableKeys](#) (page 21)

Declared In

NSMapTable.h

NSAllocateCollectable

Allocates collectable memory.

```
void *__strong NSAllocateCollectable (
    NSUInteger size,
    NSUInteger options
);
```

Parameters

size

The number of bytes of memory to allocate.

options

0 or `NSScannedOption`: A value of 0 allocates nonscanned memory; a value of `NSScannedOption` allocates scanned memory.

Return Value

A pointer to the allocated memory, or `NULL` if the function is unable to allocate the requested memory.

Availability

Available in Mac OS X v10.4 and later.

Declared In

`NSZone.h`

NSAllocateMemoryPages

Allocates a new block of memory.

```
void * NSAllocateMemoryPages (
    NSUInteger bytes
);
```

Discussion

Allocates the integral number of pages whose total size is closest to, but not less than, *byteCount*. The allocated pages are guaranteed to be filled with zeros. If the allocation fails, raises `NSInvalidArgumentException`.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSCopyMemoryPages](#) (page 39)

[NSDeallocateMemoryPages](#) (page 45)

Declared In

`NSZone.h`

NSAllocateObject

Creates and returns a new instance of a given class.

```
id NSAllocateObject (
    Class aClass,
    NSUInteger extraBytes,
    NSZone *zone
);
```

Parameters

aClass

The class of which to create an instance.

extraBytes

The number of extra bytes required for indexed instance variables (this value is typically 0).

zone

The zone in which to create the new instance (pass `NULL` to specify the default zone).

Return Value

A new instance of *aClass* or *nil* if an instance could not be created.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSDeallocateObject](#) (page 45)

Declared In

NSObject.h

NSAssert

Generates an assertion if a given condition is false.

```
#define NSAssert(condition, desc)
```

Parameters

condition

An expression that evaluates to YES or NO.

desc

An NSString object that contains an error message describing the failure condition.

Discussion

The NSAssert macro evaluates the condition and serves as a front end to the assertion handler.

Each thread has its own assertion handler, which is an object of class NSAssertionHandler. When invoked, an assertion handler prints an error message that includes the method and class names (or the function name). It then raises an NSInternalInconsistencyException exception. If *condition* evaluates to NO, the macro invokes `handleFailureInMethod:object:file:lineNumber:description:` on the assertion handler for the current thread, passing *desc* as the description string.

This macro should be used only within Objective-C methods.

Assertions are disabled if the preprocessor macro `NS_BLOCK_ASSERTIONS` is defined.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSLog](#) (page 76)

[NSLogv](#) (page 77)

[NSAssert1](#) (page 25)

[NSCAssert](#) (page 30)

[NSCParameterAssert](#) (page 41)

[NSParameterAssert](#) (page 91)

Related Sample Code

CocoaVideoFrameToGWorld

CocoaVideoFrameToNSImage

ColorMatching

OpenGLScreenSnapshot

SGDevices

Declared In

NSException.h

NSAssert1

Generates an assertion if a given condition is false.

```
#define NSAssert1(condition, desc, arg1)
```

Parameters

condition

An expression that evaluates to YES or NO.

desc

An NSString object that contains a printf-style string containing an error message describing the failure condition and a placeholder for a single argument.

arg1

An argument to be inserted, in place, into *desc*.

Discussion

The NSAssert1 macro evaluates the condition and serves as a front end to the assertion handler.

Each thread has its own assertion handler, which is an object of class NSAssertionHandler. When invoked, an assertion handler prints an error message that includes the method and class names (or the function name). It then raises an NSInternalInconsistencyException exception. If *condition* evaluates to NO, the macro invokes `handleFailureInMethod:object:file:lineNumber:description:` on the assertion handler for the current thread, passing *desc* as the description string and *arg1* as a substitution variable.

This macro should be used only within Objective-C methods.

Assertions are disabled if the preprocessor macro `NS_BLOCK_ASSERTIONS` is defined.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSLog](#) (page 76)

[NSLogv](#) (page 77)

[NSAssert](#) (page 24)

[NSAssert2](#) (page 26)

[NSAssert3](#) (page 27)

[NSAssert4](#) (page 28)

[NSAssert5](#) (page 29)

[NSCAssert](#) (page 30)

[NSCParameterAssert](#) (page 41)

[NSParameterAssert](#) (page 91)

Related Sample Code

CocoaDVDPlayer

Core Data HTML Store

GeekGameBoard
OpenGLScreenSnapshot

Declared In

NSException.h

NSAssert2

Generates an assertion if a given condition is false.

```
#define NSAssert2(condition, desc, arg1, arg2)
```

Parameters

condition

An expression that evaluates to YES or NO.

desc

An NSString object that contains a printf-style string containing an error message describing the failure condition and placeholders for two arguments.

arg1

An argument to be inserted, in place, into *desc*.

arg2

An argument to be inserted, in place, into *desc*.

Discussion

The `NSAssert2` macro evaluates the condition and serves as a front end to the assertion handler.

Each thread has its own assertion handler, which is an object of class `NSAssertionHandler`. When invoked, an assertion handler prints an error message that includes the method and class names (or the function name). It then raises an `NSInternalInconsistencyException` exception. If *condition* evaluates to NO, the macro invokes `handleFailureInMethod:object:file:lineNumber:description:` on the assertion handler for the current thread, passing *desc* as the description string and *arg1* and *arg2* as substitution variables.

This macro should be used only within Objective-C methods.

Assertions are disabled if the preprocessor macro `NS_BLOCK_ASSERTIONS` is defined.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSLog](#) (page 76)

[NSLogv](#) (page 77)

[NSAssert](#) (page 24)

[NSAssert1](#) (page 25)

[NSAssert3](#) (page 27)

[NSAssert4](#) (page 28)

[NSAssert5](#) (page 29)

[NSCAssert](#) (page 30)

[NSCParameterAssert](#) (page 41)

[NSParameterAssert](#) (page 91)

Related Sample Code

CoreRecipes

Declared In

NSException.h

NSAssert3

Generates an assertion if a given condition is false.

```
#define NSAssert3(condition, desc, arg1, arg2, arg3)
```

Parameters*condition*

An expression that evaluates to YES or NO.

desc

An NSString object that contains a printf-style string containing an error message describing the failure condition and placeholders for three arguments.

*arg1*An argument to be inserted, in place, into *desc*.*arg2*An argument to be inserted, in place, into *desc*.*arg3*An argument to be inserted, in place, into *desc*.**Discussion**

The NSAssert3 macro evaluates the condition and serves as a front end to the assertion handler.

Each thread has its own assertion handler, which is an object of class NSAssertionHandler. When invoked, an assertion handler prints an error message that includes the method and class names (or the function name). It then raises an NSInternalInconsistencyException exception. If *condition* evaluates to NO, the macro invokes `handleFailureInMethod:object:file:lineNumber:description:` on the assertion handler for the current thread, passing *desc* as the description string and *arg1*, *arg2*, and *arg3* as substitution variables.

This macro should be used only within Objective-C methods.

Assertions are disabled if the preprocessor macro `NS_BLOCK_ASSERTIONS` is defined.

Availability

Available in Mac OS X v10.0 and later.

See Also[NSLog](#) (page 76)[NSLogv](#) (page 77)[NSAssert](#) (page 24)[NSAssert1](#) (page 25)[NSAssert2](#) (page 26)[NSAssert4](#) (page 28)[NSAssert5](#) (page 29)[NSCAssert](#) (page 30)

[NSCParameterAssert](#) (page 41)

[NSParameterAssert](#) (page 91)

Declared In

`NSException.h`

NSAssert4

Generates an assertion if a given condition is false.

```
#define NSAssert4(condition, desc, arg1, arg2, arg3, arg4)
```

Parameters

condition

An expression that evaluates to YES or NO.

desc

An `NSString` object that contains a `printf`-style string containing an error message describing the failure condition and placeholders for four arguments.

arg1

An argument to be inserted, in place, into *desc*.

arg2

An argument to be inserted, in place, into *desc*.

arg3

An argument to be inserted, in place, into *desc*.

arg4

An argument to be inserted, in place, into *desc*.

Discussion

The `NSAssert4` macro evaluates the condition and serves as a front end to the assertion handler.

Each thread has its own assertion handler, which is an object of class `NSAssertionHandler`. When invoked, an assertion handler prints an error message that includes the method and class names (or the function name). It then raises an `NSInternalInconsistencyException` exception. If *condition* evaluates to NO, the macro invokes `handleFailureInMethod:object:file:lineNumber:description:` on the assertion handler for the current thread, passing *desc* as the description string and *arg1*, *arg2*, *arg3*, and *arg4* as substitution variables.

This macro should be used only within Objective-C methods.

Assertions are disabled if the preprocessor macro `NS_BLOCK_ASSERTIONS` is defined.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSLog](#) (page 76)

[NSLogv](#) (page 77)

[NSAssert](#) (page 24)

[NSAssert1](#) (page 25)

[NSAssert2](#) (page 26)

[NSAssert3](#) (page 27)

[NSAssert5](#) (page 29)[NSCAssert](#) (page 30)[NSCParameterAssert](#) (page 41)[NSParameterAssert](#) (page 91)**Declared In**

NSException.h

NSAssert5

Generates an assertion if a given condition is false.

```
#define NSAssert5(condition, desc, arg1, arg2, arg3, arg4, arg5)
```

Parameters*condition*

An expression that evaluates to YES or NO.

desc

An NSString object that contains a printf-style string containing an error message describing the failure condition and placeholders for five arguments.

*arg1*An argument to be inserted, in place, into *desc*.*arg2*An argument to be inserted, in place, into *desc*.*arg3*An argument to be inserted, in place, into *desc*.*arg4*An argument to be inserted, in place, into *desc*.*arg5*An argument to be inserted, in place, into *desc*.**Discussion**The `NSAssert5` macro evaluates the condition and serves as a front end to the assertion handler.

Each thread has its own assertion handler, which is an object of class `NSAssertionHandler`. When invoked, an assertion handler prints an error message that includes the method and class names (or the function name). It then raises an `NSInternalInconsistencyException` exception. If *condition* evaluates to NO, the macro invokes `handleFailureInMethod:object:file:lineNumber:description:` on the assertion handler for the current thread, passing *desc* as the description string and *arg1*, *arg2*, *arg3*, *arg4*, and *arg5* as substitution variables.

This macro should be used only within Objective-C methods.

Assertions are disabled if the preprocessor macro `NS_BLOCK_ASSERTIONS` is defined.

Availability

Available in Mac OS X v10.0 and later.

See Also[NSLog](#) (page 76)[NSLogv](#) (page 77)

[NSAssert](#) (page 24)
[NSAssert1](#) (page 25)
[NSAssert2](#) (page 26)
[NSAssert3](#) (page 27)
[NSAssert4](#) (page 28)
[NSCAssert](#) (page 30)
[NSCParameterAssert](#) (page 41)
[NSParameterAssert](#) (page 91)

Declared In

NSException.h

NSCAssert

Generates an assertion if the given condition is false.

```
NSCAssert(condition, NSString *description)
```

Discussion

Assertions evaluate a condition and, if the condition evaluates to false, call the assertion handler for the current thread, passing it a format string and a variable number of arguments. Each thread has its own assertion handler, which is an object of class `NSAssertionHandler`. When invoked, an assertion handler prints an error message that includes method and class names (or the function name). It then raises an `NSInternalInconsistencyException` exception.

The `NSCAssert` macro evaluates the condition and serves as a front end to the assertion handler. This macro should be used only within C functions. `NSCAssert` takes no arguments other than the condition and format string.

The *condition* must be an expression that evaluates to true or false. *description* is a printf-style format string that describes the failure condition.

Assertions are disabled if the preprocessor macro `NS_BLOCK_ASSERTIONS` is defined. All macros return `void`.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSLog](#) (page 76)
[NSLogv](#) (page 77)
[NSAssert](#) (page 24)
[NSCAssert1](#) (page 31)
[NSCParameterAssert](#) (page 41)
[NSParameterAssert](#) (page 91)

Related Sample Code

EnhancedAudioBurn

Declared In

NSException.h

NSCAssert1

`NSCAssert1` is one of a series of macros that generate assertions if the given condition is false.

```
NSCAssert1(condition, NSString *description, arg1)
```

Discussion

Assertions evaluate a condition and, if the condition evaluates to false, call the assertion handler for the current thread, passing it a format string and a variable number of arguments. Each thread has its own assertion handler, which is an object of class `NSAssertionHandler`. When invoked, an assertion handler prints an error message that includes method and class names (or the function name). It then raises an `NSInternalInconsistencyException` exception.

The `NSCAssert1` macro evaluates the condition and serves as a front end to the assertion handler. This macro should be used only within C functions.

The *condition* expression must evaluate to true or false. *description* is a printf-style format string that describes the failure condition. *arg1* is an argument to be inserted, in place, into the description.

Assertions are disabled if the preprocessor macro `NS_BLOCK_ASSERTIONS` is defined. All macros return `void`.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSLog](#) (page 76)

[NSLogv](#) (page 77)

[NSCAssert](#) (page 30)

[NSCAssert2](#) (page 31)

[NSCAssert3](#) (page 32)

[NSCAssert4](#) (page 33)

[NSCAssert5](#) (page 34)

[NSCParameterAssert](#) (page 41)

[NSParameterAssert](#) (page 91)

Related Sample Code

GeekGameBoard

Declared In

`NSException.h`

NSCAssert2

`NSCAssert2` is one of a series of macros that generate assertions if the given condition is false.

```
NSCAssert2(condition, NSString *description, arg1, arg2)
```

Discussion

Assertions evaluate a condition and, if the condition evaluates to false, call the assertion handler for the current thread, passing it a format string and a variable number of arguments. Each thread has its own assertion handler, which is an object of class `NSAssertionHandler`. When invoked, an assertion handler prints an error message that includes method and class names (or the function name). It then raises an `NSInternalInconsistencyException` exception.

The `NSCAssert2` macro evaluates the condition and serves as a front end to the assertion handler. This macro should be used only within C functions.

The *condition* expression must evaluate to true or false. *description* is a printf-style format string that describes the failure condition. Each *argn* is an argument to be inserted, in place, into the description.

Assertions are disabled if the preprocessor macro `NS_BLOCK_ASSERTIONS` is defined. All macros return `void`.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSLog](#) (page 76)

[NSLogv](#) (page 77)

[NSCAssert](#) (page 30)

[NSCAssert1](#) (page 31)

[NSCAssert3](#) (page 32)

[NSCAssert4](#) (page 33)

[NSCAssert5](#) (page 34)

[NSCParameterAssert](#) (page 41)

[NSParameterAssert](#) (page 91)

Declared In

`NSException.h`

NSCAssert3

`NSCAssert3` is one of a series of macros that generate assertions if the given condition is false.

```
NSCAssert3(condition, NSString *description, arg1, arg2, arg3)
```

Discussion

Assertions evaluate a condition and, if the condition evaluates to false, call the assertion handler for the current thread, passing it a format string and a variable number of arguments. Each thread has its own assertion handler, which is an object of class `NSAssertionHandler`. When invoked, an assertion handler prints an error message that includes method and class names (or the function name). It then raises an `NSInternalInconsistencyException` exception.

The `NSCAssert3` macro evaluates the condition and serves as a front end to the assertion handler. This macro should be used only within C functions.

The *condition* expression must evaluate to true or false. *description* is a printf-style format string that describes the failure condition. Each *argn* is an argument to be inserted, in place, into the description.

Assertions are disabled if the preprocessor macro `NS_BLOCK_ASSERTIONS` is defined. All macros return `void`.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSLog](#) (page 76)

[NSLogv](#) (page 77)

[NSCAssert](#) (page 30)
[NSCAssert1](#) (page 31)
[NSCAssert2](#) (page 31)
[NSCAssert4](#) (page 33)
[NSCAssert5](#) (page 34)
[NSCParameterAssert](#) (page 41)
[NSParameterAssert](#) (page 91)

Declared In

NSException.h

NSCAssert4

NSCAssert4 is one of a series of macros that generate assertions if the given condition is false.

```
NSCAssert4(condition, NSString *description, arg1, arg2, arg3, arg4)
```

Discussion

Assertions evaluate a condition and, if the condition evaluates to false, call the assertion handler for the current thread, passing it a format string and a variable number of arguments. Each thread has its own assertion handler, which is an object of class `NSAssertionHandler`. When invoked, an assertion handler prints an error message that includes method and class names (or the function name). It then raises an `NSInternalInconsistencyException` exception.

The `NSCAssert4` macro evaluates the condition and serves as a front end to the assertion handler. This macro should be used only within C functions.

The *condition* expression must evaluate to true or false. *description* is a printf-style format string that describes the failure condition. Each *argn* is an argument to be inserted, in place, into the description.

Assertions are disabled if the preprocessor macro `NS_BLOCK_ASSERTIONS` is defined. All macros return `void`.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSLog](#) (page 76)
[NSLogv](#) (page 77)
[NSCAssert](#) (page 30)
[NSCAssert1](#) (page 31)
[NSCAssert2](#) (page 31)
[NSCAssert3](#) (page 32)
[NSCAssert5](#) (page 34)
[NSCParameterAssert](#) (page 41)
[NSParameterAssert](#) (page 91)

Declared In

NSException.h

NSCAssert5

`NSCAssert5` is one of a series of macros that generate assertions if the given condition is false.

```
NSCAssert5(condition, NSString *description, arg1, arg2, arg3, arg4, arg5)
```

Discussion

Assertions evaluate a condition and, if the condition evaluates to false, call the assertion handler for the current thread, passing it a format string and a variable number of arguments. Each thread has its own assertion handler, which is an object of class `NSAssertionHandler`. When invoked, an assertion handler prints an error message that includes method and class names (or the function name). It then raises an `NSInternalInconsistencyException` exception.

The `NSCAssert5` macro evaluates the condition and serves as a front end to the assertion handler. This macro should be used only within C functions.

The *condition* expression must evaluate to true or false. *description* is a printf-style format string that describes the failure condition. Each *argn* is an argument to be inserted, in place, into the description.

Assertions are disabled if the preprocessor macro `NS_BLOCK_ASSERTIONS` is defined. All macros return `void`.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSLog](#) (page 76)

[NSLogv](#) (page 77)

[NSCAssert](#) (page 30)

[NSCAssert1](#) (page 31)

[NSCAssert2](#) (page 31)

[NSCAssert3](#) (page 32)

[NSCAssert4](#) (page 33)

[NSCParameterAssert](#) (page 41)

[NSParameterAssert](#) (page 91)

Declared In

`NSException.h`

NSClassFromString

Obtains a class by name.

```
Class NSClassFromString (
    NSString *aClassName
);
```

Parameters

aClassName

The name of a class.

Return Value

The class object named by *aClassName*, or `nil` if no class by that name is currently loaded. If *aClassName* is `nil`, returns `nil`.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSStringFromClass](#) (page 104)

[NSProtocolFromString](#) (page 94)

[NSSelectorFromString](#) (page 100)

Related Sample Code

GeekGameBoard

QuickLookSketch

Sketch+Accessibility

Sketch-112

Declared In

NSObjCRuntime.h

NSCompareHashTables

Returns a Boolean value that indicates whether the elements of two hash tables are equal.

```
BOOL NSCompareHashTables (
    NSHashTable *table1,
    NSHashTable *table2
);
```

Return Value

YES if the two hash tables are equal—that is, if each element of *table1* is in *table2* and the two tables are the same size, otherwise NO.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSCreateHashTable](#) (page 42)

[NSCreateHashTableWithZone](#) (page 42)

Declared In

NSHashTable.h

NSCompareMapTables

Compares the elements of two map tables for equality.

```
BOOL NSCompareMapTables (
    NSMapTable *table1,
    NSMapTable *table2
);
```

Return Value

YES if the keys and corresponding values of *table1* and *table2* are the same, and the two tables are the same size, otherwise NO.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSCreateMapTable](#) (page 43)

[NSCreateMapTableWithZone](#) (page 43)

Declared In

NSMapTable.h

NSContainsRect

Returns a Boolean value that indicates whether one rectangle completely encloses another.

```
BOOL NSContainsRect (
    NSRect aRect,
    NSRect bRect
);
```

Return Value

YES if *aRect* completely encloses *bRect*. For this condition to be true, *bRect* cannot be empty, and must not extend beyond *aRect* in any direction.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

GLUT

Declared In

NSGeometry.h

NSConvertHostDoubleToSwapped

Performs a type conversion.

```
NSSwappedDouble NSConvertHostDoubleToSwapped (
    double x
);
```

Discussion

Converts the double value in *x* to a value whose bytes can be swapped. This function does not actually swap the bytes of *x*. You should not need to call this function directly.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapHostDoubleToBig](#) (page 112)

[NSSwapHostDoubleToLittle](#) (page 112)

Declared In

NSByteOrder.h

NSConvertHostFloatToSwapped

Performs a type conversion.

```

NSSwappedFloat NSConvertHostFloatToSwapped (
    float x
);

```

Discussion

Converts the float value in *x* to a value whose bytes can be swapped. This function does not actually swap the bytes of *x*. You should not need to call this function directly.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapHostFloatToBig](#) (page 113)

[NSSwapHostFloatToLittle](#) (page 113)

Declared In

NSByteOrder.h

NSConvertSwappedDoubleToHost

Performs a type conversion.

```

double NSConvertSwappedDoubleToHost (
    NSSwappedDouble x
);

```

Discussion

Converts the value in *x* to a double value. This function does not actually swap the bytes of *x*. You should not need to call this function directly.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapBigDoubleToHost](#) (page 108)

[NSSwapLittleDoubleToHost](#) (page 117)

Declared In

NSByteOrder.h

NSConvertSwappedFloatToHost

Performs a type conversion.

```
float NSConvertSwappedFloatToHost (
    NSSwappedFloat x
);
```

Discussion

Converts the value in *x* to a float value. This function does not actually swap the bytes of *x*. You should not need to call this function directly.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapBigFloatToHost](#) (page 109)

[NSSwapLittleFloatToHost](#) (page 118)

Declared In

NSByteOrder.h

NSCopyHashTableWithZone

Performs a shallow copy of the specified hash table.

```
NSHashTable * NSCopyHashTableWithZone (
    NSHashTable *table,
    NSZone *zone
);
```

Return Value

A pointer to a new copy of *table*, created in *zone* and containing pointers to the data elements of *table*.

Discussion

If *zone* is NULL, the new table is created in the default zone.

The new table adopts the callback functions of *table* and calls the `hash` and `retain` callback functions as appropriate when inserting elements into the new table.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSCreateHashTable](#) (page 42)

[NSCreateHashTableWithZone](#) (page 42)

[NSHashTableCallbacks](#) (structure)

Declared In

NSHashTable.h

NSCopyMapTableWithZone

Performs a shallow copy of the specified map table.

```

NSMutableDictionary * NSCopyMapTableWithZone (
    NSMutableDictionary *table,
    NSZone *zone
);

```

Return Value

A pointer to a new copy of *table*, created in *zone* and containing pointers to the keys and values of *table*.

Discussion

If *zone* is `NULL`, the new table is created in the default zone.

The new table adopts the callback functions of *table* and calls the `hash` and `retain` callback functions as appropriate when inserting elements into the new table.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSMutableDictionary](#) (page 43)

[NSMutableDictionaryWithZone](#) (page 43)

[NSMutableDictionaryKeyCallbacks](#) (structure)

[NSMutableDictionaryValueCallbacks](#) (structure)

Declared In

`NSMutableDictionary.h`

NSCopyMemoryPages

Copies a block of memory.

```

void NSCopyMemoryPages (
    const void *source,
    void *dest,
    NSUInteger bytes
);

```

Discussion

Copies (or copies on write) *byteCount* bytes from *source* to *destination*.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSAllocateMemoryPages](#) (page 23)

[NSDeallocateMemoryPages](#) (page 45)

Declared In

`NSZone.h`

NSCopyObject

Creates an exact copy of an object.

```
id NSCopyObject (
    id object,
    NSUInteger extraBytes,
    NSZone *zone
);
```

Parameters*object*

The object to copy.

extraBytes

The number of extra bytes required for indexed instance variables (this value is typically 0).

*zone*The zone in which to create the new instance (pass `NULL` to specify the default zone).**Return Value**A new object that's an exact copy of *anObject*, or `nil` if *object* is `nil` or if *object* could not be copied.**Special Considerations**

This function is dangerous and very difficult to use correctly. It's use as part of `copyWithZone:` by any class that can be subclassed, is highly error prone. Under GC or when using Objective-C 2.0, the zone is completely ignored.

This function is likely to be deprecated after Mac OS X 10.6.

Availability

Available in Mac OS X v10.0 and later.

See Also[NSAllocateObject](#) (page 23)[NSDeallocateObject](#) (page 45)**Declared In**

NSObject.h

NSCountHashTable

Returns the number of elements in a hash table.

```
NSUInteger NSCountHashTable (
    NSHashTable *table
);
```

Return ValueThe number of elements currently in *table*.**Availability**

Available in Mac OS X v10.0 and later.

Declared In

NSHashTable.h

NSCountMapTable

Returns the number of elements in a map table.

```

NSUInteger NSCountMapTable (
    NSMapTable *table
);

```

Parameters

table

A reference to a map table structure.

Return Value

The number of key-value pairs currently in *table*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSMapTable.h

NSCParameterAssert

Evaluates the specified parameter.

```

NSCParameterAssert(condition)

```

Discussion

Assertions evaluate a condition and, if the condition evaluates to false, call the assertion handler for the current thread, passing it a format string and a variable number of arguments. Each thread has its own assertion handler, which is an object of class `NSAssertionHandler`. When invoked, an assertion handler prints an error message that includes method and class names (or the function name). It then raises an `NSInternalInconsistencyException` exception.

This macro validates a parameter for a C function. Simply provide the parameter as the condition argument. The macro evaluates the parameter and, if the parameter evaluates to false, logs an error message that includes the parameter and then raises an exception.

Assertions are disabled if the preprocessor macro `NS_BLOCK_ASSERTIONS` is defined. All macros return `void`.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSLog](#) (page 76)

[NSLogv](#) (page 77)

[NSAssert](#) (page 24)

[NSCAssert](#) (page 30)

[NSParameterAssert](#) (page 91)

Related Sample Code

GeekGameBoard

GLUT

Declared In

NSException.h

NSCreateHashTable

Creates and returns a new hash table.

```
NSHashTable * NSCreateHashTable (
    NSHashTableCallbacks callbacks,
    NSUInteger capacity
);
```

Return Value

A pointer to an NSHashTable created in the default zone.

Discussion

The table's size is dependent on (but generally not equal to) *capacity*. If *capacity* is 0, a small hash table is created. The NSHashTableCallbacks structure *callbacks* has five pointers to functions, with the following defaults: pointer hashing, if *hash* is NULL; pointer equality, if *isEqual* is NULL; no callback upon adding an element, if *retain* is NULL; no callback upon removing an element, if *release* is NULL; and a function returning a pointer's hexadecimal value as a string, if *describe* is NULL. The hashing function must be defined such that if two data elements are equal, as defined by the comparison function, the values produced by hashing on these elements must also be equal. Also, data elements must remain invariant if the value of the hashing function depends on them; for example, if the hashing function operates directly on the characters of a string, that string can't change.

Availability

Available in Mac OS X v10.0 and later.

See Also[NSCopyHashTableWithZone](#) (page 38)[NSCreateHashTableWithZone](#) (page 42)**Declared In**

NSHashTable.h

NSCreateHashTableWithZone

Creates a new hash table in a given zone.

```
NSHashTable * NSCreateHashTableWithZone (
    NSHashTableCallbacks callbacks,
    NSUInteger capacity,
    NSZone *zone
);
```

Return ValueA pointer to a new hash table created in the specified zone. If *zone* is NULL, the hash table is created in the default zone.**Discussion**

The table's size is dependent on (but generally not equal to) *capacity*. If *capacity* is 0, a small hash table is created. The NSHashTableCallbacks structure *callbacks* has five pointers to functions, with the following defaults: pointer hashing, if *hash* is NULL; pointer equality, if *isEqual* is NULL; no callback upon

adding an element, if `retain` is `NULL`; no callback upon removing an element, if `release` is `NULL`; and a function returning a pointer's hexadecimal value as a string, if `describe` is `NULL`. The hashing function must be defined such that if two data elements are equal, as defined by the comparison function, the values produced by hashing on these elements must also be equal. Also, data elements must remain invariant if the value of the hashing function depends on them; for example, if the hashing function operates directly on the characters of a string, that string can't change.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSCreateHashTable](#) (page 42)

Declared In

`NSHashTable.h`

NSCreateMapTable

Creates a new map table in the default zone.

```
NSMapTable * NSCreateMapTable (
    NSMapTableKeyCallbacks keyCallbacks,
    NSMapTableValueCallbacks valueCallbacks,
    NSUInteger capacity
);
```

Discussion

Creates and returns a pointer to an `NSMapTable` structure in the default zone; the table's size is dependent on (but generally not equal to) `capacity`. If `capacity` is 0, a small map table is created. The `NSMapTableKeyCallbacks` arguments are structures that are very similar to the callback structure used by [NSCreateHashTable](#) (page 42)—they have the same defaults as documented for that function.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSCopyMapTableWithZone](#) (page 38)

[NSCreateMapTableWithZone](#) (page 43)

Related Sample Code

Sketch+Accessibility

Declared In

`NSMapTable.h`

NSCreateMapTableWithZone

Creates a new map table in the specified zone.

```

NSMutableDictionary * NSCreateMapTableWithZone (
    NSMutableDictionaryKeyCallBacks keyCallBacks,
    NSMutableDictionaryValueCallBacks valueCallBacks,
    NSUInteger capacity,
    NSZone *zone
);

```

Return Value

A new map table is allocated in *zone*. If *zone* is NULL, the hash table is created in the default zone.

Discussion

The table's size is dependent on (but generally not equal to) *capacity*. If *capacity* is 0, a small map table is created. The `NSMutableDictionaryKeyCallBacks` arguments are structures that are very similar to the callback structure used by [NSCreateHashTable](#) (page 42); in fact, they have the same defaults as documented for that function.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSCopyMapTableWithZone](#) (page 38)

[NSCreateMapTable](#) (page 43)

Declared In

`NSMutableDictionary.h`

NSCreateZone

Creates a new zone.

```

NSZone * NSCreateZone (
    NSUInteger startSize,
    NSUInteger granularity,
    BOOL canFree
);

```

Return Value

A pointer to a new zone of *startSize* bytes, which will grow and shrink by *granularity* bytes. If *canFree* is 0, the allocator will never free memory, and `malloc` will be fast. Returns NULL if a new zone could not be created.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSDefaultMallocZone](#) (page 53)

[NSRecycleZone](#) (page 97)

[NSSetZoneName](#) (page 101)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

Declared In

`NSZone.h`

NSDeallocateMemoryPages

Deallocates the specified block of memory.

```
void NSDeallocateMemoryPages (
    void *ptr,
    NSUInteger bytes
);
```

Discussion

This function deallocates memory that was allocated with `NSAllocateMemoryPages`.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSCopyMemoryPages](#) (page 39)

[NSAllocateMemoryPages](#) (page 23)

Declared In

NSZone.h

NSDeallocateObject

Destroys an existing object.

```
void NSDeallocateObject (
    id object
);
```

Parameters

object

An object.

Discussion

This function deallocates *object*, which must have been allocated using `NSAllocateObject`.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSAllocateObject](#) (page 23)

Declared In

NSObject.h

NSDecimalAdd

Adds two decimal values.

```

NSCalculationError NSDecimalAdd (
    NSDecimal *result,
    const NSDecimal *leftOperand,
    const NSDecimal *rightOperand,
    NSRoundingMode roundingMode
);

```

Discussion

Adds *leftOperand* to *rightOperand* and stores the sum in *result*.

An NSDecimal can represent a number with up to 38 significant digits. If a number is more precise than that, it must be rounded off. *roundingMode* determines how to round it off. There are four possible rounding modes:

NSRoundDown	Round return values down.
NSRoundUp	Round return values up.
NSRoundPlain	Round to the closest possible return value; when caught halfway between two positive numbers, round up; when caught between two negative numbers, round down.
NSRoundBankers	Round to the closest possible return value; when halfway between two possibilities, return the possibility whose last digit is even.

The return value indicates whether any machine limitations were encountered in the addition. If none were encountered, the function returns `NSCalculationNoError`. Otherwise it may return one of the following values: `NSCalculationLossOfPrecision`, `NSCalculationOverflow` or `NSCalculationUnderflow`. For descriptions of all these error conditions, see `exceptionDuringOperation:error:leftOperand:rightOperand:` in `NSDecimalNumberBehaviors`.

For more information, see *Number and Value Programming Topics*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSDecimal.h`

NSDecimalCompact

Compacts the decimal structure for efficiency.

```

void NSDecimalCompact (
    NSDecimal *number
);

```

Discussion

Formats *number* so that calculations using it will take up as little memory as possible. All the `NSDecimal...` arithmetic functions expect compact NSDecimal arguments.

For more information, see *Number and Value Programming Topics*.

Availability

Available in Mac OS X v10.0 and later.

Declared In
NSDecimal.h

NSDecimalCompare

Compares two decimal values.

```
NSComparisonResult NSDecimalCompare (  
    const NSDecimal *leftOperand,  
    const NSDecimal *rightOperand  
);
```

Return Value

NSOrderedDescending if *leftOperand* is bigger than *rightOperand*; NSOrderedAscending if *rightOperand* is bigger than *leftOperand*; or NSOrderedSame if the two operands are equal.

Discussion

For more information, see *Number and Value Programming Topics*.

Availability

Available in Mac OS X v10.0 and later.

Declared In
NSDecimal.h

NSDecimalCopy

Copies the value of a decimal number.

```
void NSDecimalCopy (  
    NSDecimal *destination,  
    const NSDecimal *source  
);
```

Discussion

Copies the value in *source* to *destination*.

For more information, see *Number and Value Programming Topics*.

Availability

Available in Mac OS X v10.0 and later.

Declared In
NSDecimal.h

NSDecimalDivide

Divides one decimal value by another.

```

NSCalculationError NSDecimalDivide (
    NSDecimal *result,
    const NSDecimal *leftOperand,
    const NSDecimal *rightOperand,
    NSRoundingMode roundingMode
);

```

Discussion

Divides *leftOperand* by *rightOperand* and stores the quotient, possibly rounded off according to *roundingMode*, in *result*. If *rightOperand* is 0, returns `NSDivideByZero`.

For explanations of the possible return values and rounding modes, see [NSDecimalAdd](#) (page 45).

Note that repeating decimals or numbers with a mantissa larger than 38 digits cannot be represented precisely.

For more information, see *Number and Value Programming Topics*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSDecimal.h`

NSDecimalIsNotANumber

Returns a Boolean that indicates whether a given decimal contains a valid number.

```

BOOL NSDecimalIsNotANumber (
    const NSDecimal *dcm
);

```

Return Value

YES if the value in *decimal* represents a valid number, otherwise NO.

For more information, see *Number and Value Programming Topics*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSDecimal.h`

NSDecimalMultiply

Multiplies two decimal numbers together.


```

NSCalculationError NSDecimalMultiply (
    NSDecimal *result,
    const NSDecimal *leftOperand,
    const NSDecimal *rightOperand,
    NSRoundingMode roundingMode
);

```

Discussion

Multiplies *rightOperand* by *leftOperand* and stores the product, possibly rounded off according to *roundingMode*, in *result*.

For explanations of the possible return values and rounding modes, see [NSDecimalAdd](#) (page 45).

For more information, see *Number and Value Programming Topics*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSDecimal.h

NSDecimalMultiplyByPowerOf10

Multiplies a decimal by the specified power of 10.

```

NSCalculationError NSDecimalMultiplyByPowerOf10 (
    NSDecimal *result,
    const NSDecimal *number,
    short power,
    NSRoundingMode roundingMode
);

```

Discussion

Multiplies *number* by *power* of 10 and stores the product, possibly rounded off according to *roundingMode*, in *result*.

For explanations of the possible return values and rounding modes, see [NSDecimalAdd](#) (page 45).

For more information, see *Number and Value Programming Topics*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSDecimal.h

NSDecimalNormalize

Normalizes the internal format of two decimal numbers to simplify later operations.

```

NSCalculationError NSDecimalNormalize (
    NSDecimal *number1,
    NSDecimal *number2,
    NSRoundingMode roundingMode
);

```

Discussion

An NSDecimal is represented in memory as a mantissa and an exponent, expressing the value mantissa x 10^{exponent}. A number can have many NSDecimal representations; for example, the following table lists several valid NSDecimal representations for the number 100:

Mantissa	Exponent
100	0
10	1
1	2

Format *number1* and *number2* so that they have equal exponents. This format makes addition and subtraction very convenient. Both [NSDecimalAdd](#) (page 45) and [NSDecimalSubtract](#) (page 52) call NSDecimalNormalize. You may want to use it if you write more complicated addition or subtraction routines.

For explanations of the possible return values, see [NSDecimalAdd](#) (page 45).

For more information, see *Number and Value Programming Topics*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSDecimal.h

NSDecimalPower

Raises the decimal value to the specified power.

```

NSCalculationError NSDecimalPower (
    NSDecimal *result,
    const NSDecimal *number,
    NSInteger power,
    NSRoundingMode roundingMode
);

```

Discussion

Raises *number* to *power*, possibly rounding off according to *roundingMode*, and stores the resulting value in *result*.

For explanations of the possible return values and rounding modes, see [NSDecimalAdd](#) (page 45).

For more information, see *Number and Value Programming Topics*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSDecimal.h

NSDecimalRound

Rounds off the decimal value.

```
void NSDecimalRound (
    NSDecimal *result,
    const NSDecimal *number,
    NSInteger scale,
    NSRoundingMode roundingMode
);
```

Discussion

Rounds *number* off according to the parameters *scale* and *roundingMode* and stores the result in *result*.

The *scale* value specifies the number of digits *result* can have after its decimal point. *roundingMode* specifies the way that number is rounded off. There are four possible values for *roundingMode*: NSRoundDown, NSRoundUp, NSRoundPlain, and NSRoundBankers. For thorough discussions of *scale* and *roundingMode*, see NSDecimalNumberBehaviors.

For more information, see *Number and Value Programming Topics*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSDecimal.h

NSDecimalString

Returns a string representation of the decimal value.

```
NSString * NSDecimalString (
    const NSDecimal *dcm,
    id locale
);
```

Discussion

Returns a string representation of *decimal*. *locale* determines the format of the decimal separator.

For more information, see *Number and Value Programming Topics*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSDecimal.h

NSDecimalSubtract

Subtracts one decimal value from another.

```

NSCalculationError NSDecimalSubtract (
    NSDecimal *result,
    const NSDecimal *leftOperand,
    const NSDecimal *rightOperand,
    NSRoundingMode roundingMode
);

```

Discussion

Subtracts *rightOperand* from *leftOperand* and stores the difference, possibly rounded off according to *roundingMode*, in *result*.

For explanations of the possible return values and rounding modes, see [NSDecimalAdd](#) (page 45).

For more information, see *Number and Value Programming Topics*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSDecimal.h

NSDecrementExtraRefCountWasZero

Decrements the specified object's reference count.

```

BOOL NSDecrementExtraRefCountWasZero (
    id object
);

```

Parameters

object

An object.

Return Value

NO if *anObject* had an extra reference count, or YES if *anObject* didn't have an extra reference count—indicating that the object should be deallocated (with `dealloc`).

Discussion

Decrements the “extra reference” count of *anObject*. Newly created objects have only one actual reference, so that a single release message results in the object being deallocated. Extra references are those beyond the single original reference and are usually created by sending the object a retain message. Your code should generally not use these functions unless it is overriding the `retain` or `release` methods.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSExtraRefCount](#) (page 57)

[NSIncrementExtraRefCount](#) (page 65)

Declared In

NSObject.h

NSDefaultMallocZone

Returns the default zone.

```
NSZone * NSDefaultMallocZone (void);
```

Return Value

The default zone, which is created automatically at startup.

Discussion

This zone is used by the standard C function `malloc`.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSCreateZone](#) (page 44)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

Declared In

NSZone.h

NSDivideRect

Divides a rectangle into two new rectangles.

```
void NSDivideRect (
    NSRect inRect,
    NSRect *slice,
    NSRect *rem,
    CGFloat amount,
    NSRectEdge edge
);
```

Discussion

Creates two rectangles—*slice* and *rem*—from *inRect*, by dividing *inRect* with a line that's parallel to the side of *inRect* specified by *edge*. The size of *slice* is determined by *amount*, which specifies the distance from *edge*.

slice and *rem* must not be NULL.

For more information, see `NSRectEdge`.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSInsetRect](#) (page 66)

[NSIntegralRect](#) (page 67)

[NSOffsetRect](#) (page 89)

Related Sample Code

DragNDropOutlineView

ImageBackground
UIKitMovieShuffler
STUCAuthoringDeviceCocoaSample
TrackBall

Declared In

NSGeometry.h

NSEndHashTableEnumeration

Used when finished with an enumerator.

```
void NSEndHashTableEnumeration (  
    NSHashEnumerator *enumerator  
);
```

Discussion

This function should be called when you have finished using the enumeration struct *enumerator*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSHashTable.h

NSEndMapTableEnumeration

Used when finished with an enumerator.

```
void NSEndMapTableEnumeration (  
    NSMapEnumerator *enumerator  
);
```

Discussion

This function should be called when you have finished using the enumeration struct *enumerator*.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

Sketch+Accessibility

Declared In

NSMapTable.h

NSEnumerateHashTable

Creates an enumerator for the specified hash table.

```
NSHashEnumerator NSEnumerateHashTable (
    NSHashTable *table
);
```

Return Value

An `NSHashEnumerator` structure that will cause successive elements of *table* to be returned each time this enumerator is passed to `NSNextHashEnumeratorItem`.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSNextHashEnumeratorItem](#) (page 88)

Declared In

`NSHashTable.h`

NSEnumerateMapTable

Creates an enumerator for the specified map table.

```
NSMapEnumerator NSEnumerateMapTable (
    NSMapTable *table
);
```

Parameters

table

A reference to a map table structure.

Return Value

An `NSMapEnumerator` structure that will cause successive key-value pairs of *table* to be visited each time this enumerator is passed to `NSNextMapEnumeratorPair`.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSNextMapEnumeratorPair](#) (page 89)

[NSMapMember](#) (page 83)

[NSMapGet](#) (page 80)

[NSAllMapTableKeys](#) (page 21)

[NSAllMapTableValues](#) (page 22)

Related Sample Code

Sketch+Accessibility

Declared In

`NSMapTable.h`

NSEqualPoints

Returns a Boolean value that indicates whether two points are equal.

```

BOOL NSEqualPoints (
    NSPoint aPoint,
    NSPoint bPoint
);

```

Return Value

YES if the two points *aPoint* and *bPoint* are identical, otherwise NO.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

DotViewUndo

DragItemAround

iChatTheater

Quartz Composer WWDC 2005 TextEdit

Sketch-112

Declared In

NSGeometry.h

NSEqualRanges

Returns a Boolean value that indicates whether two given ranges are equal.

```

BOOL NSEqualRanges (
    NSRange range1,
    NSRange range2
);

```

Return Value

YES if *range1* and *range2* have the same locations and lengths.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

AnimatedTableView

Declared In

NSRange.h

NSEqualRects

Returns a Boolean value that indicates whether the two rectangles are equal.

```

BOOL NSEqualRects (
    NSRect aRect,
    NSRect bRect
);

```

Return Value

YES if *aRect* and *bRect* are identical, otherwise NO.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

ImageKitDemo

JSPong

Quartz Composer WWDC 2005 TextEdit

Sketch+Accessibility

Sketch-112

Declared In

NSGeometry.h

NSEqualSizes

Returns a Boolean that indicates whether two size values are equal.

```
BOOL NSEqualSizes (
    NSSize aSize,
    NSSize bSize
);
```

Return Value

YES if *aSize* and *bSize* are identical, otherwise NO.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

QTKitCreateMovie

QTQuartzPlayer

Quartz Composer QCTV

Quartz Composer WWDC 2005 TextEdit

Sketch-112

Declared In

NSGeometry.h

NSEExtraRefCount

Returns the specified object's reference count.

```
NSUInteger NSEExtraRefCount (
    id object
);
```

Parameters

object

An object.

Return Value

The current reference count of *object*.

Discussion

This function is used in conjunction with [NSIncrementExtraRefCount](#) (page 65) and [NSDecrementExtraRefCountWasZero](#) (page 52) in situations where you need to override an object's retain and release methods.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSObject.h

NSFileTypeForHFSTypeCode

Returns a string encoding a file type code.

```
NSString * NSFileTypeForHFSTypeCode (
    OSType hfsFileTypeCode
);
```

Parameters

hfsFileTypeCode

An HFS file type code.

Return Value

A string that encodes *hfsFileTypeCode*.

Discussion

For more information, see HFS File Types.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSHFSTypes.h

NSFreeHashTable

Deletes the specified hash table.

```
void NSFreeHashTable (
    NSHashTable *table
);
```

Discussion

Releases each element of the specified hash table and frees the table itself.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSResetHashTable](#) (page 98)

Declared In

NSHashTable.h

NSFreeMapTable

Deletes the specified map table.

```
void NSFreeMapTable (
    NSMapTable *table
);
```

Parameters

table

A reference to a map table structure.

Discussion

Releases each key and value of the specified map table and frees the table itself.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSResetMapTable](#) (page 98)

Related Sample Code

Sketch+Accessibility

Declared In

NSMapTable.h

NSFullUserName

Returns a string containing the full name of the current user.

```
NSString * NSFullUserName (void);
```

Return Value

A string containing the full name of the current user.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSUserName](#) (page 123)

Declared In

NSPathUtilities.h

NSGetSizeAndAlignment

Obtains the actual size and the aligned size of an encoded type.

```
const char * NSGetSizeAndAlignment (
    const char *typePtr,
    NSUInteger *sizep,
    NSUInteger *alignp
);
```

Discussion

Obtains the actual size and the aligned size of the first data type represented by *typePtr* and returns a pointer to the position of the next data type in *typePtr*. You can specify `NULL` for either *sizep* or *alignp* to ignore the corresponding information.

The value returned in *alignp* is the aligned size of the data type; for example, on some platforms, the aligned size of a `char` might be 2 bytes while the actual physical size is 1 byte.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSObjCRuntime.h`

NSGetUncaughtExceptionHandler

Returns the top-level error handler.

```
NSUncaughtExceptionHandler * NSGetUncaughtExceptionHandler (void);
```

Return Value

A pointer to the top-level error-handling function where you can perform last-minute logging before the program terminates.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSetUncaughtExceptionHandler](#) (page 101)

Declared In

`NSException.h`

NSHashGet

Returns an element of the hash table.

```
void * NSHashGet (
    NSHashTable *table,
    const void *pointer
);
```

Return Value

The pointer in the table that matches *pointer* (as defined by the `isEqual` callback function). If there is no matching element, returns `NULL`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSHashTable.h

NSHashInsert

Adds an element to the specified hash table.

```
void NSHashInsert (
    NSHashTable *table,
    const void *pointer
);
```

Discussion

Inserts *pointer*, which must not be NULL, into *table*. If *pointer* matches an item already in the table, the previous pointer is released using the `release` callback function that was specified when the table was created.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSHashRemove](#) (page 62)

[NSHashInsertKnownAbsent](#) (page 62)

[NSHashInsertIfAbsent](#) (page 61)

Declared In

NSHashTable.h

NSHashInsertIfAbsent

Adds an element to the specified hash table only if the table does not already contain the element.

```
void * NSHashInsertIfAbsent (
    NSHashTable *table,
    const void *pointer
);
```

Return Value

If *pointer* matches an item already in *table*, returns the preexisting pointer; otherwise, *pointer* is added to the *table* and returns NULL.

Discussion

You must not specify NULL for *pointer*.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSHashRemove](#) (page 62)

[NSHashInsert](#) (page 61)

[NSHashInsertKnownAbsent](#) (page 62)

Declared In

NSHashTable.h

NSHashInsertKnownAbsent

Adds an element to the specified hash table.

```
void NSHashInsertKnownAbsent (
    NSHashTable *table,
    const void *pointer
);
```

Discussion

Inserts *pointer*, which must not be NULL, into *table*. Unlike `NSHashInsert`, this function raises `NSInvalidArgumentException` if *table* already includes an element that matches *pointer*.

Availability

Available in Mac OS X v10.0 and later.

See Also[NSHashRemove](#) (page 62)[NSHashInsert](#) (page 61)[NSHashInsertIfAbsent](#) (page 61)**Declared In**

NSHashTable.h

NSHashRemove

Removes an element from the specified hash table.

```
void NSHashRemove (
    NSHashTable *table,
    const void *pointer
);
```

Discussion

If *pointer* matches an item already in *table*, this function releases the preexisting item.

Availability

Available in Mac OS X v10.0 and later.

See Also[NSHashInsert](#) (page 61)[NSHashInsertKnownAbsent](#) (page 62)[NSHashInsertIfAbsent](#) (page 61)**Declared In**

NSHashTable.h

NSHeight

Returns the height of a given rectangle.

```
CGFloat NSHeight (
    NSRect aRect
);
```

Return Value

The height of *aRect*.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSMaxX](#) (page 84)

[NSMaxY](#) (page 85)

[NSMidX](#) (page 85)

[NSMidY](#) (page 86)

[NSMinX](#) (page 86)

[NSMinY](#) (page 87)

[NSWidth](#) (page 123)

Related Sample Code

[AnimatedTableView](#)

[GLUT](#)

[iChatTheater](#)

[iSpend](#)

[Rulers](#)

Declared In

[NSGeometry.h](#)

NSHFSTypeCodeFromFileType

Returns a file type code.

```
OSType NSHFSTypeCodeFromFileType (
    NSString *fileTypeString
);
```

Parameters

fileTypeString

A string of the sort encoded by [NSFileTypeForHFSTypeCode\(\)](#).

Return Value

The HFS file type code corresponding to *fileTypeString*, or 0 if it cannot be found.

Discussion

For more information, see [HFS File Types](#).

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSHFSFileTypes.h

NSHFSTypeOfFile

Returns a string encoding a file type.

```
NSString * NSHFSTypeOfFile (
    NSString *fullFilePath
);
```

Parameters*fullFilePath*

The full absolute path of a file.

Return ValueA string that encodes *fullFilePath*'s HFS file type, or `nil` if the operation was not successful**Discussion**

For more information, see HFS File Types.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

DeskPictAppDockMenu

Declared In

NSHFSFileTypes.h

NSHomeDirectory

Returns the path to the current user's home directory.

```
NSString * NSHomeDirectory (void);
```

Return Value

The path to the current user's home directory.

DiscussionFor more information on file-system utilities, see *Low-Level File Management Programming Topics*.**Availability**

Available in Mac OS X v10.0 and later.

See Also[NSFullUserName](#) (page 59)[NSUserName](#) (page 123)[NSHomeDirectoryForUser](#) (page 65)**Related Sample Code**

Quartz Composer WWDC 2005 TextEdit

Declared In

NSPathUtilities.h

NSHomeDirectoryForUser

Returns the path to a given user's home directory.

```
NSString * NSHomeDirectoryForUser (
    NSString *userName
);
```

Parameters*userName*

The name of a user.

Return ValueThe path to the home directory for the user specified by *userName*.**Discussion**For more information on file system utilities, see *Low-Level File Management Programming Topics*.**Availability**

Available in Mac OS X v10.0 and later.

See Also[NSFullUserName](#) (page 59)[NSUserName](#) (page 123)[NSHomeDirectory](#) (page 64)**Declared In**

NSPathUtilities.h

NSHostByteOrder

Returns the endian format.

```
long NSHostByteOrder (void);
```

Return ValueThe endian format, either `NS_LittleEndian` or `NS_BigEndian`.**Availability**

Available in Mac OS X v10.0 and later.

Related Sample Code

GLUT

Declared In

NSByteOrder.h

NSIncrementExtraRefCount

Increments the specified object's reference count.

```
void NSIncrementExtraRefCount (
    id object
);
```

Parameters*object*

An object.

Discussion

This function increments the “extra reference” count of *object*. Newly created objects have only one actual reference, so that a single release message results in the object being deallocated. Extra references are those beyond the single original reference and are usually created by sending the object a retain message. Your code should generally not use these functions unless it is overriding the retain or release methods.

Availability

Available in Mac OS X v10.0 and later.

See Also[NSExtraRefCount](#) (page 57)[NSDecrementExtraRefCountWasZero](#) (page 52)**Declared In**

NSObject.h

NSInsetRect

Insets a rectangle by a specified amount.

```
NSRect NSInsetRect (
    NSRect aRect,
    CGFloat dx,
    CGFloat dy
);
```

Return Value

A copy of *aRect*, altered by moving the two sides that are parallel to the y axis inward by *dx*, and the two sides parallel to the x axis inwards by *dy*.

Availability

Available in Mac OS X v10.0 and later.

See Also[NSDivideRect](#) (page 53)[NSIntegralRect](#) (page 67)[NSOffsetRect](#) (page 89)**Related Sample Code**

CocoaSlides

ImageBrowserViewAppearance

ImageKitDemo

Quartz Composer WWDC 2005 TextEdit

Sketch-112

Declared In

NSGeometry.h

NSIntegralRect

Adjusts the sides of a rectangle to integer values.

```
NSRect NSIntegralRect (
    NSRect aRect
);
```

Return Value

A copy of *aRect*, expanded outward just enough to ensure that none of its four defining values (x, y, width, and height) have fractional parts. If the width or height of *aRect* is 0 or negative, this function returns a rectangle with origin at (0.0, 0.0) and with zero width and height.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSDivideRect](#) (page 53)

[NSInsetRect](#) (page 66)

[NSOffsetRect](#) (page 89)

Related Sample Code

AnimatingViews

FilterDemo

ImageApp

PDF Annotation Editor

PDFKitLinker2

Declared In

NSGeometry.h

NSIntersectionRange

Returns the intersection of the specified ranges.

```
NSRange NSIntersectionRange (
    NSRange range1,
    NSRange range2
);
```

Return Value

A range describing the intersection of *range1* and *range2*—that is, a range containing the indices that exist in both ranges.

Discussion

If the returned range's length field is 0, then the two ranges don't intersect, and the value of the location field is undefined.

Availability

Available in Mac OS X v10.0 and later.

See Also[NSUnionRange](#) (page 122)**Related Sample Code**

ClipboardViewer

LayoutManagerDemo

Declared In

NSRange.h

NSIntersectionRect

Calculates the intersection of two rectangles.

```
NSRect NSIntersectionRect (
    NSRect aRect,
    NSRect bRect
);
```

Return Value

The graphic intersection of *aRect* and *bRect*. If the two rectangles don't overlap, the returned rectangle has its origin at (0.0, 0.0) and zero width and height (including situations where the intersection is a point or a line segment).

Availability

Available in Mac OS X v10.0 and later.

See Also[NSUnionRect](#) (page 122)**Related Sample Code**

Cropped Image

FilterDemo

GLUT

Rulers

Sketch-112

Declared In

NSGeometry.h

NSIntersectsRect

Returns a Boolean value that indicates whether two rectangles intersect.

```
BOOL NSIntersectsRect (
    NSRect aRect,
    NSRect bRect
);
```

Return Value

YES if *aRect* intersects *bRect*, otherwise NO. Returns NO if either *aRect* or *bRect* has a width or height that is 0.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSIntersectionRect](#) (page 68)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

Sketch+Accessibility

Sketch-112

TargetGallery

Worm

Declared In

NSGeometry.h

NSIsEmptyRect

Returns a Boolean value that indicates whether a given rectangle is empty.

```
BOOL NSIsEmptyRect (
    NSRect aRect
);
```

Return Value

YES if *aRect* encloses no area at all—that is, if its width or height is 0 or negative, otherwise NO.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

Cropped Image

From A View to A Movie

From A View to A Picture

GLUT

Sketch-112

Declared In

NSGeometry.h

NSJavaBundleCleanup

This function has been deprecated.

```
void NSJavaBundleCleanup (
    NSBundle *bundle,
    NSDictionary *plist
);
```

Availability

Available in Mac OS X v10.2 through Mac OS X v10.5.

Deprecated in Mac OS X v10.5.

Declared In

NSJavaSetup.h

NSJavaBundleSetup

This function has been deprecated.

```
id NSJavaBundleSetup (
    NSBundle *bundle,
    NSDictionary *plist
);
```

Availability

Available in Mac OS X v10.2 through Mac OS X v10.5.

Deprecated in Mac OS X v10.5.

Declared In

NSJavaSetup.h

NSJavaClassesForBundle

Loads the Java classes located in the specified bundle.

```
NSArray * NSJavaClassesForBundle (
    NSBundle *bundle,
    BOOL usesyscl,
    id *vm
);
```

Discussion

Loads and returns the Java classes in the specified bundle. If the Java virtual machine is not loaded, load it first. A reference to the Java virtual machine is returned in the *vm* parameter. You can pass *nil* for the *vm* parameter if you do not want this information. Pass *NO* for *usesyscl* if you want to use a new instance of the class loader to load the classes; otherwise, the system can reuse an existing instance of the class loader. If you pass *NO* for *usesyscl*, the new class loader will be released when you are done with it; otherwise, the class loader will be cached for use next time.

Availability

Available in Mac OS X v10.2 through Mac OS X v10.5.

Deprecated in Mac OS X v10.5.

Declared In

NSJavaSetup.h

NSJavaClassesFromPath

Loads the Java classes located at the specified path.

```
NSArray * NSJavaClassesFromPath (
    NSArray *path,
    NSArray *wanted,
    BOOL usesyscl,
    id *vm
);
```

Discussion

Loads and returns the Java classes in the specified bundle. If the Java virtual machine is not loaded, load it first. A reference to the Java virtual machine is returned in the *vm* parameter. You can pass *nil* for the *vm* parameter if you do not want this information. Pass an array of names of classes to load in the *wanted* parameter. If you pass *nil* for the *wanted* parameter, all classes at the specified path will be loaded. Pass *NO* for *usesyscl* if you want to use a new instance of the class loader to load the classes; otherwise, the system can reuse an existing instance of the class loader. If you pass *NO* for *usesyscl*, the new class loader will be released when you are done with it; otherwise, the class loader will be cached for use next time.

Availability

Available in Mac OS X v10.2 through Mac OS X v10.5.

Deprecated in Mac OS X v10.5.

Declared In

NSJavaSetup.h

NSJavaNeedsToLoadClasses

Returns a Boolean value that indicates whether a virtual machine is needed or if Java classes are provided.

```
BOOL NSJavaNeedsToLoadClasses (
    NSDictionary *plist
);
```

Discussion

Returns *YES* if a virtual machine is needed or if a virtual machine already exists and there's an indication that Java classes are provided.

Availability

Available in Mac OS X v10.0 through Mac OS X v10.5.

Deprecated in Mac OS X v10.5.

Declared In

NSJavaSetup.h

NSJavaNeedsVirtualMachine

Returns a Boolean value that indicates whether a Java virtual machine is required.

```
BOOL NSJavaNeedsVirtualMachine (
    NSDictionary *plist
);
```

Discussion

Returns *YES* if *plist* contains a key saying that it requires Java.

Availability

Available in Mac OS X v10.0 through Mac OS X v10.5.

Deprecated in Mac OS X v10.5.

Declared In

NSJavaSetup.h

NSJavaObjectNamedInPath

Creates an instance of the named class using the class loader previously specified at the given path.

```
id NSJavaObjectNamedInPath (
    NSString *name,
    NSArray *path
);
```

Discussion

Returns a new instance of the class *name*. The class loader must be already be set up for the specified *path* (you can do this using a function such as [NSJavaClassesFromPath](#) (page 70)).

Availability

Available in Mac OS X v10.2 through Mac OS X v10.5.

Deprecated in Mac OS X v10.5.

Declared In

NSJavaSetup.h

NSJavaProvidesClasses

Returns a Boolean value that indicates whether Java classes are provided.

```
BOOL NSJavaProvidesClasses (
    NSDictionary *plist
);
```

Discussion

Returns YES if *plist* contains an NSJavaPath key.

Availability

Available in Mac OS X v10.0 through Mac OS X v10.5.

Deprecated in Mac OS X v10.5.

Declared In

NSJavaSetup.h

NSJavaSetup

Loads the Java virtual machine with specified parameters.


```
id NSJavaSetup (
    NSDictionary *plist
);
```

Discussion

Part of the Java-to-Objective-C bridge. You normally shouldn't use it yourself.

You can pass `nil` for the `plist` dictionary, in which case the Java virtual machine will not be loaded, so you should probably just use [NSJavaSetupVirtualMachine](#) (page 73) instead. The `plist` dictionary may contain the following key-value pairs.

- `NSJavaRoot`—An `NSString` indicating the root of the location where the application's classes are.
- `NSJavaPath`—An `NSArray` of `NSStrings`, each string containing one component of a class path whose components will be prepended by `NSJavaRoot` if they are not absolute locations.
- `NSJavaUserPath`—An `NSString` indicating another segment of the class path so that the application developer can customize where the class loader should search for classes. When searching for classes, this path is searched after the application's class path so that one cannot replace the classes used by the application.
- `NSJavaLibraryPath`—An `NSArray` of `NSStrings`, each string containing one component of a path to search for dynamic shared libraries needed by Java wrappers.
- `NSJavaClasses`—An `NSArray` of `NSStrings`, each string containing the name of one class that the VM should load so that their associated frameworks will be loaded.

Availability

Available in Mac OS X v10.2 through Mac OS X v10.5.

Deprecated in Mac OS X v10.5.

Declared In

`NSJavaSetup.h`

NSJavaSetupVirtualMachine

Sets up the Java virtual machine.

```
id NSJavaSetupVirtualMachine (void);
```

Discussion

Sets up and returns a reference to the Java virtual machine.

Availability

Available in Mac OS X v10.2 through Mac OS X v10.5.

Deprecated in Mac OS X v10.5.

Declared In

`NSJavaSetup.h`

NSLocalizedString

Returns a localized version of a string.

```
NSString *NSLocalizedString(NSString *key, NSString *comment)
```

Return Value

The result of invoking `localizedStringForKey:value:table:` on the main bundle and a `nil` table.

Discussion

You can specify Unicode characters in *key* using `\\Uxxxx`—see the `-u` option for the `genstrings` utility.

For more information, see `NSBundle`.

Special Considerations

In Mac OS X v10.4 and earlier, to ensure correct parsing by the `genstrings` utility, the *key* parameter must not contain any high-ASCII characters.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

EnhancedAudioBurn

GridCalendar

PhotoSearch

Quartz Composer WWDC 2005 TextEdit

Spotlighter

Declared In

`NSBundle.h`

NSLocalizedStringFromTable

Returns a localized version of a string.

```
NSString *NSLocalizedStringFromTable(NSString *key, NSString *tableName, NSString *comment)
```

Return Value

The result of invoking `localizedStringForKey:value:table:` on the main bundle, passing it the specified *key* and *tableName*.

Discussion

You can specify Unicode characters in *key* using `\\Uxxxx`—see the `-u` option for the `genstrings` utility.

For more information, see `NSBundle`.

Special Considerations

In Mac OS X v10.4 and earlier, to ensure correct parsing by the `genstrings` utility, the *key* parameter must not contain any high-ASCII characters.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

BindingsJoystick

Mountains

Quartz Composer WWDC 2005 TextEdit
Sketch+Accessibility

Declared In

NSBundle.h

NSLocalizedStringFromTableInBundle

Returns a localized version of a string.

```
NSString *NSLocalizedStringFromTableInBundle(NSString *key, NSString *tableName,
NSBundle *bundle, NSString *comment)
```

Return Value

The result of invoking `localizedStringForKey:value:table: on bundle`, passing it the specified *key* and *tableName*.

Discussion

You can specify Unicode characters in *key* using `\\Uxxxx`—see the `-u` option for for the `genstrings` utility.

For more information, see `NSBundle`.

Special Considerations

In Mac OS X v10.4 and earlier, to ensure correct parsing by the `genstrings` utility, the *key* parameter must not contain any high-ASCII characters.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

GLUT

Declared In

NSBundle.h

NSLocalizedStringWithDefaultValue

Returns a localized version of a string.

```
NSString *NSLocalizedStringWithDefaultValue(NSString *key, NSString *tableName,
NSBundle *bundle, NSString *value, NSString *comment)
```

Return Value

The result of invoking `localizedStringForKey:value:table: on bundle`, passing it the specified *key*, *value*, and *tableName*.

Discussion

You can specify Unicode characters in *key* using `\\Uxxxx`—see the `-u` option for for the `genstrings` utility.

If you use `genstrings` to parse your code for localizable strings, you can use this method to specify an initial value that is different from *key*.

For more information, see `NSBundle`.

Special Considerations

In Mac OS X v10.4 and earlier, to ensure correct parsing by the `genstrings` utility, the `key` parameter must not contain any high-ASCII characters.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`NSBundle.h`

NSLocationInRange

Returns a Boolean value that indicates whether a specified position is in a given range.

```
BOOL NSLocationInRange (
    NSUInteger loc,
    NSRange range
);
```

Return Value

YES if `loc` lies within `range`—that is, if it's greater than or equal to `range.location` and less than `range.location` plus `range.length`.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

`TextInputView`

Declared In

`NSRange.h`

NSLog

Logs an error message to the Apple System Log facility.

```
void NSLog (
    NSString *format,
    ...
);
```

Discussion

Simply calls [NSLogv](#) (page 77), passing it a variable number of arguments.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSLogv](#) (page 77)

Related Sample Code

`CameraBrowser`

`From A View to A Movie`

From A View to A Picture
 GLSLShowpiece
 GLUT

Declared In

NSObjCRuntime.h

NSLogPageSize

Returns the binary log of the page size.

```
NSUInteger NSLogPageSize (void);
```

Return Value

The binary log of the page size.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSRoundDownToMultipleOfPageSize](#) (page 99)

[NSRoundUpToMultipleOfPageSize](#) (page 99)

[NSPageSize](#) (page 90)

Declared In

NSZone.h

NSLogv

Logs an error message to the Apple System Log facility.

```
void NSLogv (
    NSString *format,
    va_list args
);
```

Discussion

Logs an error message to the Apple System Log facility (see `man 3 asl`). If the `STDERR_FILENO` file descriptor has been redirected away from the default or is going to a `tty`, it will also be written there. If you want to direct output elsewhere, you need to use a custom logging facility.

The message consists of a timestamp and the process ID prefixed to the string you pass in. You compose this string with a format string, *format*, and one or more arguments to be inserted into the string. The format specification allowed by these functions is that which is understood by `NSString`'s formatting capabilities (which is not necessarily the set of format escapes and flags understood by `printf`). The supported format specifiers are described in `String Format Specifiers`. A final hard return is added to the error message if one is not present in the format.

In general, you should use the [NSLog](#) (page 76) function instead of calling this function directly. If you do use this function directly, you must have prepared the variable argument list in the *args* argument by calling the standard C macro `va_start`. Upon completion, you must similarly call the standard C macro `va_end` for this list.

Output from `NSLogv` is serialized, in that only one thread in a process can be doing the writing/logging described above at a time. All attempts at writing/logging a message complete before the next thread can begin its attempts.

The effects of `NSLogv` are not serialized with subsystems other than those discussed above (such as the standard I/O package) and do not produce side effects on those subsystems (such as causing buffered output to be flushed, which may be undesirable).

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSLog](#) (page 76)

Related Sample Code

OutputBins2PDE

Declared In

NSObjCRuntime.h

NSMakeCollectable

Makes a newly allocated Core Foundation object eligible for collection.

```
NS_INLINE id NSMakeCollectable(CFTypeRef cf) {
    return cf ? (id)CFMakeCollectable(cf) : nil;
}
```

Discussion

This function is a wrapper for `CFMakeCollectable`, but its return type is `id`—avoiding the need for casting when using Cocoa objects.

This function may be useful when returning Core Foundation objects in code that must support both garbage-collected and non-garbage-collected environments, as illustrated in the following example.

```
- (CFDateRef)foo {
    CFDateRef aCFDate;
    // ...
    return [NSMakeCollectable(aCFDate) autorelease];
}
```

`CFTypeRef` style objects are garbage collected, yet only sometime after the last `CFRelease` is performed. Particularly for fully-bridged `CFTypeRef` objects such as `CFStrings` and collections (such as `CFDictionary`), you must call either `CFMakeCollectable` or the more type safe `NSMakeCollectable`, preferably right upon allocation.

Availability

Available in Mac OS X v10.5 and later.

Declared In

NSZone.h

NSMakePoint

Creates a new `NSPoint` from the specified values.

```
NSPoint NSMakePoint (
    CGFloat x,
    CGFloat y
);
```

Return Value

An `NSPoint` having the coordinates `x` and `y`.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

ClockControl

GLUT

Sketch+Accessibility

Sketch-112

WhackedTV

Declared In

`NSGeometry.h`

NSMakeRange

Creates a new `NSRange` from the specified values.

```
NSRange NSMakeRange (
    NSUInteger loc,
    NSUInteger len
);
```

Return Value

An `NSRange` with location *location* and length *length*.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

CoreRecipes

DemoAssistant

Quartz Composer WWDC 2005 TextEdit

STUCAuthoringDeviceCocoaSample

TextInputView

Declared In

`NSRange.h`

NSMakeRect

Creates a new `NSRect` from the specified values.

```
NSRect NSMakeRect (
    CGFloat x,
    CGFloat y,
    CGFloat w,
    CGFloat h
);
```

Return Value

An `NSRect` having the specified origin of $[x, y]$ and size of $[w, h]$.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

FilterDemo

FunHouse

GLSLShowpiece

ImageKitDemo

WhackedTV

Declared In

`NSGeometry.h`

NSMakeSize

Returns a new `NSSize` from the specified values.

```
NSSize NSMakeSize (
    CGFloat w,
    CGFloat h
);
```

Return Value

An `NSSize` having the specified *width* and *height*.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

CIRAWFilterSample

From A View to A Movie

FunHouse

QTQuartzPlayer

Sketch-112

Declared In

`NSGeometry.h`

NSMapGet

Returns a map table value for the specified key.


```
void * NSMapGet (
    NSMapTable *table,
    const void *key
);
```

Return Value

The value that *table* maps to *key*, or NULL if *table* doesn't contain *key*.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSMapMember](#) (page 83)

[NSEnumerateMapTable](#) (page 55)

[NSNextMapEnumeratorPair](#) (page 89)

[NSAllMapTableKeys](#) (page 21)

[NSAllMapTableValues](#) (page 22)

Related Sample Code

Sketch+Accessibility

Declared In

NSMapTable.h

NSMapInsert

Inserts a key-value pair into the specified table.

```
void NSMapInsert (
    NSMapTable *table,
    const void *key,
    const void *value
);
```

Discussion

Inserts *key* and *value* into *table*. If *key* matches a key already in *table*, *value* is retained and the previous value is released, using the `retain` and `release` callback functions that were specified when the table was created. Raises `NSInvalidArgumentException` if *key* is equal to the `notAKeyMarker` field of the table's `NSMapTableKeyCallbacks` structure.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSMapRemove](#) (page 83)

[NSMapInsertIfAbsent](#) (page 82)

[NSMapInsertKnownAbsent](#) (page 82)

Related Sample Code

Sketch+Accessibility

Declared In

NSMapTable.h

NSMapInsertIfAbsent

Inserts a key-value pair into the specified table.

```
void * NSMapInsertIfAbsent (
    NSMapTable *table,
    const void *key,
    const void *value
);
```

Return Value

If *key* matches a key already in *table*, the preexisting key; otherwise, *key* and *value* are added to *table* and returns NULL.

Discussion

Raises `NSInvalidArgumentException` if *key* is equal to the `notAKeyMarker` field of the table's `NSMapTableKeyCallBacks` structure.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSMapRemove](#) (page 83)

[NSMapInsert](#) (page 81)

[NSMapInsertKnownAbsent](#) (page 82)

Declared In

`NSMapTable.h`

NSMapInsertKnownAbsent

Inserts a key-value pair into the specified table if the pair had not been previously added.

```
void NSMapInsertKnownAbsent (
    NSMapTable *table,
    const void *key,
    const void *value
);
```

Discussion

Inserts *key* (which must not be `notAKeyMarker`) and *value* into *table*. Unlike `NSMapInsert`, this function raises `NSInvalidArgumentException` if *table* already includes a key that matches *key*.

key is compared with `notAKeyMarker` using pointer comparison; if *key* is identical to `notAKeyMarker`, raises `NSInvalidArgumentException`.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSMapRemove](#) (page 83)

[NSMapInsert](#) (page 81)

[NSMapInsertIfAbsent](#) (page 82)

Declared In

NSMapTable.h

NSMapMember

Indicates whether a given table contains a given key.

```

BOOL NSMapMember (
    NSMapTable *table,
    const void *key,
    void **originalKey,
    void **value
);

```

Return ValueYES if *table* contains a key equal to *key*, otherwise NO.**Discussion**

If *table* contains a key equal to *key*, *originalKey* is set to *key*, and *value* is set to the value that *table* maps to *key*.

Availability

Available in Mac OS X v10.0 and later.

See Also[NSMapGet](#) (page 80)[NSEnumerateMapTable](#) (page 55)[NSNextMapEnumeratorPair](#) (page 89)[NSAllMapTableKeys](#) (page 21)[NSAllMapTableValues](#) (page 22)**Declared In**

NSMapTable.h

NSMapRemove

Removes a key and corresponding value from the specified table.

```

void NSMapRemove (
    NSMapTable *table,
    const void *key
);

```

Discussion

If *key* matches a key already in *table*, this function releases the preexisting key and its corresponding value.

Availability

Available in Mac OS X v10.0 and later.

See Also[NSMapInsert](#) (page 81)[NSMapInsertIfAbsent](#) (page 82)[NSMapInsertKnownAbsent](#) (page 82)

Declared In

NSMapTable.h

NSMaxRange

Returns the sum of the location and length of the range.

```
NSUInteger NSMaxRange (
    NSRange range
);
```

Return ValueThe sum of the location and length of the range—that is, `range.location + range.length`.**Availability**

Available in Mac OS X v10.0 and later.

Related Sample Code

ClipboardViewer

GeekGameBoard

iSpend

Quartz Composer WWDC 2005 TextEdit

TextLinks

Declared In

NSRange.h

NSMaxX

Returns the largest x coordinate of a given rectangle.

```
CGFloat NSMaxX (
    NSRect aRect
);
```

Return ValueThe largest x coordinate value within *aRect*.**Availability**

Available in Mac OS X v10.0 and later.

See Also[NSWidth](#) (page 123)[NSHeight](#) (page 63)[NSMaxY](#) (page 85)**Related Sample Code**

AnimatedTableView

BezierPathLab

Rulers

Sketch+Accessibility

Sketch-112

Declared In

NSGeometry.h

NSMaxY

Returns the largest y coordinate of a given rectangle.

```
CGFloat NSMaxY (  
    NSRect aRect  
);
```

Return Value

The largest y coordinate value within *aRect*.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSWidth](#) (page 123)

[NSHeight](#) (page 63)

[NSMaxX](#) (page 84)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

Rulers

Sketch+Accessibility

Sketch-112

ZipBrowser

Declared In

NSGeometry.h

NSMidX

Returns the x coordinate of a given rectangle's midpoint.

```
CGFloat NSMidX (  
    NSRect aRect  
);
```

Return Value

Returns the x coordinate of the center of *aRect*.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSWidth](#) (page 123)

[NSHeight](#) (page 63)

[NSMidY](#) (page 86)

Related Sample Code

QTQuartzPlayer
 QuickLookSketch
 Sketch+Accessibility
 Sketch-112
 TrackBall

Declared In

NSGeometry.h

NSMidY

Returns the y coordinate of a given rectangle's midpoint.

```
CGFloat NSMidY (
    NSRect aRect
);
```

Return Value

The y coordinate of *aRect*'s center point.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSWidth](#) (page 123)

[NSHeight](#) (page 63)

[NSMidX](#) (page 85)

Related Sample Code

AnimatedTableView
 QuickLookSketch
 Sketch+Accessibility
 Sketch-112
 TrackBall

Declared In

NSGeometry.h

NSMinX

Returns the smallest x coordinate of a given rectangle.

```
CGFloat NSMinX (
    NSRect aRect
);
```

Return Value

The smallest x coordinate value within *aRect*.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSWidth](#) (page 123)

[NSHeight](#) (page 63)

[NSMinY](#) (page 87)

Related Sample Code

AnimatedTableView

PhotoSearch

Rulers

Sketch+Accessibility

Sketch-112

Declared In

NSGeometry.h

NSMinY

Returns the smallest y coordinate of a given rectangle.

```
CGFloat NSMinY (  
    NSRect aRect  
);
```

Return Value

The smallest y coordinate value within *aRect*.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSWidth](#) (page 123)

[NSHeight](#) (page 63)

[NSMinX](#) (page 86)

Related Sample Code

QTQuartzPlayer

QuickLookSketch

Rulers

Sketch+Accessibility

Sketch-112

Declared In

NSGeometry.h

NSMouseInRect

Returns a Boolean value that indicates whether the point is in the specified rectangle.

```

BOOL NSMouseInRect (
    NSPoint aPoint,
    NSRect aRect,
    BOOL flipped
);

```

Return Value

YES if the hot spot of the cursor lies inside a given rectangle, otherwise NO.

Discussion

This method assumes an unscaled and unrotated coordinate system. Specify YES for *isFlipped* if the underlying view uses a flipped coordinate system.

Point-in-rectangle functions generally assume that the bottom edge of a rectangle is outside of the rectangle boundaries, while the upper edge is inside the boundaries. This method views *aRect* from the point of view of the user—that is, this method always treats the bottom edge of the rectangle as the one closest to the bottom edge of the user’s screen. By making this adjustment, this function ensures consistent mouse-detection behavior from the user’s perspective.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSPointInRect](#) (page 93)

Related Sample Code

DragNDropOutlineView

GLUT

ImageMap

ImageMapExample

PhotoSearch

Declared In

NSGeometry.h

NSNextHashEnumeratorItem

Returns the next hash-table element in the enumeration.

```

void * NSNextHashEnumeratorItem (
    NSHashEnumerator *enumerator
);

```

Return Value

The next element in the table that *enumerator* is associated with, or NULL if *enumerator* has already iterated over all the elements.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSEnumerateHashTable](#) (page 54)

Declared In

NSHashTable.h

NSNextMapEnumeratorPair

Returns a Boolean value that indicates whether the next map-table pair in the enumeration are set.

```
BOOL NSNextMapEnumeratorPair (
    NSMapEnumerator *enumerator,
    void **key,
    void **value
);
```

Return Value

NO if *enumerator* has already iterated over all the elements in the table that *enumerator* is associated with; otherwise, sets *key* and *value* to match the next key-value pair in the table and returns YES.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSEnumerateMapTable](#) (page 55)

[NSMapMember](#) (page 83)

[NSMapGet](#) (page 80)

[NSAllMapTableKeys](#) (page 21)

[NSAllMapTableValues](#) (page 22)

Related Sample Code

Sketch+Accessibility

Declared In

NSMapTable.h

NSOffsetRect

Offsets the rectangle by the specified amount.

```
NSRect NSOffsetRect (
    NSRect aRect,
    CGFloat dX,
    CGFloat dY
);
```

Return Value

A copy of *aRect*, with its location shifted by *dX* along the x axis and by *dY* along the y axis.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSDivideRect](#) (page 53)

[NSInsetRect](#) (page 66)

[NSIntegralRect](#) (page 67)

Related Sample Code

PDFView

QuickLookSketch

Sketch+Accessibility

Sketch-112

TextLinks

Declared In

NSGeometry.h

NSOpenStepRootDirectory

Returns the root directory of the user's system.

```
NSString * NSOpenStepRootDirectory (void);
```

Return Value

A string identifying the root directory of the user's system.

Discussion

For more information on file system utilities, see *Low-Level File Management Programming Topics*.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSHomeDirectory](#) (page 64)

[NSHomeDirectoryForUser](#) (page 65)

Declared In

NSPathUtilities.h

NSPageSize

Returns the number of bytes in a page.

```
NSUInteger NSPageSize (void);
```

Return Value

The number of bytes in a page.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSRoundDownToMultipleOfPageSize](#) (page 99)

[NSRoundUpToMultipleOfPageSize](#) (page 99)

[NSLogPageSize](#) (page 77)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

Declared In

NSZone.h

NSParameterAssert

Validates the specified parameter.

NSParameterAssert(condition)

Discussion

Assertions evaluate a condition and, if the condition evaluates to false, call the assertion handler for the current thread, passing it a format string and a variable number of arguments. Each thread has its own assertion handler, which is an object of class `NSAssertionHandler`. When invoked, an assertion handler prints an error message that includes method and class names (or the function name). It then raises an `NSInternalInconsistencyException` exception.

This macro validates a parameter for an Objective-C method. Simply provide the parameter as the *condition* argument. The macro evaluates the parameter and, if it is false, it logs an error message that includes the parameter and then raises an exception.

Assertions are disabled if the preprocessor macro `NS_BLOCK_ASSERTIONS` is defined. All assertion macros return void.

Availability

Available in Mac OS X v10.0 and later.

See Also[NSLog](#) (page 76)[NSLogv](#) (page 77)[NSAssert](#) (page 24)[NSCAssert](#) (page 30)[NSCParameterAssert](#) (page 41)**Related Sample Code**

GeekGameBoard

QuickLookSketch

Sketch+Accessibility

Sketch-112

TimelineToTC

Declared In

NSException.h

NSPointFromCGPointReturns an `NSPoint` typecast from a `CGPoint`.

```

NSRect NSRectFromCGRect(CGRect cgregt) {
    union _ {NSRect ns; CGRect cg;};
    return ((union _ *)&cgregt)->ns;
}

```

Return Value

An `NSPoint` typecast from a `CGPoint`.

Availability

Available in Mac OS X v10.5 and later.

See Also

[NSPointToCGPoint](#) (page 93)

[NSRectFromCGRect](#) (page 96)

[NSSizeFromCGSize](#) (page 102)

Related Sample Code

LightTable

Declared In

NSGeometry.h

NSPointFromString

Returns a point from a text-based representation.

```

NSPoint NSPointFromString (
    NSString *aString
);

```

Parameters

aString

A string of the form “{x, y}”.

Return Value

If *aString* is of the form “{x, y}” an `NSPoint` structure that uses x and y as the x and y coordinates, in that order.

If *aString* only contains a single number, it is used as the x coordinate. If *aString* does not contain any numbers, returns an `NSPoint` object whose x and y coordinates are both 0.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSStringFromPoint](#) (page 105)

Related Sample Code

QuickLookSketch

Sketch+Accessibility

Declared In

NSGeometry.h

NSPointInRect

Returns a Boolean value that indicates whether a given point is in a given rectangle.

```

BOOL NSPointInRect (
    NSPoint aPoint,
    NSRect aRect
);

```

Return Value

YES if *aPoint* is located within the rectangle represented by *aRect*, otherwise NO.

Discussion

Point-in-rectangle functions generally assume that the “upper” and “left” edges of a rectangle are inside the rectangle boundaries, while the “lower” and “right” edges are outside the boundaries. This method treats the “upper” and “left” edges of the rectangle as the ones containing the origin of the rectangle.

Special Considerations

The meanings of “upper” and “lower” (and “left” and “right”) are relative to the current coordinate system and the location of the rectangle. For a rectangle of positive height located in positive x and y coordinates:

- In the default Mac OS X desktop coordinate system—where the origin is at the bottom left—the rectangle edge closest to the bottom of the screen is the “upper” edge (and is considered inside the rectangle).
- On iOS and in a flipped coordinate system on Mac OS X desktop—where the origin is at the top left—the rectangle edge closest to the bottom of the screen is the “lower” edge (and is considered outside the rectangle).

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSMouseInRect](#) (page 87)

Related Sample Code

[AnimatedTableView](#)

[FunkyOverlayWindow](#)

[LiveVideoMixer2](#)

[LiveVideoMixer3](#)

[Sketch-112](#)

Declared In

[NSGeometry.h](#)

NSPointToCGPoint

Returns a `CGPoint` typecast from an `NSPoint`.

```
CGPoint NSPointToCGPoint(NSPoint nspoint) {
    union _ {NSPoint ns; CGPoint cg;};
    return ((union _ *)&nspoint)->cg;
}
```

Return Value

A `CGPoint` typecast from an `NSPoint`.

Availability

Available in Mac OS X v10.5 and later.

See Also

[NSPointFromCGPoint](#) (page 91)

[NSRectToCGRect](#) (page 97)

[NSSizeToCGSize](#) (page 103)

Related Sample Code

GeekGameBoard

LightTable

Declared In

`NSGeometry.h`

NSProtocolFromString

Returns a the protocol with a given name.

```
Protocol *NSProtocolFromString (
    NSString *namestr
);
```

Parameters

namestr

The name of a protocol.

Return Value

The protocol object named by *namestr*, or `nil` if no protocol by that name is currently loaded. If *namestr* is `nil`, returns `nil`.

Availability

Available in Mac OS X v10.5 and later.

See Also

[NSStringFromProtocol](#) (page 106)

[NSClassFromString](#) (page 34)

[NSSelectorFromString](#) (page 100)

Declared In

`NSObjCRuntime.h`

NSRangeFromString

Returns a range from a textual representation.

```
NSRange NSRangeFromString (
    NSString *aString
);
```

Discussion

Scans *aString* for two integers which are used as the location and length values, in that order, to create an NSRange struct. If *aString* only contains a single integer, it is used as the location value. If *aString* does not contain any integers, this function returns an NSRange struct whose location and length values are both 0.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSStringFromRange](#) (page 106)

Declared In

NSRange.h

NSReallocateCollectable

Reallocates collectable memory.

```
void *__strong NSReallocateCollectable (
    void *ptr,
    NSUInteger size,
    NSUInteger options
);
```

Discussion

Changes the size of the block of memory pointed to by *ptr* to *size* bytes. It may allocate new memory to replace the old, in which case it moves the contents of the old memory block to the new block, up to a maximum of *size* bytes.

options can be 0 or `NSScannedOption`: A value of 0 allocates nonscanned memory; a value of `NSScannedOption` allocates scanned memory.

This function returns NULL if it's unable to allocate the requested memory.

Availability

Available in Mac OS X v10.4 and later.

Declared In

NSZone.h

NSRealMemoryAvailable

Returns information about the user's system.

```
NSUInteger NSRealMemoryAvailable (void);
```

Return Value

The number of bytes available in RAM.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSZone.h

NSRectFromCGRect

Returns an `NSRect` typecast from a `CGRect`.

```
NSRect NSRectFromCGRect(CGRect cgrect) {
    union _ {NSRect ns; CGRect cg;};
    return ((union _ *)&cgrect)->ns;
}
```

Return Value

An `NSRect` typecast from a `CGRect`.

Availability

Available in Mac OS X v10.5 and later.

See Also

[NSRectToCGRect](#) (page 97)

[NSPointFromCGPoint](#) (page 91)

[NSSizeFromCGSize](#) (page 102)

Related Sample Code

iChatTheater

LightTable

Declared In

NSGeometry.h

NSRectFromString

Returns a rectangle from a text-based representation.

```
NSRect NSRectFromString (
    NSString *aString
);
```

Discussion

Scans *aString* for four numbers which are used as the x and y coordinates and the width and height, in that order, to create an `NSPoint` object. If *aString* does not contain four numbers, those numbers that were scanned are used, and 0 is used for the remaining values. If *aString* does not contain any numbers, this function returns an `NSRect` object with a rectangle whose origin is (0, 0) and width and height are both 0.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSStringFromRect](#) (page 107)

Related Sample Code

DynamicProperties
 LightTable
 QuickLookSketch
 Sketch+Accessibility

Declared In

NSGeometry.h

NSRectToCGRect

Returns a CGRect typecast from an NSRect.

```
CGRect NSRectToCGRect(NSRect nsrect) {
    union _ {NSRect ns; CGRect cg;};
    return ((union _ *)&nsrect)->cg;
}
```

Return Value

A CGRect typecast from an NSRect.

Availability

Available in Mac OS X v10.5 and later.

See Also

[NSRectFromCGRect](#) (page 96)
[NSPointToCGPoint](#) (page 93)
[NSSizeToCGSize](#) (page 103)

Related Sample Code

CoreTextArcCocoa
 From A View to A Movie
 From A View to A Picture
 LightTable
 Quartz 2D Transformer

Declared In

NSGeometry.h

NSRecycleZone

Frees memory in a zone.

```
void NSRecycleZone (
    NSZone *zone
);
```

Discussion

Frees *zone* after adding any of its pointers still in use to the default zone. (This strategy prevents retained objects from being inadvertently destroyed.)

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSCreateZone](#) (page 44)

[NSZoneMalloc](#) (page 126)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

Declared In

NSZone.h

NSResetHashTable

Deletes the elements of the specified hash table.

```
void NSResetHashTable (
    NSHashTable *table
);
```

Discussion

Releases each element but doesn't deallocate *table*. This function is useful for preserving the capacity of *table*.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSFreeHashTable](#) (page 58)

Declared In

NSHashTable.h

NSResetMapTable

Deletes the elements of the specified map table.

```
void NSResetMapTable (
    NSMapTable *table
);
```

Parameters

table

A reference to a map table structure.

Discussion

Releases each key and value but doesn't deallocate *table*. This method is useful for preserving the capacity of *table*.

Availability

Available in Mac OS X v10.0 and later.

See Also[NSFreeMapTable](#) (page 59)**Declared In**

NSMapTable.h

NSRoundDownToMultipleOfPageSize

Returns the specified number of bytes rounded down to a multiple of the page size.

```

NSUInteger NSRoundDownToMultipleOfPageSize (
    NSUInteger bytes
);

```

Return ValueIn bytes, the multiple of the page size that is closest to, but not greater than, *byteCount* (that is, the number of bytes rounded down to a multiple of the page size).**Availability**

Available in Mac OS X v10.0 and later.

See Also[NSPageSize](#) (page 90)[NSLogPageSize](#) (page 77)[NSRoundUpToMultipleOfPageSize](#) (page 99)**Declared In**

NSZone.h

NSRoundUpToMultipleOfPageSize

Returns the specified number of bytes rounded up to a multiple of the page size.

```

NSUInteger NSRoundUpToMultipleOfPageSize (
    NSUInteger bytes
);

```

Return ValueIn bytes, the multiple of the page size that is closest to, but not less than, *byteCount* (that is, the number of bytes rounded up to a multiple of the page size).**Availability**

Available in Mac OS X v10.0 and later.

See Also[NSPageSize](#) (page 90)[NSLogPageSize](#) (page 77)[NSRoundDownToMultipleOfPageSize](#) (page 99)**Declared In**

NSZone.h

NSSearchPathForDirectoriesInDomains

Creates a list of directory search paths.

```
NSArray * NSSearchPathForDirectoriesInDomains (
    NSSearchPathDirectory directory,
    NSSearchPathDomainMask domainMask,
    BOOL expandTilde
);
```

Discussion

Creates a list of path strings for the specified directories in the specified domains. The list is in the order in which you should search the directories. If *expandTilde* is YES, tildes are expanded as described in *stringByExpandingTildeInPath*.

For more information on file system utilities, see *Locating Directories on the System*.

Note: The directory returned by this method may not exist. This method simply gives you the appropriate location for the requested directory. Depending on the application's needs, it may be up to the developer to create the appropriate directory and any in between.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

Core Data HTML Store
 CoreRecipes
 DesktopImage
 From A View to A Movie
 From A View to A Picture

Declared In

NSPathUtilities.h

NSSelectorFromString

Returns the selector with a given name.

```
SEL NSSelectorFromString (
    NSString *aSelectorName
);
```

Parameters

aSelectorName

A string of any length, with any characters, that represents the name of a selector.

Return Value

The selector named by *aSelectorName*. If *aSelectorName* is nil, or cannot be converted to UTF-8 (this should be only due to insufficient memory), returns (SEL)0.

Discussion

To make a selector, `NSStringFromClass` passes a UTF-8 encoded character representation of `aSelectorName` to `sel_registerName` and returns the value returned by that function. Note, therefore, that if the selector does not exist it is registered and the newly-registered selector is returned.

Recall that a colon (":") is part of a method name; `setHeight` is not the same as `setHeight:`. For more about methods names, see *Objects, Classes, and Messaging* in *The Objective-C Programming Language*.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSStringFromSelector](#) (page 107)

[NSProtocolFromString](#) (page 94)

[NSClassFromString](#) (page 34)

Related Sample Code

CoreRecipes

ImageMap

ImageMapExample

Declared In

`NSObjCRuntime.h`

NSSetUncaughtExceptionHandler

Changes the top-level error handler.

```
void NSSetUncaughtExceptionHandler (
    NSUncaughtExceptionHandler *
);
```

Discussion

Sets the top-level error-handling function where you can perform last-minute logging before the program terminates.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSetUncaughtExceptionHandler](#) (page 60)

`reportException:` (`NSApplication`)

Declared In

`NSException.h`

NSSetZoneName

Sets the name of the specified zone.

```
void NSSetZoneName (
    NSZone *zone,
    NSString *name
);
```

Discussion

Sets the name of *zone* to *name*, which can aid in debugging.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSZoneName](#) (page 126)

Related Sample Code

Quartz Composer WWDC 2005 TextEdit

Declared In

NSZone.h

NSShouldRetainWithZone

Indicates whether an object should be retained.

```
BOOL NSShouldRetainWithZone (
    id anObject,
    NSZone *requestedZone
);
```

Parameters

anObject

An object.

requestedZone

A memory zone.

Return Value

Returns YES if *requestedZone* is NULL, the default zone, or the zone in which *anObject* was allocated; otherwise NO.

Discussion

This function is typically called from inside an NSObject's `copyWithZone:`, when deciding whether to retain *anObject* as opposed to making a copy of it.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSObject.h

NSSizeFromCGSize

Returns an `NSSize` typecast from a `CGSize`.

```

NSSize NSSizeFromCGSize(CGSize cgs) {
    return (*(NSSize *)&(cgs));
}

```

Return Value

An `NSSize` typecast from a `CGSize`.

Availability

Available in Mac OS X v10.5 and later.

See Also

[NSSizeToCGSize](#) (page 103)

[NSPointFromCGPoint](#) (page 91)

[NSRectFromCGRect](#) (page 96)

Related Sample Code

[AnimatedTableView](#)

Declared In

`NSGeometry.h`

NSSizeFromString

Returns an `NSSize` from a text-based representation.

```

NSSize NSSizeFromString (
    NSString *aString
);

```

Discussion

Scans *aString* for two numbers which are used as the width and height, in that order, to create an `NSSize` struct. If *aString* only contains a single number, it is used as the width. The *aString* argument should be formatted like the output of [NSStringFromSize](#) (page 108), for example, `@{10,20}`. If *aString* does not contain any numbers, this function returns an `NSSize` struct whose width and height are both 0.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSStringFromSize](#) (page 108)

Declared In

`NSGeometry.h`

NSSizeToCGSize

Returns a `CGSize` typecast from an `NSSize`.

```

CGSize NSSizeToCGSize(NSSize nsize) {
    return (*(CGSize *)&(nsize));
}

```

Return Value

A `CGSize` typecast from an `NSSize`.

Availability

Available in Mac OS X v10.5 and later.

See Also

[NSSizeFromCGSize](#) (page 102)

[NSPointToCGPoint](#) (page 93)

[NSRectToCGRect](#) (page 97)

Related Sample Code

LightTable

Quartz 2D Shadings

Declared In

NSGeometry.h

NSStringFromClass

Returns the name of a class as a string.

```
NSString * NSStringFromClass (
    Class aClass
);
```

Parameters

aClass

A class.

Return Value

A string containing the name of *aClass*. If *aClass* is nil, returns nil.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSClassFromString](#) (page 34)

[NSStringFromProtocol](#) (page 106)

[NSStringFromSelector](#) (page 107)

Related Sample Code

FunHouse

QuickLookSketch

Sketch+Accessibility

Sketch-112

ToolbarSample

Declared In

NSObjCRuntime.h

NSStringFromHashTable

Returns a string describing the hash table's contents.


```
NSString * NSStringFromHashTable (
    NSHashTable *table
);
```

Return Value

A string describing *table*'s contents.

Discussion

The function iterates over the elements of *table*, and for each one appends the string returned by the `describe` callback function. If `NULL` was specified for the callback function, the hexadecimal value of each pointer is added to the string.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSHashTable.h`

NSStringFromMapTable

Returns a string describing the map table's contents.

```
NSString * NSStringFromMapTable (
    NSMapTable *table
);
```

Parameters

table

A reference to a map table structure.

Return Value

A string describing the map table's contents.

Discussion

The function iterates over the key-value pairs of *table* and for each one appends the string "*a = b;\n*", where *a* and *b* are the key and value strings returned by the corresponding `describe` callback functions. If `NULL` was specified for the callback function, *a* and *b* are the key and value pointers, expressed as hexadecimal numbers.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSMapTable.h`

NSStringFromPoint

Returns a string representation of a point.

```
NSString * NSStringFromPoint (
    NSPoint aPoint
);
```

Parameters*aPoint*

A point structure.

Return ValueA string of the form “{*a*, *b*}”, where *a* and *b* are the x and y coordinates of *aPoint*.**Availability**

Available in Mac OS X v10.0 and later.

See Also[NSPointFromString](#) (page 92)**Declared In**

NSGeometry.h

NSStringFromProtocol

Returns the name of a protocol as a string.

```
NSString * NSStringFromProtocol (
    Protocol *proto
);
```

Parameters*proto*

A protocol.

Return ValueA string containing the name of *proto*.**Availability**

Available in Mac OS X v10.5 and later.

See Also[NSProtocolFromString](#) (page 94)[NSStringFromClass](#) (page 104)[NSStringFromSelector](#) (page 107)**Declared In**

NSObjCRuntime.h

NSStringFromRange

Returns a string representation of a range.

```
NSString * NSStringFromRange (
    NSRange range
);
```

Return Value

A string of the form “{*a*, *b*}”, where *a* and *b* are non-negative integers representing *aRange*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSRange.h

NSStringFromRect

Returns a string representation of a rectangle.

```
NSString * NSStringFromRect (
    NSRect aRect
);
```

Discussion

Returns a string of the form “{{*a*, *b*}, {*c*, *d*}}”, where *a*, *b*, *c*, and *d* are the x and y coordinates and the width and height, respectively, of *aRect*.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSRectFromString](#) (page 96)

Related Sample Code

DynamicProperties

LightTable

Declared In

NSGeometry.h

NSStringFromSelector

Returns a string representation of a given selector.

```
NSString * NSStringFromSelector (
    SEL aSelector
);
```

Parameters

aSelector

A selector.

Return Value

A string representation of *aSelector*.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSelectorFromString](#) (page 100)

[NSStringFromProtocol](#) (page 106)

[NSStringFromClass](#) (page 104)

Related Sample Code

CallJS

EnhancedAudioBurn

QT Capture Widget

SpecialPictureProtocol

WebKitPluginWithJavaScript

Declared In

NSObjCRuntime.h

NSStringFromSize

Returns a string representation of a size.

```
NSString * NSStringFromSize (
    NSSize aSize
);
```

Return Value

A string of the form “{a, b}”, where a and b are the width and height, respectively, of *aSize*.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSizeFromString](#) (page 103)

Declared In

NSGeometry.h

NSSwapBigDoubleToHost

A utility for swapping the bytes of a number.

```
double NSSwapBigDoubleToHost (
    NSSwappedDouble x
);
```

Discussion

Converts the big-endian value in *x* to the current endian format and returns the resulting value. If it is necessary to swap the bytes of *x*, this function calls [NSSwapDouble](#) (page 111) to perform the swap.

Availability

Available in Mac OS X v10.0 and later.

See Also[NSSwapHostDoubleToBig](#) (page 112)[NSSwapLittleDoubleToHost](#) (page 117)**Declared In**

NSByteOrder.h

NSSwapBigFloatToHost

A utility for swapping the bytes of a number.

```
float NSSwapBigFloatToHost (
    NSSwappedFloat x
);
```

Discussion

Converts the big-endian value in *x* to the current endian format and returns the resulting value. If it is necessary to swap the bytes of *x*, this function calls [NSSwapFloat](#) (page 111) to perform the swap.

Availability

Available in Mac OS X v10.0 and later.

See Also[NSSwapHostFloatToBig](#) (page 113)[NSSwapLittleFloatToHost](#) (page 118)**Declared In**

NSByteOrder.h

NSSwapBigIntToHost

A utility for swapping the bytes of a number.

```
unsigned int NSSwapBigIntToHost (
    unsigned int x
);
```

Discussion

Converts the big-endian value in *x* to the current endian format and returns the resulting value. If it is necessary to swap the bytes of *x*, this function calls [NSSwapInt](#) (page 117) to perform the swap.

Availability

Available in Mac OS X v10.0 and later.

See Also[NSSwapHostIntToBig](#) (page 113)[NSSwapLittleIntToHost](#) (page 118)**Declared In**

NSByteOrder.h

NSSwapBigLongLongToHost

A utility for swapping the bytes of a number.

```

unsigned long long NSSwapBigLongLongToHost (
    unsigned long long x
);

```

Discussion

Converts the big-endian value in *x* to the current endian format and returns the resulting value. If it is necessary to swap the bytes of *x*, this function calls [NSSwapLongLong](#) (page 121) to perform the swap.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapHostLongLongToBig](#) (page 114)

[NSSwapLittleLongLongToHost](#) (page 119)

Declared In

NSByteOrder.h

NSSwapBigLongToHost

A utility for swapping the bytes of a number.

```

unsigned long NSSwapBigLongToHost (
    unsigned long x
);

```

Discussion

Converts the big-endian value in *x* to the current endian format and returns the resulting value. If it is necessary to swap the bytes of *x*, this function calls [NSSwapLong](#) (page 120) to perform the swap.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapHostLongToBig](#) (page 115)

[NSSwapLittleLongToHost](#) (page 119)

Declared In

NSByteOrder.h

NSSwapBigShortToHost

A utility for swapping the bytes of a number.

```
unsigned short NSSwapBigShortToHost (
    unsigned short x
);
```

Discussion

Converts the big-endian value in *x* to the current endian format and returns the resulting value. If it is necessary to swap the bytes of *x*, this function calls [NSSwapShort](#) (page 121) to perform the swap.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapHostShortToBig](#) (page 116)

[NSSwapLittleShortToHost](#) (page 120)

Declared In

NSByteOrder.h

NSSwapDouble

A utility for swapping the bytes of a number.

```
NSSwappedDouble NSSwapDouble (
    NSSwappedDouble x
);
```

Discussion

Swaps the bytes of *x* and returns the resulting value. Bytes are swapped from each low-order position to the corresponding high-order position and vice versa. For example, if the bytes of *x* are numbered from 1 to 8, this function swaps bytes 1 and 8, bytes 2 and 7, bytes 3 and 6, and bytes 4 and 5.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapLongLong](#) (page 121)

[NSSwapFloat](#) (page 111)

Declared In

NSByteOrder.h

NSSwapFloat

A utility for swapping the bytes of a number.

```
NSSwappedFloat NSSwapFloat (
    NSSwappedFloat x
);
```

Discussion

Swaps the bytes of *x* and returns the resulting value. Bytes are swapped from each low-order position to the corresponding high-order position and vice versa. For example, if the bytes of *x* are numbered from 1 to 4, this function swaps bytes 1 and 4, and bytes 2 and 3.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapLong](#) (page 120)

[NSSwapDouble](#) (page 111)

Declared In

NSByteOrder.h

NSSwapHostDoubleToBig

A utility for swapping the bytes of a number.

```
NSSwappedDouble NSSwapHostDoubleToBig (
    double x
);
```

Discussion

Converts the value in x , specified in the current endian format, to big-endian format and returns the resulting value. If it is necessary to swap the bytes, this function calls [NSSwapDouble](#) (page 111) to perform the swap.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapBigDoubleToHost](#) (page 108)

[NSSwapHostDoubleToLittle](#) (page 112)

Declared In

NSByteOrder.h

NSSwapHostDoubleToLittle

A utility for swapping the bytes of a number.

```
NSSwappedDouble NSSwapHostDoubleToLittle (
    double x
);
```

Discussion

Converts the value in x , specified in the current endian format, to little-endian format and returns the resulting value. If it is necessary to swap the bytes, this function calls [NSSwapDouble](#) (page 111) to perform the swap.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapLittleDoubleToHost](#) (page 117)

[NSSwapHostDoubleToBig](#) (page 112)

Declared In

NSByteOrder.h

NSSwapHostFloatToBig

A utility for swapping the bytes of a number.

```
NSSwappedFloat NSSwapHostFloatToBig (  
    float x  
);
```

Discussion

Converts the value in *x*, specified in the current endian format, to big-endian format and returns the resulting value. If it is necessary to swap the bytes, this function calls [NSSwapFloat](#) (page 111) to perform the swap.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapBigFloatToHost](#) (page 109)

[NSSwapHostFloatToLittle](#) (page 113)

Declared In

NSByteOrder.h

NSSwapHostFloatToLittle

A utility for swapping the bytes of a number.

```
NSSwappedFloat NSSwapHostFloatToLittle (  
    float x  
);
```

Discussion

Converts the value in *x*, specified in the current endian format, to little-endian format and returns the resulting value. If it is necessary to swap the bytes, this function calls [NSSwapFloat](#) (page 111) to perform the swap.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapLittleFloatToHost](#) (page 118)

[NSSwapHostFloatToBig](#) (page 113)

Declared In

NSByteOrder.h

NSSwapHostIntToBig

A utility for swapping the bytes of a number.

```
unsigned int NSSwapHostIntToBig (
    unsigned int x
);
```

Discussion

Converts the value in x , specified in the current endian format, to big-endian format and returns the resulting value. If it is necessary to swap the bytes, this function calls [NSSwapInt](#) (page 117) to perform the swap.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapBigIntToHost](#) (page 109)

[NSSwapHostIntToLittle](#) (page 114)

Related Sample Code

QTMetadataEditor

Declared In

NSByteOrder.h

NSSwapHostIntToLittle

A utility for swapping the bytes of a number.

```
unsigned int NSSwapHostIntToLittle (
    unsigned int x
);
```

Discussion

Converts the value in x , specified in the current endian format, to little-endian format and returns the resulting value. If it is necessary to swap the bytes, this function calls [NSSwapInt](#) (page 117) to perform the swap.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapLittleIntToHost](#) (page 118)

[NSSwapHostIntToBig](#) (page 113)

Declared In

NSByteOrder.h

NSSwapHostLongLongToBig

A utility for swapping the bytes of a number.

```
unsigned long long NSSwapHostLongLongToBig (
    unsigned long long x
);
```

Discussion

Converts the value in x , specified in the current endian format, to big-endian format and returns the resulting value. If it is necessary to swap the bytes, this function calls [NSSwapLongLong](#) (page 121) to perform the swap.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapBigLongLongToHost](#) (page 110)

[NSSwapHostLongLongToLittle](#) (page 115)

Declared In

NSByteOrder.h

NSSwapHostLongLongToLittle

A utility for swapping the bytes of a number.

```
unsigned long long NSSwapHostLongLongToLittle (
    unsigned long long x
);
```

Discussion

Converts the value in x , specified in the current endian format, to little-endian format and returns the resulting value. If it is necessary to swap the bytes, this function calls [NSSwapLongLong](#) (page 121) to perform the swap.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapLittleLongLongToHost](#) (page 119)

[NSSwapHostLongLongToBig](#) (page 114)

Declared In

NSByteOrder.h

NSSwapHostLongToBig

A utility for swapping the bytes of a number.

```
unsigned long NSSwapHostLongToBig (
    unsigned long x
);
```

Discussion

Converts the value in x , specified in the current endian format, to big-endian format and returns the resulting value. If it is necessary to swap the bytes, this function calls [NSSwapLong](#) (page 120) to perform the swap.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapBigLongToHost](#) (page 110)

[NSSwapHostLongToLittle](#) (page 116)

Declared In

NSByteOrder.h

NSSwapHostLongToLittle

A utility for swapping the bytes of a number.

```
unsigned long NSSwapHostLongToLittle (
    unsigned long x
);
```

Discussion

Converts the value in x , specified in the current endian format, to little-endian format and returns the resulting value. If it is necessary to swap the bytes, this function calls [NSSwapLong](#) (page 120) to perform the swap.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapLittleLongToHost](#) (page 119)

[NSSwapHostLongToBig](#) (page 115)

Declared In

NSByteOrder.h

NSSwapHostShortToBig

A utility for swapping the bytes of a number.

```
unsigned short NSSwapHostShortToBig (
    unsigned short x
);
```

Discussion

Converts the value in x , specified in the current endian format, to big-endian format and returns the resulting value. If it is necessary to swap the bytes, this function calls [NSSwapShort](#) (page 121) to perform the swap.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapBigShortToHost](#) (page 110)

[NSSwapHostShortToLittle](#) (page 117)

Declared In

NSByteOrder.h

NSSwapHostShortToLittle

A utility for swapping the bytes of a number.

```
unsigned short NSSwapHostShortToLittle (  
    unsigned short x  
);
```

Discussion

Converts the value in *x*, specified in the current endian format, to little-endian format and returns the resulting value. If it is necessary to swap the bytes, this function calls [NSSwapShort](#) (page 121) to perform the swap.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapLittleShortToHost](#) (page 120)

[NSSwapHostShortToBig](#) (page 116)

Related Sample Code

AudioBurn

Declared In

NSByteOrder.h

NSSwapInt

A utility for swapping the bytes of a number.

```
unsigned int NSSwapInt (  
    unsigned int inv  
);
```

Discussion

Swaps the bytes of *inv* and returns the resulting value. Bytes are swapped from each low-order position to the corresponding high-order position and vice versa. For example, if the bytes of *inv* are numbered from 1 to 4, this function swaps bytes 1 and 4, and bytes 2 and 3.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapShort](#) (page 121)

[NSSwapLong](#) (page 120)

[NSSwapLongLong](#) (page 121)

Declared In

NSByteOrder.h

NSSwapLittleDoubleToHost

A utility for swapping the bytes of a number.

```
double NSSwapLittleDoubleToHost (
    NSSwappedDouble x
);
```

Discussion

Converts the little-endian formatted value in *x* to the current endian format and returns the resulting value. If it is necessary to swap the bytes of *x*, this function calls [NSSwapDouble](#) (page 111) to perform the swap.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapHostDoubleToLittle](#) (page 112)

[NSSwapBigDoubleToHost](#) (page 108)

[NSConvertSwappedDoubleToHost](#) (page 37)

Declared In

NSByteOrder.h

NSSwapLittleFloatToHost

A utility for swapping the bytes of a number.

```
float NSSwapLittleFloatToHost (
    NSSwappedFloat x
);
```

Discussion

Converts the little-endian formatted value in *x* to the current endian format and returns the resulting value. If it is necessary to swap the bytes of *x*, this function calls [NSSwapFloat](#) (page 111) to perform the swap.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapHostFloatToLittle](#) (page 113)

[NSSwapBigFloatToHost](#) (page 109)

[NSConvertSwappedFloatToHost](#) (page 37)

Declared In

NSByteOrder.h

NSSwapLittleIntToHost

A utility for swapping the bytes of a number.

```
unsigned int NSSwapLittleIntToHost (
    unsigned int x
);
```

Discussion

Converts the little-endian formatted value in *x* to the current endian format and returns the resulting value. If it is necessary to swap the bytes, this function calls [NSSwapInt](#) (page 117) to perform the swap.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapHostIntToLittle](#) (page 114)

[NSSwapBigIntToHost](#) (page 109)

Related Sample Code

ZipBrowser

Declared In

NSByteOrder.h

NSSwapLittleLongLongToHost

A utility for swapping the bytes of a number.

```
unsigned long long NSSwapLittleLongLongToHost (
    unsigned long long x
);
```

Discussion

Converts the little-endian formatted value in *x* to the current endian format and returns the resulting value. If it is necessary to swap the bytes, this function calls [NSSwapLongLong](#) (page 121) to perform the swap.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapHostLongLongToLittle](#) (page 115)

[NSSwapBigLongLongToHost](#) (page 110)

Declared In

NSByteOrder.h

NSSwapLittleLongToHost

A utility for swapping the bytes of a number.

```
unsigned long NSSwapLittleLongToHost (
    unsigned long x
);
```

Discussion

Converts the little-endian formatted value in *x* to the current endian format and returns the resulting value. If it is necessary to swap the bytes of *x*, this function calls [NSSwapLong](#) (page 120) to perform the swap.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapHostLongToLittle](#) (page 116)

[NSSwapBigLongToHost](#) (page 110)

[NSSwapLong](#) (page 120)

Declared In

NSByteOrder.h

NSSwapLittleShortToHost

A utility for swapping the bytes of a number.

```
unsigned short NSSwapLittleShortToHost (
    unsigned short x
);
```

Discussion

Converts the little-endian formatted value in *x* to the current endian format and returns the resulting value. If it is necessary to swap the bytes of *x*, this function calls [NSSwapShort](#) (page 121) to perform the swap.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapHostShortToLittle](#) (page 117)

[NSSwapBigShortToHost](#) (page 110)

Related Sample Code

ZipBrowser

Declared In

NSByteOrder.h

NSSwapLong

A utility for swapping the bytes of a number.

```
unsigned long NSSwapLong (
    unsigned long inv
);
```

Discussion

Swaps the bytes of *inv* and returns the resulting value. Bytes are swapped from each low-order position to the corresponding high-order position and vice versa. For example, if the bytes of *inv* are numbered from 1 to 4, this function swaps bytes 1 and 4, and bytes 2 and 3.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapLongLong](#) (page 121)

[NSSwapInt](#) (page 117)

[NSSwapFloat](#) (page 111)

Declared In

NSByteOrder.h

NSSwapLongLong

A utility for swapping the bytes of a number.

```
unsigned long long NSSwapLongLong (
    unsigned long long inv
);
```

Discussion

Swaps the bytes of *inv* and returns the resulting value. Bytes are swapped from each low-order position to the corresponding high-order position and vice versa. For example, if the bytes of *inv* are numbered from 1 to 8, this function swaps bytes 1 and 8, bytes 2 and 7, bytes 3 and 6, and bytes 4 and 5.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapLong](#) (page 120)

[NSSwapDouble](#) (page 111)

Declared In

NSByteOrder.h

NSSwapShort

A utility for swapping the bytes of a number.

```
unsigned short NSSwapShort (
    unsigned short inv
);
```

Discussion

Swaps the low-order and high-order bytes of *inv* and returns the resulting value.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSwapInt](#) (page 117)

[NSSwapLong](#) (page 120)

Declared In

NSByteOrder.h

NSTemporaryDirectory

Returns the path of the temporary directory for the current user.

```
NSString * NSTemporaryDirectory (void);
```

Return Value

A string containing the path of the temporary directory for the current user. If no such directory is currently available, returns `nil`.

Discussion

For more information on file system utilities, see *Low-Level File Management Programming Topics*.

The temporary directory is determined by `confstr(3)` passing the `_CS_DARWIN_USER_TEMP_DIR` flag. The erase rules are whatever match that directory.

See the `NSFileManager` method `URLForDirectory:inDomain:appropriateForURL:create:error:` for an alternate (and more flexible) means of finding the correct temporary directory.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSearchPathForDirectoriesInDomains](#) (page 100)

[NSHomeDirectory](#) (page 64)

Related Sample Code

AbstractTree

Core Data HTML Store

Image Kit with Core Data

QTRecorder

SpotlightFortunes

Declared In

`NSPathUtilities.h`

NSUnionRange

Returns the union of the specified ranges.

```
NSRange NSUnionRange (
    NSRange range1,
    NSRange range2
);
```

Return Value

A range covering all indices in and between `range1` and `range2`. If one range is completely contained in the other, the returned range is equal to the larger range.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSIntersectionRange](#) (page 67)

Declared In

`NSRange.h`

NSUnionRect

Calculates the union of two rectangles.

```
NSRect NSUnionRect (
    NSRect aRect,
    NSRect bRect
);
```

Discussion

Returns the smallest rectangle that completely encloses both *aRect* and *bRect*. If one of the rectangles has 0 (or negative) width or height, a copy of the other rectangle is returned; but if both have 0 (or negative) width or height, the returned rectangle has its origin at (0.0, 0.0) and has 0 width and height.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSIntersectionRect](#) (page 68)

Related Sample Code

QuickLookSketch

Reducer

Sketch+Accessibility

Sketch-112

Worm

Declared In

NSGeometry.h

NSUserName

Returns the logon name of the current user.

```
NSString * NSUserName (void);
```

Return Value

The logon name of the current user.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSFullUserName](#) (page 59)

[NSHomeDirectory](#) (page 64)

[NSHomeDirectoryForUser](#) (page 65)

Declared In

NSPathUtilities.h

NSWidth

Returns the width of the specified rectangle.

```
CGFloat NSWidth (
    NSRect aRect
);
```

Return Value

The width of *aRect*.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSMaxX](#) (page 84)

[NSMaxY](#) (page 85)

[NSMidX](#) (page 85)

[NSMidY](#) (page 86)

[NSMinX](#) (page 86)

[NSMinY](#) (page 87)

[NSHeight](#) (page 63)

Related Sample Code

GLUT

iChatTheater

PhotoSearch

Rulers

TrackBall

Declared In

NSGeometry.h

NSZoneCalloc

Allocates memory in a zone.

```
void * NSZoneCalloc (
    NSZone *zone,
    NSUInteger numElems,
    NSUInteger byteSize
);
```

Discussion

Allocates enough memory from *zone* for *numElems* elements, each with a size *numBytes* bytes, and returns a pointer to the allocated memory. The memory is initialized with zeros. This function returns NULL if it was unable to allocate the requested memory.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSDefaultMallocZone](#) (page 53)

[NSRecycleZone](#) (page 97)

[NSZoneFree](#) (page 125)

[NSZoneMalloc](#) (page 126)

[NSZoneRealloc](#) (page 127)

Declared In

NSZone.h

NSZoneFree

Deallocates a block of memory in the specified zone.

```
void NSZoneFree (
    NSZone *zone,
    void *ptr
);
```

Discussion

Returns memory to the *zone* from which it was allocated. The standard C function `free` does the same, but spends time finding which zone the memory belongs to.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSRecycleZone](#) (page 97)

[NSZoneMalloc](#) (page 126)

[NSZoneCalloc](#) (page 124)

[NSZoneRealloc](#) (page 127)

Related Sample Code

AudioBurn

Declared In

NSZone.h

NSZoneFromPointer

Gets the zone for a given block of memory.

```
NSZone * NSZoneFromPointer (
    void *ptr
);
```

Return Value

The zone for the block of memory indicated by *pointer*, or NULL if the block was not allocated from a zone.

Discussion

pointer must be one that was returned by a prior call to an allocation function.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSZoneCalloc](#) (page 124)

[NSZoneMalloc](#) (page 126)

[NSZoneRealloc](#) (page 127)

Declared In

NSZone.h

NSZoneMalloc

Allocates memory in a zone.

```
void * NSZoneMalloc (
    NSZone *zone,
    NSUInteger size
);
```

Discussion

Allocates *size* bytes in *zone* and returns a pointer to the allocated memory. This function returns NULL if it was unable to allocate the requested memory.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSDefaultMallocZone](#) (page 53)

[NSRecycleZone](#) (page 97)

[NSZoneFree](#) (page 125)

[NSZoneCalloc](#) (page 124)

[NSZoneRealloc](#) (page 127)

Related Sample Code

AudioBurn

Declared In

NSZone.h

NSZoneName

Returns the name of the specified zone.

```
NSString * NSZoneName (
    NSZone *zone
);
```

Return Value

A string containing the name associated with *zone*. If *zone* is *nil*, the default zone is used. If no name is associated with *zone*, the returned string is empty.

Availability

Available in Mac OS X v10.0 and later.

See Also

[NSSetZoneName](#) (page 101)

Declared In

NSZone.h

NSZoneRealloc

Allocates memory in a zone.

```
void * NSZoneRealloc (
    NSZone *zone,
    void *ptr,
    NSUInteger size
);
```

Discussion

Changes the size of the block of memory pointed to by *ptr* to *size* bytes. It may allocate new memory to replace the old, in which case it moves the contents of the old memory block to the new block, up to a maximum of *size* bytes. *ptr* may be NULL. This function returns NULL if it was unable to allocate the requested memory.

Availability

Available in Mac OS X v10.0 and later.

See Also[NSDefaultMallocZone](#) (page 53)[NSRecycleZone](#) (page 97)[NSZoneFree](#) (page 125)[NSZoneCalloc](#) (page 124)[NSZoneMalloc](#) (page 126)**Declared In**

NSZone.h

NS_DURING

Marks the start of the exception-handling domain.

NS_DURING

Discussion

The NS_DURING macro marks the start of the exception-handling domain for a section of code. (The [NS_HANDLER](#) (page 128) macro marks the end of the domain.) Within the exception-handling domain you can raise an exception, giving the local exception handler (or lower exception handlers) a chance to handle it.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

EnhancedAudioBurn

XMLBrowser

Declared In

NSException.h

NS_ENDHANDLER

Marks the end of the local event handler.

NS_ENDHANDLER

Discussion

The NS_ENDHANDLER marks the end of a section of code that is a local exception handler. (The NS_HANDLER (page 128) macros marks the beginning of this section.) If an exception is raised in the exception handling domain marked off by the NS_DURING (page 127) and NS_HANDLER (page 128), the local exception handler (if specified) is given a chance to handle the exception.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

EnhancedAudioBurn

XMLBrowser

Declared In

NSException.h

NS_HANDLER

Marks the end of the exception-handling domain and the start of the local exception handler.

NS_HANDLER

Discussion

The NS_HANDLER macro marks end of a section of code that is an exception-handling domain while at the same time marking the beginning of a section of code that is a local exception handler for that domain. (The NS_DURING (page 127) macro marks the beginning of the exception-handling domain; the NS_ENDHANDLER (page 128) marks the end of the local exception handler.) If an exception is raised in the exception-handling domain, the local exception handler is first given the chance to handle the exception before lower-level handlers are given a chance.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

EnhancedAudioBurn

XMLBrowser

Declared In

NSException.h

NS_VALUEReturn

Permits program control to exit from an exception-handling domain with a value of a specified type.


```
NS_VALUEReturn(val, type)
```

Parameters

val

A value to preserve beyond the exception-handling domain.

type

The type of the value specified in *val*.

Discussion

The `NS_VALUEReturn` macro returns program control to the caller out of the exception-handling domain—that is, a section of code between the `NS_DURING` (page 127) and `NS_HANDLER` (page 128) macros that might raise an exception. The specified value (of the specified type) is returned to the caller. The standard `return` statement does not work as expected in the exception-handling domain.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSException.h`

NS_VOIDRETURN

Permits program control to exit from an exception-handling domain.

```
NS_VOIDRETURN
```

Discussion

The `NS_VOIDRETURN` macro returns program control to the caller out of the exception-handling domain—that is, a section of code between the `NS_DURING` (page 127) and `NS_HANDLER` (page 128) macros that might raise an exception. The standard `return` statement does not work as expected in the exception-handling domain.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSException.h`

Document Revision History

This table describes the changes to *Foundation Functions Reference*.

Date	Notes
2010-03-24	Corrected the description of <code>NSRectToCGRect</code> to show the actual implementation.
2009-10-11	Clarified that a directory returned by <code>NSSearchPathForDirectoriesInDomains()</code> may not exist.
2009-10-19	Added usage warning to <code>NSCopyObject()</code> .
2009-05-26	Corrected typographical errors.
2008-10-15	Corrected discussions of <code>NSMapInsert</code> and <code>NSTemporaryDirectory</code> ; clarified behavior of <code>NSPointInRect</code> .
2008-09-09	Updated descriptions of the <code>NSLocalizedString</code> , <code>NSLocalizedStringFromTable</code> , <code>NSLocalizedStringFromTableInBundle</code> , and <code>NSLocalizedStringWithDefaultValue</code> for Mac OS X v10.5.
2008-07-11	Corrected descriptions of the functions <code>NSRoundUpToMultipleOfPageSize</code> and <code>NSRoundDownToMultipleOfPageSize</code> .
2008-02-08	Corrected minor typographical errors.
2007-07-23	Updated to include API introduced in Mac OS X v10.5.
2006-09-05	Corrected broken link.
2006-06-28	Removed references to retired document.
2006-05-23	First publication of this content as a separate document.

REVISION HISTORY

Document Revision History