
NSDate Class Reference

Data Management: Dates, Times, & Numbers





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

Numbers is a trademark of Apple Inc.

OpenGL is a registered trademark of Silicon Graphics, Inc.

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

Simultaneously published in the United States and Canada.

Even though Apple has reviewed this document, APPLE MAKES NO WARRANTY OR REPRESENTATION,

EITHER EXPRESS OR IMPLIED, WITH RESPECT TO THIS DOCUMENT, ITS QUALITY, ACCURACY, MERCHANTABILITY, OR FITNESS FOR A PARTICULAR PURPOSE. AS A RESULT, THIS DOCUMENT IS PROVIDED "AS IS," AND YOU, THE READER, ARE ASSUMING THE ENTIRE RISK AS TO ITS QUALITY AND ACCURACY.

IN NO EVENT WILL APPLE BE LIABLE FOR DIRECT, INDIRECT, SPECIAL, INCIDENTAL, OR CONSEQUENTIAL DAMAGES RESULTING FROM ANY DEFECT OR INACCURACY IN THIS DOCUMENT, even if advised of the possibility of such damages.

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

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

Contents

NSDate Class Reference 5

Overview	5
Subclassing Notes	6
Adopted Protocols	7
Tasks	7
Creating and Initializing Date Objects	7
Getting Temporal Boundaries	8
Comparing Dates	8
Getting Time Intervals	8
Adding a Time Interval	9
Representing Dates as Strings	9
Converting to an NSDateObject Object	9
Class Methods	9
date	9
dateWithNaturalLanguageString:	10
dateWithNaturalLanguageString:locale:	10
dateWithString:	11
dateWithTimeInterval:sinceDate:	12
dateWithTimeIntervalSince1970:	12
dateWithTimeIntervalSinceNow:	13
dateWithTimeIntervalSinceReferenceDate:	13
distantFuture	14
distantPast	15
timeIntervalSinceReferenceDate	15
Instance Methods	16
compare:	16
dateByAddingTimeInterval:	16
dateWithCalendarFormat:timeZone:	17
description	18
descriptionWithCalendarFormat:timeZone:locale:	18
descriptionWithLocale:	19
earlierDate:	20
init	20
initWithString:	20
initWithTimeInterval:sinceDate:	21
initWithTimeIntervalSince1970:	22
initWithTimeIntervalSinceNow:	22
initWithTimeIntervalSinceReferenceDate:	23
isEqualToDate:	23
laterDate:	24
timeIntervalSince1970	24

- timeIntervalSinceDate: 25
- timeIntervalSinceNow 25
- timeIntervalSinceReferenceDate 26
- Constants 26
 - NSTimeIntervalSince1970 26
- Notifications 27
 - NSSystemClockDidChangeNotification 27

Appendix A [Deprecated NSDate Methods 29](#)

- Deprecated in Mac OS X v10.6 29
 - addTimeInterval: 29

[Document Revision History 31](#)

[Index 33](#)

NSDate Class Reference

Inherits from	NSObject
Conforms to	NSCoding NSCopying NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in Mac OS X v10.0 and later.
Declared in	NSDate.h NSDate.h
Companion guides	Date and Time Programming Guide for Cocoa Property List Programming Guide
Related sample code	CalendarItems NewsReader Quartz Composer WWDC 2005 TextEdit Reminders ScriptingBridgeiCal

Overview

`NSDate` objects represent a single point in time. `NSDate` is a class cluster; its single public superclass, `NSDate`, declares the programmatic interface for specific and relative time values. The objects you create using `NSDate` are referred to as date objects. They are immutable objects. Because of the nature of class clusters, objects returned by the `NSDate` class are instances not of that abstract class but of one of its private subclasses. Although a date object's class is private, its interface is public, as declared by the abstract superclass `NSDate`. Generally, you instantiate a suitable date object by invoking one of the `date...` class methods.

`NSDate` is an abstract class that provides behavior for creating dates, comparing dates, representing dates, computing intervals, and similar functionality. `NSDate` presents a programmatic interface through which suitable date objects are requested and returned. Date objects returned from `NSDate` are lightweight and immutable since they represent an invariant point in time. This class is designed to provide the foundation for arbitrary calendrical representations.

The sole primitive method of `NSDate`, [timeIntervalSinceReferenceDate](#) (page 26), provides the basis for all the other methods in the `NSDate` interface. This method returns a time value relative to an absolute reference date—the first instant of 1 January 2001, GMT.

`NSDate` provides several methods to interpret and to create string representations of dates (for example, [dateWithNaturalLanguageString:locale:](#) (page 10) and [descriptionWithLocale:](#) (page 19)). In general, on Mac OS X v10.4 and later you should use an instance of `NSDateFormatter` to parse and generate strings using the methods `dateFromString:` and `stringFromDate:`—see [Date Formatters](#) for more details.

`NSDate` models the change from the Julian to the Gregorian calendar in October 1582, and calendrical calculations performed in conjunction with `NSCalendar` take this transition into account. Note, however, that some locales adopted the Gregorian calendar at other times; for example, Great Britain didn't switch over until September 1752.

`NSDate` is “toll-free bridged” with its Cocoa Foundation counterpart, *CFDate Reference*. This means that the Core Foundation type is interchangeable in function or method calls with the bridged Foundation object. Therefore, in a method where you see an `NSDate *` parameter, you can pass a `CFDateRef`, and in a function where you see a `CFDateRef` parameter, you can pass an `NSDate` instance (you cast one type to the other to suppress compiler warnings). See [Interchangeable Data Types](#) for more information on toll-free bridging.

Subclassing Notes

The major reason for subclassing `NSDate` is to create a class with convenience methods for working with a particular calendrical system. But you could also require a custom `NSDate` class for other reasons, such as to get a date and time value that provides a finer temporal granularity.

Methods to Override

If you want to subclass `NSDate` to obtain behavior different than that provided by the private or public subclasses, you must do these things:

- Declare a suitable instance variable to hold the date and time value (relative to an absolute reference date).
- Override the [timeIntervalSinceReferenceDate](#) (page 26) instance method to provide the correct date and time value based on your instance variable.
- Override [initWithTimeIntervalSinceReferenceDate:](#) (page 23), the designated initializer method.

If you are creating a subclass that represents a calendrical system, you must also define methods that partition past and future periods into the units of this calendar.

Because the `NSDate` class adopts the `NSCopying` and `NSCoding` protocols, your subclass must also implement all of the methods in these protocols.

Special Considerations

Your subclass may use a different reference date than the absolute reference date used by `NSDate` (the first instance of 1 January 2001, GMT). If it does, it must still use the absolute reference date in its implementations of the methods [timeIntervalSinceReferenceDate](#) (page 26) and [initWithTimeIntervalSinceReferenceDate:](#) (page 23). That is, the reference date referred to in the titles of these methods is the absolute reference date. If you do not use the absolute reference date in these methods, comparisons between `NSDate` objects of your subclass and `NSDate` objects of a private subclass will not work.

Adopted Protocols

NSCoding

`encodeWithCoder:`

`initWithCoder:`

NSCopying

`copyWithZone:`

Tasks

Creating and Initializing Date Objects

+ `date` (page 9)

Creates and returns a new date set to the current date and time.

+ `dateWithNaturalLanguageString:` (page 10)

Creates and returns an NSDate object set to the date and time specified by a given string.

+ `dateWithNaturalLanguageString:locale:` (page 10)

Creates and returns an NSDate object set to the date and time specified by a given string.

+ `dateWithString:` (page 11)

Creates and returns an NSDate object with a date and time value specified by a given string in the international string representation format (YYYY-MM-DD HH:MM:SS ±HHMM).

+ `dateWithTimeIntervalSinceNow:` (page 13)

Creates and returns an NSDate object set to a given number of seconds from the current date and time.

+ `dateWithTimeInterval:sinceDate:` (page 12)

Creates and returns an NSDate object set to a given number of seconds from the specified date.

+ `dateWithTimeIntervalSinceReferenceDate:` (page 13)

Creates and returns an NSDate object set to a given number of seconds from the first instant of 1 January 2001, GMT.

+ `dateWithTimeIntervalSince1970:` (page 12)

Creates and returns an NSDate object set to the given number of seconds from the first instant of 1 January 1970, GMT.

- `init` (page 20)

Returns an NSDate object initialized to the current date and time.

- `initWithString:` (page 20)

Returns an NSDate object initialized with a date and time value specified by a given string in the international string representation format.

- `initWithTimeIntervalSinceNow:` (page 22)

Returns an NSDate object initialized relative to the current date and time by a given number of seconds.

- `initWithTimeInterval:sinceDate:` (page 21)

Returns an NSDate object initialized relative to another given date by a given number of seconds.

- [initWithTimeIntervalSinceReferenceDate:](#) (page 23)
Returns an NSDate object initialized relative the first instant of 1 January 2001, GMT by a given number of seconds.
- [initWithTimeIntervalSince1970:](#) (page 22)
Returns an NSDate object set to the given number of seconds from the first instant of 1 January 1970, GMT.

Getting Temporal Boundaries

- + [distantFuture](#) (page 14)
Creates and returns an NSDate object representing a date in the distant future.
- + [distantPast](#) (page 15)
Creates and returns an NSDate object representing a date in the distant past.

Comparing Dates

- [isEqualToDate:](#) (page 23)
Returns a Boolean value that indicates whether a given object is an NSDate object and exactly equal the receiver.
- [earlierDate:](#) (page 20)
Returns the earlier of the receiver and another given date.
- [laterDate:](#) (page 24)
Returns the later of the receiver and another given date.
- [compare:](#) (page 16)
Returns an NSComparisonResult value that indicates the temporal ordering of the receiver and another given date.

Getting Time Intervals

- [timeIntervalSinceDate:](#) (page 25)
Returns the interval between the receiver and another given date.
- [timeIntervalSinceNow](#) (page 25)
Returns the interval between the receiver and the current date and time.
- + [timeIntervalSinceReferenceDate](#) (page 15)
Returns the interval between the first instant of 1 January 2001, GMT and the current date and time.
- [timeIntervalSinceReferenceDate](#) (page 26)
Returns the interval between the receiver and the first instant of 1 January 2001, GMT.
- [timeIntervalSince1970](#) (page 24)
Returns the interval between the receiver and the first instant of 1 January 1970, GMT.

Adding a Time Interval

- [dateByAddingTimeInterval:](#) (page 16)
Returns a new NSDate object that is set to a given number of seconds relative to the receiver.
- [addTimeInterval:](#) (page 29) **Deprecated in Mac OS X v10.6**
Returns a new NSDate object that is set to a given number of seconds relative to the receiver. (**Deprecated.** This method has been replaced by [dateByAddingTimeInterval:](#) (page 16).)

Representing Dates as Strings

- [description](#) (page 18)
Returns a string representation of the receiver.
- [descriptionWithCalendarFormat:timeZone:locale:](#) (page 18)
Returns a string representation of the receiver, formatted as specified by given conversion specifiers.
- [descriptionWithLocale:](#) (page 19)
Returns a string representation of the receiver using the given locale.

Converting to an NSDate object

- [dateWithCalendarFormat:timeZone:](#) (page 17)
Converts the receiver to an NSDate object with a given format string and time zone.

Class Methods

date

Creates and returns a new date set to the current date and time.

```
+ (id)date
```

Return Value

A new date object set to the current date and time.

Discussion

This method uses the default initializer method for the class, [init](#) (page 20).

The following code sample shows how to use `date` to get the current date:

```
NSDate *today = [NSDate date];
```

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

Clock Control

ClockControl

DatePicker
Reminders
With and Without Bindings

Declared In

NSDate.h

dateWithNaturalLanguageString:

Creates and returns an NSDate object set to the date and time specified by a given string.

```
+ (id)dateWithNaturalLanguageString:(NSString *)string
```

Parameters

string

A string that contains a colloquial specification of a date, such as “last Tuesday at dinner,” “3pm December 31, 2001,” “12/31/01,” or “31/12/01.”

Return Value

A new NSDate object set to the current date and time specified by *string*.

Discussion

This method supports only a limited set of colloquial phrases, primarily in English. It may give unexpected results, and its use is strongly discouraged.

In parsing the string, this method uses the date and time preferences stored in the user’s defaults database. (See [dateWithNaturalLanguageString:locale:](#) (page 10) for a list of the specific items used.)

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

Core Data HTML Store
Reminders

Declared In

NSCalendarDate.h

dateWithNaturalLanguageString:locale:

Creates and returns an NSDate object set to the date and time specified by a given string.

```
+ (id)dateWithNaturalLanguageString:(NSString *)string locale:(id)localeDictionary
```

Parameters

string

A string that contains a colloquial specification of a date, such as “last Tuesday at dinner,” “3pm December 31, 2001,” “12/31/01,” or “31/12/01.”

localeDictionary

An `NSDictionary` object containing locale data. To use the user's preferences, you can use `[[NSUserDefaults standardUserDefaults] dictionaryRepresentation]`.

If you pass `nil` or an instance of `NSLocale`, `NSDate` uses the system default locale—this is not the same as the current user's locale.

Return Value

A new `NSDate` object set to the date and time specified by *string* as interpreted according to *localeDictionary*.

Discussion

This method supports only a limited set of colloquial phrases, primarily in English. It may give unexpected results, and its use is strongly discouraged.

The keys and values that represent the locale data from *localeDictionary* are used when parsing the string. In addition to the locale keys listed in the class description, these keys are used when parsing natural language strings:

```

NSDateTimeOrdering
NSEarlierTimeDesignations
NSHourNameDesignations
NSLaterTimeDesignations
NSNextDayDesignations
NSNextNextDayDesignations
NSPriorDayDesignations
NSThisDayDesignations
NSYearMonthWeekDesignations

```

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [dateWithNaturalLanguageString:](#) (page 10)

Declared In

`NSCalendarDate.h`

dateWithString:

Creates and returns an `NSDate` object with a date and time value specified by a given string in the international string representation format (YYYY-MM-DD HH:MM:SS ±HHMM).

```
+ (id)dateWithString:(NSString *)aString
```

Parameters

aString

A string that specifies a date and time value in the international string representation format—YYYY-MM-DD HH:MM:SS ±HHMM, where ±HHMM is a time zone offset in hours and minutes from GMT (for example, "2001-03-24 10:45:32 +0600").

You must specify all fields of the format string, including the time zone offset, which must have a plus or minus sign prefix.

Return Value

An NSDate object with a date and time value specified by *aString*.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [initWithString:](#) (page 20)

Declared In

NSCalendarDate.h

dateWithTimeInterval:sinceDate:

Creates and returns an NSDate object set to a given number of seconds from the specified date.

```
+ (id)dateWithTimeInterval:(NSTimeInterval)seconds sinceDate:(NSDate *)date
```

Parameters

seconds

The number of seconds to add to *date*. Use a negative argument to specify a date and time before *date*.

date

The date.

Return Value

An NSDate object set to *seconds* seconds from *date*.

Availability

Available in Mac OS X v10.6 and later.

Declared In

NSDate.h

dateWithTimeIntervalSince1970:

Creates and returns an NSDate object set to the given number of seconds from the first instant of 1 January 1970, GMT.

```
+ (id)dateWithTimeIntervalSince1970:(NSTimeInterval)seconds
```

Parameters

seconds

The number of seconds from the reference date, 1 January 1970, GMT, for the new date. Use a negative argument to specify a date before this date.

Return Value

An NSDate object set to *seconds* seconds from the reference date.

Discussion

This method is useful for creating NSDate objects from `time_t` values returned by BSD system functions.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [timeIntervalSince1970](#) (page 24)

Related Sample Code

SharedMemory

Declared In

NSDate.h

dateWithTimeIntervalSinceNow:

Creates and returns an NSDate object set to a given number of seconds from the current date and time.

```
+ (id)dateWithTimeIntervalSinceNow:(NSTimeInterval)seconds
```

Parameters

seconds

The number of seconds from the current date and time for the new date. Use a negative value to specify a date before the current date.

Return Value

An NSDate object set to *seconds* seconds from the current date and time.

Availability

Available in Mac OS X v10.0 and later.

See Also

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

Related Sample Code

GLUT

IdentitySample

SimpleThreads

TrivialThreads

WhackedTV

Declared In

NSDate.h

dateWithTimeIntervalSinceReferenceDate:

Creates and returns an NSDate object set to a given number of seconds from the first instant of 1 January 2001, GMT.

```
+ (id)dateWithTimeIntervalSinceReferenceDate:(NSTimeInterval)seconds
```

Parameters

seconds

The number of seconds from the absolute reference date (the first instant of 1 January 2001, GMT) for the new date. Use a negative argument to specify a date and time before the reference date.

Return Value

An NSDate object set to *seconds* seconds from the absolute reference date.

Availability

Available in Mac OS X v10.0 and later.

See Also

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

Related Sample Code

GridCalendar

NewsReader

PhotoSearch

Declared In

NSDate.h

distantFuture

Creates and returns an NSDate object representing a date in the distant future.

```
+ (id)distantFuture
```

Return Value

An NSDate object representing a date in the distant future (in terms of centuries).

Discussion

You can pass this value when an NSDate object is required to have the date argument essentially ignored. For example, the NSWindow method `nextEventMatchingMask:untilDate:inMode:dequeue:` returns `nil` if an event specified in the event mask does not happen before the specified date. You can use the object returned by `distantFuture` as the date argument to wait indefinitely for the event to occur.

```
myEvent = [myWindow nextEventMatchingMask:myEventMask
            untilDate:[NSDate distantFuture]
            inMode:NSDefaultRunLoopMode
            dequeue:YES];
```

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [distantPast](#) (page 15)

Related Sample Code

CalendarItems

CIAnnotation

DatePicker

LiveVideoMixer

SeeMyFriends

Declared In

NSDate.h

distantPast

Creates and returns an `NSDate` object representing a date in the distant past.

```
+ (id)distantPast
```

Return Value

An `NSDate` object representing a date in the distant past (in terms of centuries).

Discussion

You can use this object as a control date, a guaranteed temporal boundary.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [distantFuture](#) (page 14)

Related Sample Code

CIVideoDemoGL

GLChildWindowDemo

GLUT

ThreadsExportMovie

Vertex Optimization

Declared In

`NSDate.h`

timeIntervalSinceReferenceDate

Returns the interval between the first instant of 1 January 2001, GMT and the current date and time.

```
+ (NSTimeInterval)timeIntervalSinceReferenceDate
```

Return Value

The interval between the system's absolute reference date (the first instant of 1 January 2001, GMT) and the current date and time.

Discussion

This method is the primitive method for `NSDate`. If you subclass `NSDate`, you must override this method with your own implementation for it.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [timeIntervalSinceReferenceDate](#) (page 26)

- [timeIntervalSinceDate:](#) (page 25)

- [timeIntervalSince1970](#) (page 24)

- [timeIntervalSinceNow](#) (page 25)

Declared In

`NSDate.h`

Instance Methods

compare:

Returns an `NSComparisonResult` value that indicates the temporal ordering of the receiver and another given date.

```
- (NSComparisonResult)compare:(NSDate *)anotherDate
```

Parameters

anotherDate

The date with which to compare the receiver.

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

Return Value

If:

- The receiver and *anotherDate* are exactly equal to each other, `NSOrderedSame`
- The receiver is later in time than *anotherDate*, `NSOrderedDescending`
- The receiver is earlier in time than *anotherDate*, `NSOrderedAscending`.

Discussion

This method detects sub-second differences between dates. If you want to compare dates with a less fine granularity, use [timeIntervalSinceDate:](#) (page 25) to compare the two dates.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [earlierDate:](#) (page 20)
- `isEqual:` (NSObject protocol)
- [laterDate:](#) (page 24)

Related Sample Code

Reminders

Declared In

`NSDate.h`

dateByAddingTimeInterval:

Returns a new `NSDate` object that is set to a given number of seconds relative to the receiver.

```
- (id)dateByAddingTimeInterval:(NSTimeInterval)seconds
```

Parameters

seconds

The number of seconds to add to the receiver. Use a negative value for seconds to have the returned object specify a date before the receiver.

Return Value

A new `NSDate` object that is set to *seconds* seconds relative to the receiver. The date returned might have a representation different from the receiver's.

Availability

Available in Mac OS X v10.6 and later.

See Also

- [initWithTimeInterval:sinceDate:](#) (page 21)
- [timeIntervalSinceDate:](#) (page 25)

Declared In

NSDate.h

dateWithCalendarFormat:timeZone:

Converts the receiver to an `NSDate` object with a given format string and time zone.

```
- (NSDate *)dateWithCalendarFormat:(NSString *)formatString
    timeZone:(NSTimeZone *)timeZone
```

Parameters

formatString

The format for the returned string (see [Converting Dates to Strings](#) for a discussion of how to create the format string). Pass `nil` to use the default format string, “%Y-%m-%d %H:%M:%S %Z” (this conforms to the international format YYYY-MM-DD HH:MM:SS ±HHMM.)

timeZone

The time zone for the new calendar date. Pass `nil` to use the default time zone—specific to the current locale.

Return Value

A new `NSDate` object bound to *formatString* and the time zone *timeZone*.

Special Considerations

Important: `NSDate` is slated for deprecation, and its use is strongly discouraged.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [description](#) (page 18)
- [descriptionWithCalendarFormat:timeZone:locale:](#) (page 18)
- [descriptionWithLocale:](#) (page 19)

`dateWithString:calendarFormat:` (`NSDate`)

Related Sample Code

Birthdays

Declared In

NSDate.h

description

Returns a string representation of the receiver.

```
- (NSString *)description
```

Return Value

A string representation of the receiver in the international format `YYYY-MM-DD HH:MM:SS ±HHMM`, where `±HHMM` represents the time zone offset in hours and minutes from GMT (for example, “2001-03-24 10:45:32 +0600”).

Availability

Available in Mac OS X v10.0 and later.

See Also

- [descriptionWithLocale:](#) (page 19)

Related Sample Code

SourceView

Declared In

NSDate.h

descriptionWithCalendarFormat:timeZone:locale:

Returns a string representation of the receiver, formatted as specified by given conversion specifiers.

```
- (NSString *)descriptionWithCalendarFormat:(NSString *)formatString
    timeZone:(NSTimeZone *)aTimeZone locale:(id)localeDictionary
```

Parameters

formatString

The format for the returned string (see [Converting Dates to Strings](#) for a discussion of how to create the format string). Pass `nil` to use the default format string, “%Y-%m-%d %H:%M:%S %z” (this conforms to the international format `YYYY-MM-DD HH:MM:SS ±HHMM`).

aTimeZone

The time zone in which to represent the receiver. Pass `nil` to use the default time zone—specific to the current locale.

localeDictionary

An `NSDictionary` object containing locale data. To use the user's preferences, you can use `[[NSUserDefaults standardUserDefaults] dictionaryRepresentation]`.

If you pass `nil` or an instance of `NSLocale`, `NSDate` uses the system default locale—this is not the same as the current user's locale.

Return Value

A string representation of the receiver, formatted as specified by the given conversion specifiers.

Discussion

There are several problems with the implementation of this method that cannot be fixed for compatibility reasons. To format a date correctly, you should consider using a date formatter object instead (see `NSDateFormatter` and *Data Formatting Programming Guide for Cocoa*).

You could use this method to print the current time as follows:

```

sprintf(aString, "The current time is %s\n", [[[NSDate date]
    descriptionWithCalendarFormat:@"%H:%M:%S %Z" timeZone:nil
    locale:[NSUserDefaults standardUserDefaults] dictionaryRepresentation]]
    UTF8String]);

```

Availability

Available in Mac OS X v10.0 and later.

See Also

- [description](#) (page 18)

`descriptionWithCalendarFormat:locale:` (NSDate)

- [descriptionWithLocale:](#) (page 19)

Related Sample Code

SharedMemory

Declared In

NSDate.h

descriptionWithLocale:

Returns a string representation of the receiver using the given locale.

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

Parameters

locale

An `NSLocale` object.

If you pass `nil`, `NSDate` formats the date in the same way as the [description](#) (page 18) method.

On Mac OS X v10.4 and earlier, this parameter was an `NSDictionary` object. If you pass in an `NSDictionary` object on Mac OS X v10.5, `NSDate` uses the default user locale—the same as if you passed in `[NSLocale currentLocale]`.

Return Value

A string representation of the receiver, using the given locale, or if the locale argument is `nil`, in the international format `YYYY-MM-DD HH:MM:SS ±HHMM`, where `±HHMM` represents the time zone offset in hours and minutes from GMT (for example, "2001-03-24 10:45:32 +0600")

Special Considerations

On Mac OS X v10.4 and earlier, *localeDictionary* is an `NSDictionary` object containing locale data. To use the user's preferences, you can use `[[NSUserDefaults standardUserDefaults] dictionaryRepresentation]`.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [description](#) (page 18)

Declared In

NSDate.h

earlierDate:

Returns the earlier of the receiver and another given date.

```
- (NSDate *)earlierDate:(NSDate *)anotherDate
```

Parameters

anotherDate

The date with which to compare the receiver.

Return Value

The earlier of the receiver and *anotherDate*, determined using [timeIntervalSinceDate:](#) (page 25). If the receiver and *anotherDate* represent the same date, returns the receiver.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [compare:](#) (page 16)
- `isEqual:` (NSObject protocol)
- [laterDate:](#) (page 24)

Declared In

NSDate.h

init

Returns an NSDate object initialized to the current date and time.

```
- (id)init
```

Return Value

An NSDate object initialized to the current date and time.

Discussion

This method uses the designated initializer, [initWithTimeIntervalSinceReferenceDate:](#) (page 23).

Availability

Available in Mac OS X v10.0 and later.

See Also

- + [date](#) (page 9)
- [initWithTimeIntervalSinceReferenceDate:](#) (page 23)

Declared In

NSDate.h

initWithString:

Returns an NSDate object initialized with a date and time value specified by a given string in the international string representation format.

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

Parameters*description*

A string that specifies a date and time value in the international string representation format—YYYY-MM-DD HH:MM:SS ±HHMM, where ±HHMM is a time zone offset in hours and minutes from GMT (for example, “2001-03-24 10:45:32 +0600”).

You must specify all fields of the format string, including the time zone offset, which must have a plus or minus sign prefix.

Return Value

An NSDate object initialized with a date and time value specified by *aString*.

Discussion

This method uses the designated initializer, [initWithTimeIntervalSinceReferenceDate:](#) (page 23).

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [dateWithString:](#) (page 11)

- [description](#) (page 18)

Declared In

NSCalendarDate.h

initWithTimeInterval:sinceDate:

Returns an NSDate object initialized relative to another given date by a given number of seconds.

```
- (id)initWithTimeInterval:(NSTimeInterval)seconds sinceDate:(NSDate *)refDate
```

Parameters*seconds*

The number of seconds to add to *refDate*. A negative value means the receiver will be earlier than *refDate*.

refDate

The reference date.

Return Value

An NSDate object initialized relative to *refDate* by *seconds* seconds.

Discussion

This method uses the designated initializer, [initWithTimeIntervalSinceReferenceDate:](#) (page 23).

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

ScriptingBridgeiCal

Declared In

NSDate.h

initWithTimeIntervalSince1970:

Returns an NSDate object set to the given number of seconds from the first instant of 1 January 1970, GMT.

```
- (id)initWithTimeIntervalSince1970:(NSTimeInterval)seconds
```

Parameters

seconds

The number of seconds from the reference date, 1 January 1970, GMT, for the new date. Use a negative argument to specify a date before this date.

Return Value

An NSDate object set to *seconds* seconds from the reference date.

Discussion

This method is useful for creating NSDate objects from `time_t` values returned by BSD system functions.

Availability

Available in Mac OS X v10.6 and later.

Declared In

NSDate.h

initWithTimeIntervalSinceNow:

Returns an NSDate object initialized relative to the current date and time by a given number of seconds.

```
- (id)initWithTimeIntervalSinceNow:(NSTimeInterval)seconds
```

Parameters

seconds

The number of seconds from relative to the current date and time to which the receiver should be initialized. A negative value means the returned object will represent a date in the past.

Return Value

An NSDate object initialized relative to the current date and time by *seconds* seconds.

Discussion

This method uses the designated initializer, [initWithTimeIntervalSinceReferenceDate:](#) (page 23).

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [dateWithTimeIntervalSinceNow:](#) (page 13)

Related Sample Code

PDFKitLinker2

SimpleScriptingProperties

Vertex Optimization

Declared In

NSDate.h

initWithTimeIntervalSinceReferenceDate:

Returns an `NSDate` object initialized relative the first instant of 1 January 2001, GMT by a given number of seconds.

- (id)initWithTimeIntervalSinceReferenceDate:(NSTimeInterval)seconds

Parameters

seconds

The number of seconds to add to the reference date (the first instant of 1 January 2001, GMT). A negative value means the receiver will be earlier than the reference date.

Return Value

An `NSDate` object initialized relative to the absolute reference date by *seconds* seconds.

Discussion

This method is the designated initializer for the `NSDate` class and is declared primarily for the use of subclasses of `NSDate`. When you subclass `NSDate` to create a concrete date class, you must override this method.

Availability

Available in Mac OS X v10.0 and later.

See Also

+ [dateWithTimeIntervalSinceReferenceDate:](#) (page 13)

Declared In

`NSDate.h`

isEqualToDate:

Returns a Boolean value that indicates whether a given object is an `NSDate` object and exactly equal the receiver.

- (BOOL)isEqualToDate:(NSDate *)anotherDate

Parameters

anotherDate

The date to compare with the receiver.

Return Value

YES if the *anotherDate* is an `NSDate` object and is exactly equal to the receiver, otherwise NO.

Discussion

This method detects sub-second differences between dates. If you want to compare dates with a less fine granularity, use [timeIntervalSinceDate:](#) (page 25) to compare the two dates.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [compare:](#) (page 16)
- [earlierDate:](#) (page 20)
- `isEqual:` (NSObject protocol)
- [laterDate:](#) (page 24)

Declared In

NSDate.h

laterDate:

Returns the later of the receiver and another given date.

- (NSDate *)laterDate:(NSDate *)*anotherDate***Parameters***anotherDate*

The date with which to compare the receiver.

Return ValueThe later of the receiver and *anotherDate*, determined using [timeIntervalSinceDate:](#) (page 25). If the receiver and *anotherDate* represent the same date, returns the receiver.**Availability**

Available in Mac OS X v10.0 and later.

See Also

- [compare:](#) (page 16)
- [earlierDate:](#) (page 20)
- [isEqual:](#) (NSObject protocol)

Declared In

NSDate.h

timeIntervalSince1970

Returns the interval between the receiver and the first instant of 1 January 1970, GMT.

- (NSTimeInterval)timeIntervalSince1970

Return Value

The interval between the receiver and the reference date, 1 January 1970, GMT. If the receiver is earlier than the reference date, the value is negative.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [timeIntervalSinceDate:](#) (page 25)
- [timeIntervalSinceNow:](#) (page 25)
- [timeIntervalSinceReferenceDate:](#) (page 26)
- + [timeIntervalSinceReferenceDate:](#) (page 15)

Declared In

NSDate.h

timeIntervalSinceDate:

Returns the interval between the receiver and another given date.

```
- (NSTimeInterval)timeIntervalSinceDate:(NSDate *)anotherDate
```

Parameters

anotherDate

The date with which to compare the receiver.

Return Value

The interval between the receiver and *anotherDate*. If the receiver is earlier than *anotherDate*, the return value is negative.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [timeIntervalSince1970](#) (page 24)
- [timeIntervalSinceNow](#) (page 25)
- [timeIntervalSinceReferenceDate](#) (page 26)

Related Sample Code

URL CacheInfo

Declared In

NSDate.h

timeIntervalSinceNow

Returns the interval between the receiver and the current date and time.

```
- (NSTimeInterval)timeIntervalSinceNow
```

Return Value

The interval between the receiver and the current date and time. If the receiver is earlier than the current date and time, the return value is negative.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [timeIntervalSinceDate:](#) (page 25)
- [timeIntervalSince1970](#) (page 24)
- [timeIntervalSinceReferenceDate](#) (page 26)

Related Sample Code

NewsReader

Declared In

NSDate.h

timeIntervalSinceReferenceDate

Returns the interval between the receiver and the first instant of 1 January 2001, GMT.

- (NSTimeInterval)timeIntervalSinceReferenceDate

Return Value

The interval between the receiver and the system's absolute reference date (the first instant of 1 January 2001, GMT). If the receiver is earlier than the reference date, the value is negative.

Availability

Available in Mac OS X v10.0 and later.

See Also

- [timeIntervalSinceDate](#): (page 25)
- [timeIntervalSinceNow](#) (page 25)
- + [timeIntervalSinceReferenceDate](#) (page 15)

Related Sample Code

CircleView
 From A View to A Movie
 GLSL Basics Cocoa
 GLUT
 Quartz Composer WWDC 2005 TextEdit

Declared In

NSDate.h

Constants

NSTimeIntervalSince1970

NSDate provides a constant that specifies the number of seconds from 1 January 1970 to the reference date, 1 January 2001.

```
#define NSTimeIntervalSince1970 978307200.0
```

Constants

NSTimeIntervalSince1970

The number of seconds from 1 January 1970 to the reference date, 1 January 2001.

Available in Mac OS X v10.0 and later.

Declared in NSDate.h.

Discussion

1 January 1970 is the epoch (or starting point) for Unix time.

Declared In

NSDate.h

Notifications

NSSystemClockDidChangeNotification

Posted whenever the system clock is changed. This can be initiated by a call to `settimeofday()` or the user changing values in the Date and Time Preference panel. The notification object is `null`. This notification does not contain a *userInfo* dictionary.

Availability

Available in Mac OS X v10.6 and later.

Declared In

NSDate.h

Deprecated NSDate Methods

A method identified as deprecated has been superseded and may become unsupported in the future.

Deprecated in Mac OS X v10.6

addTimeInterval:

Returns a new `NSDate` object that is set to a given number of seconds relative to the receiver. (Deprecated in Mac OS X v10.6. This method has been replaced by `dateByAddingTimeInterval:` (page 16).)

```
- (id)addTimeInterval:(NSTimeInterval)seconds
```

Parameters

seconds

The number of seconds to add to the receiver. Use a negative value for seconds to have the returned object specify a date before the receiver.

Return Value

A new `NSDate` object that is set to *seconds* seconds relative to the receiver. The date returned might have a representation different from the receiver's.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.6.

See Also

- `initWithTimeInterval:sinceDate:` (page 21)
- `timeIntervalSinceDate:` (page 25)
- `dateByAddingTimeInterval:` (page 16)

Declared In

`NSDate.h`

Document Revision History

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

Date	Notes
2009-08-17	Updated for Mac OS X v 10.6. Added new methods. Deprecated addTimeInterval:. Added new notification.
2008-10-15	Revised documentation of description and descriptionWithLocale: methods.
2008-06-09	Added a warning to methods related to NSDate that it is slated for deprecation.
2007-10-31	Updated the definitions of the laterDate: and earlierDate: methods.
2007-03-06	Added notes regarding transition between Julian and Gregorian calendar.
2007-02-27	Updated for Mac OS X V10.5.
2006-05-23	Corrected sentence fragment in class description.
	First publication of this content as a separate document.

REVISION HISTORY

Document Revision History

Index

A

`addTimeInterval`: [instance method 29](#)

C

`compare`: [instance method 16](#)

D

`date` [class method 9](#)
`dateByAddingTimeInterval`: [instance method 16](#)
`dateWithCalendarFormat:timeZone`: [instance method 17](#)
`dateWithNaturalLanguageString`: [class method 10](#)
`dateWithNaturalLanguageString:locale`: [class method 10](#)
`dateWithString`: [class method 11](#)
`dateWithTimeInterval:sinceDate`: [class method 12](#)
`dateWithTimeIntervalSince1970`: [class method 12](#)
`dateWithTimeIntervalSinceNow`: [class method 13](#)
`dateWithTimeIntervalSinceReferenceDate`: [class method 13](#)
`description` [instance method 18](#)
`descriptionWithCalendarFormat:timeZone:locale`: [instance method 18](#)
`descriptionWithLocale`: [instance method 19](#)
`distantFuture` [class method 14](#)
`distantPast` [class method 15](#)

E

`earlierDate`: [instance method 20](#)

I

`init` [instance method 20](#)
`initWithString`: [instance method 20](#)
`initWithTimeInterval:sinceDate`: [instance method 21](#)
`initWithTimeIntervalSince1970`: [instance method 22](#)
`initWithTimeIntervalSinceNow`: [instance method 22](#)
`initWithTimeIntervalSinceReferenceDate`: [instance method 23](#)
`isEqualToDate`: [instance method 23](#)

L

`laterDate`: [instance method 24](#)

N

`NSSystemClockDidChangeNotification` [notification 27](#)
`NSTimeIntervalSince1970` [26](#)
`NSTimeIntervalSince1970` [constant 26](#)

T

`timeIntervalSince1970` [instance method 24](#)
`timeIntervalSinceDate`: [instance method 25](#)
`timeIntervalSinceNow` [instance method 25](#)
`timeIntervalSinceReferenceDate` [class method 15](#)
`timeIntervalSinceReferenceDate` [instance method 26](#)