NSNumberFormatter Class Reference

Cocoa > Data Management

2008-11-19

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

Simultaneously published in the United States and Canada.

Contents

NSNumberFormatter Class Reference

Inherits from	NSFormatter : NSObject
Conforms to	NSCoding (NSFormatter) NSCopying (NSFormatter) NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.0 and later.
Companion guide	Data Formatting Programming Guide for Cocoa
Declared in	NSNumberFormatter.h
Related sample code	CoreRecipes Grady NumberInput_IMKit_Sample Quartz Composer QCTV TrackBall

Overview

Instances of `NSNumberFormatter` format the textual representation of cells that contain `NSNumber` objects and convert textual representations of numeric values into `NSNumber` objects. The representation encompasses integers, floats, and doubles; floats and doubles can be formatted to a specified decimal position. `NSNumberFormatter` objects can also impose ranges on the numeric values cells can accept.

Many new methods were added to `NSNumberFormatter` for Mac OS X v10.4 with the intent of making the class interface more like that of `CFNumberFormatter`, the Core Foundation service on which the class is based. The behavior of an `NSNumberFormatter` object can conform either to the range of behaviors existing prior to Mac OS X v10.4 or to the range of behavior since that release. (Methods added for and since Mac OS X v10.4 are indicated by a method's availability statement.) You can determine the current formatter behavior with the `formatterBehavior` (page 21) method and you can set the formatter behavior with the `setFormatterBehavior:` (page 43) method.

iPhone OS Note: iPhone OS supports only the modern 10.4+ behavior. 10.0-style methods and format strings are not available on iPhone OS.

Important: The pre-Mac OS X v10.4 methods of `NSNumberFormatter` are not compatible with the methods added for Mac OS X v10.4. An `NSNumberFormatter` object should not invoke methods in these different behavior groups indiscriminately. Use the old-style methods if you have configured the number-formatter behavior to be `NSNumberFormatterBehavior10_0`. Use the new methods instead of the older-style ones if you have configured the number-formatter behavior to be `NSNumberFormatterBehavior10_4`.

Nomenclature note: `NSNumberFormatter` provides several methods (such as `setMaximumFractionDigits:` (page 48)) that allow you to manage the number of **fraction digits** allowed as input by an instance: "fraction digits" are the numbers after the decimal separator (in English locales typically referred to as the "decimal point").

Tasks

Configuring Formatter Behavior and Style

- `setFormatterBehavior:` (page 43)
  Sets the formatter behavior of the receiver.
- `formatterBehavior` (page 21)
  Returns an `NSNumberFormatterBehavior` constant that indicates the formatter behavior of the receiver.
+ `setDefaultFormatterBehavior:` (page 16)
  Sets the default formatter behavior for new instances of `NSNumberFormatter` .
+ `defaultFormatterBehavior` (page 15)
  Returns an `NSNumberFormatterBehavior` constant that indicates default formatter behavior for new instances of `NSNumberFormatter`.
- `setNumberStyle:` (page 54)
  Sets the number style used by the receiver.
- `numberStyle` (page 33)
  Returns the number-formatter style of the receiver.
- `setGeneratesDecimalNumbers:` (page 44)
  Controls whether the receiver creates instances of `NSDecimalNumber` when it converts strings to number objects.
- `generatesDecimalNumbers` (page 22)
  Returns a Boolean value that indicates whether the receiver creates instances of `NSDecimalNumber` when it converts strings to number objects.

Converting Between Numbers and Strings

- `getObjectValue:forString:range:error:` (page 22)
  Returns by reference a cell-content object after creating it from a range of characters in a given string.

- `numberFromString:` (page 32)
  Returns an `NSNumber` object created by parsing a given string.
- `stringFromNumber:` (page 64)
  Returns a string containing the formatted value of the provided number object.

Managing Localization of Numbers

- `setLocalizesFormat:` (page 47)
  Sets whether the dollar sign character (`$`), decimal separator character (`.`), and thousand separator character (`,`) are converted to appropriately localized characters as specified by the user's localization preference.
- `localizesFormat` (page 26)
  Returns a Boolean value that indicates whether the receiver localizes formats.
- `setLocale:` (page 46)
  Sets the locale of the receiver.
- `locale` (page 25)
  Returns the locale of the receiver.

Configuring Rounding Behavior

- `setRoundingBehavior:` (page 58)
  Sets the rounding behavior used by the receiver.
- `roundingBehavior` (page 36)
  Returns an `NSDecimalNumberHandler` object indicating the rounding behavior of the receiver.
- `setRoundingIncrement:` (page 59)
  Sets the rounding increment used by the receiver.
- `roundingIncrement` (page 36)
  Returns the rounding increment used by the receiver.
- `setRoundingMode:` (page 59)
  Sets the rounding mode used by the receiver.
- `roundingMode` (page 37)
  Returns the rounding mode used by the receiver.

Configuring Numeric Formats

- `setFormat:` (page 42)
  Sets the receiver's format.
- `formatWidth` (page 22)
  Returns the format width of the receiver.
- `setNegativeFormat:` (page 52)
  Sets the format the receiver uses to display negative values.
- `negativeFormat` (page 30)
  Returns the format used by the receiver to display negative numbers.

- `setPositiveFormat:` (page 57)
  Sets the format the receiver uses to display positive values.
- `positiveFormat` (page 35)
  Returns the format used by the receiver to display positive numbers.
- `setFormatWidth:` (page 43)
  Sets the format width used by the receiver.
- `format` (page 21)
  Returns the format used by the receiver.
- `setMultiplier:` (page 51)
  Sets the multiplier of the receiver.
- `multiplier` (page 30)
  Returns the multiplier used by the receiver as an `NSNumber` object.

Configuring Numeric Symbols

- `percentSymbol` (page 34)
  Returns the string that the receiver uses to represent the percent symbol.
- `setPercentSymbol:` (page 56)
  Sets the string used by the receiver to represent the percent symbol.
- `perMillSymbol` (page 34)
  Returns the string that the receiver uses for the per-thousands symbol.
- `setPerMillSymbol:` (page 56)
  Sets the string used by the receiver to represent the per-mill (per-thousand) symbol.
- `minusSign` (page 29)
  Returns the string the receiver uses to represent the minus sign.
- `setMinusSign:` (page 51)
  Sets the string used by the receiver for the minus sign.
- `plusSign` (page 34)
  Returns the string the receiver uses for the plus sign.
- `setPlusSign:` (page 56)
  Sets the string used by the receiver to represent the plus sign.
- `exponentSymbol` (page 20)
  Returns the string the receiver uses as an exponent symbol.
- `setExponentSymbol:` (page 42)
  Sets the string used by the receiver to represent the exponent symbol.
- `zeroSymbol` (page 69)
  Returns the string the receiver uses as the symbol to show the value zero.
- `setZeroSymbol:` (page 64)
  Sets the string the receiver uses as the symbol to show the value zero.
- `nilSymbol` (page 32)
  Returns the string the receiver uses to represent a `nil` value.
- `setNilSymbol:` (page 53)
  Sets the string the receiver uses to represent `nil` values.

- `notANumberSymbol` (page 32)
  Returns the symbol the receiver uses to represent NaN (“not a number”) when it converts values.
- `setNotANumberSymbol:` (page 54)
  Sets the string the receiver uses to represent NaN (“not a number”).
- `negativeInfinitySymbol` (page 30)
  Returns the symbol the receiver uses to represent negative infinity.
- `setNegativeInfinitySymbol:` (page 52)
  Sets the string used by the receiver for the negative infinity symbol.
- `positiveInfinitySymbol` (page 35)
  Returns the string the receiver uses for the positive infinity symbol.
- `setPositiveInfinitySymbol:` (page 57)
  Sets the string used by the receiver for the positive infinity symbol.

Configuring the Format of Currency

- `setCurrencySymbol:` (page 41)
  Sets the string used by the receiver as a local currency symbol.
- `currencySymbol` (page 20)
  Returns the receiver’s local currency symbol.
- `setCurrencyCode:` (page 40)
  Sets the receiver’s currency code.
- `currencyCode` (page 19)
  Returns the receiver’s currency code as a string.
- `setInternationalCurrencySymbol:` (page 46)
  Sets the string used by the receiver for the international currency symbol.
- `internationalCurrencySymbol` (page 24)
  Returns the international currency symbol used by the receiver.
- `setCurrencyGroupingSeparator:` (page 40)
  Sets the currency grouping separator for the receiver.
- `currencyGroupingSeparator` (page 19)
  Returns the currency grouping separator for the receiver.

Configuring Numeric Prefixes and Suffixes

- `setPositivePrefix:` (page 57)
  Sets the string the receiver uses as the prefix for positive values.
- `positivePrefix` (page 35)
  Returns the string the receiver uses as the prefix for positive values.
- `setPositiveSuffix:` (page 58)
  Sets the string the receiver uses as the suffix for positive values.
- `positiveSuffix` (page 36)
  Returns the string the receiver uses as the suffix for positive values.

- `setNegativePrefix:` (page 53)
  Sets the string the receiver uses as a prefix for negative values.
- `negativePrefix` (page 31)
  Returns the string the receiver inserts as a prefix to negative values.
- `setNegativeSuffix:` (page 53)
  Sets the string the receiver uses as a suffix for negative values.
- `negativeSuffix` (page 31)
  Returns the string the receiver adds as a suffix to negative values.

Configuring the Display of Numeric Values

- `setTextAttributesForNegativeValues:` (page 60)
  Sets the text attributes to be used in displaying negative values .
- `textAttributesForNegativeValues` (page 65)
  Returns a dictionary containing the text attributes that have been set for negative values.
- `setTextAttributesForPositiveValues:` (page 62)
  Sets the text attributes to be used in displaying positive values.
- `textAttributesForPositiveValues` (page 67)
  Returns a dictionary containing the text attributes that have been set for positive values.
- `setAttributedStringForZero:` (page 39)
  Sets the attributed string that the receiver uses to display zero values.
- `attributedStringForZero` (page 18)
  Returns the attributed string used to display zero values.
- `setTextAttributesForZero:` (page 62)
  Sets the text attributes used to display a zero value.
- `textAttributesForZero` (page 67)
  Returns a dictionary containing the text attributes used to display a value of zero.
- `setAttributedStringForNil:` (page 38)
  Sets the attributed string the receiver uses to display `nil` values.
- `attributedStringForNil` (page 17)
  Returns the attributed string used to display `nil` values.
- `setTextAttributesForNil:` (page 61)
  Sets the text attributes used to display the `nil` symbol.
- `textAttributesForNil` (page 66)
  Returns a dictionary containing the text attributes used to display the `nil` symbol.
- `setAttributedStringForNotANumber:` (page 39)
  Sets the attributed string the receiver uses to display “not a number” values.
- `attributedStringForNotANumber` (page 18)
  Returns the attributed string used to display “not a number” values.
- `setTextAttributesForNotANumber:` (page 61)
  Sets the text attributes used to display the NaN ("not a number") string.
- `textAttributesForNotANumber` (page 66)
  Returns a dictionary containing the text attributes used to display the NaN ("not a number") symbol.

- `setTextAttributesForPositiveInfinity:` (page 62)
  Sets the text attributes used to display the positive infinity symbol.
- `textAttributesForPositiveInfinity` (page 66)
  Returns a dictionary containing the text attributes used to display the positive infinity symbol.
- `setTextAttributesForNegativeInfinity:` (page 60)
  Sets the text attributes used to display the negative infinity symbol.
- `textAttributesForNegativeInfinity` (page 65)
  Returns a dictionary containing the text attributes used to display the negative infinity string.

Configuring Separators and Grouping Size

- `setGroupingSeparator:` (page 44)
  Specifies the string used by the receiver for a grouping separator.
- `groupingSeparator` (page 23)
  Returns a string containing the receiver's grouping separator.
- `setUsesGroupingSeparator:` (page 63)
  Controls whether the receiver displays the grouping separator.
- `usesGroupingSeparator` (page 68)
  Returns a Boolean value that indicates whether the receiver uses the grouping separator.
- `setThousandSeparator:` (page 63)
  Sets the character the receiver uses as a thousand separator.
- `thousandSeparator` (page 67)
  Returns a string containing the character the receiver uses to represent thousand separators.
- `setHasThousandSeparators:` (page 45)
  Sets whether the receiver uses thousand separators.
- `hasThousandSeparators` (page 24)
  Returns a Boolean value that indicates whether the receiver's format includes thousand separators.
- `setDecimalSeparator:` (page 41)
  Sets the character the receiver uses as a decimal separator.
- `decimalSeparator` (page 20)
  Returns a string containing the character the receiver uses to represent decimal separators.
- `setAlwaysShowsDecimalSeparator:` (page 38)
  Controls whether the receiver always shows the decimal separator, even for integer numbers.
- `alwaysShowsDecimalSeparator` (page 17)
  Returns a Boolean value that indicates whether the receiver always shows a decimal separator, even if the number is an integer.
- `setCurrencyDecimalSeparator:` (page 40)
  Sets the string used by the receiver as a decimal separator.
- `currencyDecimalSeparator` (page 19)
  Returns the receiver's currency decimal separator as a string.
- `setGroupingSize:` (page 45)
  Sets the grouping size of the receiver.
- `groupingSize` (page 23)
  Returns the receiver's primary grouping size.

- `setSecondaryGroupingSize:` (page 59)
  Sets the secondary grouping size of the receiver.
- `secondaryGroupingSize` (page 37)
  Returns the size of secondary groupings for the receiver.

Managing the Padding of Numbers

- `setPaddingCharacter:` (page 55)
  Sets the string that the receiver uses to pad numbers in the formatted string representation.
- `paddingCharacter` (page 33)
  Returns a string containing the padding character for the receiver.
- `setPaddingPosition:` (page 55)
  Sets the padding position used by the receiver.
- `paddingPosition` (page 33)
  Returns the padding position of the receiver.

Managing Input Attributes

- `setAllowsFloats:` (page 38)
  Sets whether the receiver allows as input floating-point values (that is, values that include the period character [.]).
- `allowsFloats` (page 16)
  Returns a Boolean value that indicates whether the receiver allows floating-point values as input.
- `setMinimum:` (page 49)
  Sets the lowest number the receiver allows as input.
- `minimum` (page 28)
  Returns the lowest number allowed as input by the receiver.
- `setMaximum:` (page 47)
  Sets the highest number the receiver allows as input.
- `maximum` (page 26)
  Returns the highest number allowed as input by the receiver.
- `setMinimumIntegerDigits:` (page 50)
  Sets the minimum number of integer digits allowed as input by the receiver.
- `minimumIntegerDigits` (page 29)
  Returns the minimum number of integer digits allowed as input by the receiver.
- `setMinimumFractionDigits:` (page 50)
  Sets the minimum number of digits after the decimal separator allowed as input by the receiver.
- `minimumFractionDigits` (page 28)
  Returns the minimum number of digits after the decimal separator allowed as input by the receiver.
- `setMaximumIntegerDigits:` (page 48)
  Sets the maximum number of integer digits allowed as input by the receiver.
- `maximumIntegerDigits` (page 27)
  Returns the maximum number of integer digits allowed as input by the receiver.

- `setMaximumFractionDigits:` (page 48)
  Sets the maximum number of digits after the decimal separator allowed as input by the receiver.
- `maximumFractionDigits` (page 27)
  Returns the maximum number of digits after the decimal separator allowed as input by the receiver.

Configuring Significant Digits

- `setUsesSignificantDigits:` (page 64)
  Sets whether the receiver uses significant digits.
- `usesSignificantDigits` (page 68)
  Returns a Boolean value that indicates whether the receiver uses significant digits.
- `setMinimumSignificantDigits:` (page 51)
  Sets the minimum number of significant digits for the receiver.
- `minimumSignificantDigits` (page 29)
  Returns the minimum number of significant digits for the receiver.
- `setMaximumSignificantDigits:` (page 49)
  Sets the maximum number of significant digits for the receiver.
- `maximumSignificantDigits` (page 27)
  Returns the maximum number of significant digits for the receiver.

Managing Leniency Behavior

- `setLenient:` (page 46)
  Sets whether the receiver will use heuristics to guess at the number which is intended by a string.
- `isLenient` (page 25)
  Returns a Boolean value that indicates whether the receiver uses heuristics to guess at the number which is intended by a string.

Managing the Validation of Partial Numeric Strings

- `setPartialStringValidationEnabled:` (page 55)
  Sets whether partial string validation is enabled for the receiver.
- `isPartialStringValidationEnabled` (page 25)
  Returns a Boolean value that indicates whether partial string validation is enabled.

Class Methods

defaultFormatterBehavior

Returns an `NSNumberFormatterBehavior` constant that indicates default formatter behavior for new instances of `NSNumberFormatter`.

```
+ (NSNumberFormatterBehavior)defaultFormatterBehavior
```

Return Value
An `NSNumberFormatterBehavior` constant that indicates default formatter behavior for new instances of `NSNumberFormatter`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`+ setDefaultFormatterBehavior:` (page 16)

Declared In
`NSNumberFormatter.h`

setDefaultFormatterBehavior:

Sets the default formatter behavior for new instances of `NSNumberFormatter`.

```
+ (void)setDefaultFormatterBehavior:(NSNumberFormatterBehavior)behavior
```

Parameters
behavior
An `NSNumberFormatterBehavior` constant that indicates the revision of the class providing the default behavior.

Availability
Available in Mac OS X v10.4 and later.

See Also
`+ defaultFormatterBehavior` (page 15)

Related Sample Code
NumberInput_IMKit_Sample

Declared In
`NSNumberFormatter.h`

Instance Methods

allowsFloats

Returns a Boolean value that indicates whether the receiver allows floating-point values as input.

```
- (BOOL)allowsFloats
```

Return Value
`YES` if the receiver allows as input floating-point values (that is, values that include the period character [`.`]), otherwise `NO`.

Discussion
When this method returns `NO`, only integer values can be provided as input. The default is `YES`.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setAllowsFloats:` (page 38)

Declared In
`NSNumberFormatter.h`

alwaysShowsDecimalSeparator

Returns a Boolean value that indicates whether the receiver always shows a decimal separator, even if the number is an integer.

```
- (BOOL)alwaysShowsDecimalSeparator
```

Return Value
`YES` if the receiver always shows a decimal separator, even if the number is an integer, otherwise `NO`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setAlwaysShowsDecimalSeparator:` (page 38)

Declared In
`NSNumberFormatter.h`

attributedStringForNil

Returns the attributed string used to display `nil` values.

```
- (NSAttributedString *)attributedStringForNil
```

Return Value
The attributed string used to display `nil` values.

Discussion
By default `nil` values are displayed as an empty string.

Special Considerations

This method is for use with formatters using `NSNumberFormatterBehavior10_0` behavior.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setAttributedStringForNil:` (page 38)

Declared In
NSNumberFormatter.h

attributedStringForNotANumber

Returns the attributed string used to display “not a number” values.

```
- (NSAttributedString *)attributedStringForNotANumber
```

Return Value
The attributed string used to display “not a number” values.

Discussion
By default “not a number” values are displayed as the string “NaN”.

Special Considerations

This method is for use with formatters using `NSNumberFormatterBehavior10_0` behavior.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setAttributedStringForNotANumber:` (page 39)

Declared In
NSNumberFormatter.h

attributedStringForZero

Returns the attributed string used to display zero values.

```
- (NSAttributedString *)attributedStringForZero
```

Return Value
The attributed string used to display zero values.

Discussion
By default zero values are displayed according to the format specified for positive values; for more discussion of this subject see *Data Formatting Programming Guide for Cocoa*.

Special Considerations

This method is for use with formatters using `NSNumberFormatterBehavior10_0` behavior.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setAttributedStringForZero:` (page 39)

Declared In
NSNumberFormatter.h

currencyCode

Returns the receiver's currency code as a string.

```
- (NSString *)currencyCode
```

Return Value
The receiver's currency code as a string.

Discussion
A currency code is a three-letter code that is, in most cases, composed of a country's two-character Internet country code plus an extra character to denote the currency unit. For example, the currency code for the Australian dollar is "AUD". Currency codes are based on the ISO 4217 standard.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setCurrencyCode:` (page 40)

Declared In
`NSNumberFormatter.h`

currencyDecimalSeparator

Returns the receiver's currency decimal separator as a string.

```
- (NSString *)currencyDecimalSeparator
```

Return Value
The receiver's currency decimal separator as a string.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- currencyDecimalSeparator` (page 19)

Declared In
`NSNumberFormatter.h`

currencyGroupingSeparator

Returns the currency grouping separator for the receiver.

```
- (NSString *)currencyGroupingSeparator
```

Return Value
The currency grouping separator for the receiver.

Availability
Available in Mac OS X v10.5 and later.

Declared In
NSNumberFormatter.h

currencySymbol

Returns the receiver's local currency symbol.

```
- (NSString *)currencySymbol
```

Discussion
A country typically has a local currency symbol and an international currency symbol. The local symbol is used within the country, while the international currency symbol is used in international contexts to specify that country's currency unambiguously. The local currency symbol is often represented by a Unicode code point.

Availability
Available in Mac OS X v10.4 and later.

See Also
- internationalCurrencySymbol (page 24)
- setCurrencySymbol: (page 41)

Declared In
NSNumberFormatter.h

decimalSeparator

Returns a string containing the character the receiver uses to represent decimal separators.

```
- (NSString *)decimalSeparator
```

Return Value
A string containing the character the receiver uses to represent decimal separators.

Discussion
The return value doesn't indicate whether decimal separators are enabled.

Availability
Available in Mac OS X v10.0 and later.

See Also
- setDecimalSeparator: (page 41)

Declared In
NSNumberFormatter.h

exponentSymbol

Returns the string the receiver uses as an exponent symbol.

```
- (NSString *)exponentSymbol
```

Return Value
The string the receiver uses as an exponent symbol.

Discussion
The exponent symbol is the “E” or “e” in the scientific notation of numbers, as in 1.0e+56.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setExponentSymbol:` (page 42)

Declared In
`NSNumberFormatter.h`

format

Returns the format used by the receiver.

```
- (NSString *)format
```

Return Value
The format used by the receiver.

Special Considerations

This method is for use with formatters using `NSNumberFormatterBehavior10_0` behavior.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setFormat:` (page 42)

Declared In
`NSNumberFormatter.h`

formatterBehavior

Returns an `NSNumberFormatterBehavior` constant that indicates the formatter behavior of the receiver.

```
- (NSNumberFormatterBehavior)formatterBehavior
```

Return Value
An `NSNumberFormatterBehavior` constant that indicates the formatter behavior of the receiver.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setFormatterBehavior:` (page 43)

Declared In
`NSNumberFormatter.h`

formatWidth

Returns the format width of the receiver.

```
- (NSUInteger)formatWidth
```

Discussion
The format width is the number of characters of a formatted number within a string that is either left justified or right justified based on the value returned from `paddingPosition` (page 33).

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setFormatWidth:` (page 43)

Declared In
`NSNumberFormatter.h`

generatesDecimalNumbers

Returns a Boolean value that indicates whether the receiver creates instances of `NSDecimalNumber` when it converts strings to number objects.

```
- (BOOL)generatesDecimalNumbers
```

Return Value
`YES` if the receiver creates instances of `NSDecimalNumber` when it converts strings to number objects, `NO` if it creates instance of `NSNumber`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setGeneratesDecimalNumbers:` (page 44)

Declared In
`NSNumberFormatter.h`

getObjectValue:forString:range:error:

Returns by reference a cell-content object after creating it from a range of characters in a given string.

```
- (BOOL)getObjectValue:(out id *)anObject forString:(NSString *)aString range:(inout
    NSRange *)rangep error:(out NSError **)error
```

Parameters

anObject

On return, contains an instance of `NSDecimalNumber` or `NSNumber` based on the current value of `generatesDecimalNumbers` (page 22). The default is to return `NSDecimalNumber` instances

aString

A string object with the range of characters specified in *rangep* that is used to create *anObject*.

rangep

A range of characters in *aString*. On return, contains the actual range of characters used to create the object.

error

If an error occurs, upon return contains an `NSError` object that explains the reason why the conversion failed. If you pass in `nil` for *error* you are indicating that you are not interested in error information.

Return Value
`YES` if the conversion from string to cell-content object was successful, otherwise `NO`.

Discussion
If there is an error, the delegate (if any) of the control object managing the cell can then respond to the failure in the `NSControl` delegation method `control:didFailToFormatString:errorDescription:`.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `numberFromString:` (page 32)
- `stringFromNumber:` (page 64)

Declared In
`NSNumberFormatter.h`

groupingSeparator

Returns a string containing the receiver's grouping separator.

```
- (NSString *)groupingSeparator
```

Return Value
A string containing the receiver's grouping separator.

Discussion
For example, the grouping separator used in the United States is the comma ("10,000") whereas in France it is the period ("10.000").

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setGroupingSeparator:` (page 44)

Declared In
`NSNumberFormatter.h`

groupingSize

Returns the receiver's primary grouping size.

```
- (NSUInteger)groupingSize
```

Return Value
The receiver's primary grouping size.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setGroupingSize:` (page 45)

Declared In
`NSNumberFormatter.h`

hasThousandSeparators

Returns a Boolean value that indicates whether the receiver's format includes thousand separators.

```
- (BOOL)hasThousandSeparators
```

Return Value
`YES` if the receiver's format includes thousand separators, otherwise `NO`.

Discussion
The default is `NO`.

Special Considerations

This method is for use with formatters using `NSNumberFormatterBehavior10_0` behavior.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setHasThousandSeparators:` (page 45)

Declared In
`NSNumberFormatter.h`

internationalCurrencySymbol

Returns the international currency symbol used by the receiver.

```
- (NSString *)internationalCurrencySymbol
```

Discussion
A country typically has a local currency symbol and an international currency symbol. The local symbol is used within the country, while the international currency symbol is used in international contexts to specify that country's currency unambiguously. The international currency symbol is often represented by a Unicode code point.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `currencySymbol` (page 20)

- setInternationalCurrencySymbol: (page 46)

Declared In
NSNumberFormatter.h

isLenient

Returns a Boolean value that indicates whether the receiver uses heuristics to guess at the number which is intended by a string.

```
- (BOOL)isLenient
```

Return Value
YES if the receiver uses heuristics to guess at the number which is intended by the string; otherwise NO.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setLenient: (page 46)

Declared In
NSNumberFormatter.h

isPartialStringValidationEnabled

Returns a Boolean value that indicates whether partial string validation is enabled.

```
- (BOOL)isPartialStringValidationEnabled
```

Return Value
YES if partial string validation is enabled, otherwise NO.

Availability
Available in Mac OS X v10.5 and later.

See Also
- setPartialStringValidationEnabled: (page 55)

Declared In
NSNumberFormatter.h

locale

Returns the locale of the receiver.

```
- (NSLocale *)locale
```

Return Value
The locale of the receiver.

Discussion
A number formatter's locale specifies default localization attributes, such as ISO country and language codes, currency code, calendar, system of measurement, and decimal separator.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setLocale:` (page 46)

Declared In
`NSNumberFormatter.h`

localizesFormat

Returns a Boolean value that indicates whether the receiver localizes formats.

```
- (BOOL)localizesFormat
```

Return Value
`YES` if the receiver localizes formats, otherwise `NO`.

Discussion
The default is `NO`.

Special Considerations

This method is for use with formatters using `NSNumberFormatterBehavior10_0` behavior.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setLocalizesFormat:` (page 47)

Declared In
`NSNumberFormatter.h`

maximum

Returns the highest number allowed as input by the receiver.

```
- (NSNumber *)maximum
```

Return Value
The highest number allowed as input by the receiver or `nil`, meaning no limit.

Discussion
For versions prior to Mac OS X v10.4 (and number-formatter behavior set to `NSNumberFormatterBehavior10_0`) this method returns an `NSDecimalNumber` object.

Availability
Available in Mac OS X v10.4 and later.

Version returning NSDecimalNumber available prior to Mac OS X v10.4.

See Also
- `- setMaximum:` (page 47)
- `+ setDefaultFormatterBehavior:` (page 16)
- `- formatterBehavior` (page 21)
- `- setFormatterBehavior:` (page 43)

Declared In
`NSNumberFormatter.h`

maximumFractionDigits

Returns the maximum number of digits after the decimal separator allowed as input by the receiver.

```
- (NSUInteger)maximumFractionDigits
```

Return Value
The maximum number of digits after the decimal separator allowed as input by the receiver.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `- setMaximumFractionDigits:` (page 48)

Declared In
`NSNumberFormatter.h`

maximumIntegerDigits

Returns the maximum number of integer digits allowed as input by the receiver.

```
- (NSUInteger)maximumIntegerDigits
```

Return Value
The maximum number of integer digits allowed as input by the receiver.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `- setMaximumIntegerDigits:` (page 48)

Declared In
`NSNumberFormatter.h`

maximumSignificantDigits

Returns the maximum number of significant digits for the receiver.

```
- (NSUInteger)maximumSignificantDigits
```

Return Value
The maximum number of significant digits for the receiver.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setMaximumSignificantDigits:` (page 49)
- `minimumSignificantDigits` (page 29)
- `usesSignificantDigits` (page 68)

Declared In
`NSNumberFormatter.h`

minimum

Returns the lowest number allowed as input by the receiver.

```
- (NSNumber *)minimum
```

Return Value
The lowest number allowed as input by the receiver or `nil`, meaning no limit.

Discussion
For versions prior to Mac OS X v10.4 (and number-formatter behavior set to `NSNumberFormatterBehavior10_0`) this method returns an `NSDecimalNumber` object.

Availability
Available in Mac OS X v10.4 and later. Version returning NSDecimalNumber available prior to Mac OS X v10.4.

See Also
- `setMinimum:` (page 49)
- `setDefaultFormatterBehavior:` (page 16)
- `formatterBehavior` (page 21)
- `setFormatterBehavior:` (page 43)

Declared In
`NSNumberFormatter.h`

minimumFractionDigits

Returns the minimum number of digits after the decimal separator allowed as input by the receiver.

```
- (NSUInteger)minimumFractionDigits
```

Return Value
The minimum number of digits after the decimal separator allowed as input by the receiver.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setMinimumFractionDigits:` (page 50)

Declared In
`NSNumberFormatter.h`

minimumIntegerDigits

Returns the minimum number of integer digits allowed as input by the receiver.

```
- (NSUInteger)minimumIntegerDigits
```

Return Value
The minimum number of integer digits allowed as input by the receiver.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setMinimumIntegerDigits:` (page 50)

Declared In
`NSNumberFormatter.h`

minimumSignificantDigits

Returns the minimum number of significant digits for the receiver.

```
- (NSUInteger)minimumSignificantDigits
```

Return Value
The minimum number of significant digits for the receiver.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `setMinimumSignificantDigits:` (page 51)
- `maximumSignificantDigits` (page 27)
- `usesSignificantDigits` (page 68)

Declared In
`NSNumberFormatter.h`

minusSign

Returns the string the receiver uses to represent the minus sign.

```
- (NSString *)minusSign
```

Return Value
The string that represents the receiver's minus sign.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setMinusSign:` (page 51)

Declared In
`NSNumberFormatter.h`

multiplier

Returns the multiplier used by the receiver as an `NSNumber` object.

```
- (NSNumber *)multiplier
```

Discussion
A multiplier is a factor used in conversions between numbers and strings (that is, numbers as stored and numbers as displayed). When the input value is a string, the multiplier is used to divide, and when the input value is a number, the multiplier is used to multiply. These operations allow the formatted values to be different from the values that a program manipulates internally.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setMultiplier:` (page 51)

Declared In
`NSNumberFormatter.h`

negativeFormat

Returns the format used by the receiver to display negative numbers.

```
- (NSString *)negativeFormat
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setNegativeFormat:` (page 52)
- `setFormat:` (page 42)

Declared In
`NSNumberFormatter.h`

negativeInfinitySymbol

Returns the symbol the receiver uses to represent negative infinity.

```
- (NSString *)negativeInfinitySymbol
```

Return Value
The symbol the receiver uses to represent negative infinity.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setNegativeInfinitySymbol:` (page 52)

Declared In
`NSNumberFormatter.h`

negativePrefix

Returns the string the receiver inserts as a prefix to negative values.

```
- (NSString *)negativePrefix
```

Return Value
The string the receiver inserts as a prefix to negative values.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `negativeSuffix` (page 31)
- `setNegativePrefix:` (page 53)

Declared In
`NSNumberFormatter.h`

negativeSuffix

Returns the string the receiver adds as a suffix to negative values.

```
- (NSString *)negativeSuffix
```

Return Value
The string the receiver adds as a suffix to negative values.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `negativePrefix` (page 31)
- `setNegativeSuffix:` (page 53)

Declared In
`NSNumberFormatter.h`

nilSymbol

Returns the string the receiver uses to represent a `nil` value.

```
- (NSString *)nilSymbol
```

Return Value
The string the receiver uses to represent a `nil` value.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setNilSymbol:` (page 53)

Declared In
`NSNumberFormatter.h`

notANumberSymbol

Returns the symbol the receiver uses to represent NaN ("not a number") when it converts values.

```
- (NSString *)notANumberSymbol
```

Return Value
The symbol the receiver uses to represent NaN ("not a number") when it converts values.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setNotANumberSymbol:` (page 54)

Declared In
`NSNumberFormatter.h`

numberFromString:

Returns an `NSNumber` object created by parsing a given string.

```
- (NSNumber *)numberFromString:(NSString *)string
```

Parameters
string
An `NSString` object that is parsed to generate the returned number object.

Return Value
An `NSNumber` object created by parsing *string* using the receiver's format.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- stringFromNumber:` (page 64)

Related Sample Code
NumberInput_IMKit_Sample

Declared In
`NSNumberFormatter.h`

numberStyle

Returns the number-formatter style of the receiver.

```
- (NSNumberFormatterStyle)numberStyle
```

Return Value
An `NSNumberFormatterStyle` constant that indicates the number-formatter style of the receiver.

Discussion
Styles are essentially predetermined sets of values for certain properties. Examples of number-formatter styles are those used for decimal values, percentage values, and currency.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setNumberStyle:` (page 54)

Declared In
`NSNumberFormatter.h`

paddingCharacter

Returns a string containing the padding character for the receiver.

```
- (NSString *)paddingCharacter
```

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setPaddingCharacter:` (page 55)

Declared In
`NSNumberFormatter.h`

paddingPosition

Returns the padding position of the receiver.

```
- (NSNumberFormatterPadPosition)paddingPosition
```

Discussion
The returned constant indicates whether the padding is before or after the number's prefix or suffix.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setPaddingPosition:` (page 55)

Declared In
`NSNumberFormatter.h`

percentSymbol

Returns the string that the receiver uses to represent the percent symbol.

```
- (NSString *)percentSymbol
```

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setPercentSymbol:` (page 56)

Declared In
`NSNumberFormatter.h`

perMillSymbol

Returns the string that the receiver uses for the per-thousands symbol.

```
- (NSString *)perMillSymbol
```

Return Value
The string that the receiver uses for the per-thousands symbol.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setPerMillSymbol:` (page 56)

Declared In
`NSNumberFormatter.h`

plusSign

Returns the string the receiver uses for the plus sign.

```
- (NSString *)plusSign
```

Return Value
The string the receiver uses for the plus sign.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setPlusSign:` (page 56)

Declared In
`NSNumberFormatter.h`

positiveFormat

Returns the format used by the receiver to display positive numbers.

```
- (NSString *)positiveFormat
```

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setPositiveFormat:` (page 57)
- `setFormat:` (page 42)

Declared In
`NSNumberFormatter.h`

positiveInfinitySymbol

Returns the string the receiver uses for the positive infinity symbol.

```
- (NSString *)positiveInfinitySymbol
```

Return Value
The string the receiver uses for the positive infinity symbol.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setPositiveInfinitySymbol:` (page 57)

Declared In
`NSNumberFormatter.h`

positivePrefix

Returns the string the receiver uses as the prefix for positive values.

```
- (NSString *)positivePrefix
```

Return Value
The string the receiver uses as the prefix for positive values.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setPositivePrefix:` (page 57)

Declared In
`NSNumberFormatter.h`

positiveSuffix

Returns the string the receiver uses as the suffix for positive values.

```
- (NSString *)positiveSuffix
```

Return Value
The string the receiver uses as the suffix for positive values.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setPositiveSuffix:` (page 58)

Declared In
`NSNumberFormatter.h`

roundingBehavior

Returns an `NSDecimalNumberHandler` object indicating the rounding behavior of the receiver.

```
- (NSDecimalNumberHandler *)roundingBehavior
```

Return Value
An `NSDecimalNumberHandler` object indicating the rounding behavior of the receiver.

Special Considerations
This method is for use with formatters using `NSNumberFormatterBehavior10_0` behavior.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setRoundingBehavior:` (page 58)

Declared In
`NSNumberFormatter.h`

roundingIncrement

Returns the rounding increment used by the receiver.

```
- (NSNumber *)roundingIncrement
```

Return Value
The rounding increment used by the receiver.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setRoundingIncrement: (page 59)

Declared In
NSNumberFormatter.h

roundingMode

Returns the rounding mode used by the receiver.

```
- (NSNumberFormatterRoundingMode)roundingMode
```

Return Value
The rounding mode used by the receiver.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setRoundingMode: (page 59)

Declared In
NSNumberFormatter.h

secondaryGroupingSize

Returns the size of secondary groupings for the receiver.

```
- (NSUInteger)secondaryGroupingSize
```

Return Value
The size of secondary groupings for the receiver.

Discussion
Some locales allow the specification of another grouping size for larger numbers. For example, some locales may represent a number such as 61, 242, 378.46 (as in the United States) as 6,12,42,378.46. In this case, the secondary grouping size (covering the groups of digits furthest from the decimal point) is 2.

Availability
Available in Mac OS X v10.4 and later.

See Also
- setSecondaryGroupingSize: (page 59)

Declared In
NSNumberFormatter.h

setAllowsFloats:

Sets whether the receiver allows as input floating-point values (that is, values that include the period character [.]).

```
- (void)setAllowsFloats:(BOOL)flag
```

Parameters

flag

YES if the receiver allows floating-point values, NO otherwise.

Discussion
By default, floating point values are allowed as input.

Availability
Available in Mac OS X v10.0 and later.

See Also
- allowsFloats (page 16)

Related Sample Code
Quartz Composer QCTV

Declared In
NSNumberFormatter.h

setAlwaysShowsDecimalSeparator:

Controls whether the receiver always shows the decimal separator, even for integer numbers.

```
- (void)setAlwaysShowsDecimalSeparator:(BOOL)flag
```

Parameters

flag

YES if the receiver should always show the decimal separator, NO otherwise.

Availability
Available in Mac OS X v10.4 and later.

See Also
- alwaysShowsDecimalSeparator (page 17)

Declared In
NSNumberFormatter.h

setAttributedStringForNil:

Sets the attributed string the receiver uses to display nil values.

```
- (void)setAttributedStringForNil:(NSAttributedString *)newAttributedString
```

Parameters

newAttributedString

An NSAttributedString object that the receiver uses to display nil values.

Special Considerations

This method is for use with formatters using `NSNumberFormatterBehavior10_0` behavior.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- attributedStringForNil` (page 17)

Declared In
`NSNumberFormatter.h`

setAttributedStringForNotANumber:

Sets the attributed string the receiver uses to display “not a number” values.

```
- (void)setAttributedStringForNotANumber:(NSAttributedString *)newAttributedString
```

Parameters

newAttributedString
An `NSAttributedString` object that the receiver uses to display NaN values.

Special Considerations

This method is for use with formatters using `NSNumberFormatterBehavior10_0` behavior.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- attributedStringForNotANumber` (page 18)

Declared In
`NSNumberFormatter.h`

setAttributedStringForZero:

Sets the attributed string that the receiver uses to display zero values.

```
- (void)setAttributedStringForZero:(NSAttributedString *)newAttributedString
```

Parameters

newAttributedString
An `NSAttributedString` object that the receiver uses to display zero values.

Special Considerations

This method is for use with formatters using `NSNumberFormatterBehavior10_0` behavior.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- attributedStringForZero` (page 18)

Declared In
NSNumberFormatter.h

setCurrencyCode:

Sets the receiver's currency code.

```
- (void)setCurrencyCode:(NSString *)string
```

Parameters

string

A string specifying the receiver's new currency code.

Discussion
A currency code is a three-letter code that is, in most cases, composed of a country's two-character Internet country code plus an extra character to denote the currency unit. For example, the currency code for the Australian dollar is "AUD". Currency codes are based on the ISO 4217 standard.

Availability
Available in Mac OS X v10.4 and later.

See Also
- currencyCode (page 19)

Declared In
NSNumberFormatter.h

setCurrencyDecimalSeparator:

Sets the string used by the receiver as a decimal separator.

```
- (void)setCurrencyDecimalSeparator:(NSString *)string
```

Parameters

string

The string to use as the currency decimal separator.

Availability
Available in Mac OS X v10.4 and later.

See Also
- currencyDecimalSeparator (page 19)

Declared In
NSNumberFormatter.h

setCurrencyGroupingSeparator:

Sets the currency grouping separator for the receiver.

```
- (NSString *)setCurrencyGroupingSeparator:(NSString *)string
```

Parameters

string

The currency grouping separator for the receiver.

Availability

Available in Mac OS X v10.5 and later.

Declared In

NSNumberFormatter.h

setCurrencySymbol:

Sets the string used by the receiver as a local currency symbol.

```
- (void)setCurrencySymbol:(NSString *)string
```

Parameters

string

A string that represents a local currency symbol.

Discussion

The local symbol is used within the country, while the international currency symbol is used in international contexts to specify that country's currency unambiguously. The local currency symbol is often represented by a Unicode code point.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `currencySymbol` (page 20)
- `setInternationalCurrencySymbol:` (page 46)

Declared In

NSNumberFormatter.h

setDecimalSeparator:

Sets the character the receiver uses as a decimal separator.

```
- (void)setDecimalSeparator:(NSString *)newSeparator
```

Parameters

newSeparator

The string that specifies the decimal-separator character to use. If *newSeparator* contains multiple characters, only the first one is used.

Discussion

If you don't have decimal separators enabled through another means (such as `setFormat:` (page 42)), using this method enables them.

Availability

Available in Mac OS X v10.0 and later.

See Also
- `decimalSeparator` (page 20)
- `formatterBehavior` (page 21)

Declared In
`NSNumberFormatter.h`

setExponentSymbol:

Sets the string used by the receiver to represent the exponent symbol.

```
- (void)setExponentSymbol:(NSString *)string
```

Parameters

string

A string that represents an exponent symbol.

Discussion
The exponent symbol is the “E” or “e” in the scientific notation of numbers, as in 1.0e+56.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `exponentSymbol` (page 20)

Declared In
`NSNumberFormatter.h`

setFormat:

Sets the receiver’s format.

```
- (void)setFormat:(NSString *)aFormat
```

Parameters

aFormat

A string that can consist of one, two, or three parts separated by “;”. The first part of the string represents the positive format, the second part of the string represents the zero value, and the last part of the string represents the negative format. If the string has just two parts, the first one becomes the positive format, and the second one becomes the negative format. If the string has just one part, it becomes the positive format, and default formats are provided for zero and negative values based on the positive format. For more discussion of this subject, see *Data Formatting Programming Guide for Cocoa*.

Discussion
The following code excerpt shows the three different approaches for setting an `NSNumberFormatter` object’s format using `setFormat:`:

```
NSNumberFormatter *numberFormatter =
    [[[NSNumberFormatter alloc] init] autorelease];

// specify just positive format
[numberFormatter setFormat:@"$#,##0.00"];
```

```
// specify positive and negative formats
[numberFormatter setFormat:@"$#,##0.00;($#,##0.00)"];

// specify positive, zero, and negative formats
[numberFormatter setFormat:@"$#,###.00;0.00;($#,##0.00)"];
```

Special Considerations

This method is for use with formatters using `NSNumberFormatterBehavior10_0` behavior.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- format` (page 21)

Declared In
`NSNumberFormatter.h`

setFormatterBehavior:

Sets the formatter behavior of the receiver.

```
- (void)setFormatterBehavior:(NSNumberFormatterBehavior)behavior
```

Parameters

behavior

An `NSNumberFormatterBehavior` constant that indicates the revision of the `NSNumberFormatter` class providing the current behavior.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- formatterBehavior` (page 21)

Related Sample Code
TrackBall

Declared In
`NSNumberFormatter.h`

setFormatWidth:

Sets the format width used by the receiver.

```
- (void)setFormatWidth:(NSUInteger)number
```

Parameters

number

An integer that specifies the format width.

Discussion
The format width is the number of characters of a formatted number within a string that is either left justified or right justified based on the value returned from `paddingPosition` (page 33).

Availability
Available in Mac OS X v10.4 and later.

See Also
`- formatWidth` (page 22)

Declared In
`NSNumberFormatter.h`

setGeneratesDecimalNumbers:

Controls whether the receiver creates instances of `NSDecimalNumber` when it converts strings to number objects.

```
- (void)setGeneratesDecimalNumbers:(BOOL)flag
```

Parameters

flag

`YES` if the receiver should generate `NSDecimalNumber` instances, `NO` if it should generate `NSNumber` instances.

Discussion
The default is `YES`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- generatesDecimalNumbers` (page 22)

Declared In
`NSNumberFormatter.h`

setGroupingSeparator:

Specifies the string used by the receiver for a grouping separator.

```
- (void)setGroupingSeparator:(NSString *)string
```

Parameters

string

A string that specifies the grouping separator to use.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- groupingSeparator` (page 23)

Declared In
NSNumberFormatter.h

setGroupingSize:

Sets the grouping size of the receiver.

```
- (void)setGroupingSize:(NSUInteger)numDigits
```

Parameters

numDigits

An integer that specifies the grouping size.

Availability
Available in Mac OS X v10.4 and later.

See Also
- groupingSize (page 23)

Declared In
NSNumberFormatter.h

setHasThousandSeparators:

Sets whether the receiver uses thousand separators.

```
- (void)setHasThousandSeparators:(BOOL)flag
```

Parameters

flag

When *flag* is NO, thousand separators are disabled for both positive and negative formats (even if you've set them through another means, such as setFormat: (page 42)). When *flag* is YES, thousand separators are used.

Discussion
In addition to using this method to add thousand separators to your format, you can also use it to disable thousand separators if you've set them using another method. The default is NO (though you in effect change this setting to YES when you set thousand separators through any means, such as setFormat: (page 42)).

Special Considerations

This method is for use with formatters using NSNumberFormatterBehavior10_0 behavior.

Availability
Available in Mac OS X v10.0 and later.

See Also
- hasThousandSeparators (page 24)

Declared In
NSNumberFormatter.h

setInternationalCurrencySymbol:

Sets the string used by the receiver for the international currency symbol.

```
- (void)setInternationalCurrencySymbol:(NSString *)string
```

Parameters

string

A string that represents an international currency symbol.

Discussion

The local symbol is used within the country, while the international currency symbol is used in international contexts to specify that country's currency unambiguously. The local currency symbol is often represented by a Unicode code point.

Availability

Available in Mac OS X v10.4 and later.

See Also

- internationalCurrencySymbol (page 24)

Declared In

NSNumberFormatter.h

setLenient:

Sets whether the receiver will use heuristics to guess at the number which is intended by a string.

```
- (void)setLenient:(BOOL)b
```

Parameters

b

YES if the receiver will use heuristics to guess at the number which is intended by the string; otherwise NO.

Discussion

If the formatter is set to be lenient, as with any guessing it may get the result number wrong (that is, a number other than that which was intended).

Availability

Available in Mac OS X v10.5 and later.

See Also

- isLenient (page 25)

Declared In

NSNumberFormatter.h

setLocale:

Sets the locale of the receiver.

```
- (void)setLocale:(NSLocale *)theLocale
```

Parameters

theLocale

An `NSLocale` object representing the new locale of the receiver.

Discussion

The locale determines the default values for many formatter attributes, such as ISO country and language codes, currency code, calendar, system of measurement, and decimal separator.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `locale` (page 25)

Declared In

`NSNumberFormatter.h`

setLocalizesFormat:

Sets whether the dollar sign character (`$`), decimal separator character (`.`), and thousand separator character (`,`) are converted to appropriately localized characters as specified by the user's localization preference.

```
- (void)setLocalizesFormat:(BOOL)flag
```

Parameters

flag

`YES` if these characters are converted to the localized equivalents, `NO` otherwise.

Discussion

While the currency-symbol part of this feature may be useful in certain types of applications, it's probably more likely that you would tie a particular application to a particular currency (that is, that you would "hard-code" the currency symbol and separators instead of having them dynamically change based on the user's configuration). The reason for this, of course, is that `NSNumberFormatter` doesn't perform currency conversions, it just formats numeric data. You wouldn't want one user interpreting the value `"56324"` as US currency and another user who's accessing the same data interpreting it as Japanese currency, simply based on each user's localization preferences.

Special Considerations

This method is for use with formatters using `NSNumberFormatterBehavior10_0` behavior.

Availability

Available in Mac OS X v10.0 and later.

See Also

- `localizesFormat` (page 26)

Declared In

`NSNumberFormatter.h`

setMaximum:

Sets the highest number the receiver allows as input.

```
- (void)setMaximum:(NSNumber *)aMaximum
```

Parameters

aMaximum

A number object that specifies a maximum input value.

Discussion

If *aMaximum* is `nil`, checking for the maximum value is disabled. For versions prior to Mac OS X v10.4 (and number-formatter behavior set to `NSNumberFormatterBehavior10_0`) this method requires an `NSDecimalNumber` argument.

Availability

Available in Mac OS X v10.4 and later. Version requiring NSDecimalNumber argument available prior to Mac OS X v10.4.

See Also

- `maximum` (page 26)
- `+ setDefaultFormatterBehavior:` (page 16)
- `formatterBehavior` (page 21)
- `setFormatterBehavior:` (page 43)

Related Sample Code

Quartz Composer QCTV

Declared In

`NSNumberFormatter.h`

setMaximumFractionDigits:

Sets the maximum number of digits after the decimal separator allowed as input by the receiver.

```
- (void)setMaximumFractionDigits:(NSUInteger)number
```

Parameters

number

The maximum number of digits after the decimal separator allowed as input.

Availability

Available in Mac OS X v10.4 and later.

See Also

- `maximumFractionDigits` (page 27)

Related Sample Code

TrackBall

Declared In

`NSNumberFormatter.h`

setMaximumIntegerDigits:

Sets the maximum number of integer digits allowed as input by the receiver.

```
- (void)setMaximumIntegerDigits:(NSUInteger)number
```

Parameters

number

The maximum number of integer digits allowed as input.

Availability

Available in Mac OS X v10.4 and later.

See Also

- minimumIntegerDigits (page 29)

Declared In

NSNumberFormatter.h

setMaximumSignificantDigits:

Sets the maximum number of significant digits for the receiver.

```
- (void)setMaximumSignificantDigits:(NSUInteger)number
```

Parameters

number

The maximum number of significant digits for the receiver.

Availability

Available in Mac OS X v10.5 and later.

See Also

- maximumSignificantDigits (page 27)
- setMinimumSignificantDigits: (page 51)
- usesSignificantDigits (page 68)

Declared In

NSNumberFormatter.h

setMinimum:

Sets the lowest number the receiver allows as input.

```
- (void)setMinimum:(NSNumber *)aMinimum
```

Parameters

aMinimum

A number object that specifies a minimum input value.

Discussion

If *aMinimum* is nil, checking for the minimum value is disabled. For versions prior to Mac OS X v10.4 (and number-formatter behavior set to NSNumberFormatterBehavior10_0) this method requires an NSDecimalNumber argument.

Availability

Available in Mac OS X v10.4 and later. Version requiring NSDecimalNumber argument available prior to Mac OS X v10.4.

See Also
- `- minimum` (page 28)
- `+ setDefaultFormatterBehavior:` (page 16)
- `- formatterBehavior` (page 21)
- `- setFormatterBehavior:` (page 43)

Related Sample Code
Quartz Composer QCTV

Declared In
`NSNumberFormatter.h`

setMinimumFractionDigits:

Sets the minimum number of digits after the decimal separator allowed as input by the receiver.

```
- (void)setMinimumFractionDigits:(NSUInteger)number
```

Parameters

number
The minimum number of digits after the decimal separator allowed as input.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `- minimumFractionDigits` (page 28)

Related Sample Code
TrackBall

Declared In
`NSNumberFormatter.h`

setMinimumIntegerDigits:

Sets the minimum number of integer digits allowed as input by the receiver.

```
- (void)setMinimumIntegerDigits:(NSUInteger)number
```

Parameters

number
The minimum number of integer digits allowed as input.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `- minimumIntegerDigits` (page 29)

Related Sample Code
TrackBall

Declared In
NSNumberFormatter.h

setMinimumSignificantDigits:

Sets the minimum number of significant digits for the receiver.

```
- (void)setMinimumSignificantDigits:(NSUInteger)number
```

Parameters

number
The minimum number of significant digits for the receiver.

Availability
Available in Mac OS X v10.5 and later.

See Also
- minimumSignificantDigits (page 29)
- setMaximumSignificantDigits: (page 49)
- usesSignificantDigits (page 68)

Declared In
NSNumberFormatter.h

setMinusSign:

Sets the string used by the receiver for the minus sign.

```
- (void)setMinusSign:(NSString *)string
```

Parameters

string
A string that represents a minus sign.

Availability
Available in Mac OS X v10.4 and later.

See Also
- minusSign (page 29)

Declared In
NSNumberFormatter.h

setMultiplier:

Sets the multiplier of the receiver.

```
- (void)setMultiplier:(NSNumber *)number
```

Parameters

number
A number object that represents a multiplier.

Discussion
A multiplier is a factor used in conversions between numbers and strings (that is, numbers as stored and numbers as displayed). When the input value is a string, the multiplier is used to divide, and when the input value is a number, the multiplier is used to multiply. These operations allow the formatted values to be different from the values that a program manipulates internally.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `multiplier` (page 30)

Declared In
`NSNumberFormatter.h`

setNegativeFormat:

Sets the format the receiver uses to display negative values.

```
- (void)setNegativeFormat:(NSString *)aFormat
```

Parameters

aFormat
A string that specifies the format for negative values.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `negativeFormat` (page 30)
- `setFormat:` (page 42)

Declared In
`NSNumberFormatter.h`

setNegativeInfinitySymbol:

Sets the string used by the receiver for the negative infinity symbol.

```
- (void)setNegativeInfinitySymbol:(NSString *)string
```

Parameters

string
A string that represents a negative infinity symbol.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `negativeInfinitySymbol` (page 30)

Declared In
`NSNumberFormatter.h`

setNegativePrefix:

Sets the string the receiver uses as a prefix for negative values.

```
- (void)setNegativePrefix:(NSString *)string
```

Parameters

string

A string to use as the prefix for negative values.

Availability

Available in Mac OS X v10.4 and later.

See Also

- negativePrefix (page 31)

Declared In

NSNumberFormatter.h

setNegativeSuffix:

Sets the string the receiver uses as a suffix for negative values.

```
- (void)setNegativeSuffix:(NSString *)string
```

Parameters

string

A string to use as the suffix for negative values.

Availability

Available in Mac OS X v10.4 and later.

See Also

- negativeSuffix (page 31)

Declared In

NSNumberFormatter.h

setNilSymbol:

Sets the string the receiver uses to represent `nil` values.

```
- (void)setNilSymbol:(NSString *)string
```

Parameters

string

A string that represents a `nil` value.

Availability

Available in Mac OS X v10.4 and later.

See Also

- nilSymbol (page 32)

Declared In
NSNumberFormatter.h

setNotANumberSymbol:

Sets the string the receiver uses to represent NaN (“not a number”).

```
- (void)setNotANumberSymbol:(NSString *)string
```

Parameters

string

A string that represents a NaN symbol.

Availability
Available in Mac OS X v10.4 and later.

See Also
- notANumberSymbol (page 32)

Declared In
NSNumberFormatter.h

setNumberStyle:

Sets the number style used by the receiver.

```
- (void)setNumberStyle:(NSNumberFormatterStyle)style
```

Parameters

style

An NSNumberFormatterStyle constant that specifies a formatter style.

Discussion
Styles are essentially predetermined sets of values for certain properties. Examples of number-formatter styles are those used for decimal values, percentage values, and currency.

Availability
Available in Mac OS X v10.4 and later.

See Also
- numberStyle (page 33)

Related Sample Code
Grady
Mountains
NumberInput_IMKit_Sample
TrackBall

Declared In
NSNumberFormatter.h

setPaddingCharacter:

Sets the string that the receiver uses to pad numbers in the formatted string representation.

```
- (void)setPaddingCharacter:(NSString *)string
```

Parameters

string

A string containing a padding character (or characters).

Availability

Available in Mac OS X v10.4 and later.

See Also

- paddingCharacter (page 33)

Declared In

NSNumberFormatter.h

setPaddingPosition:

Sets the padding position used by the receiver.

```
- (void)setPaddingPosition:(NSNumberFormatterPadPosition)position
```

Parameters

position

An NSNumberFormatterPadPosition constant that indicates a padding position (before or after prefix or suffix).

Availability

Available in Mac OS X v10.4 and later.

See Also

- paddingPosition (page 33)

Declared In

NSNumberFormatter.h

setPartialStringValidationEnabled:

Sets whether partial string validation is enabled for the receiver.

```
- (void)setPartialStringValidationEnabled:(BOOL)b
```

Parameters

b

YES if partial string validation is enabled, otherwise NO.

Availability

Available in Mac OS X v10.5 and later.

See Also

- isPartialStringValidationEnabled (page 25)

Declared In
NSNumberFormatter.h

setPercentSymbol:

Sets the string used by the receiver to represent the percent symbol.

```
- (void)setPercentSymbol:(NSString *)string
```

Parameters

string
A string that represents a percent symbol.

Availability
Available in Mac OS X v10.4 and later.

See Also
- percentSymbol (page 34)

Declared In
NSNumberFormatter.h

setPerMillSymbol:

Sets the string used by the receiver to represent the per-mill (per-thousand) symbol.

```
- (void)setPerMillSymbol:(NSString *)string
```

Parameters

string
A string that represents a per-mill symbol.

Availability
Available in Mac OS X v10.4 and later.

See Also
- perMillSymbol (page 34)

Declared In
NSNumberFormatter.h

setPlusSign:

Sets the string used by the receiver to represent the plus sign.

```
- (void)setPlusSign:(NSString *)string
```

Parameters

string
A string that represents a plus sign.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `plusSign` (page 34)

Declared In
`NSNumberFormatter.h`

setPositiveFormat:

Sets the format the receiver uses to display positive values.

```
- (void)setPositiveFormat:(NSString *)aFormat
```

Parameters

aFormat
A string that specifies the format for positive values.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `positiveFormat` (page 35)
- `setFormat:` (page 42)

Declared In
`NSNumberFormatter.h`

setPositiveInfinitySymbol:

Sets the string used by the receiver for the positive infinity symbol.

```
- (void)setPositiveInfinitySymbol:(NSString *)string
```

Parameters

string
A string that represents a positive infinity symbol.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `positiveInfinitySymbol` (page 35)

Declared In
`NSNumberFormatter.h`

setPositivePrefix:

Sets the string the receiver uses as the prefix for positive values.

```
- (void)setPositivePrefix:(NSString *)string
```

Parameters

string

A string to use as the prefix for positive values.

Availability

Available in Mac OS X v10.4 and later.

See Also

- positivePrefix (page 35)

Declared In

NSNumberFormatter.h

setPositiveSuffix:

Sets the string the receiver uses as the suffix for positive values.

```
- (void)setPositiveSuffix:(NSString *)string
```

Parameters

string

A string to use as the suffix for positive values.

Availability

Available in Mac OS X v10.4 and later.

See Also

- positiveSuffix (page 36)

Declared In

NSNumberFormatter.h

setRoundingBehavior:

Sets the rounding behavior used by the receiver.

```
- (void)setRoundingBehavior:(NSDecimalNumberHandler *)newRoundingBehavior
```

Parameters

newRoundingBehavior

An NSDecimalNumberHandler object representing a rounding behavior.

Special Considerations

This method is for use with formatters using NSNumberFormatterBehavior10_0 behavior.

Availability

Available in Mac OS X v10.0 and later.

See Also

- roundingBehavior (page 36)

Declared In

NSNumberFormatter.h

setRoundingIncrement:

Sets the rounding increment used by the receiver.

```
- (void)setRoundingIncrement:(NSNumber *)number
```

Parameters

number

A number object specifying a rounding increment.

Availability
Available in Mac OS X v10.4 and later.

See Also
- roundingIncrement (page 36)

Declared In
NSNumberFormatter.h

setRoundingMode:

Sets the rounding mode used by the receiver.

```
- (void)setRoundingMode:(NSNumberFormatterRoundingMode)mode
```

Parameters

mode

An NSNumberFormatterRoundingMode constant that indicates a rounding mode.

Availability
Available in Mac OS X v10.4 and later.

See Also
- roundingMode (page 37)

Declared In
NSNumberFormatter.h

setSecondaryGroupingSize:

Sets the secondary grouping size of the receiver.

```
- (void)setSecondaryGroupingSize:(NSUInteger)number
```

Parameters

number

An integer that specifies the size of secondary groupings.

Discussion
Some locales allow the specification of another grouping size for larger numbers. For example, some locales may represent a number such as 61, 242, 378.46 (as in the United States) as 6,12,42,378.46. In this case, the secondary grouping size (covering the groups of digits furthest from the decimal point) is 2.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `secondaryGroupingSize` (page 37)

Declared In
`NSNumberFormatter.h`

setTextAttributesForNegativeInfinity:

Sets the text attributes used to display the negative infinity symbol.

```
- (void)setTextAttributesForNegativeInfinity:(NSDictionary *)newAttributes
```

Parameters

newAttributes

A dictionary containing text attributes for the display of the negative infinity symbol.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `textAttributesForNegativeInfinity` (page 65)
- `setNegativeInfinitySymbol:` (page 52)

Declared In
`NSNumberFormatter.h`

setTextAttributesForNegativeValues:

Sets the text attributes to be used in displaying negative values .

```
- (void)setTextAttributesForNegativeValues:(NSDictionary *)newAttributes
```

Parameters

newAttributes

A dictionary containing properties for the display of negative values.

Discussion
For example, this code excerpt causes negative values to be displayed in red:

```
NSNumberFormatter *numberFormatter =
    [[[NSNumberFormatter alloc] init] autorelease];
NSMutableDictionary *newAttrs = [NSMutableDictionary dictionary];

[numberFormatter setFormat:@"$#,##0.00;($#,##0.00)"];
[newAttrs setObject:[NSColor redColor] forKey:@"NSColor"];
[numberFormatter setTextAttributesForNegativeValues:newAttrs];
[[textField cell] setFormatter:numberFormatter];
```

An even simpler way to cause negative values to be displayed in red is to include the constant `[Red]` in your format string, as shown in this example:

```
[numberFormatter setFormat:@"$#,##0.00;[Red]($#,##0.00)"];
```

When you set a value's text attributes to use color, the color appears only when the value's cell doesn't have input focus. When the cell has input focus, the value is displayed in standard black.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `textAttributesForNegativeValues` (page 65)

Declared In
`NSNumberFormatter.h`

setTextAttributesForNil:

Sets the text attributes used to display the `nil` symbol.

```
- (void)setTextAttributesForNil:(NSDictionary *)newAttributes
```

Parameters

newAttributes
A dictionary containing text attributes for the display of the `nil` symbol.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `textAttributesForNil` (page 66)
- `nilSymbol` (page 32)

Declared In
`NSNumberFormatter.h`

setTextAttributesForNotANumber:

Sets the text attributes used to display the NaN ("not a number") string.

```
- (void)setTextAttributesForNotANumber:(NSDictionary *)newAttributes
```

Parameters

newAttributes
A dictionary containing text attributes for the display of the NaN symbol.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setTextAttributesForNotANumber:` (page 61)
- `notANumberSymbol` (page 32)

Declared In
`NSNumberFormatter.h`

setTextAttributesForPositiveInfinity:

Sets the text attributes used to display the positive infinity symbol.

```
- (void)setTextAttributesForPositiveInfinity:(NSDictionary *)newAttributes
```

Parameters

newAttributes

A dictionary containing text attributes for the display of the positive infinity symbol.

Availability

Available in Mac OS X v10.4 and later.

See Also

- positiveInfinitySymbol (page 35)
- textAttributesForPositiveInfinity (page 66)

Declared In

NSNumberFormatter.h

setTextAttributesForPositiveValues:

Sets the text attributes to be used in displaying positive values.

```
- (void)setTextAttributesForPositiveValues:(NSDictionary *)newAttributes
```

Parameters

newAttributes

A dictionary containing text attributes for the display of positive values.

Discussion

See setTextAttributesForNegativeValues: (page 60) for an example of how a related method might be used.

Availability

Available in Mac OS X v10.0 and later.

See Also

- textAttributesForPositiveValues (page 67)

Declared In

NSNumberFormatter.h

setTextAttributesForZero:

Sets the text attributes used to display a zero value.

```
- (void)setTextAttributesForZero:(NSDictionary *)newAttributes
```

Parameters

newAttributes

A dictionary containing text attributes for the display of zero values.

Availability

Available in Mac OS X v10.4 and later.

See Also
- `textAttributesForZero` (page 67)

Declared In
`NSNumberFormatter.h`

setThousandSeparator:

Sets the character the receiver uses as a thousand separator.

```
- (void)setThousandSeparator:(NSString *)newSeparator
```

Parameters

newSeparator

A string that specifies the thousand-separator character to use. If *newSeparator* contains multiple characters, only the first one is used.

Discussion
If you don't have thousand separators enabled through any other means (such as `setFormat:` (page 42)), using this method enables them.

Special Considerations

This method is for use with formatters using `NSNumberFormatterBehavior10_0` behavior.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `thousandSeparator` (page 67)

Declared In
`NSNumberFormatter.h`

setUsesGroupingSeparator:

Controls whether the receiver displays the grouping separator.

```
- (void)setUsesGroupingSeparator:(BOOL)flag
```

Parameters

flag

`YES` if the receiver should display the grouping separator, `NO` otherwise.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `usesGroupingSeparator` (page 68)

Declared In
`NSNumberFormatter.h`

setUsesSignificantDigits:

Sets whether the receiver uses significant digits.

```
- (void)setUsesSignificantDigits:(BOOL)b
```

Parameters

b

`YES` if the receiver uses significant digits, otherwise `NO`.

Availability
Available in Mac OS X v10.5 and later.

See Also
- `usesSignificantDigits` (page 68)
- `setMaximumSignificantDigits:` (page 49)
- `setMinimumSignificantDigits:` (page 51)

Declared In
`NSNumberFormatter.h`

setZeroSymbol:

Sets the string the receiver uses as the symbol to show the value zero.

```
- (void)setZeroSymbol:(NSString *)string
```

Parameters

string

The string the receiver uses as the symbol to show the value zero.

Discussion
By default this is `0`; you might want to set it to, for example, " - ", similar to the way that a spreadsheet might when a column is defined as accounting.

Special Considerations

On Mac OS X v10.4, this method works correctly for 10_0-style number formatters but does not work correctly for 10_4-style number formatters. You can work around the problem by subclassing and overriding the methods that convert between strings and numbers to look for the zero cases first and provide different behavior, invoking `super` when not zero.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `zeroSymbol` (page 69)

Declared In
`NSNumberFormatter.h`

stringFromNumber:

Returns a string containing the formatted value of the provided number object.

```
- (NSString *)stringFromNumber:(NSNumber *)number
```

Parameters

number

An `NSNumber` object that is parsed to create the returned string object.

Return Value

A string containing the formatted value of *number* using the receiver's current settings.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- numberFromString:` (page 32)

Related Sample Code

Mountains

NumberInput_IMKit_Sample

Declared In

`NSNumberFormatter.h`

textAttributesForNegativeInfinity

Returns a dictionary containing the text attributes used to display the negative infinity string.

```
- (NSDictionary *)textAttributesForNegativeInfinity
```

Return Value

A dictionary containing the text attributes used to display the negative infinity string.

Availability

Available in Mac OS X v10.4 and later.

See Also

`- setTextAttributesForNegativeInfinity:` (page 60)

Declared In

`NSNumberFormatter.h`

textAttributesForNegativeValues

Returns a dictionary containing the text attributes that have been set for negative values.

```
- (NSDictionary *)textAttributesForNegativeValues
```

Return Value

A dictionary containing the text attributes that have been set for negative values.

Availability

Available in Mac OS X v10.0 and later.

See Also

`- setTextAttributesForNegativeValues:` (page 60)

Declared In
NSNumberFormatter.h

textAttributesForNil

Returns a dictionary containing the text attributes used to display the `nil` symbol.

```
- (NSDictionary *)textAttributesForNil
```

Return Value
A dictionary containing the text attributes used to display the `nil` symbol.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setTextAttributesForNil:` (page 61)

Declared In
NSNumberFormatter.h

textAttributesForNotANumber

Returns a dictionary containing the text attributes used to display the NaN ("not a number") symbol.

```
- (NSDictionary *)textAttributesForNotANumber
```

Return Value
A dictionary containing the text attributes used to display the NaN ("not a number") symbol.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setTextAttributesForNotANumber:` (page 61)
- `notANumberSymbol` (page 32)

Declared In
NSNumberFormatter.h

textAttributesForPositiveInfinity

Returns a dictionary containing the text attributes used to display the positive infinity symbol.

```
- (NSDictionary *)textAttributesForPositiveInfinity
```

Return Value
A dictionary containing the text attributes used to display the positive infinity symbol.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setTextAttributesForPositiveInfinity:` (page 62)
- `positiveInfinitySymbol` (page 35)

Declared In
`NSNumberFormatter.h`

textAttributesForPositiveValues

Returns a dictionary containing the text attributes that have been set for positive values.

```
- (NSDictionary *)textAttributesForPositiveValues
```

Return Value
A dictionary containing the text attributes that have been set for positive values.

Availability
Available in Mac OS X v10.0 and later.

See Also
- `setTextAttributesForPositiveValues:` (page 62)

Declared In
`NSNumberFormatter.h`

textAttributesForZero

Returns a dictionary containing the text attributes used to display a value of zero.

```
- (NSDictionary *)textAttributesForZero
```

Return Value
A dictionary containing the text attributes used to display a value of zero.

Availability
Available in Mac OS X v10.4 and later.

See Also
- `setTextAttributesForZero:` (page 62)

Declared In
`NSNumberFormatter.h`

thousandSeparator

Returns a string containing the character the receiver uses to represent thousand separators.

```
- (NSString *)thousandSeparator
```

Return Value
A string containing the character the receiver uses to represent thousand separators.

Discussion
By default this is the comma character (,). Note that the return value doesn't indicate whether thousand separators are enabled.

Special Considerations

This method is for use with formatters using `NSNumberFormatterBehavior10_0` behavior.

Availability
Available in Mac OS X v10.0 and later.

See Also
`- setThousandSeparator:` (page 63)

Declared In
`NSNumberFormatter.h`

usesGroupingSeparator

Returns a Boolean value that indicates whether the receiver uses the grouping separator.

```
- (BOOL)usesGroupingSeparator
```

Return Value
`YES` if the receiver uses the grouping separator, otherwise `NO`.

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setUsesGroupingSeparator:` (page 63)

Declared In
`NSNumberFormatter.h`

usesSignificantDigits

Returns a Boolean value that indicates whether the receiver uses significant digits.

```
- (BOOL)usesSignificantDigits
```

Return Value
`YES` if the receiver uses significant digits, otherwise `NO`.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- setUsesSignificantDigits:` (page 64)
`- maximumSignificantDigits` (page 27)
`- minimumSignificantDigits` (page 29)

Declared In
`NSNumberFormatter.h`

zeroSymbol

Returns the string the receiver uses as the symbol to show the value zero.

```
- (NSString *)zeroSymbol
```

Return Value
The string the receiver uses as the symbol to show the value zero.

Discussion
For a discussion of how this is used, see `setZeroSymbol:` (page 64).

Availability
Available in Mac OS X v10.4 and later.

See Also
`- setZeroSymbol:` (page 64)

Declared In
`NSNumberFormatter.h`

Constants

NSNumberFormatterStyle

These constants specify predefined number format styles.

```
typedef enum {
    NSNumberFormatterNoStyle = kCFNumberFormatterNoStyle,
    NSNumberFormatterDecimalStyle = kCFNumberFormatterDecimalStyle,
    NSNumberFormatterCurrencyStyle = kCFNumberFormatterCurrencyStyle,
    NSNumberFormatterPercentStyle = kCFNumberFormatterPercentStyle,
    NSNumberFormatterScientificStyle = kCFNumberFormatterScientificStyle,
    NSNumberFormatterSpellOutStyle = kCFNumberFormatterSpellOutStyle
} NSNumberFormatterStyle;
```

Constants

`NSNumberFormatterNoStyle`
Specifies no style.
Available in Mac OS X v10.4 and later.
Declared in `NSNumberFormatter.h`.

`NSNumberFormatterDecimalStyle`
Specifies a decimal style format.
Available in Mac OS X v10.4 and later.
Declared in `NSNumberFormatter.h`.

`NSNumberFormatterCurrencyStyle`
Specifies a currency style format.
Available in Mac OS X v10.4 and later.
Declared in `NSNumberFormatter.h`.

`NSNumberFormatterPercentStyle`
Specifies a percent style format.

Available in Mac OS X v10.4 and later.

Declared in `NSNumberFormatter.h`.

`NSNumberFormatterScientificStyle`
Specifies a scientific style format.

Available in Mac OS X v10.4 and later.

Declared in `NSNumberFormatter.h`.

`NSNumberFormatterSpellOutStyle`
Specifies a spell-out format; for example, “23” becomes “twenty-three”.

Available in Mac OS X v10.4 and later.

Declared in `NSNumberFormatter.h`.

Discussion
These constants are used by the `numberStyle` (page 33) and `setNumberStyle:` (page 54) methods.

Declared In
`NSNumberFormatter.h`

NSNumberFormatterBehavior

These constants specify the behavior of a number formatter.

```
typedef enum {
    NSNumberFormatterBehaviorDefault = 0,
    NSNumberFormatterBehavior10_0 = 1000,
    NSNumberFormatterBehavior10_4 = 1040,
} NSNumberFormatterBehavior;
```

Constants
`NSNumberFormatterBehaviorDefault`
The number-formatter behavior set as the default for new instances. You can set the default formatter behavior with the class method `setDefaultFormatterBehavior:` (page 16).

Available in Mac OS X v10.4 and later.

Declared in `NSNumberFormatter.h`.

`NSNumberFormatterBehavior10_0`
The number-formatter behavior as it existed prior to Mac OS X v10.4.

Available in Mac OS X v10.4 and later.

Declared in `NSNumberFormatter.h`.

`NSNumberFormatterBehavior10_4`
The number-formatter behavior since Mac OS X v10.4.

Available in Mac OS X v10.4 and later.

Declared in `NSNumberFormatter.h`.

Discussion
These constants are returned by the `defaultFormatterBehavior` (page 15) class method and the `formatterBehavior` (page 21) instance methods; you set them with the `setDefaultFormatterBehavior:` (page 16) class method and the `setFormatterBehavior:` (page 43) instance method.

Declared In
NSNumberFormatter.h

NSNumberFormatterPadPosition

These constants are used to specify how numbers should be padded.

```
typedef enum {
    NSNumberFormatterPadBeforePrefix = kCFNumberFormatterPadBeforePrefix,
    NSNumberFormatterPadAfterPrefix = kCFNumberFormatterPadAfterPrefix,
    NSNumberFormatterPadBeforeSuffix = kCFNumberFormatterPadBeforeSuffix,
    NSNumberFormatterPadAfterSuffix = kCFNumberFormatterPadAfterSuffix
} NSNumberFormatterPadPosition;
```

Constants

`NSNumberFormatterPadBeforePrefix`
Specifies that the padding should occur before the prefix.
Available in Mac OS X v10.4 and later.
Declared in `NSNumberFormatter.h`.

`NSNumberFormatterPadAfterPrefix`
Specifies that the padding should occur after the prefix.
Available in Mac OS X v10.4 and later.
Declared in `NSNumberFormatter.h`.

`NSNumberFormatterPadBeforeSuffix`
Specifies that the padding should occur before the suffix.
Available in Mac OS X v10.4 and later.
Declared in `NSNumberFormatter.h`.

`NSNumberFormatterPadAfterSuffix`
Specifies that the padding should occur after the suffix.
Available in Mac OS X v10.4 and later.
Declared in `NSNumberFormatter.h`.

Discussion
These constants are used by the `paddingPosition` (page 33) and `setPaddingPosition:` (page 55) methods.

Declared In
NSNumberFormatter.h

NSNumberFormatterRoundingMode

These constants are used to specify how numbers should be rounded.

```
typedef enum {
    NSNumberFormatterRoundCeiling = kCFNumberFormatterRoundCeiling,
    NSNumberFormatterRoundFloor = kCFNumberFormatterRoundFloor,
    NSNumberFormatterRoundDown = kCFNumberFormatterRoundDown,
    NSNumberFormatterRoundUp = kCFNumberFormatterRoundUp,
    NSNumberFormatterRoundHalfEven = kCFNumberFormatterRoundHalfEven,
    NSNumberFormatterRoundHalfDown = kCFNumberFormatterRoundHalfDown,
    NSNumberFormatterRoundHalfUp = kCFNumberFormatterRoundHalfUp
} NSNumberFormatterRoundingMode;
```

Constants

`NSNumberFormatterRoundCeiling`

Round up to next larger number with the proper number of digits after the decimal separator.

Available in Mac OS X v10.4 and later.

Declared in `NSNumberFormatter.h`.

`NSNumberFormatterRoundFloor`

Round down to next smaller number with the proper number of digits after the decimal separator.

Available in Mac OS X v10.4 and later.

Declared in `NSNumberFormatter.h`.

`NSNumberFormatterRoundDown`

Round down to next smaller number with the proper number of digits after the decimal separator.

Available in Mac OS X v10.4 and later.

Declared in `NSNumberFormatter.h`.

`NSNumberFormatterRoundHalfEven`

Round the last digit, when followed by a 5, toward an even digit (.25 -> .2, .35 -> .4)

Available in Mac OS X v10.4 and later.

Declared in `NSNumberFormatter.h`.

`NSNumberFormatterRoundUp`

Round up to next larger number with the proper number of digits after the decimal separator.

Available in Mac OS X v10.4 and later.

Declared in `NSNumberFormatter.h`.

`NSNumberFormatterRoundHalfDown`

Round down when a 5 follows putative last digit.

Available in Mac OS X v10.4 and later.

Declared in `NSNumberFormatter.h`.

`NSNumberFormatterRoundHalfUp`

Round up when a 5 follows putative last digit.

Available in Mac OS X v10.4 and later.

Declared in `NSNumberFormatter.h`.

Declared In

`NSNumberFormatter.h`

These constants are used by the `roundingMode` (page 37)and `setRoundingMode:` (page 59) methods.

Document Revision History

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

Date	Notes
2008-11-19	Added note about supported behaviors for iPhone OS.
2008-10-15	Revised descriptions of isLenient and setLenient: methods.
2007-10-31	Clarified the use of the setZeroSymbol: method.
2007-02-26	Updated to include new API in Mac OS X v10.5.
2007-01-08	Corrected minor typographical errors.
2006-05-23	Corrected declarations of enum types.
	First publication of this content as a separate document.

Document Revision History

Index

N

P

R

S

T

U

Z