
NSDate Class Reference

(Not Recommended)





Apple Inc.
© 2009 Apple Inc.
All rights reserved.

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

The Apple logo is a trademark of Apple Inc.

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

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

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

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

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

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION, EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY

DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

THE WARRANTY AND REMEDIES SET FORTH ABOVE ARE EXCLUSIVE AND IN LIEU OF ALL OTHERS, ORAL OR WRITTEN, EXPRESS OR IMPLIED. No Apple dealer, agent, or employee is authorized to make any modification, extension, or addition to this warranty.

Some states do not allow the exclusion or limitation of implied warranties or liability for incidental or consequential damages, so the above limitation or exclusion may not apply to you. This warranty gives you specific legal rights, and you may also have other rights which vary from state to state.

Contents

NSDate Class Reference (Not Recommended) 5

Overview	5
The Calendar Format	6
Locales and String Representations of Calendar Dates	6
Subclassing Notes	7
Tasks	7
Creating an NSDate Instance	7
Initializing an NSDate Instance	7
Retrieving Date Elements	7
Adjusting a Date	8
Computing Date Intervals	8
Representing Dates as Strings	8
Getting and Setting Calendar Formats	9
Managing the Time Zone	9
Class Methods	9
calendarDate	9
dateWithString:calendarFormat:	9
dateWithString:calendarFormat:locale:	10
dateWithYear:month:day:hour:minute:second:timeZone:	11
Instance Methods	12
calendarFormat	12
dateByAddingYears:months:days:hours:minutes:seconds:	12
dayOfCommonEra	13
dayOfMonth	14
dayOfWeek	14
dayOfYear	15
description	15
descriptionWithCalendarFormat:	16
descriptionWithCalendarFormat:locale:	16
descriptionWithLocale:	17
hourOfDay	18
initWithString:	18
initWithString:calendarFormat:	19
initWithString:calendarFormat:locale:	19
initWithYear:month:day:hour:minute:second:timeZone:	20
minuteOfHour	21
monthOfYear	21
secondOfMinute	22
setCalendarFormat:	22
setTimeZone:	23
timeZone	23

yearOfCommonEra 24

years:months:days:hours:minutes:seconds:sinceDate: 24

Document Revision History 27

Index 29

NSDate Class Reference (Not Recommended)

Inherits from	NSDate : NSObject
Conforms to	NSCoding (NSDate) NSCopying (NSDate) NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	NSDate.h
Companion guides	Date and Time Programming Guide for Cocoa Data Formatting Programming Guide for Cocoa

Important: NSDate is deprecated in Mac OS X v10.6 and later. For calendrical calculations, you should use suitable combinations of NSDate, NSDateComponents, as described in Calendars, Date Components, and Calendar Units.

Overview

NSDate is a public subclass of NSDate that represents concrete date objects and performs date computations based on the Gregorian calendar. These objects associate a time interval with a time zone and are especially suited for representing and manipulating dates according to western calendrical systems.

An NSDate object stores a date as the number of seconds relative to the absolute reference date (the first instance of 1 January 2001, GMT). Use the associated time zone to change how the NSDate object prints its time interval. The time zone does not change how the time interval is stored. Because the value is stored independently of the time zone, you can accurately compare NSDate objects with any other NSDate objects or use them to create other NSDate objects. It also means that you can track a date across different time zones; that is, you can create a new NSDate object with a different time zone to see how the particular date is represented in that time zone.

Important: NSDate uses the Gregorian calendar for all of time, even before it was actually adopted. NSDate's version of the Gregorian calendar uses the Julian calendar before October 4, 1582. If you need to accurately deal with dates prior to October 4, 1582, you should use NSDate.

NSDate provides both class and instance methods for creating objects. Some of these methods allow you to initialize NSDate objects from strings, while others create objects from sets of integers corresponding to the standard time values (months, hours, seconds, and so on).

To retrieve conventional elements of an `NSCalendarDate` object, use the `...Of...` methods. For example, `dayOfWeek` (page 14) returns a number that indicates the day of the week (0 is Sunday). The `monthOfYear` (page 21) method returns a number from 1 through 12 that indicates the month.

The Calendar Format

Each `NSCalendarDate` object has a calendar format associated with it. This format is a string that contains date conversion specifiers that are very similar to those used in the standard C library function `strftime()`. `NSCalendarDate` interprets dates that are represented as strings conforming to this format. You can set the default format for an `NSCalendarDate` object at initialization time or using the `setCalendarFormat:` (page 22) method. Several methods allow you to specify formats other than the one bound to the object.

The date conversion specifiers cover a range of date conventions. See *Converting Dates to Strings* in *Date and Time Programming Guide for Cocoa* for the list of specifiers.

Locales and String Representations of Calendar Dates

`NSCalendarDate` provides several `description...` methods for representing dates as strings. These methods—`description` (page 15), `descriptionWithLocale:` (page 17), `descriptionWithCalendarFormat:` (page 16), and `descriptionWithCalendarFormat:locale:` (page 16)—take an implicit or explicit calendar format. The locale information affects the returned string. If you use `descriptionWithLocale:` or `descriptionWithCalendarFormat:locale:`, you may specify a locale dictionary. `NSCalendarDate` accesses the locale information as an `NSDictionary` object. The following keys in the locale dictionary affect `NSCalendarDate`:

<code>NSTimeDateFormatString</code>	
<code>NSAMPMDesignation</code>	
<code>NSMonthNameArray</code>	
<code>NSShortMonthNameArray</code>	
<code>NSWeekDayNameArray</code>	
<code>NSShortWeekDayNameArray</code>	

If a `description...` method does not have a locale parameter or if you pass `nil` as the locale to a method that takes a locale argument, `NSCalendarDate` uses the system default locale. The default locale—sometimes called the "root" locale—is a generic English-like locale. Typically you should instead use the user's preferences. You can obtain a dictionary representation of the user's standard user defaults using the `NSUserDefaults` method `dictionaryRepresentation`, as illustrated in the following example:

```
NSCalendarDate *calendarDate = [[NSCalendarDate alloc]
initWithTimeIntervalSinceReferenceDate:uploadedTime];
[calendarDate descriptionWithLocale:[NSUserDefaults standardUserDefaults]
dictionaryRepresentation]];
// ...
[calendarDate release];
```

Subclassing Notes

If you subclass `NSCalendarDate` and override `description` (page 15), you should also override `descriptionWithLocale:` (page 17). The `stringWithFormat:` method of `NSString` uses `descriptionWithLocale:` (page 17) instead of `description` when you use the `%@` conversion specifier to get a string representation of an `NSCalendarDate` object. That is, this message:

```
[NSString stringWithFormat:@"The current date and time are %@",
    [MyNSCalendarDateSubclass date]]
```

invokes `descriptionWithLocale:` (page 17).

Tasks

Creating an NSCalendarDate Instance

- + `calendarDate` (page 9)
Creates and returns a calendar date initialized to the current date and time.
- + `dateWithString:calendarFormat:` (page 9)
Creates and returns a calendar date initialized with the date given as a string in a specified format.
- + `dateWithString:calendarFormat:locale:` (page 10)
Creates and returns a calendar date initialized with the date given as a string in a specified format and interpreted using a given locale.
- + `dateWithYear:month:day:hour:minute:second:timeZone:` (page 11)
Creates and returns a calendar date initialized with specified values for year, month, day, hour, minute, second, and time zone.

Initializing an NSCalendarDate Instance

- `initWithString:` (page 18)
Returns a calendar date initialized with the date specified as a string in the default calendar format.
- `initWithString:calendarFormat:` (page 19)
Returns a calendar date initialized with the date given as a string in a specified format.
- `initWithString:calendarFormat:locale:` (page 19)
Returns a calendar date initialized with the date given as a string in a specified format and interpreted using a given locale.
- `initWithYear:month:day:hour:minute:second:timeZone:` (page 20)
Returns a calendar date initialized with specified values for year, month, day, hour, minute, second, and time zone.

Retrieving Date Elements

- `dayOfCommonEra` (page 13)
Returns the number of days between the receiver and the beginning of the Common Era.

- [dayOfMonth](#) (page 14)
Returns the day of the month (1 through 31) of the receiver.
- [dayOfWeek](#) (page 14)
Returns the day of the week (0 through 6) of the receiver.
- [dayOfYear](#) (page 15)
Returns the day of the year (1 through 366) of the receiver.
- [hourOfDay](#) (page 18)
Returns the hour (0 through 23) of the receiver.
- [minuteOfHour](#) (page 21)
Returns the minute (0 through 59) of the receiver.
- [monthOfYear](#) (page 21)
Returns the month of the year (1 through 12) of the receiver.
- [secondOfMinute](#) (page 22)
Returns the second (0 through 59) of the receiver.
- [yearOfCommonEra](#) (page 24)
Returns the year, including the century, of the receiver.

Adjusting a Date

- [dateByAddingYears:months:days:hours:minutes:seconds:](#) (page 12)
Returns a new calendar date that represents the date of the receiver updated with given offsets.

Computing Date Intervals

- [years:months:days:hours:minutes:seconds:sinceDate:](#) (page 24)
Computes the calendrical time difference between the receiver and a given date.

Representing Dates as Strings

- [description](#) (page 15)
Returns a string representation of the receiver formatted as specified by the receiver's default calendar format.
- [descriptionWithCalendarFormat:](#) (page 16)
Returns a string representation of the receiver.
- [descriptionWithCalendarFormat:locale:](#) (page 16)
Returns a string representation of the receiver formatted according to given conversion specifiers and represented according to given locale information.
- [descriptionWithLocale:](#) (page 17)
Returns a string representation of the receiver formatted as specified by the receiver's default calendar format and represented according to the given locale information.

Getting and Setting Calendar Formats

- [calendarFormat](#) (page 12)
Returns the receiver's default calendar format.
- [setCalendarFormat:](#) (page 22)
Sets the default calendar format for the receiver.

Managing the Time Zone

- [setTimeZone:](#) (page 23)
Sets the time zone for the receiver.
- [timeZone](#) (page 23)
Returns the time zone object associated with the receiver.

Class Methods

calendarDate

Creates and returns a calendar date initialized to the current date and time.

```
+ (id)calendarDate
```

Return Value

A new calendar date initialized to the current date and time.

Availability

Available in Mac OS X v10.0 and later.

See Also

```
+ date (NSDate)
```

Declared In

```
NSCalendarDate.h
```

dateWithString:calendarFormat:

Creates and returns a calendar date initialized with the date given as a string in a specified format.

```
+ (id)dateWithString:(NSString *)description calendarFormat:(NSString *)format
```

Parameters

description

A string containing a description of a date in the format specified by *format*.

format

A string used to interpret *description* and as the default calendar format for the new object. *format* consists of conversion specifiers similar to those used in `strftime()`. See *Converting Dates to Strings*, in *Date and Time Programming Guide for Cocoa* for more details.

Return Value

A new calendar date initialized with the date specified in *description*. Returns *nil* if *description* does not match *format* exactly.

Discussion

The following example shows how to get a calendar date with a temporal value corresponding to the form “Friday, 1 July 2001, 11:45 AM.”:

```
NSCalendarDate *today = [NSCalendarDate
    dateWithString:@"Friday, 1 July 2001, 11:45 AM"
    calendarFormat:@"%A, %d %B %Y, %I:%M %p"];
```

If you include a time zone in the *description* parameter, this method verifies it and can substitute an alternative time zone. If the method does supply a new time zone, it applies the difference in offsets-from-GMT values between the substituted and the original time zones to the calendar date being created.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [dateWithString:calendarFormat:locale:](#) (page 10)

- [calendarFormat](#) (page 12)

- [initWithString:calendarFormat:](#) (page 19)

Declared In

NSCalendarDate.h

dateWithString:calendarFormat:locale:

Creates and returns a calendar date initialized with the date given as a string in a specified format and interpreted using a given locale.

```
+ (id)dateWithString:(NSString *)description calendarFormat:(NSString *)format
    locale:(id)localeDictionary
```

Parameters

description

A string containing a description of a date in the format specified by *format*.

format

A string used to interpret *description* and as the default calendar format for the new object. *format* consists of conversion specifiers similar to those used in `strftime()`. See [Converting Dates to Strings](#), in *Date and Time Programming Guide for Cocoa* for more details.

localeDictionary

A dictionary that contains keys and values to represent the locale data to use when parsing *description*. See [“Locales and String Representations of Calendar Dates”](#) (page 6) for a list of the appropriate keys.

Return Value

A new calendar date initialized with the date specified by *description* and interpreted using the locale data in *localeDictionary*. Returns *nil* if *description* does not exactly match *format*.

Availability

Available in Mac OS X v10.0 and later.

See Also

- + [dateWithString:calendarFormat:](#) (page 9)
- [calendarFormat](#) (page 12)
- [initWithString:calendarFormat:locale:](#) (page 19)

Declared In

NSCalendarDate.h

dateWithYear:month:day:hour:minute:second:timeZone:

Creates and returns a calendar date initialized with specified values for year, month, day, hour, minute, second, and time zone.

```
+ (id)dateWithYear:(NSInteger)year month:(NSUInteger)month day:(NSInteger)day
    hour:(NSUInteger)hour minute:(NSUInteger)minute second:(NSInteger)second
    timeZone:(NSTimeZone *)aTimeZone
```

Parameters

year

The year for the new date. The value must include the century (for example, 1999 instead of 99).

month

The month for the new date. Valid values are 1 through 12.

day

The day for the new date. Valid values are 1 through 31.

hour

The hour for the new date. Valid values are 0 through 23.

minute

The minute for the new date. Valid values are 0 through 59.

second

The second for the new date. Valid values are 0 through 59.

aTimeZone

The time zone for the new date.

Return Value

A new calendar date initialized with the specified values for year, month, day, hour, minute, second, and time zone.

Discussion

On days when daylight savings time “falls back,” there are two 1:30 AMs. If you use this method, there is no way to create the *second* 1:30 AM. Instead, you should create the first and then use [dateByAddingYears:months:days:hours:minutes:seconds:](#) (page 12) to add an hour.

The following code fragment shows a calendar date created for 4 July 2001, 9 PM, Eastern Standard Time (`timeZoneWithName:` returns the `NSTimeZone` object that represents the time zone with the specified name):

```
NSCalendarDate *fireworks = [NSCalendarDate dateWithYear:2001
    month:7 day:4 hour:21 minute:0 second:0
    timeZone:[NSTimeZone timeZoneWithAbbreviation:@"EST"]];
```

Availability

Available in Mac OS X v10.0 and later.

See Also

- [initWithYear:month:day:hour:minute:second:timeZone:](#) (page 20)

Declared In

NSCalendarDate.h

Instance Methods

calendarFormat

Returns the receiver's default calendar format.

```
- (NSString *)calendarFormat
```

Return Value

The receiver's default calendar format (used when the format is unspecified).

Discussion

You can set this format when you create the calendar date using one of the class methods [dateWithString:calendarFormat:](#) (page 9) or [dateWithString:calendarFormat:locale:](#) (page 10), or you can change the format using the instance method [setCalendarFormat:](#) (page 22). If you do not specify a default calendar format, `NSCalendarDate` substitutes its own default: an international format of “%Y-%m-%d %H:%M:%S %Z” (for example, 2001-03-24 16:45:12 +0900). See *Converting Dates to Strings*, in *Date and Time Programming Guide for Cocoa* for more information on what a calendar format contains.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [descriptionWithLocale:](#) (page 17)

Declared In

NSCalendarDate.h

dateByAddingYears:months:days:hours:minutes:seconds:

Returns a new calendar date that represents the date of the receiver updated with given offsets.

```
- (NSCalendarDate *)dateByAddingYears:(NSInteger)year months:(NSInteger)month
    days:(NSInteger)day hours:(NSInteger)hour minutes:(NSInteger)minute
    seconds:(NSInteger)second
```

Parameters

year

The number of years to add to the receiver. The value may be negative to indicate a time in the past.

month

The number of months to add to the receiver. The value may be negative to indicate a time in the past.

day

The number of days to add to the receiver. The value may be negative to indicate a time in the past.

hour

The number of hours to add to the receiver. The value may be negative to indicate a time in the past.

minute

The number of minutes to add to the receiver. The value may be negative to indicate a time in the past.

second

The number of seconds to add to the receiver. The value may be negative to indicate a time in the past.

Return Value

A new calendar date that represents the date of the receiver updated with the year, month, day, hour, minute, and second offsets specified in the parameters.

Discussion

The parameter values are applied in a left-to-right order: *year* first, then *month*, then *day*, and so on. So, adding one month, four days to 27 April results in 31 May, not 1 June.

This method preserves “clock time” across changes in daylight saving time zones and leap years. If you add one day to 2:30 AM on the day before daylight saving time “springs ahead,” it will actually result in 1:30 AM on the next day (which is one day, or 24 hours, later).

The following code fragment shows a calendar date created with a date a week later than an existing calendar date:

```
NSDate *now = [NSDate calendarDate];
NSDate *nextWeek = [now dateByAddingYears:0 months:0
                    days:7 hours:0 minutes:0 seconds:0];
```

Availability

Available in Mac OS X v10.0 and later.

See Also

- [years:months:days:hours:minutes:seconds:sinceDate:](#) (page 24)

Declared In

NSDate.h

dayOfCommonEra

Returns the number of days between the receiver and the beginning of the Common Era.

- (NSInteger)dayOfCommonEra

Return Value

The number of days between the receiver and the beginning of the Common Era.

Discussion

The base year of the Common Era is 1 C.E. (which is the same as 1 A.D.).

Availability

Available in Mac OS X v10.0 and later.

See Also

- [yearOfCommonEra](#) (page 24)

Declared In

NSDate.h

dayOfMonth

Returns the day of the month (1 through 31) of the receiver.

- (NSInteger)dayOfMonth

Return Value

The day of the month (1 through 31) of the receiver.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [dayOfWeek](#) (page 14)
- [dayOfYear](#) (page 15)
- [hourOfDay](#) (page 18)
- [minuteOfHour](#) (page 21)
- [monthOfYear](#) (page 21)
- [secondOfMinute](#) (page 22)

Declared In

NSDate.h

dayOfWeek

Returns the day of the week (0 through 6) of the receiver.

- (NSInteger)dayOfWeek

Return Value

The day of the week (0 through 6) of the receiver. 0 indicates Sunday.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [dayOfMonth](#) (page 14)
- [dayOfYear](#) (page 15)
- [hourOfDay](#) (page 18)
- [minuteOfHour](#) (page 21)
- [monthOfYear](#) (page 21)
- [secondOfMinute](#) (page 22)

Declared In

NSCalendarDate.h

dayOfYear

Returns the day of the year (1 through 366) of the receiver.

- (NSInteger)dayOfYear

Return Value

The day of the year (1 through 366) of the receiver.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [dayOfMonth](#) (page 14)
- [dayOfWeek](#) (page 14)
- [hourOfDay](#) (page 18)
- [minuteOfHour](#) (page 21)
- [monthOfYear](#) (page 21)
- [secondOfMinute](#) (page 22)

Declared In

NSCalendarDate.h

description

Returns a string representation of the receiver formatted as specified by the receiver's default calendar format.

- (NSString *)description

Return Value

A string representation of the receiver, formatted as specified by the receiver's default calendar format.

Discussion

You can find out what the default calendar format is using the method [calendarFormat](#) (page 12). See ["Locales and String Representations of Calendar Dates"](#) (page 6) for information on locales and this method.

Because `NSDate` implements [descriptionWithLocale:](#) (page 17), `descriptionWithLocale:` is used to print the date when you use the `%@` conversion specifier. That is, the following statement invokes `descriptionWithLocale:`, not `description`:

```
NSLog(@"The current date and time is %@", [NSDate date]);
```

Availability

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

See Also

- [descriptionWithCalendarFormat:](#) (page 16)
- [descriptionWithCalendarFormat:locale:](#) (page 16)
- [descriptionWithLocale:](#) (page 17)

- [setDescriptionWithCalendarFormat:](#) (page 22)

Declared In

NSCalendarDate.h

descriptionWithCalendarFormat:

Returns a string representation of the receiver.

```
- (NSString *)descriptionWithCalendarFormat:(NSString *)format
```

Parameters

format

The format for the description. See [Converting Dates to Strings](#), in *Date and Time Programming Guide for Cocoa* for a listing of specifiers.

Return Value

A string representation of the receiver, formatted as specified by the conversion specifiers in the calendar format string *format*.

Discussion

See [“Locales and String Representations of Calendar Dates”](#) (page 6) for information on locales and this method.

The following example shows how to create a description of the current date in the same format as “Tues 3/24/01 3:30 PM”:

```
NSDate *now = [NSDate date];
NSString *nowAsString =
    [now descriptionWithCalendarFormat:@"%a %m/%d/%y %I:%M %p"];
```

Availability

Available in Mac OS X v10.0 and later.

See Also

- [description](#) (page 15)
- [descriptionWithCalendarFormat:locale:](#) (page 16)
- [descriptionWithLocale:](#) (page 17)

Declared In

NSCalendarDate.h

descriptionWithCalendarFormat:locale:

Returns a string representation of the receiver formatted according to given conversion specifiers and represented according to given locale information.

```
- (NSString *)descriptionWithCalendarFormat:(NSString *)format
    locale:(id)localeDictionary
```

Parameters*format*

The format for the description. See *Converting Dates to Strings*, in *Date and Time Programming Guide for Cocoa* for a list of specifiers.

localeDictionary

A dictionary that contains keys and values to represent the locale data to use when creating the description. See [“Locales and String Representations of Calendar Dates”](#) (page 6) for further details.

Return Value

A string representation of the receiver, formatted according to the conversion specifiers in *format* and represented according to the locale information in *localeDictionary*.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [description](#) (page 15)
- [descriptionWithCalendarFormat:](#) (page 16)
- [descriptionWithLocale:](#) (page 17)

Declared In

NSCalendarDate.h

descriptionWithLocale:

Returns a string representation of the receiver formatted as specified by the receiver’s default calendar format and represented according to the given locale information.

```
- (NSString *)descriptionWithLocale:(id)localeDictionary
```

Parameters*localeDictionary*

A dictionary that contains keys and values to represent the locale data to use when creating the description. See [“Locales and String Representations of Calendar Dates”](#) (page 6) for further details.

Return Value

A string representation of the receiver formatted as specified by the receiver’s default calendar format and represented according to the locale information in *localeDictionary*.

Discussion

You can find out what the default calendar format is using the method [calendarFormat](#) (page 12).

This method is used to print an `NSDate` object when the `%@` conversion specifier is used. That is, this statement invokes `descriptionWithLocale::`

```
NSLog(@"The current date and time is %@", [NSDate date]);
```

Availability

Available in Mac OS X v10.0 and later.

See Also

- [description](#) (page 15)
- [descriptionWithCalendarFormat:](#) (page 16)
- [descriptionWithCalendarFormat:locale:](#) (page 16)

- [setCalendarFormat:](#) (page 22)

Declared In

NSCalendarDate.h

hourOfDay

Returns the hour (0 through 23) of the receiver.

- (NSInteger)hourOfDay

Return Value

The hour (0 through 23) of the receiver.

Discussion

On daylight saving time “fall back” days, a value of 1 is returned for two consecutive hours, but with a different time zone (the first in daylight saving time and the second in standard time).

Availability

Available in Mac OS X v10.0 and later.

See Also

- [dayOfMonth](#) (page 14)
- [dayOfWeek](#) (page 14)
- [dayOfYear](#) (page 15)
- [minuteOfHour](#) (page 21)
- [monthOfYear](#) (page 21)
- [secondOfMinute](#) (page 22)

Declared In

NSCalendarDate.h

initWithString:

Returns a calendar date initialized with the date specified as a string in the default calendar format.

- (id)initWithString:(NSString *)*description*

Parameters

description

The description of the new date. The string must conform to the default calendar format “%Y-%m-%d %H:%M:%S %z” (for example, 2001-03-24 16:45:12 +0900). See [Converting Dates to Strings](#), in [Date and Time Programming Guide for Cocoa](#) for a discussion of date conversion specifiers.

Return Value

A calendar date initialized with the date specified by *description*. Returns *nil* if *description* does not exactly match the default calendar format.

Availability

Available in Mac OS X v10.0 and later.

Declared In

NSCalendarDate.h

initWithString:calendarFormat:

Returns a calendar date initialized with the date given as a string in a specified format.

```
- (id)initWithString:(NSString *)description calendarFormat:(NSString *)format
```

Parameters*description*A string containing a description of a date in the format specified by *format*.*format*A string used to interpret *description* and as the default calendar format for the new object. *format* consists of conversion specifiers similar to those used in `strftime()`. See *Converting Dates to Strings*, in *Date and Time Programming Guide for Cocoa* for more details.**Discussion**

The following example shows how to initialize a calendar date with a string of the form “03.24.01 22:00 PST”:

```
NSCalendarDate *newDate = [[NSCalendarDate alloc]
initWithString:@"03.24.01 22:00 PST"
calendarFormat:@"%m.%d.%y %H:%M %Z"];
```

Availability

Available in Mac OS X v10.0 and later.

See Also+ [dateWithString:calendarFormat:](#) (page 9)- [calendarFormat](#) (page 12)**Declared In**

NSCalendarDate.h

initWithString:calendarFormat:locale:

Returns a calendar date initialized with the date given as a string in a specified format and interpreted using a given locale.

```
- (id)initWithString:(NSString *)description calendarFormat:(NSString *)format
locale:(id)localeDictionary
```

Parameters*description*A string containing a description of a date in the format specified by *format*.*format*A string used to interpret *description* and as the default calendar format for the new object. *format* consists of conversion specifiers similar to those used in `strftime()`. See *Converting Dates to Strings*, in *Date and Time Programming Guide for Cocoa* for more details.

localeDictionary

A dictionary that contains keys and values to represent the locale data to use when parsing *description*. See “[Locales and String Representations of Calendar Dates](#)” (page 6) for a list of the appropriate keys.

Return Value

A calendar date initialized with the date specified in the string *description*. Returns `nil` if you specify a locale dictionary that has a month name array with more than 12 elements or a day name array with more than 7 arguments.

Availability

Available in Mac OS X v10.0 and later.

See Also

- + [dateWithString:calendarFormat:locale:](#) (page 10)
- [calendarFormat](#) (page 12)

Declared In

NSCalendarDate.h

initWithYear:month:day:hour:minute:second:timeZone:

Returns a calendar date initialized with specified values for year, month, day, hour, minute, second, and time zone.

```
- (id)initWithYear:(NSInteger)year month:(NSUInteger)month day:(NSUInteger)day
    hour:(NSUInteger)hour minute:(NSUInteger)minute second:(NSUInteger)second
    timeZone:(NSTimeZone *)aTimeZone
```

Parameters

year

The year for the new date. The value must include the century (for example, 1999 instead of 99).

month

The month for the new date. Valid values are 1 through 12.

day

The day for the new date. Valid values are 1 through 31.

hour

The hour for the new date. Valid values are 0 through 23.

minute

The minute for the new date. Valid values are 0 through 59.

second

The second for the new date. Valid values are 0 through 59.

aTimeZone

The time zone for the new date.

Return Value

A calendar date initialized with the specified values for year, month, day, hour, minute, second, and time zone.

Discussion

On days when daylight saving time “falls back,” there are two 1:30 AMs. If you use this method there is no way to create the *second* 1:30 AM. Instead, you should create the first and then use [dateByAddingYears:months:days:hours:minutes:seconds:](#) (page 12) to add an hour.

The following code fragment shows a calendar date created with a date of 4 July 2001, 9 PM, Eastern Standard Time (`timeZoneWithName:` returns the `NSTimeZone` object that represents the time zone with the specified name):

```
NSCalendarDate *fireworks = [[[NSCalendarDate alloc] initWithYear:2001
    month:7 day:4 hour:21 minute:0 second:0
    timeZone:[NSTimeZone timeZoneWithAbbreviation:@"EST"]] autorelease];
```

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [dateWithYear:month:day:hour:minute:second:timeZone:](#) (page 11)

Declared In

NSCalendarDate.h

minuteOfHour

Returns the minute (0 through 59) of the receiver.

- (NSInteger)minuteOfHour

Return Value

The minute (0 through 59) of the receiver.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [dayOfMonth](#) (page 14)
- [dayOfWeek](#) (page 14)
- [dayOfYear](#) (page 15)
- [hourOfDay](#) (page 18)
- [monthOfYear](#) (page 21)
- [secondOfMinute](#) (page 22)

Declared In

NSCalendarDate.h

monthOfYear

Returns the month of the year (1 through 12) of the receiver.

- (NSInteger)monthOfYear

Return Value

The month of the year (1 through 12) of the receiver.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [dayOfMonth](#) (page 14)
- [dayOfWeek](#) (page 14)
- [dayOfYear](#) (page 15)
- [hourOfDay](#) (page 18)
- [minuteOfHour](#) (page 21)
- [secondOfMinute](#) (page 22)

Declared In

NSDate.h

secondOfMinute

Returns the second (0 through 59) of the receiver.

- (NSInteger)secondOfMinute

Return Value

The seconds value (0 through 59) of the receiver.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [dayOfMonth](#) (page 14)
- [dayOfWeek](#) (page 14)
- [dayOfYear](#) (page 15)
- [hourOfDay](#) (page 18)
- [minuteOfHour](#) (page 21)
- [monthOfYear](#) (page 21)

Declared In

NSDate.h

setCalendarFormat:

Sets the default calendar format for the receiver.

- (void)setCalendarFormat:(NSString *)*format*

Parameters

format

The default calendar format for the receiver. See [Converting Dates to Strings](#), in *Date and Time Programming Guide for Cocoa* for a list of the date conversion specifiers.

Discussion

A calendar format is a string formatted with date conversion specifiers. If you do not specify a calendar format for an object, `NSCalendarDate` substitutes its own default. The default is the international format of “%Y-%m-%d %H:%M:%S %z” (for example, 2001-03-24 16:45:12 +0900).

Availability

Available in Mac OS X v10.0 and later.

See Also

- [calendarFormat](#) (page 12)
- [description](#) (page 15)
- [descriptionWithLocale:](#) (page 17)

Declared In

`NSCalendarDate.h`

setTimeZone:

Sets the time zone for the receiver.

```
- (void)setTimeZone:(NSTimeZone *)aTimeZone
```

Parameters

aTimeZone

The time zone for the receiver.

Discussion

If you do not specify a time zone for an object at initialization time, `NSCalendarDate` uses the default time zone for the locale.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [timeZone](#) (page 23)

Declared In

`NSCalendarDate.h`

timeZone

Returns the time zone object associated with the receiver.

```
- (NSTimeZone *)timeZone
```

Return Value

The time zone object associated with the receiver.

Discussion

You can set the time zone when you create the calendar date using the class methods [dateWithString:calendarFormat:](#) (page 9) or [dateWithString:calendarFormat:locale:](#) (page 10) by including the time zone in the description and format parameters. Or you can explicitly set the time

zone to an `NSTimeZone` object using `dateWithYear:month:day:hour:minute:second:timeZone:` (page 11). If you do not specify a time zone for an object at initialization time, `NSCalendarDate` uses the default time zone for the locale.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [setTimeZone:](#) (page 23)

Declared In

`NSCalendarDate.h`

yearOfCommonEra

Returns the year, including the century, of the receiver.

```
- (NSInteger)yearOfCommonEra
```

Return Value

The year, including the century, of the receiver (for example, 1995). The base year of the Common Era is 1 C.E. (which is the same as 1 A.D.).

Availability

Available in Mac OS X v10.0 and later.

See Also

- [dayOfCommonEra](#) (page 13)

Declared In

`NSCalendarDate.h`

years:months:days:hours:minutes:seconds:sinceDate:

Computes the calendrical time difference between the receiver and a given date.

```
- (void)years:(NSInteger *)yearsPointer months:(NSInteger *)monthsPointer
    days:(NSInteger *)daysPointer hours:(NSInteger *)hoursPointer minutes:(NSInteger
    *)minutesPointer seconds:(NSInteger *)secondsPointer sinceDate:(NSCalendarDate
    *)date
```

Parameters

yearsPointer

Upon return, contains the number of years between the receiver and *date*. Pass `NULL` to ignore this component.

monthsPointer

Upon return, contains the number of months between the receiver and *date*. Pass `NULL` to ignore this component.

daysPointer

Upon return, contains the number of days between the receiver and *date*. Pass `NULL` to ignore this component.

hoursPointer

Upon return, contains the number of hours between the receiver and *date*. Pass `NULL` to ignore this component.

minutesPointer

Upon return, contains the number of minutes between the receiver and *date*. Pass `NULL` to ignore this component.

secondsPointer

Upon return, contains the number of seconds between the receiver and *date*. Pass `NULL` to ignore this component.

date

The date with which to compare the receiver. The value must not be `nil`, otherwise an exception is raised.

Discussion

You can choose any representation you wish for the time difference by passing `NULL` for arguments you want to ignore, other than *date*. The following example illustrates how to compute the difference in months, days, and years between two dates.

```
NSCalendarDate *momsBDay = [NSCalendarDate dateWithYear:1936
                             month:1 day:8 hour:7 minute:30 second:0
                             timeZone:[NSTimeZone timeZoneWithAbbreviation:@"EST"]];
NSCalendarDate *dateOfBirth = [NSCalendarDate dateWithYear:1965
                             month:12 day:7 hour:17 minute:25 second:0
                             timeZone:[NSTimeZone timeZoneWithAbbreviation:@"EST"]];
int years, months, days;
```

```
[dateOfBirth years:&years months:&months days:&days hours:NULL
             minutes:NULL seconds:NULL sinceDate:momsBDay];
```

This returns 29 years, 10 months, and 29 days. To express the years in terms of months, pass `NULL` for the years argument:

```
[dateOfBirth years:NULL months:&months days:&days hours:NULL
             minutes:NULL seconds:NULL sinceDate:momsBDay];
```

This returns 358 months and 29 days.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [dateByAddingYears:months:days:hours:minutes:seconds:](#) (page 12)

Declared In

NSCalendarDate.h

Document Revision History

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

Date	Notes
2009-05-21	Added statement to introduction that all methods in this class are deprecated as of Mac OS X v10.6.
2008-05-04	Added note that use is discouraged.
2007-10-31	Corrected typographical errors.
2007-03-06	Added warning about lack of support for Julian calendar.
2006-06-28	Enhanced the discussion of locale dictionaries.
2006-05-23	First publication of this content as a separate document.

REVISION HISTORY

Document Revision History

Index

C

calendarDate **class method** [9](#)
calendarFormat **instance method** [12](#)

D

dateByAddingYears:months:days:hours:minutes:
seconds: **instance method** [12](#)
dateWithString:calendarFormat: **class method** [9](#)
dateWithString:calendarFormat:locale: **class
method** [10](#)
dateWithYear:month:day:hour:minute:second:
timeZone: **class method** [11](#)
dayOfCommonEra **instance method** [13](#)
dayOfMonth **instance method** [14](#)
dayOfWeek **instance method** [14](#)
dayOfYear **instance method** [15](#)
description **instance method** [15](#)
descriptionWithCalendarFormat: **instance method**
[16](#)
descriptionWithCalendarFormat:locale: **instance
method** [16](#)
descriptionWithLocale: **instance method** [17](#)

H

hourOfDay **instance method** [18](#)

I

initWithString: **instance method** [18](#)
initWithString:calendarFormat: **instance method**
[19](#)
initWithString:calendarFormat:locale: **instance
method** [19](#)

initWithYear:month:day:hour:minute:second:
timeZone: **instance method** [20](#)

M

minuteOfHour **instance method** [21](#)
monthOfYear **instance method** [21](#)

S

secondOfMinute **instance method** [22](#)
setCalendarFormat: **instance method** [22](#)
setTimeZone: **instance method** [23](#)

T

timeZone **instance method** [23](#)

Y

yearOfCommonEra **instance method** [24](#)
years:months:days:hours:minutes:seconds:sinceDate:
instance method [24](#)