

---

# NSDecimalNumberBehaviors Protocol Reference

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



2006-05-23



Apple Inc.  
© 2006 Apple Computer, Inc.  
All rights reserved.

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

The Apple logo is a trademark of Apple Inc.

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

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

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

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

Apple, the Apple logo, Cocoa, and Mac 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

---

## **NSDecimalNumberBehaviors Protocol Reference 5**

---

Overview 5

Tasks 5

    Rounding 5

    Handling Errors 5

Instance Methods 6

    exceptionDuringOperation:error:leftOperand:rightOperand: 6

    roundingMode 6

    scale 7

Constants 7

    NSRoundingMode 7

    NSCalculationError 9

---

## **Document Revision History 11**

---

## **Index 13**

---



# NSDecimalNumberBehaviors Protocol Reference

---

<b>Adopted by</b>	NSDecimalNumberHandler
<b>Framework</b>	/System/Library/Frameworks/Foundation.framework
<b>Availability</b>	Available in Mac OS X v10.0 and later.
<b>Companion guide</b>	Number and Value Programming Topics for Cocoa
<b>Declared in</b>	NSDecimal.h NSDecimalNumber.h

## Overview

The `NSDecimalBehaviors` protocol declares three methods that control the discretionary aspects of working with `NSDecimalNumber` objects.

The `scale` (page 7) and `roundingMode` (page 6) methods determine the precision of `NSDecimalNumber`'s return values and the way in which those values should be rounded to fit that precision. The `exceptionDuringOperation:error:leftOperand:rightOperand:` (page 6) method determines the way in which an `NSDecimalNumber` object should handle different calculation errors.

For an example of a class that adopts the `NSDecimalBehaviors` protocol, see the specification for `NSDecimalNumberHandler`.

## Tasks

### Rounding

- `roundingMode` (page 6)  
Returns the way that `NSDecimalNumber`'s `decimalNumberBy...` methods round their return values.
- `scale` (page 7)  
Returns the number of digits allowed after the decimal separator.

### Handling Errors

- `exceptionDuringOperation:error:leftOperand:rightOperand:` (page 6)  
Specifies what an `NSDecimalNumber` object will do when it encounters an error.

## Instance Methods

### exceptionDuringOperation:error:leftOperand:rightOperand:

Specifies what an `NSDecimalNumber` object will do when it encounters an error.

```
- (NSDecimalNumber *)exceptionDuringOperation:(SEL)method
    error:(NSCalculationError)error leftOperand:(NSDecimalNumber *)leftOperand
    rightOperand:(NSDecimalNumber *)rightOperand
```

#### Parameters

*method*

The method that was being executed when the error occurred.

*error*

The type of error that was generated.

*leftOperand*

The left operand.

*rightOperand*

The right operand.

#### Discussion

There are four possible values for *error*, described in [NSCalculationError](#) (page 9). The first three have to do with limits on the ability of `NSDecimalNumber` to represent decimal numbers. An `NSDecimalNumber` object can represent any number that can be expressed as mantissa x 10<sup>exponent</sup>, where mantissa is a decimal integer up to 38 digits long, and exponent is between -256 and 256. The fourth results from the caller trying to divide by 0.

In implementing `exceptionDuringOperation:error:leftOperand:rightOperand:`, you can handle each of these errors in several ways:

- Raise an exception. For an explanation of exceptions, see *Exception Programming Topics for Cocoa*.
- Return `nil`. The calling method will return its value as though no error had occurred. If *error* is `NSCalculationLossOfPrecision`, *method* will return an imprecise value—that is, one constrained to 38 significant digits. If *error* is `NSCalculationUnderflow` or `NSCalculationOverflow`, *method* will return `NSDecimalNumber's notANumber`. You shouldn't return `nil` if *error* is `NSDivideByZero`.
- Correct the error and return a valid `NSDecimalNumber` object. The calling method will use this as its own return value.

#### Availability

Available in Mac OS X v10.0 and later.

#### Declared In

`NSDecimalNumber.h`

### roundingMode

Returns the way that `NSDecimalNumber's decimalNumberBy...` methods round their return values.

```
- (NSRoundingMode)roundingMode
```

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

NSDecimalNumber.h

**scale**

Returns the number of digits allowed after the decimal separator.

```
- (short)scale
```

**Return Value**

The number of digits allowed after the decimal separator.

**Discussion**

This method limits the precision of the values returned by NSDecimalNumber's decimalNumberBy... methods. If `scale` returns a negative value, it affects the digits before the decimal separator as well. If `scale` returns `NSDecimalNoScale`, the number of digits is unlimited.

Assuming that `roundingMode` (page 6) returns `NSRoundPlain`, different values of `scale` have the following effects on the number 123.456:

Scale	Return Value
<code>NSDecimalNoScale</code>	123.456
2	123.45
0	123
-2	100

**Availability**

Available in Mac OS X v10.0 and later.

**Declared In**

NSDecimalNumber.h

## Constants

**NSRoundingMode**

These constants specify rounding behaviors.

```
typedef enum {
    NSRoundPlain,
    NSRoundDown,
    NSRoundUp,
    NSRoundBankers
} NSRoundingMode;
```

**Constants****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.

Available in Mac OS X v10.0 and later.

Declared in NSDecimal.h.

**NSRoundDown**

Round return values down.

Available in Mac OS X v10.0 and later.

Declared in NSDecimal.h.

**NSRoundUp**

Round return values up.

Available in Mac OS X v10.0 and later.

Declared in NSDecimal.h.

**NSRoundBankers**

Round to the closest possible return value; when halfway between two possibilities, return the possibility whose last digit is even.

In practice, this means that, over the long run, numbers will be rounded up as often as they are rounded down; there will be no systematic bias.

Available in Mac OS X v10.0 and later.

Declared in NSDecimal.h.

**Discussion**

The rounding mode matters only if the `scale` (page 7) method sets a limit on the precision of `NSDecimalNumber` return values. It has no effect if `scale` returns `NSDecimalNoScale`. Assuming that `scale` (page 7) returns 1, the rounding mode has the following effects on various original values:

Original Value	NSRoundPlain	NSRoundDown	NSRoundUp	NSRoundBankers
1.24	1.2	1.2	1.3	1.2
1.26	1.3	1.2	1.3	1.3
1.25	1.3	1.2	1.3	1.2
1.35	1.4	1.3	1.4	1.4
-1.35	-1.4	-1.4	-1.3	-1.4

**Availability**

Available in Mac OS X version 10.0 and later.

**Declared In**

NSDecimal.h



## NSCalculationError

Calculation error constants used to describe an error in `exceptionDuringOperation:error:leftOperand:rightOperand:` (page 6).

```
typedef enum {
    NSCalculationNoError = 0,
    NSCalculationLossOfPrecision,
    NSCalculationUnderflow,
    NSCalculationOverflow,
    NSCalculationDivideByZero
} NSCalculationError;
```

### Constants

`NSCalculationNoError`

No error occurred.

Available in Mac OS X v10.0 and later.

Declared in `NSDecimal.h`.

`NSCalculationLossOfPrecision`

The number can't be represented in 38 significant digits.

Available in Mac OS X v10.0 and later.

Declared in `NSDecimal.h`.

`NSCalculationOverflow`

The number is too large to represent.

Available in Mac OS X v10.0 and later.

Declared in `NSDecimal.h`.

`NSCalculationUnderflow`

The number is too small to represent.

Available in Mac OS X v10.0 and later.

Declared in `NSDecimal.h`.

`NSCalculationDivideByZero`

The caller tried to divide by 0.

Available in Mac OS X v10.0 and later.

Declared in `NSDecimal.h`.

### Availability

Available in Mac OS X version 10.0 and later.

### Declared In

`NSDecimal.h`



# Document Revision History

---

This table describes the changes to *NSDecimalNumberBehaviors Protocol Reference*.

Date	Notes
2006-05-23	Added definition of NSCalculationNoError.
	First publication of this content as a separate document.

**REVISION HISTORY**

Document Revision History

# Index

---

## E

---

exceptionDuringOperation:error:leftOperand:  
rightOperand: **protocol instance method** [6](#)

## N

---

NSCalculationDivideByZero **constant** [9](#)  
NSCalculationError **data type** [9](#)  
NSCalculationLossOfPrecision **constant** [9](#)  
NSCalculationNoError **constant** [9](#)  
NSCalculationOverflow **constant** [9](#)  
NSCalculationUnderflow **constant** [9](#)  
NSRoundBankers **constant** [8](#)  
NSRoundDown **constant** [8](#)  
NSRoundingMode **data type** [7](#)  
NSRoundPlain **constant** [8](#)  
NSRoundUp **constant** [8](#)

## R

---

roundingMode **protocol instance method** [6](#)

## S

---

scale **protocol instance method** [7](#)