# NSCalendar Class Reference

**Cocoa > Data Management**

**2009-02-04**

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



Simultaneously published in the United States and Canada.

# Contents

# NSCalendar Class Reference

| | |
|---|---|
| **Inherits from** | NSObject |
| **Conforms to** | NSCoding<br>NSCopying<br>NSObject (NSObject) |
| **Framework** | /System/Library/Frameworks/Foundation.framework |
| **Availability** | Available in Mac OS X v10.4 and later. |
| **Declared in** | NSCalendar.h |
| **Companion guides** | Date and Time Programming Guide for Cocoa<br>Data Formatting Programming Guide for Cocoa |
| **Related sample code** | Birthdays<br>Mountains<br>Reminders |

## Overview

Calendars encapsulate information about systems of reckoning time in which the beginning, length, and divisions of a year are defined. They provide information about the calendar and support for calendrical computations such as determining the range of a given calendrical unit and adding units to a given absolute time.

In a calendar, day, week, weekday, month, and year numbers are generally 1-based, but there may be calendar-specific exceptions. Ordinal numbers, where they occur, are 1-based. Some calendars represented by this API may have to map their basic unit concepts into year/month/week/day/… nomenclature. For example, a calendar composed of 4 quarters in a year instead of 12 months uses the month unit to represent quarters. The particular values of the unit are defined by each calendar, and are not necessarily consistent with values for that unit in another calendar.

To do calendar arithmetic, you use `NSDate` objects in conjunction with a calendar. For example, to convert between a decomposed date in one calendar and another calendar, you must first convert the decomposed elements into a date using the first calendar, then decompose it using the second. `NSDate` provides the absolute scale and epoch (reference point) for dates and times, which can then be rendered into a particular calendar, for calendrical computations or user display.

Two `NSCalendar` methods that return a date object, `dateFromComponents:` (page 12), `dateByAddingComponents:toDate:options:` (page 11), take as a parameter an `NSDateComponents` object that describes the calendrical components required for the computation. You can provide as many components as you need (or choose to). When there is incomplete information to compute an absolute time,

default values similar to `0` and `1` are usually chosen by a calendar, but this is a calendar-specific choice. If you provide inconsistent information, calendar-specific disambiguation is performed (which may involve ignoring one or more of the parameters). Related methods (`components:fromDate:` (page 9) and `components:fromDate:toDate:options:` (page 10)) take a bit mask parameter that specifies which components to calculate when returning an `NSDateComponents` object. The bit mask is composed of `NSCalendarUnit` constants (see "Constants" (page 19)).

`NSCalendar` is "toll-free bridged" with its Core Foundation counterpart, CFCalendar. 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 `NSCalendar *` parameter, you can pass in a `CFCalendarRef`, and in a function where you see a `CFCalendarRef` parameter, you can pass in an `NSCalendar` instance. See Interchangeable Data Types for more information on toll-free bridging.

# Tasks

## System Locale Information

`+ currentCalendar` (page 8)
Returns the logical calendar for the current user.

`+ autoupdatingCurrentCalendar` (page 7)
Returns the current logical calendar for the current user.

## Initializing a Calendar

`- initWithCalendarIdentifier:` (page 13)
Initializes a newly-allocated `NSCalendar` object for the calendar specified by a given identifier.

`- setFirstWeekday:` (page 17)
Sets the index of the first weekday for the receiver.

`- setLocale:` (page 17)
Sets the locale for the receiver.

`- setMinimumDaysInFirstWeek:` (page 18)
Sets the minimum number of days in the first week of the receiver.

`- setTimeZone:` (page 18)
Sets the time zone for the receiver.

## Getting Information About a Calendar

`- calendarIdentifier` (page 8)
Returns the identifier for the receiver.

`- firstWeekday` (page 13)
Returns the index of the first weekday of the receiver.

`- locale` (page 13)
Returns the locale for the receiver.

- `maximumRangeOfUnit:` (page 14)
  The maximum range limits of the values that a given unit can take on in the receive
- `minimumDaysInFirstWeek` (page 14)
  Returns the minimum number of days in the first week of the receiver.
- `minimumRangeOfUnit:` (page 15)
  Returns the minimum range limits of the values that a given unit can take on in the receiver.
- `ordinalityOfUnit:inUnit:forDate:` (page 15)
  Returns, for a given absolute time, the ordinal number of a smaller calendar unit (such as a day) within a specified larger calendar unit (such as a week).
- `rangeOfUnit:inUnit:forDate:` (page 16)
  Returns the range of absolute time values that a smaller calendar unit (such as a day) can take on in a larger calendar unit (such as a month) that includes a specified absolute time.
- `rangeOfUnit:startDate:interval:forDate:` (page 16)
  Returns by reference the starting time and duration of a given calendar unit that contains a given date.
- `timeZone` (page 19)
  Returns the time zone for the receiver.

### Calendrical Calculations

- `components:fromDate:` (page 9)
  Returns a `NSDateComponents` object containing a given date decomposed into specified components.
- `components:fromDate:toDate:options:` (page 10)
  Returns, as an `NSDateComponents` object using specified components, the difference between two supplied dates.
- `dateByAddingComponents:toDate:options:` (page 11)
  Returns a new `NSDate` object representing the absolute time calculated by adding given components to a given date.
- `dateFromComponents:` (page 12)
  Returns a new `NSDate` object representing the absolute time calculated from given components.

## Class Methods

### autoupdatingCurrentCalendar

Returns the current logical calendar for the current user.

```
+ (id)autoupdatingCurrentCalendar
```

**Return Value**
The current logical calendar for the current user.

**Discussion**
Settings you get from this calendar do change as the user's settings change (contrast with `currentCalendar` (page 8)).

Note that if you cache values based on the calendar or related information those caches will of course not be automatically updated by the updating of the calendar object.

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

**See Also**
`+ currentCalendar` (page 8)
`- initWithCalendarIdentifier:` (page 13)
`- calendarIdentifier` (page 8)

**Declared In**
`NSCalendar.h`

### currentCalendar

Returns the logical calendar for the current user.

```
+ (id)currentCalendar
```

**Return Value**
The logical calendar for the current user.

**Discussion**
The returned calendar is formed from the settings for the current user's chosen system locale overlaid with any custom settings the user has specified in System Preferences. Settings you get from this calendar do not change as System Preferences are changed, so that your operations are consistent (contrast with `autoupdatingCurrentCalendar` (page 7)).

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

**See Also**
`+ autoupdatingCurrentCalendar` (page 7)
`- initWithCalendarIdentifier:` (page 13)
`- calendarIdentifier` (page 8)

**Declared In**
`NSCalendar.h`

## Instance Methods

### calendarIdentifier

Returns the identifier for the receiver.

```
- (NSString *)calendarIdentifier
```

**Return Value**
The identifier for the receiver. For valid identifiers, see `NSLocale`.

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

**See Also**
`+ autoupdatingCurrentCalendar` (page 7)
`- initWithCalendarIdentifier:` (page 13)

**Related Sample Code**
Mountains

**Declared In**
`NSCalendar.h`

## components:fromDate:

Returns a `NSDateComponents` object containing a given date decomposed into specified components.

```
- (NSDateComponents *)components:(NSUInteger)unitFlags fromDate:(NSDate *)date
```

**Parameters**

*`unitFlags`*
The components into which to decompose *`date`*—a bitwise `OR` of `NSCalendarUnit` constants.

*`date`*
The date for which to perform the calculation.

**Return Value**
An `NSDateComponents` object containing *`date`* decomposed into the components specified by *`unitFlags`*. Returns `nil` if *`date`* falls outside of the defined range of the receiver or if the computation cannot be performed

**Discussion**
The Weekday ordinality, when requested, refers to the next larger (than Week) of the requested units. Some computations can take a relatively long time.

The following example shows how to use this method to determine the current year, month, and day, using an existing calendar (`gregorian`):

```
unsigned unitFlags = NSYearCalendarUnit | NSMonthCalendarUnit |
NSDayCalendarUnit;
NSDate *date = [NSDate date];
NSDateComponents *comps = [gregorian components:unitFlags fromDate:date];
```

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

**See Also**
`- dateFromComponents:` (page 12)
`- components:fromDate:toDate:options:` (page 10)
`- dateByAddingComponents:toDate:options:` (page 11)

**Related Sample Code**
Birthdays

**Declared In**
NSCalendar.h

## components:fromDate:toDate:options:

Returns, as an NSDateComponents object using specified components, the difference between two supplied dates.

```
- (NSDateComponents *)components:(NSUInteger)unitFlags fromDate:(NSDate
    *)startingDate toDate:(NSDate *)resultDate options:(NSUInteger)opts
```

**Parameters**

*unitFlags*
: Specifies the components for the returned NSDateComponents object—a bitwise OR of NSCalendarUnit constants.

*startingDate*
: The start date for the calculation.

*resultDate*
: The end date for the calculation.

*opts*
: Options for the calculation.

  If you specify a "wrap" option (NSWrapCalendarComponents), the specified components are incremented and wrap around to zero/one on overflow, but do not cause higher units to be incremented. When the wrap option is false, overflow in a unit carries into the higher units, as in typical addition.

**Return Value**
An NSDateComponents object whose components are specified by *unitFlags* and calculated from the difference between the *resultDate* and *startDate* using the options specified by *opts*. Returns nil if either date falls outside the defined range of the receiver or if the computation cannot be performed.

**Discussion**
The result is lossy if there is not a small enough unit requested to hold the full precision of the difference. Some operations can be ambiguous, and the behavior of the computation is calendar-specific, but generally larger components will be computed before smaller components; for example, in the Gregorian calendar a result might be 1 month and 5 days instead of, for example, 0 months and 35 days. The resulting component values may be negative if *resultDate* is before *startDate*.

The following example shows how to get the approximate number of months and days between two dates using an existing calendar (gregorian):

```
NSDate *startDate = ...;
NSDate *endDate = ...;
unsigned int unitFlags = NSMonthCalendarUnit | NSDayCalendarUnit;
NSDateComponents *comps = [gregorian components:unitFlags fromDate:startDate
toDate:endDate  options:0];
int months = [comps month];
int days = [comps day];
```

Note that some computations can take a relatively long time.

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

**See Also**
- `dateByAddingComponents:toDate:options:` (page 11)
- `dateFromComponents:` (page 12)

**Declared In**
`NSCalendar.h`

## dateByAddingComponents:toDate:options:

Returns a new `NSDate` object representing the absolute time calculated by adding given components to a given date.

```
- (NSDate *)dateByAddingComponents:(NSDateComponents *)comps toDate:(NSDate *)date
    options:(NSUInteger)opts
```

**Parameters**

*comps*

The components to add to *date*.

*date*

The date to which *comps* are added.

*opts*

Options for the calculation. See "`NSDateComponents wrapping behavior`" (page 21) for possible values. Pass `0` to specify no options.

If you specify no options (you pass `0`), overflow in a unit carries into the higher units (as in typical addition).

**Return Value**
A new `NSDate` object representing the absolute time calculated by adding to *date* the calendrical components specified by *comps* using the options specified by *opts*. Returns `nil` if *date* falls outside the defined range of the receiver or if the computation cannot be performed.

**Discussion**
Some operations can be ambiguous, and the behavior of the computation is calendar-specific, but generally components are added in the order specified.

The following example shows how to add 2 months and 3 days to the current date and time using an existing calendar (`gregorian`):

```
NSDate *currentDate = [NSDate date];
NSDateComponents *comps = [[NSDateComponents alloc] init];
[comps setMonth:2];
[comps setDay:3];
NSDate *date = [gregorian dateByAddingComponents:comps toDate:currentDate
options:0];
[comps release];
```

Note that some computations can take a relatively long time.

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

**See Also**
- `dateFromComponents:` (page 12)

- components:fromDate:toDate:options: (page 10)

**Declared In**
NSCalendar.h

## dateFromComponents:

Returns a new NSDate object representing the absolute time calculated from given components.

- (NSDate *)dateFromComponents:(NSDateComponents *)*comps*

**Parameters**

*comps*

The components from which to calculate the returned date.

**Return Value**
A new NSDate object representing the absolute time calculated from *comps*. Returns nil if the receiver cannot convert the components given in *comps* into an absolute time. The method also returns nil and for out-of-range values.

**Discussion**
When there are insufficient components provided to completely specify an absolute time, a calendar uses default values of its choice. When there is inconsistent information, a calendar may ignore some of the components parameters or the method may return nil. Unnecessary components are ignored (for example, Day takes precedence over Weekday and Weekday ordinals).

The following example shows how to use this method to create a date object to represent 14:10:00 on 6 January 1965, for a given calendar (gregorian).

```
NSDateComponents *comps = [[NSDateComponents alloc] init];
[comps setYear:1965];
[comps setMonth:1];
[comps setDay:6];
[comps setHour:14];
[comps setMinute:10];
[comps setSecond:0];
NSDate *date = [gregorian dateFromComponents:comps];
[comps release];
```

Note that some computations can take a relatively long time to perform.

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

**See Also**
- components:fromDate: (page 9)
- dateFromComponents: (page 12)

**Related Sample Code**
Reminders

**Declared In**
NSCalendar.h

## firstWeekday

Returns the index of the first weekday of the receiver.

```
- (NSUInteger)firstWeekday
```

**Return Value**
The index of the first weekday of the receiver.

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

**See Also**
- `setFirstWeekday:` (page 17)

**Declared In**
`NSCalendar.h`

## initWithCalendarIdentifier:

Initializes a newly-allocated `NSCalendar` object for the calendar specified by a given identifier.

```
- (id)initWithCalendarIdentifier:(NSString *)string
```

**Parameters**
*string*
The identifier for the new calendar. For valid identifiers, see `NSLocale`.

**Return Value**
The initialized calendar, or `nil` if the identifier is unknown (if, for example, it is either an unrecognized string or the calendar is not supported by the current version of the operating system).

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

**See Also**
- `+ autoupdatingCurrentCalendar` (page 7)
- `- calendarIdentifier` (page 8)

**Related Sample Code**
Birthdays
Mountains
Reminders

**Declared In**
`NSCalendar.h`

## locale

Returns the locale for the receiver.

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

**Return Value**
The locale for the receiver.

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

**See Also**
- `setLocale:` (page 17)

**Declared In**
`NSCalendar.h`

## maximumRangeOfUnit:

The maximum range limits of the values that a given unit can take on in the receive

```
- (NSRange)maximumRangeOfUnit:(NSCalendarUnit)unit
```

**Parameters**

*unit*

The unit for which the maximum range is returned.

**Return Value**
The maximum range limits of the values that the unit specified by *unit* can take on in the receiver.

**Discussion**
As an example, in the Gregorian calendar the maximum range of values for the Day unit is 1-31.

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

**See Also**
- `minimumRangeOfUnit:` (page 15)

**Declared In**
`NSCalendar.h`

## minimumDaysInFirstWeek

Returns the minimum number of days in the first week of the receiver.

```
- (NSUInteger)minimumDaysInFirstWeek
```

**Return Value**
The minimum number of days in the first week of the receiver

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

**See Also**
- `setMinimumDaysInFirstWeek:` (page 18)

**Declared In**
`NSCalendar.h`

## minimumRangeOfUnit:

Returns the minimum range limits of the values that a given unit can take on in the receiver.

```
- (NSRange)minimumRangeOfUnit:(NSCalendarUnit)unit
```

**Parameters**

*unit*

The unit for which the maximum range is returned.

**Return Value**

The minimum range limits of the values that the unit specified by *unit* can take on in the receiver.

**Discussion**

As an example, in the Gregorian calendar the minimum range of values for the Day unit is 1-28.

**Availability**

Available in Mac OS X v10.4 and later.

**See Also**

- maximumRangeOfUnit: (page 14)

**Declared In**

NSCalendar.h

## ordinalityOfUnit:inUnit:forDate:

Returns, for a given absolute time, the ordinal number of a smaller calendar unit (such as a day) within a specified larger calendar unit (such as a week).

```
- (NSUInteger)ordinalityOfUnit:(NSCalendarUnit)smaller inUnit:(NSCalendarUnit)larger
    forDate:(NSDate *)date
```

**Parameters**

*smaller*

The smaller calendar unit

*larger*

The larger calendar unit

*date*

The absolute time for which the calculation is performed

**Return Value**

The ordinal number of *smaller* within *larger* at the time specified by *date*. Returns NSNotFound if *larger* is not logically bigger than *smaller* in the calendar, or the given combination of units does not make sense (or is a computation which is undefined).

**Discussion**

The ordinality is in most cases not the same as the decomposed value of the unit. Typically return values are 1 and greater. For example, the time 00:45 is in the first hour of the day, and for units Hour and Day respectively, the result would be 1. An exception is the week-in-month calculation, which returns 0 for days before the first week in the month containing the date.

Note that some computations can take a relatively long time.

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

**See Also**
- `rangeOfUnit:inUnit:forDate:` (page 16)
- `rangeOfUnit:startDate:interval:forDate:` (page 16)

**Declared In**
`NSCalendar.h`

## rangeOfUnit:inUnit:forDate:

Returns the range of absolute time values that a smaller calendar unit (such as a day) can take on in a larger calendar unit (such as a month) that includes a specified absolute time.

```
- (NSRange)rangeOfUnit:(NSCalendarUnit)smaller inUnit:(NSCalendarUnit)larger
    forDate:(NSDate *)date
```

**Parameters**

*smaller*
The smaller calendar unit.

*larger*
The larger calendar unit.

*date*
The absolute time for which the calculation is performed.

**Return Value**
The range of absolute time values *smaller* can take on in *larger* at the time specified by *date*. Returns {NSNotFound, NSNotFound} if *larger* is not logically bigger than *smaller* in the calendar, or the given combination of units does not make sense (or is a computation which is undefined).

**Discussion**
You can use this method to calculate, for example, the range the Day unit can take on in the Month in which *date* lies.

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

**See Also**
- `rangeOfUnit:startDate:interval:forDate:` (page 16)
- `ordinalityOfUnit:inUnit:forDate:` (page 15)

**Related Sample Code**
Birthdays

**Declared In**
`NSCalendar.h`

## rangeOfUnit:startDate:interval:forDate:

Returns by reference the starting time and duration of a given calendar unit that contains a given date.

```
- (BOOL)rangeOfUnit:(NSCalendarUnit)unit startDate:(NSDate **)datep
    interval:(NSTimeInterval *)tip forDate:(NSDate *)date
```

**Parameters**

*unit*

A calendar unit (see "Calendar Units" (page 19) for possible values).

*datep*

Upon return, contains the starting time of the calendar unit *unit* that contains the date *date*

*tip*

Upon return, contains the duration of the calendar unit *unit* that contains the date *date*

*date*

A date.

**Return Value**

`YES` if the starting time and duration of a unit could be calculated, otherwise `NO`.

**Availability**

Available in Mac OS X v10.5 and later.

**See Also**

- `rangeOfUnit:inUnit:forDate:` (page 16)
- `ordinalityOfUnit:inUnit:forDate:` (page 15)

**Declared In**

`NSCalendar.h`

## setFirstWeekday:

Sets the index of the first weekday for the receiver.

```
- (void)setFirstWeekday:(NSUInteger)weekday
```

**Parameters**

*weekday*

The first weekday for the receiver.

**Availability**

Available in Mac OS X v10.4 and later.

**See Also**

- `firstWeekday` (page 13)

**Declared In**

`NSCalendar.h`

## setLocale:

Sets the locale for the receiver.

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

**Parameters**

*locale*

The locale for the receiver.

**Availability**

Available in Mac OS X v10.4 and later.

**See Also**

- locale (page 13)

**Declared In**

NSCalendar.h

## setMinimumDaysInFirstWeek:

Sets the minimum number of days in the first week of the receiver.

```
- (void)setMinimumDaysInFirstWeek:(NSUInteger)mdw
```

**Parameters**

*mdw*

The minimum number of days in the first week of the receiver.

**Availability**

Available in Mac OS X v10.4 and later.

**See Also**

- minimumDaysInFirstWeek (page 14)

**Declared In**

NSCalendar.h

## setTimeZone:

Sets the time zone for the receiver.

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

**Parameters**

*tz*

The time zone for the receiver.

**Availability**

Available in Mac OS X v10.4 and later.

**See Also**

- timeZone (page 19)

**Declared In**

NSCalendar.h

### timeZone

Returns the time zone for the receiver.

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

**Return Value**
The time zone for the receiver.

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

**See Also**
`- setTimeZone:` (page 18)

**Declared In**
`NSCalendar.h`

## Constants

### NSCalendarUnit

Defines a type used to specify calendrical units such as day and month.

```
typedef NSUInteger NSCalendarUnit;
```

**Discussion**
See "Calendar Units" (page 19) for possible values.

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

**Declared In**
`NSCalendar.h`

### Calendar Units

Specify calendrical units such as day and month.

```
enum {
    NSEraCalendarUnit = kCFCalendarUnitEra,
    NSYearCalendarUnit = kCFCalendarUnitYear,
    NSMonthCalendarUnit = kCFCalendarUnitMonth,
    NSDayCalendarUnit = kCFCalendarUnitDay,
    NSHourCalendarUnit = kCFCalendarUnitHour,
    NSMinuteCalendarUnit = kCFCalendarUnitMinute,
    NSSecondCalendarUnit = kCFCalendarUnitSecond,
    NSWeekCalendarUnit = kCFCalendarUnitWeek,
    NSWeekdayCalendarUnit = kCFCalendarUnitWeekday,
    NSWeekdayOrdinalCalendarUnit = kCFCalendarUnitWeekdayOrdinal
};
```

**Constants**

`NSEraCalendarUnit`

Specifies the era unit.

The corresponding value is an `int`. Equal to `kCFCalendarUnitEra`.

Available in Mac OS X v10.4 and later.

Declared in `NSCalendar.h`.

`NSYearCalendarUnit`

Specifies the year unit.

The corresponding value is an `int`. Equal to `kCFCalendarUnitYear`.

Available in Mac OS X v10.4 and later.

Declared in `NSCalendar.h`.

`NSMonthCalendarUnit`

Specifies the month unit.

The corresponding value is an `int`. Equal to `kCFCalendarUnitMonth`.

Available in Mac OS X v10.4 and later.

Declared in `NSCalendar.h`.

`NSDayCalendarUnit`

Specifies the day unit.

The corresponding value is an `int`. Equal to `kCFCalendarUnitDay`.

Available in Mac OS X v10.4 and later.

Declared in `NSCalendar.h`.

`NSHourCalendarUnit`

Specifies the hour unit.

The corresponding value is an `int`. Equal to `kCFCalendarUnitHour`.

Available in Mac OS X v10.4 and later.

Declared in `NSCalendar.h`.

`NSMinuteCalendarUnit`

Specifies the minute unit.

The corresponding value is an `int`. Equal to `kCFCalendarUnitMinute`.

Available in Mac OS X v10.4 and later.

Declared in `NSCalendar.h`.

`NSSecondCalendarUnit`
Specifies the second unit.

The corresponding value is a `double`. Equal to `kCFCalendarUnitSecond`.

Available in Mac OS X v10.4 and later.

Declared in `NSCalendar.h`.

`NSWeekCalendarUnit`
Specifies the week unit.

The corresponding value is an `int`. Equal to `kCFCalendarUnitWeek`.

Available in Mac OS X v10.4 and later.

Declared in `NSCalendar.h`.

`NSWeekdayCalendarUnit`
Specifies the weekday unit.

The corresponding value is an `int`. Equal to `kCFCalendarUnitWeekday`. The weekday units are the numbers 1 through N (where for the Gregorian calendar N=7 and 1 is Sunday).

Available in Mac OS X v10.4 and later.

Declared in `NSCalendar.h`.

`NSWeekdayOrdinalCalendarUnit`
Specifies the ordinal weekday unit.

The corresponding value is an `int`. Equal to `kCFCalendarUnitWeekdayOrdinal`. The weekday ordinal unit describes ordinal position within the month unit of the corresponding weekday unit. For example, in the Gregorian calendar a weekday ordinal unit of 2 for a weekday unit 3 indicates "the second Tuesday in the month".

Available in Mac OS X v10.4 and later.

Declared in `NSCalendar.h`.

**Discussion**
Calendar units may be used as a bit mask to specify a combination of units. Values in this `enum` are equal to the corresponding constants in the `CFCalendarUnit enum`.

**Declared In**
`NSCalendar.h`

## NSDateComponents wrapping behavior

The wrapping option specifies wrapping behavior for calculations involving `NSDateComponents` objects.

```
enum
{
    NSWrapCalendarComponents = kCFCalendarComponentsWrap,
};
```

**Constants**
`NSWrapCalendarComponents`
Specifies that the components specified for an `NSDateComponents` object should be incremented and wrap around to zero/one on overflow, but should not cause higher units to be incremented.

Available in Mac OS X v10.4 and later.

Declared in `NSCalendar.h`.

**Declared In**
`NSCalendar.h`

# Document Revision History

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

| Date | Notes |
| --- | --- |
| 2009-02-04 | Clarified return value of ordinalityOfUnit:inUnit:forDate:. |
| 2008-06-09 | Removed reference to NSCalendarDate, which is slated for deprecation. Added explanation of weekday ordinal unit. |
| 2007-06-06 | Corrected discussion of currentCalendar. |
| 2007-05-21 | Included API introduced in Mac OS X v10.5. |
| 2007-03-06 | Noted that NSCalendarDate does not implement Julian calendar. |
| 2006-10-03 | Clarified options argument to dateByAddingComponents:toDate:options:. |
| 2006-05-23 | First publication of this content as a separate document. |

Document Revision History

# Index

## T