NSDate Class Reference

Developer

Contents

NSDate Class Reference

Inherits from	NSObject
Conforms to	NSCopying NSSecureCoding NSObject (NSObject)
Framework	/System/Library/Frameworks/Foundation.framework
Availability	Available in iOS 2.0 and later.
Declared in	NSDate.h
Companion guides	Date and Time Programming Guide Property List Programming Guide
Related sample code	MVCNetworking TableViewSuite TopSongs URLCache XMLPerformance

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 23), 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.

To parse strings containing dates and to generate string representations of a date, you should use an instance of NSDateFormatter using the methods `dateFromString:` and `stringFromDate:` respectively—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, CFDateRef. See "Toll-Free Bridging" 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 23) instance method to provide the correct date and time value based on your instance variable.
- Override `initWithTimeIntervalSinceReferenceDate:` (page 19), 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 23) and `initWithTimeIntervalSinceReferenceDate:` (page 19). 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.

Tasks

Creating and Initializing Date Objects

`+ date` (page 8)
  Creates and returns a new date set to the current date and time.

`+ dateWithTimeIntervalSinceNow:` (page 10)
  Creates and returns an `NSDate` object set to a given number of seconds from the current date and time.

`+ dateWithTimeInterval:sinceDate:` (page 9)
  Creates and returns an `NSDate` object set to a given number of seconds from the specified date.

`+ dateWithTimeIntervalSinceReferenceDate:` (page 11)
  Creates and returns an `NSDate` object set to a given number of seconds from the first instant of 1 January 2001, GMT.

`+ dateWithTimeIntervalSince1970:` (page 10)
  Creates and returns an `NSDate` object set to the given number of seconds from the first instant of 1 January 1970, GMT.

`– init` (page 17)
  Returns an `NSDate` object initialized to the current date and time.

`– initWithTimeIntervalSinceNow:` (page 19)
  Returns an `NSDate` object initialized relative to the current date and time by a given number of seconds.

`– initWithTimeInterval:sinceDate:` (page 17)
  Returns an `NSDate` object initialized relative to another given date by a given number of seconds.

– `initWithTimeIntervalSinceReferenceDate:` (page 19)
Returns an `NSDate` object initialized relative the first instant of 1 January 2001, GMT by a given number of seconds.

– `initWithTimeIntervalSince1970:` (page 18)
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 11)
Creates and returns an `NSDate` object representing a date in the distant future.

+ `distantPast` (page 12)
Creates and returns an `NSDate` object representing a date in the distant past.

Comparing Dates

– `isEqualToDate:` (page 20)
Returns a Boolean value that indicates whether a given object is an `NSDate` object and exactly equal the receiver.

– `earlierDate:` (page 16)
Returns the earlier of the receiver and another given date.

– `laterDate:` (page 20)
Returns the later of the receiver and another given date.

– `compare:` (page 13)
Returns an `NSComparisonResult` value that indicates the temporal ordering of the receiver and another given date.

Getting Time Intervals

– `timeIntervalSinceDate:` (page 22)
Returns the interval between the receiver and another given date.

– `timeIntervalSinceNow` (page 22)
Returns the interval between the receiver and the current date and time.

+ `timeIntervalSinceReferenceDate` (page 13)
    Returns the interval between the first instant of 1 January 2001, GMT and the current date and time.

– `timeIntervalSinceReferenceDate` (page 23)
    Returns the interval between the receiver and the first instant of 1 January 2001, GMT.

– `timeIntervalSince1970` (page 21)
    Returns the interval between the receiver and the first instant of 1 January 1970, GMT.

Adding a Time Interval

– `dateByAddingTimeInterval:` (page 14)
    Returns a new `NSDate` object that is set to a given number of seconds relative to the receiver.

– `addTimeInterval:` (page 25) Deprecated in iOS 4.0
    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 14).)

Representing Dates as Strings

– `description` (page 15)
    Returns a string representation of the receiver.

– `descriptionWithLocale:` (page 16)
    Returns a string representation of the receiver using the given locale.

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 17).

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

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

Availability
Available in iOS 2.0 and later.

Related Sample Code
Birthdays
CoreDataBooks
MVCNetworking
Regions
TableViewSuite

Declared in
`NSDate.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 iOS 4.0 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 iOS 2.0 and later.

See Also

– `timeIntervalSince1970` (page 21)

Related Sample Code
MVCNetworking

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 iOS 2.0 and later.

See Also

– `initWithTimeIntervalSinceNow:` (page 19)

Related Sample Code
Quartz Composer SQLiteQuery

SimpleEKDemo

SimpleTextInput

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 iOS 2.0 and later.

See Also

– `initWithTimeIntervalSinceReferenceDate:` (page 19)

Related Sample Code
URLCache

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 iOS 2.0 and later.

See Also

`+ distantPast` (page 12)

Related Sample Code

TopSongs

XMLPerformance

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 iOS 2.0 and later.

See Also
`+ distantFuture` (page 11)

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 iOS 2.0 and later.

See Also
`– timeIntervalSinceReferenceDate` (page 23)
`– timeIntervalSinceDate:` (page 22)
`– timeIntervalSince1970` (page 21)
`– timeIntervalSinceNow` (page 22)

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 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 22) to compare the two dates.

Availability

Available in iOS 2.0 and later.

See Also

`– earlierDate:` (page 16)
`isEqual:` (`NSObject` protocol)
`– laterDate:` (page 20)

Related Sample Code

PhotosByLocation

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 iOS 4.0 and later.

See Also

– `initWithTimeInterval:sinceDate:` (page 17)
– `timeIntervalSinceDate:` (page 22)

Related Sample Code
TableViewSuite

Declared in

`NSDate.h`

description

Returns a string representation of the receiver.

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

Return Value

A string representation of the receiver.

Discussion

The representation is not guaranteed to remain constant across different releases of the operating system. To format a date, you should use a date formatter object instead (see `NSDateFormatter` and *Data Formatting Guide*)

Availability

Available in iOS 2.0 and later.

See Also

– `descriptionWithLocale:` (page 16)

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 15) method.

On OS X v10.4 and earlier, this parameter was an `NSDictionary` object. If you pass in an `NSDictionary` object on 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 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 iOS 4.0 and later.

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 22). If the receiver and `anotherDate` represent the same date, returns the receiver.

Availability

Available in iOS 2.0 and later.

See Also

– `compare:` (page 13)
`isEqual:` (`NSObject` protocol)
– `laterDate:` (page 20)

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 19).

Availability

Available in iOS 2.0 and later.

See Also

+ `date` (page 8)
– `initWithTimeIntervalSinceReferenceDate:` (page 19)

Declared in

`NSDate.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 19).

Availability

Available in iOS 2.0 and later.

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 iOS 4.0 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 19).

Availability

Available in iOS 2.0 and later.

See Also

`+ dateWithTimeIntervalSinceNow:` (page 10)

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 iOS 2.0 and later.

See Also
`+ dateWithTimeIntervalSinceReferenceDate:` (page 11)

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 22) to compare the two dates.

Availability
Available in iOS 2.0 and later.

See Also
`– compare:` (page 13)
`– earlierDate:` (page 16)
`isEqual:` (`NSObject` protocol)
`– laterDate:` (page 20)

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 Value

The later of the receiver and `anotherDate`, determined using `timeIntervalSinceDate:` (page 22). If the receiver and `anotherDate` represent the same date, returns the receiver.

Availability

Available in iOS 2.0 and later.

See Also

`– compare:` (page 13)
`– earlierDate:` (page 16)
`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 iOS 2.0 and later.

See Also

`– timeIntervalSinceDate:` (page 22)
`– timeIntervalSinceNow` (page 22)
`– timeIntervalSinceReferenceDate` (page 23)
`+ timeIntervalSinceReferenceDate` (page 13)

Related Sample Code
GKLeaderboards

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 iOS 2.0 and later.

See Also
– `timeIntervalSince1970` (page 21)
– `timeIntervalSinceNow` (page 22)
– `timeIntervalSinceReferenceDate` (page 23)

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 iOS 2.0 and later.

See Also

– `timeIntervalSinceDate:` (page 22)
– `timeIntervalSince1970` (page 21)
– `timeIntervalSinceReferenceDate` (page 23)

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 iOS 2.0 and later.

See Also

– `timeIntervalSinceDate:` (page 22)
– `timeIntervalSinceNow` (page 22)
+ `timeIntervalSinceReferenceDate` (page 13)

Related Sample Code

aurioTouch
GKRocket
MVCNetworking
TopSongs
XMLPerformance

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 iOS 2.0 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 iOS 4.0 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 iOS 4.0

addTimeInterval:

Returns a new `NSDate` object that is set to a given number of seconds relative to the receiver. (Deprecated in iOS 4.0. This method has been replaced by `dateByAddingTimeInterval:` (page 14).)

```
– (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 iOS 2.0 and later.

Deprecated in iOS 4.0.

See Also

– `initWithTimeInterval:sinceDate:` (page 17)
– `timeIntervalSinceDate:` (page 22)
– `dateByAddingTimeInterval:` (page 14)

Declared in

`NSDate.h`

Document Revision History

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

Date	Notes
2011-04-05	Updated description of description method.
2010-04-29	Added method dateByAddingTimeInterval:.
2010-01-17	Removed inappropriate references to dateWithNaturalLanguageString:.
2009-08-17	Updated for 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 NSCalendarDate 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 OS X V10.5.
2006-05-23	Corrected sentence fragment in class description. First publication of this content as a separate document.

Apple Inc.
© 2011 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.

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.

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

Apple, the Apple logo, Cocoa, Numbers, OS X, and Quartz are trademarks of Apple Inc., registered in the U.S. and other countries.

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

iOS is a trademark or registered trademark of Cisco in the U.S. and other countries and is used under license.

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.