
NSDecimalNumber Class Reference

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





Apple Inc.
© 2007 Apple Inc.
All rights reserved.

No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, mechanical, electronic, photocopying, recording, or otherwise, without prior written permission of Apple Inc., with the following exceptions: Any person is hereby authorized to store documentation on a single computer for personal use only and to print copies of documentation for personal use provided that the documentation contains Apple's copyright notice.

The Apple logo is a trademark of Apple Inc.

Use of the "keyboard" Apple logo (Option-Shift-K) for commercial purposes without the prior written consent of Apple may constitute trademark infringement and unfair competition in violation of federal and state laws.

No licenses, express or implied, are granted with respect to any of the technology described in this document. Apple retains all intellectual property rights associated with the technology described in this document. This document is intended to assist application developers to develop applications only for Apple-labeled computers.

Every effort has been made to ensure that the information in this document is accurate. Apple is not responsible for typographical errors.

Apple Inc.
1 Infinite Loop
Cupertino, CA 95014
408-996-1010

Apple, the Apple logo, Cocoa, Mac, and Objective-C are trademarks of Apple Inc., registered in the United States and other countries.

iPhone and Numbers are trademarks of Apple Inc.

Times is a registered trademark of Heidelberger Druckmaschinen AG, available from Linotype Library GmbH.

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

NSDecimalNumber Class Reference 5

Overview	5
Tasks	5
Creating a Decimal Number	5
Initializing a Decimal Number	6
Performing Arithmetic	6
Rounding Off	7
Accessing the Value	7
Managing Behavior	7
Comparing Decimal Numbers	8
Getting Maximum and Minimum Possible Values	8
Class Methods	8
decimalNumberWithDecimal:	8
decimalNumberWithMantissa:exponent:isNegative:	8
decimalNumberWithString:	9
decimalNumberWithString:locale:	10
defaultBehavior	10
maximumDecimalNumber	11
minimumDecimalNumber	11
notANumber	12
one	12
setDefaultBehavior:	12
zero	13
Instance Methods	13
compare:	13
decimalNumberByAdding:	14
decimalNumberByAdding:withBehavior:	14
decimalNumberByDividingBy:	15
decimalNumberByDividingBy:withBehavior:	15
decimalNumberByMultiplyingBy:	15
decimalNumberByMultiplyingBy:withBehavior:	16
decimalNumberByMultiplyingByPowerOf10:	16
decimalNumberByMultiplyingByPowerOf10:withBehavior:	17
decimalNumberByRaisingToPower:	17
decimalNumberByRaisingToPower:withBehavior:	17
decimalNumberByRoundingAccordingToBehavior:	18
decimalNumberBySubtracting:	18
decimalNumberBySubtracting:withBehavior:	19
decimalValue	19
descriptionWithLocale:	19
doubleValue	20

- initWithDecimal: 20
- initWithMantissa:exponent:isNegative: 20
- initWithString: 21
- initWithString:locale: 22
- objCType 22
- Constants 22
 - NSDecimalNumber Exception Names 22

Document Revision History 25

Index 27

NSDecimalNumber Class Reference

Inherits from	NSNumber : NSValue : NSObject
Conforms to	NSCoding (NSValue) NSCopying (NSValue) NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.0 and later.
Companion guide	Number and Value Programming Topics for Cocoa
Declared in	NSDecimalNumber.h
Related sample code	BindingsJoystick Calculator Core Data HTML Store

Overview

`NSDecimalNumber`, an immutable subclass of `NSNumber`, provides an object-oriented wrapper for doing base-10 arithmetic. An instance can represent any number that can be expressed as $\text{mantissa} \times 10^{\text{exponent}}$ where `mantissa` is a decimal integer up to 38 digits long, and `exponent` is an integer from -128 through 127.

Tasks

Creating a Decimal Number

- + [decimalNumberWithDecimal:](#) (page 8)
Creates and returns an `NSDecimalNumber` object equivalent to a given `NSDecimal` structure.
- + [decimalNumberWithMantissa:exponent:isNegative:](#) (page 8)
Creates and returns an `NSDecimalNumber` object equivalent to the number specified by the arguments.
- + [decimalNumberWithString:](#) (page 9)
Creates and returns an `NSDecimalNumber` object whose value is equivalent to that in a given numeric string.

- + [decimalNumberWithString:locale:](#) (page 10)
Creates and returns an `NSDecimalNumber` object whose value is equivalent to that in a given numeric string, interpreted using a given locale.
- + [one](#) (page 12)
Returns an `NSDecimalNumber` object equivalent to the number 1.0.
- + [zero](#) (page 13)
Returns an `NSDecimalNumber` object equivalent to the number 0.0.
- + [notANumber](#) (page 12)
Returns an `NSDecimalNumber` object that specifies no number.

Initializing a Decimal Number

- [initWithDecimal:](#) (page 20)
Returns an `NSDecimalNumber` object initialized to represent a given decimal.
- [initWithMantissa:exponent:isNegative:](#) (page 20)
Returns an `NSDecimalNumber` object initialized using the given mantissa, exponent, and sign.
- [initWithString:](#) (page 21)
Returns an `NSDecimalNumber` object initialized so that its value is equivalent to that in a given numeric string.
- [initWithString:locale:](#) (page 22)
Returns an `NSDecimalNumber` object initialized so that its value is equivalent to that in a given numeric string, interpreted using a given locale.

Performing Arithmetic

- [decimalNumberByAdding:](#) (page 14)
Returns a new `NSDecimalNumber` object whose value is the sum of the receiver and another given `NSDecimalNumber` object.
- [decimalNumberBySubtracting:](#) (page 18)
Returns a new `NSDecimalNumber` object whose value is that of another given `NSDecimalNumber` object subtracted from the value of the receiver.
- [decimalNumberByMultiplyingBy:](#) (page 15)
Returns a new `NSDecimalNumber` object whose value is the value of the receiver multiplied by that of another given `NSDecimalNumber` object.
- [decimalNumberByDividingBy:](#) (page 15)
Returns a new `NSDecimalNumber` object whose value is the value of the receiver divided by that of another given `NSDecimalNumber` object.
- [decimalNumberByRaisingToPower:](#) (page 17)
Returns a new `NSDecimalNumber` object whose value is the value of the receiver raised to a given power.
- [decimalNumberByMultiplyingByPowerOf10:](#) (page 16)
Multiplies the receiver by 10^{power} and returns the product, a newly created `NSDecimalNumber` object.

- [decimalNumberByAdding:withBehavior:](#) (page 14)
Adds *decimalNumber* to the receiver and returns the sum, a newly created NSDecimalNumber object.
- [decimalNumberBySubtracting:withBehavior:](#) (page 19)
Subtracts *decimalNumber* from the receiver and returns the difference, a newly created NSDecimalNumber object.
- [decimalNumberByMultiplyingBy:withBehavior:](#) (page 16)
Multiplies the receiver by *decimalNumber* and returns the product, a newly created NSDecimalNumber object.
- [decimalNumberByDividingBy:withBehavior:](#) (page 15)
Divides the receiver by *decimalNumber* and returns the quotient, a newly created NSDecimalNumber object.
- [decimalNumberByRaisingToPower:withBehavior:](#) (page 17)
Raises the receiver to *power* and returns the result, a newly created NSDecimalNumber object.
- [decimalNumberByMultiplyingByPowerOf10:withBehavior:](#) (page 17)
Multiplies the receiver by $10^{\textit{power}}$ and returns the product, a newly created NSDecimalNumber object.

Rounding Off

- [decimalNumberByRoundingAccordingToBehavior:](#) (page 18)
Rounds the receiver off in the way specified by *behavior* and returns the result, a newly created NSDecimalNumber object.

Accessing the Value

- [decimalValue](#) (page 19)
Returns the receiver's value, expressed as an NSDecimal structure.
- [doubleValue](#) (page 20)
Returns the approximate value of the receiver as a double.
- [descriptionWithLocale:](#) (page 19)
Returns a string, specified according to a given locale, that represents the contents of the receiver.
- [objCType](#) (page 22)
Returns a C string containing the Objective-C type of the data contained in the receiver, which for an NSDecimalNumber object is always "d" (for double).

Managing Behavior

- + [defaultBehavior](#) (page 10)
Returns the way arithmetic methods, like [decimalNumberByAdding:](#) (page 14), round off and handle error conditions.
- + [setDefaultBehavior:](#) (page 12)
Specifies the way that arithmetic methods, like [decimalNumberByAdding:](#) (page 14), round off and handle error conditions.

Comparing Decimal Numbers

- [compare:](#) (page 13)

Returns an `NSComparisonResult` value that indicates the numerical ordering of the receiver and another given `NSDecimalNumber` object.

Getting Maximum and Minimum Possible Values

+ [maximumDecimalNumber](#) (page 11)

Returns the largest possible value of an `NSDecimalNumber` object.

+ [minimumDecimalNumber](#) (page 11)

Returns the smallest possible value of an `NSDecimalNumber` object.

Class Methods

decimalNumberWithDecimal:

Creates and returns an `NSDecimalNumber` object equivalent to a given `NSDecimal` structure.

```
+ (NSDecimalNumber *)decimalNumberWithDecimal:(NSDecimal)decimal
```

Parameters

decimal

An `NSDecimal` structure that specifies the value for the new decimal number object.

Return Value

An `NSDecimalNumber` object equivalent to *decimal*.

Discussion

You can initialize *decimal* programmatically or generate it using the `NSScanner` method, `scanDecimal:`

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSDecimalNumber.h`

decimalNumberWithMantissa:exponent:isNegative:

Creates and returns an `NSDecimalNumber` object equivalent to the number specified by the arguments.

```
+ (NSDecimalNumber *)decimalNumberWithMantissa:(unsigned long long)mantissa
    exponent:(short)exponent isNegative:(BOOL)isNegative
```

Parameters

mantissa

The mantissa for the new decimal number object.

exponent

The exponent for the new decimal number object.

isNegative

A Boolean value that specifies whether the sign of the number is negative.

Discussion

The arguments express a number in a kind of scientific notation that requires the mantissa to be an integer. So, for example, if the number to be represented is -12.345 , it is expressed as 12345×10^{-3} —*mantissa* is 12345; *exponent* is -3 ; and *isNegative* is YES, as illustrated by the following example.

```
NSDecimalNumber *number = [NSDecimalNumber decimalNumberWithMantissa:12345
                               exponent:-3
                               isNegative:YES];
```

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSDecimalNumber.h

decimalNumberWithString:

Creates and returns an NSDecimalNumber object whose value is equivalent to that in a given numeric string.

```
+ (NSDecimalNumber *)decimalNumberWithString:(NSString *)numericString
```

Parameters

numericString

A numeric string.

Besides digits, *numericString* can include an initial “+” or “-”; a single “E” or “e”; to indicate the exponent of a number in scientific notation; and a single NSDecimalSeparator to divide the fractional from the integral part of the number.

Return Value

An NSDecimalNumber object whose value is equivalent to *numericString*.

Discussion

Whether the NSDecimalSeparator is a period (as is used, for example, in the United States) or a comma (as is used, for example, in France) depends on the default locale.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [decimalNumberWithString:locale:](#) (page 10)

Related Sample Code

Calculator

Core Data HTML Store

Declared In

NSDecimalNumber.h

decimalNumberWithString:locale:

Creates and returns an `NSDecimalNumber` object whose value is equivalent to that in a given numeric string, interpreted using a given locale.

```
+ (NSDecimalNumber *)decimalNumberWithString:(NSString *)numericString
    locale:(NSDictionary *)locale
```

Parameters

numericString

A numeric string.

Besides digits, *numericString* can include an initial “+” or “-”; a single “E” or “e” to indicate the exponent of a number in scientific notation; and a single `NSDecimalSeparator` to divide the fractional from the integral part of the number.

locale

A dictionary that defines the locale (specifically the `NSDecimalSeparator`) to use to interpret the number in *numericString*.

Return Value

An `NSDecimalNumber` object whose value is equivalent to *numericString*.

Discussion

The *locale* parameter determines whether the `NSDecimalSeparator` is a period (as is used, for example, in the United States) or a comma (as is used, for example, in France).

The following strings show examples of acceptable values for *numericString*:

“2500.6” (or “2500,6”, depending on locale)

“-2500.6” (or “-2500,6”)

“-2.5006e3” (or “-2,5006e3”)

“-2.5006E3” (or “-2,5006E3”)

The following strings are unacceptable:

“2,500.6”

“2500 3/5”

“2.5006x10e3”

“two thousand five hundred and six tenths”

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [decimalNumberWithString:](#) (page 9)

Declared In

`NSDecimalNumber.h`

defaultBehavior

Returns the way arithmetic methods, like [decimalNumberByAdding:](#) (page 14), round off and handle error conditions.

```
+ (id < NSDecimalNumberBehaviors >)defaultBehavior
```

Discussion

By default, the arithmetic methods use the `NSRoundPlain` behavior; that is, the methods round to the closest possible return value. The methods assume your need for precision does not exceed 38 significant digits and raise exceptions when they try to divide by 0 or produce a number too big or too small to be represented.

If this default behavior doesn't suit your application, you should use methods that let you specify the behavior, like `decimalNumberByAdding:withBehavior:` (page 14). If you find yourself using a particular behavior consistently, you can specify a different default behavior with `setDefaultBehavior:` (page 12).

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSDecimalNumber.h`

maximumDecimalNumber

Returns the largest possible value of an `NSDecimalNumber` object.

```
+ (NSDecimalNumber *)maximumDecimalNumber
```

Return Value

The largest possible value of an `NSDecimalNumber` object.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [minimumDecimalNumber](#) (page 11)

Declared In

`NSDecimalNumber.h`

minimumDecimalNumber

Returns the smallest possible value of an `NSDecimalNumber` object.

```
+ (NSDecimalNumber *)minimumDecimalNumber
```

Return Value

The smallest possible value of an `NSDecimalNumber` object.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [maximumDecimalNumber](#) (page 11)

Declared In

`NSDecimalNumber.h`

notANumber

Returns an `NSDecimalNumber` object that specifies no number.

```
+ (NSDecimalNumber *)notANumber
```

Return Value

An `NSDecimalNumber` object that specifies no number.

Discussion

Any arithmetic method receiving `notANumber` as an argument returns `notANumber`.

This value can be a useful way of handling non-numeric data in an input file. This method can also be a useful response to calculation errors. For more information on calculation errors, see the `exceptionDuringOperation:error:leftOperand:rightOperand:` method description in the `NSDecimalNumberBehaviors` protocol specification.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

Calculator

Declared In

`NSDecimalNumber.h`

one

Returns an `NSDecimalNumber` object equivalent to the number 1.0.

```
+ (NSDecimalNumber *)one
```

Return Value

An `NSDecimalNumber` object equivalent to the number 1.0.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [zero](#) (page 13)

Declared In

`NSDecimalNumber.h`

setDefaultBehavior:

Specifies the way that arithmetic methods, like [decimalNumberByAdding:](#) (page 14), round off and handle error conditions.

```
+ (void)setDefaultBehavior:(id < NSDecimalNumberBehaviors >)behavior
```

Discussion

behavior must conform to the `NSDecimalNumberBehaviors` protocol.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSDecimalNumber.h

zero

Returns an NSDecimalNumber object equivalent to the number 0.0.

```
+ (NSDecimalNumber *)zero
```

Return Value

An NSDecimalNumber object equivalent to the number 0.0.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [one](#) (page 12)

Related Sample Code

BindingsJoystick

Calculator

Declared In

NSDecimalNumber.h

Instance Methods

compare:

Returns an NSComparisonResult value that indicates the numerical ordering of the receiver and another given NSDecimalNumber object.

```
- (NSComparisonResult)compare:(NSNumber *)decimalNumber
```

Parameters

decimalNumber

The number with which to compare the receiver.

This value must not be nil. If this value is nil, the behavior is undefined and may change in future versions of Mac OS X.

Return Value

NSOrderedAscending if the value of *decimalNumber* is greater than the receiver; NSOrderedSame if they're equal; and NSOrderedDescending if the value of *decimalNumber* is less than the receiver.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSDecimalNumber.h

decimalNumberByAdding:

Returns a new `NSDecimalNumber` object whose value is the sum of the receiver and another given `NSDecimalNumber` object.

```
- (NSDecimalNumber *)decimalNumberByAdding:(NSDecimalNumber *)decimalNumber
```

Parameters*decimalNumber*

The number to add to the receiver.

Return Value

A new `NSDecimalNumber` object whose value is the sum of the receiver and *decimalNumber*.

Discussion

This method uses the default behavior when handling calculation errors and rounding.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [decimalNumberByAdding:withBehavior:](#) (page 14)

+ [defaultBehavior](#) (page 10)

Related Sample Code

Calculator

Declared In

NSDecimalNumber.h

decimalNumberByAdding:withBehavior:

Adds *decimalNumber* to the receiver and returns the sum, a newly created `NSDecimalNumber` object.

```
- (NSDecimalNumber *)decimalNumberByAdding:(NSDecimalNumber *)decimalNumber
    withBehavior:(id < NSDecimalNumberBehaviors >)behavior
```

Discussion

behavior specifies the handling of calculation errors and rounding.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSDecimalNumber.h

decimalNumberByDividingBy:

Returns a new `NSDecimalNumber` object whose value is the value of the receiver divided by that of another given `NSDecimalNumber` object.

```
- (NSDecimalNumber *)decimalNumberByDividingBy:(NSDecimalNumber *)decimalNumber
```

Parameters

decimalNumber

The number by which to divide the receiver.

Return Value

A new `NSDecimalNumber` object whose value is the value of the receiver divided by *decimalNumber*.

Discussion

This method uses the default behavior when handling calculation errors and rounding.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [decimalNumberByDividingBy:withBehavior:](#) (page 15)
- + [defaultBehavior](#) (page 10)

Related Sample Code

Calculator

Declared In

`NSDecimalNumber.h`

decimalNumberByDividingBy:withBehavior:

Divides the receiver by *decimalNumber* and returns the quotient, a newly created `NSDecimalNumber` object.

```
- (NSDecimalNumber *)decimalNumberByDividingBy:(NSDecimalNumber *)decimalNumber
    withBehavior:(id < NSDecimalNumberBehaviors >)behavior
```

Discussion

behavior specifies the handling of calculation errors and rounding.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSDecimalNumber.h`

decimalNumberByMultiplyingBy:

Returns a new `NSDecimalNumber` object whose value is the value of the receiver multiplied by that of another given `NSDecimalNumber` object.

```
- (NSDecimalNumber *)decimalNumberByMultiplyingBy:(NSDecimalNumber *)decimalNumber
```

Parameters*decimalNumber*

The number by which to multiply the receiver.

Return ValueA new `NSDecimalNumber` object whose value is *decimalNumber* multiplied by the receiver.**Discussion**

This method uses the default behavior when handling calculation errors and when rounding.

Availability

Available in Mac OS X v10.0 and later.

See Also- [decimalNumberByMultiplyingBy:withBehavior:](#) (page 16)+ [defaultBehavior](#) (page 10)**Related Sample Code**

Calculator

Declared In`NSDecimalNumber.h`**decimalNumberByMultiplyingBy:withBehavior:**Multiplies the receiver by *decimalNumber* and returns the product, a newly created `NSDecimalNumber` object.

```
- (NSDecimalNumber *)decimalNumberByMultiplyingBy:(NSDecimalNumber *)decimalNumber
    withBehavior:(id < NSDecimalNumberBehaviors >)behavior
```

Discussion*behavior* specifies the handling of calculation errors and rounding.**Availability**

Available in Mac OS X v10.0 and later.

Declared In`NSDecimalNumber.h`**decimalNumberByMultiplyingByPowerOf10:**Multiplies the receiver by $10^{\textit{power}}$ and returns the product, a newly created `NSDecimalNumber` object.

```
- (NSDecimalNumber *)decimalNumberByMultiplyingByPowerOf10:(short)power
```

Discussion

This method uses the default behavior when handling calculation errors and when rounding.

Availability

Available in Mac OS X v10.0 and later.

See Also- [decimalNumberByMultiplyingByPowerOf10:withBehavior:](#) (page 17)

+ [defaultBehavior](#) (page 10)

Declared In

NSDecimalNumber.h

decimalNumberByMultiplyingByPowerOf10:withBehavior:

Multiplies the receiver by 10^{power} and returns the product, a newly created `NSDecimalNumber` object.

```
- (NSDecimalNumber *)decimalNumberByMultiplyingByPowerOf10:(short)power
    withBehavior:(id < NSDecimalNumberBehaviors >)behavior
```

Discussion

behavior specifies the handling of calculation errors and rounding.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSDecimalNumber.h

decimalNumberByRaisingToPower:

Returns a new `NSDecimalNumber` object whose value is the value of the receiver raised to a given power.

```
- (NSDecimalNumber *)decimalNumberByRaisingToPower:(NSUInteger)power
```

Parameters

power

The power to which to raise the receiver.

Return Value

A new `NSDecimalNumber` object whose value is the value of the receiver raised to the power *power*.

Discussion

This method uses the default behavior when handling calculation errors and when rounding.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [decimalNumberByRaisingToPower:withBehavior:](#) (page 17)

+ [defaultBehavior](#) (page 10)

Declared In

NSDecimalNumber.h

decimalNumberByRaisingToPower:withBehavior:

Raises the receiver to *power* and returns the result, a newly created `NSDecimalNumber` object.

```
- (NSDecimalNumber *)decimalNumberByRaisingToPower:(NSUInteger)power withBehavior:(id < NSDecimalNumberBehaviors >)behavior
```

Discussion

behavior specifies the handling of calculation errors and rounding.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSDecimalNumber.h

decimalNumberByRoundingAccordingToBehavior:

Rounds the receiver off in the way specified by *behavior* and returns the result, a newly created NSDecimalNumber object.

```
- (NSDecimalNumber *)decimalNumberByRoundingAccordingToBehavior:(id < NSDecimalNumberBehaviors >)behavior
```

Discussion

For a description of the different ways of rounding, see the `roundingMode` method in the NSDecimalNumberBehaviors protocol specification.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSDecimalNumber.h

decimalNumberBySubtracting:

Returns a new NSDecimalNumber object whose value is that of another given NSDecimalNumber object subtracted from the value of the receiver.

```
- (NSDecimalNumber *)decimalNumberBySubtracting:(NSDecimalNumber *)decimalNumber
```

Parameters

decimalNumber

The number to subtract from the receiver.

Return Value

A new NSDecimalNumber object whose value is *decimalNumber* subtracted from the receiver.

Discussion

This method uses the default behavior when handling calculation errors and when rounding.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [decimalNumberBySubtracting:withBehavior:](#) (page 19)

+ [defaultBehavior](#) (page 10)

Related Sample Code

Calculator

Declared In

NSDecimalNumber.h

decimalNumberBySubtracting:withBehavior:

Subtracts *decimalNumber* from the receiver and returns the difference, a newly created NSDecimalNumber object.

```
- (NSDecimalNumber *)decimalNumberBySubtracting:(NSDecimalNumber *)decimalNumber
    withBehavior:(id < NSDecimalNumberBehaviors >)behavior
```

Discussion

behavior specifies the handling of calculation errors and rounding.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSDecimalNumber.h

decimalValue

Returns the receiver's value, expressed as an NSDecimal structure.

```
- (NSDecimal)decimalValue
```

Return Value

The receiver's value, expressed as an NSDecimal structure.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSDecimalNumber.h

descriptionWithLocale:

Returns a string, specified according to a given locale, that represents the contents of the receiver.

```
- (NSString *)descriptionWithLocale:(NSDictionary *)locale
```

Parameters

locale

A dictionary that defines the locale (specifically the NSDecimalSeparator) to use to generate the returned string.

Return Value

A string that represents the contents of the receiver, according to *locale*.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSDecimalNumber.h

doubleValue

Returns the approximate value of the receiver as a double.

- (double)doubleValue

Return Value

The approximate value of the receiver as a double.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSDecimalNumber.h

initWithDecimal:

Returns an NSDecimalNumber object initialized to represent a given decimal.

- (id)initWithDecimal:(NSDecimal)decimal

Parameters

decimal

The value of the new object.

Return Value

An NSDecimalNumber object initialized to represent *decimal*.

Discussion

This method is the designated initializer for NSDecimalNumber.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSDecimalNumber.h

initWithMantissa:exponent:isNegative:

Returns an NSDecimalNumber object initialized using the given mantissa, exponent, and sign.

- (id)initWithMantissa:(unsigned long long)mantissa exponent:(short)exponent
isNegative:(BOOL)flag

Parameters*mantissa*

The mantissa for the new decimal number object.

exponent

The exponent for the new decimal number object.

flag

A Boolean value that specifies whether the sign of the number is negative.

Return ValueAn `NSDecimalNumber` object initialized using the given mantissa, exponent, and sign.**Discussion**

The arguments express a number in a type of scientific notation that requires the mantissa to be an integer. So, for example, if the number to be represented is 1.23, it is expressed as 123×10^{-2} —*mantissa* is 123; *exponent* is -2 ; and *isNegative*, which refers to the sign of the mantissa, is NO.

Availability

Available in Mac OS X v10.0 and later.

See Also+ [decimalNumberWithMantissa:exponent:isNegative:](#) (page 8)**Declared In**`NSDecimalNumber.h`**initWithString:**

Returns an `NSDecimalNumber` object initialized so that its value is equivalent to that in a given numeric string.

```
- (id)initWithString:(NSString *)numericString
```

Parameters*numericString*

A numeric string.

Besides digits, *numericString* can include an initial “+” or “-”; a single “E” or “e” to indicate the exponent of a number in scientific notation; and a single `NSDecimalSeparator` to divide the fractional from the integral part of the number. For a listing of acceptable and unacceptable strings, see the class method [decimalNumberWithString:locale:](#) (page 10).

Return ValueAn `NSDecimalNumber` object initialized so that its value is equivalent to that in *numericString*.**Availability**

Available in Mac OS X v10.0 and later.

Declared In`NSDecimalNumber.h`

initWithString:locale:

Returns an `NSDecimalNumber` object initialized so that its value is equivalent to that in a given numeric string, interpreted using a given locale.

```
- (id)initWithString:(NSString *)numericString locale:(NSDictionary *)locale
```

Parameters

numericString

A numeric string.

Besides digits, *numericString* can include an initial “+” or “-”; a single “E” or “e” to indicate the exponent of a number in scientific notation; and a single `NSDecimalSeparator` to divide the fractional from the integral part of the number.

locale

A dictionary that defines the locale (specifically the `NSDecimalSeparator`) to use to interpret the number in *numericString*.

Return Value

An `NSDecimalNumber` object initialized so that its value is equivalent to that in *numericString*, interpreted using *locale*.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [decimalNumberWithString:locale:](#) (page 10)

Declared In

`NSDecimalNumber.h`

objCType

Returns a C string containing the Objective-C type of the data contained in the receiver, which for an `NSDecimalNumber` object is always “d” (for double).

```
- (const char *)objCType
```

Availability

Available in Mac OS X v10.0 and later.

Declared In

`NSDecimalNumber.h`

Constants

NSDecimalNumber Exception Names

Names of the various exceptions raised by `NSDecimalNumber` to indicate computational errors.

```
extern NSString *NSDecimalNumberExactnessException;  
extern NSString *NSDecimalNumberOverflowException;  
extern NSString *NSDecimalNumberUnderflowException;  
extern NSString *NSDecimalNumberDivideByZeroException;
```

Constants

NSDecimalNumberExactnessException

The name of the exception raised if there is an exactness error.

Available in Mac OS X v10.0 and later.

Declared in NSDecimalNumber.h.

NSDecimalNumberOverflowException

The name of the exception raised on overflow.

Available in Mac OS X v10.0 and later.

Declared in NSDecimalNumber.h.

NSDecimalNumberUnderflowException

The name of the exception raised on underflow.

Available in Mac OS X v10.0 and later.

Declared in NSDecimalNumber.h.

NSDecimalNumberDivideByZeroException

The name of the exception raised on divide by zero.

Available in Mac OS X v10.0 and later.

Declared in NSDecimalNumber.h.

Declared In

NSDecimalNumber.h

Document Revision History

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

Date	Notes
2007-10-31	Updated the description of the compare: method.
2006-05-23	Incorporated constant definitions from Foundation Constants.
	First publication of this content as a separate document.

REVISION HISTORY

Document Revision History

Index

C

compare: [instance method 13](#)

D

decimalNumberByAdding: [instance method 14](#)

decimalNumberByAdding:withBehavior: [instance method 14](#)

decimalNumberByDividingBy: [instance method 15](#)

decimalNumberByDividingBy:withBehavior: [instance method 15](#)

decimalNumberByMultiplyingBy: [instance method 15](#)

decimalNumberByMultiplyingBy:withBehavior: [instance method 16](#)

decimalNumberByMultiplyingByPowerOf10: [instance method 16](#)

decimalNumberByMultiplyingByPowerOf10:withBehavior: [instance method 17](#)

decimalNumberByRaisingToPower: [instance method 17](#)

decimalNumberByRaisingToPower:withBehavior: [instance method 17](#)

decimalNumberByRoundingAccordingToBehavior: [instance method 18](#)

decimalNumberBySubtracting: [instance method 18](#)

decimalNumberBySubtracting:withBehavior: [instance method 19](#)

decimalNumberWithDecimal: [class method 8](#)

decimalNumberWithMantissa:exponent:isNegative: [class method 8](#)

decimalNumberWithString: [class method 9](#)

decimalNumberWithString:locale: [class method 10](#)

doubleValue [instance method 19](#)

defaultBehavior [class method 10](#)

descriptionWithLocale: [instance method 19](#)

doubleValue [instance method 20](#)

I

initWithDecimal: [instance method 20](#)

initWithMantissa:exponent:isNegative: [instance method 20](#)

initWithString: [instance method 21](#)

initWithString:locale: [instance method 22](#)

M

maximumDecimalNumber [class method 11](#)

minimumDecimalNumber [class method 11](#)

N

notANumber [class method 12](#)

NSDecimalNumberExceptionNames [22](#)

NSDecimalNumberDivideByZeroException [constant 23](#)

NSDecimalNumberExactnessException [constant 23](#)

NSDecimalNumberOverflowException [constant 23](#)

NSDecimalNumberUnderflowException [constant 23](#)

O

objCType [instance method 22](#)

one [class method 12](#)

S

setDefaultBehavior: [class method 12](#)

Z

zero class method [13](#)