NSCalendarDate

Inherits From:	NSDate : NSObject
Conforms To:	NSCoding, NSCopying (NSDate) NSObject (NSObject)
Declared In:	Foundation/NSDate.h

lass Description

NSCalendarDate is a public subclass of NSDate that creates concrete date objects for western calendars. These objects have time zones and calendar formats as attributes and are especially suited for representing and manipulating dates according to western calendrical systems.

An NSCalendarDate object stores a date as the number of seconds relative to the absolute reference date (the first instance of 1 January 2001, GMT). It uses an NSTimeZone object to adjust the visible representation of that date to reflect its associated time zone. The time zone object only affects the way the date is displayed; it does not affect the value stored. Because the value is stored independent of the time zone, you can accurately compare NSCalendarDates with any other NSDate objects or use them to create other NSDate objects. It also means that you can track an NSCalendarDate object across different time zones; that is, as you can change the NSTimeZone object to see how the particular date is represented in that time zone.

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

To retrieve conventional elements of an NSCalendarDate object, use the **...Of...** methods. For example, **dayOfWeek** returns a number that indicates the day of the week (0 is Sunday). The **monthOfYear** method returns a number between 1 and 12 that indicates the month.

NSCalendarDate performs date computations based on western calendrical systems, primarily the Gregorian. (The algorithms are derived from public domain software described in “Calendrical Calculations,” a two-part series by Nachum Dershowitz and Edward M Reingold in *Software — Practice and Experience*).

The Calendar Format

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

The date conversion specifiers cover a range of date conventions:

%%	a '%' character
%a	abbreviated weekday name
%A	full weekday name
%b	abbreviated month name
%B	full month name
%c	shorthand for %X %x, the locale format for date and time
%d	day of the month as a decimal number (01-31)
%e	same as %d but does not print the leading 0 for days 1 through 9
%F	milliseconds as a decimal number (000-999)
%H	hour based on a 24-hour clock as a decimal number (00-23)
%I	hour based on a 12-hour clock as a decimal number (01-12)
%j	day of the year as a decimal number (001-366)
%m	month as a decimal number (01-12)
%M	minute as a decimal number (00-59)
%p	AM/PM designation for the locale
%S	second as a decimal number (00-59)
%w	weekday as a decimal number (0-6), where Sunday is 0
%x	date using the date representation for the locale
%X	time using the time representation for the locale
%y	year without century (00-99)
%Y	year with century (such as 1990)
%Z	time zone abbreviation (such as PDT)
%z	time zone offset in hours and minutes from GMT (HHMM)

NSString Representations for NSCalendarDates

NSCalendarDate provides several **description...** methods for representing dates as strings. These methods—**description**, **descriptionWithLocale:**, **descriptionWithCalendarFormat:**, and **descriptionWithCalendarFormat:locale:**—take an implicit or explicit calendar format. The user's locale information affects the returned string. NSCalendarDate accesses the locale information as an NSDictionary. If you use **descriptionWithLocale:** or **descriptionWithCalendarFormat:locale:**, you can specify a different locale dictionary. The following keys in the locale dictionary affect NSCalendarDates:

NSTimeDateFormatString	Specifies how dates with times are printed, affecting strings that use the format specifiers %c, %X, or %x. The default is to use abbreviated months and days with a 24 hour clock, as in “Sun Jan 01 23:00:00 +6 2001”.
NSAMPMDesignation	Specifies how the morning and afternoon designations are printed, affecting strings that use the %p format specifier. The default is AM and PM.
NSMonthNameArray	Specifies the names for the months, affecting strings that use the %B format specifier.
NSShortMonthNameArray	Specifies the abbreviations for the months, affecting strings that use the %b format specifier.
NSWeekDayNameArray	Specifies the names for the days of the week, affecting strings that use the %A format specifier.
NSShortWeekDayNameArray	Specifies the abbreviations for the days of the week, affecting strings that use the %a format specifier.

For more information on the locale dictionary, see the file **Locales.rtf** in the *Foundation Framework Reference*.

If you subclass NSCalendarDate and override **description**, you should also override **descriptionWithLocale:**. NSString's **stringWithFormat:** method uses **descriptionWithLocale:** instead of **description** when you use the %@ conversion specifier to print an NSCalendarDate. That is, this message:

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

will invoke **descriptionWithLocale:**.

Method Types

Creating an NSCalendarDate instance	
	+ calendarDate
	+ dateWithString:calendarFormat:
	+ dateWithString:calendarFormat:locale:
	+ dateWithYear:month:day:hour:minute:second:timeZone:
	– initWithString:
	– initWithString:calendarFormat:
	– initWithString:calendarFormat:locale:
	– initWithYear:month:day:hour:minute:second:timeZone:
Retrieving date elements	– dayOfMonth
	– dayOfWeek
	– dayOfYear
	– hourOfDay
	– minuteOfHour
	– monthOfYear
	– secondOfMinute
	– yearOfCommonEra
	– dayOfCommonEra
Adjusting a date	– dateByAddingYears:months:days:hours:minutes:seconds:
Computing date intervals	– years:months:days:hours:minutes:seconds:sinceDate:
Representing dates as NSStrings	– description
	– descriptionWithCalendarFormat:
	– descriptionWithCalendarFormat:locale:
	– descriptionWithLocale:
Getting and setting calendar formats	– setCalendarFormat:
	– calendarFormat
Getting and setting time zones	– timeZoneDetail
	– setTimeZone:

Class Methods

calendarDate

+ (id)**calendarDate**

Creates and returns an NSCalendarDate initialized to the current date and time.

See also: + **date** (NSDate)

dateWithString:calendarFormat:

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

Creates and returns an NSCalendarDate initialized with the date specified in the string *description*. NSCalendarDate uses *format* both to interpret the *description* string and as the default calendar format for this new object. *format* consists of conversion specifiers similar to those used in **strftime()**. See the class description for a discussion of date conversion specifiers. If *description* does not match *format* exactly, this method returns **nil**.

For example, let's say your company's convention for dates on correspondence takes the form "Friday, 1 July 1994, 11:45 AM." To get an NSCalendarDate object with a temporal value corresponding to this string, you would use this statement:

```
NSCalendarDate *today = [NSCalendarDate
    dateWithString:@"Friday, 1 July 1994, 11:45 AM"
    calendarFormat:@"%A, %d %B %Y, %I:%M %p"];
```

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

See also: + **dateWithString:calendarFormat:locale:**, – **calendarFormat**, – **initWithString:calendarFormat:**

dateWithString:calendarFormat:locale:

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

Creates and returns an NSCalendarDate initialized with the date specified in the string *description*. NSCalendarDate uses *format* both to interpret the *description* string and as the default calendar format for this new object. *format* consists of conversion specifiers similar to those used in **strftime()**. The keys and values that represent the locale data in *locale* are used when parsing the string. See the class description for a list of the date conversion specifiers and appropriate locale dictionary keys. If *description* does not match *format* exactly, this method returns **nil**.

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

See also: + **dateWithString:calendarFormat:**, – **calendarFormat**, – **initWithString:calendarFormat:locale:**

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

+ (id)**dateWithYear:**(int)*year*
  month:(unsigned)*month*
  day:(unsigned)*day*
  hour:(unsigned)*hour*
  minute:(unsigned)*minute*
  second:(unsigned)*second*
  timeZone:(NSTimeZone *)*aTimeZone*

Creates and returns an NSCalendarDate initialized with the specified values for year, month, day, hour, minute, and second and the NSTimeZone object. The *year* value must include the century (for example, 1995 instead of 95). The other values are the standard ones: 1 through 12 for months, 1 through 31 for days, 0 through 23 for hours and 0 through 59 for both minutes and seconds.

The method verifies the time zone *aTimeZone* and can substitute an alternative time zone. If the method does supply a new time zone, it applies the difference in offsets-from-GMT values between the substituted and the original time zone to the NSCalendarDate being created.

The following code fragment shows an NSCalendarDate created with a date on the fourth of July, 9 PM, Eastern Standard Time (**timeZoneWithName:** returns the NSTimeZone object that represents the time zone with the specified name).

```
NSCalendarDate *fireworks = [NSCalendarDate dateWithYear:1994 month:7
    day:4 hour:21 minute:0 second:0
    timeZone:[NSTimeZone timeZoneWithName:@"EST"]];
```

See also: – **initWithYear:month:day:hour:minute:second:timeZone:**

nstance Methods

calendarFormat

– (NSString *)**calendarFormat**

Returns the receiver's default calendar format (used when the format is unspecified). You can set this format when you create the NSCalendarDate using one of the class methods **dateWithString:calendarFormat:** or **dateWithString:calendarFormat:locale:**, or you can change the format using the instance method **setCalendarFormat:**. If you do not specify a default calendar format, NSCalendarDate substitutes its own default: an international format of "%Y-%m-%d %H:%M:%S %z" (for example, 1994-01-14 16:45:12 +0900). See the class description for a discussion of date conversion specifiers.

See also: – **description**, – **descriptionWithLocale:**

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

– (NSCalendarDate *)**dateByAddingYears:**(int)*year*
months:(int)*month*
days:(int)*day*
hours:(int)*hour*
minutes:(int)*minute*
seconds:(int)*second*

Returns an NSCalendarDate that is updated with the year, month, day, hour, minute, and second offsets specified as arguments. The offsets can be positive (future) or negative (past). This method preserves "clock time" across changes in Daylight Savings Time zones and leap years. For example, adding one month to an NSCalendarDate with a time of 12 noon correctly maintains time at 12 noon.

The following code fragment shows an NSCalendarDate created with a date a week later than an existing NSCalendarDate.

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

See also: – **years:months:days:hours:minutes:seconds:sinceDate:**

dayOfCommonEra

– (int)**dayOfCommonEra**

Returns the number of days since the beginning of the Common Era. The base year of the Common Era is 1 A.C.E. (which is the same as 1 A.D.).

See also: – **dayOfMonth**, – **dayOfWeek**, – **dayOfYear**, – **hourOfDay**, – **minuteOfHour**, – **monthOfYear**, – **secondOfMinute**, – **yearOfCommonEra**

dayOfMonth

– (int)**dayOfMonth**

Returns a number that indicates the day of the month (1 through 31) of the receiver.

See also: – **dayOfCommonEra**, – **dayOfWeek**, – **dayOfYear**, – **hourOfDay**, – **minuteOfHour**, – **monthOfYear**, – **secondOfMinute**, – **yearOfCommonEra**

dayOfWeek

– (int)**dayOfWeek**

Returns a number that indicates the day of the week (0 through 6) of the receiver; 0 indicates Sunday.

See also: – **dayOfCommonEra**, – **dayOfMonth**, – **dayOfYear**, – **hourOfDay**, – **minuteOfHour**, – **monthOfYear**, – **secondOfMinute**, – **yearOfCommonEra**

dayOfYear

– (int)**dayOfYear**

Returns a number that indicates the day of the year (1 through 366) of the receiver.

See also: – **dayOfCommonEra**, – **dayOfMonth**, – **dayOfWeek**, – **hourOfDay**, – **minuteOfHour**, – **monthOfYear**, – **secondOfMinute**, – **yearOfCommonEra**

description

– (NSString *)**description**

Returns a string representation of the receiver. The string is formatted as specified by the receiver's default calendar format. You can find out what the default calendar format is using the method **calendarFormat**.

Because NSCalendarDate implements **descriptionWithLocale:**, **descriptionWithLocale:** is used to print the date when you use the %@ conversion specifier. That is, the following statement invokes **descriptionWithLocale:**, not **description**.

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

See also: – **descriptionWithCalendarFormat:**, – **descriptionWithCalendarFormat:locale:**, – **descriptionWithLocale:**, – **setCalendarFormat:**

descriptionWithCalendarFormat:

– (NSString *)**descriptionWithCalendarFormat:**(NSString *)*format*

Returns a string representation of the receiver. The string is formatted as specified by the conversion specifiers in the calendar format string *format*. The conversion specifiers cover a range of date conventions. See the class description for a listing of these specifiers.

This example displays the current date formatted as "Tues 3/1/94 3:30 PM" in a text field:

```
NSCalendarDate *now = [NSCalendarDate calendarDate];
NSString *datestr =
    [now descriptionWithCalendarFormat:@"%a %m/%d/%y %I:%M %p"];
```

```
[dateField setStringValue:datestr];
```

See also: – **description**, – **descriptionWithCalendarFormat:locale:**, – **descriptionWithLocale:**

descriptionWithCalendarFormat:locale:

– (NSString *)**descriptionWithCalendarFormat:**(NSString *)*format* **locale:**(NSDictionary *)*locale*

Returns a string representation of the receiver. The string is formatted according to the conversion specifiers in *format* and represented according to the locale information in *locale*. See the class description for a list of the date conversion specifiers and appropriate locale dictionary keys.

See also: – **description**, – **descriptionWithCalendarFormat:**, – **descriptionWithLocale:**

descriptionWithLocale:

– (NSString *)**descriptionWithLocale:**(NSDictionary *)*locale*

Returns a string representation of the receiver. The string is formatted as specified by the receiver's default calendar format and represented according to the locale information in *locale*. You can find out what the default calendar format is using the method **calendarFormat**. See the class description for a list of the locale dictionary keys that affect calendar formats.

This method is used to print the NSCalendarDate when the %@ conversion specifier is used. That is, this statement invokes **descriptionWithLocale:**.

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

See also: – **description**, – **descriptionWithCalendarFormat:**,
– **descriptionWithCalendarFormat:locale:**, – **setCalendarFormat:**

hourOfDay

– (int)**hourOfDay**

Returns the hour value (0 through 23) of the receiver. On Daylight Savings "fall back" days, a value of 1 is returned for two consecutive hours, but with a different time zone (the first in daylight savings time and the second in standard time).

See also: – **dayOfCommonEra**, – **dayOfMonth**, – **dayOfWeek**, – **dayOfYear**, – **minuteOfHour**,
– **monthOfYear**, – **secondOfMinute**, – **yearOfCommonEra**

initWithString:

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

Returns an NSCalendarDate initialized with the date specified as a string in *description*. The *description* string must conform to the default calendar format “%Y-%m-%d %H:%M:%S %z” (for example, 1994-01-14 16:45:12 +0900). If *description* does not match this format exactly, this method returns **nil**. See the class description for a discussion of date conversion specifiers.

See also: + **dateWithString:** (NSDate)

initWithString:calendarFormat:

– (id)**initWithString:**(NSString *)*description* **calendarFormat:**(NSString *)*format*

Returns an NSCalendarDate initialized with the date specified as a string in *description*. This method uses *format* both to interpret the *description* string and as the default calendar format for this object. *format* consists of conversion specifiers similar to those used in **strftime()**. See the class description for a listing of these specifiers. If *description* does not match *format* exactly, this method returns **nil**.

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

For example, let’s assume you want to initialize an NSCalendarDate object with a string obtained from a text field. This date string takes the form “03.21.94 22:00 PST”:

```
NSCalendarDate *newDate = [[[NSCalendarDate alloc]
    initWithString:[dateField stringValue]
    calendarFormat:@"%m.%d.%y %H:%M %Z"] autorelease];
```

See also: + **dateWithString:calendarFormat:, – calendarFormat**

initWithString:calendarFormat:locale:

– (id)**initWithString:**(NSString *)*description*
    calendarFormat:(NSString *)*format*
    locale:(NSDictionary *)*locale*

Returns an NSCalendarDate initialized with the date specified in the string *description*. NSCalendarDate uses *format* both to interpret the *description* string and as the default calendar format for this object. *format* consists of conversion specifiers similar to those used in **strftime()**. The keys and values that represent the locale data from *locale* are used when parsing the string. See the class description for a list of the date conversion specifiers and appropriate locale dictionary keys.

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

If you specify a locale dictionary that has a month name array with more than 12 elements or a day name array with more than 7 arguments, **initWithString:calendarFormat:locale:** returns **nil**.

See also: **+ dateWithString:calendarFormat:locale:, – calendarFormat**

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

– (id)**initWithYear:**(int)*year*
month:(unsigned)*month*
day:(unsigned)*day*
hour:(unsigned)*hour*
minute:(unsigned)*minute*
second:(unsigned)*second*
timeZone:(NSTimeZone *)*aTimeZone*

Returns an NSCalendarDate initialized with the specified values for year, month, day, hour, minute, and second and the NSTimeZone object. The *year* value must include the century (for example, 1995 instead of 95). The other values are the standard ones: 1 through 12 for months, 1 through 31 for days, 0 through 23 for hours and 0 through 59 for both minutes and seconds.

The method verifies the time zone supplied as an argument and can substitute an alternative time zone. If the method does supply a new time zone, it applies the difference in offsets-from-GMT values between the substituted and the original time zone to the NSCalendarDate being created.

The following code fragment shows an NSCalendarDate created with a date on the fourth of July, 9 PM, Eastern Standard Time (**timeZoneWithName:** returns the NSTimeZone object that represents the time zone with the specified name).

```
NSCalendarDate *fireworks = [[[NSCalendarDate alloc] initWithYear:1994
   month:7 day:4 hour:21 minute:0 second:0
   timeZone:[NSTimeZone timeZoneWithName:@"EST"]] autorelease];
```

See also: **– dateWithYear:month:day:hour:minute:second:timeZone:**

minuteOfHour

– (int)**minuteOfHour**

Returns the minutes value (0 through 59) of the receiver.

See also: – **dayOfCommonEra**, – **dayOfMonth**, – **dayOfWeek**, – **dayOfYear**, – **hourOfDay**, – **monthOfYear**, – **secondOfMinute**, – **yearOfCommonEra**

monthOfYear

– (int)**monthOfYear**

Returns a number that indicates the month of the year (1 through 12) of the receiver.

See also: – **dayOfCommonEra**, – **dayOfMonth**, – **dayOfWeek**, – **dayOfYear**, – **hourOfDay**, – **minuteOfHour**, – **secondOfMinute**, – **yearOfCommonEra**

secondOfMinute

– (int)**secondOfMinute**

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

See also: – **dayOfCommonEra**, – **dayOfMonth**, – **dayOfWeek**, – **dayOfYear**, – **hourOfDay**, – **minuteOfHour**, – **monthOfYear**, – **yearOfCommonEra**

setCalendarFormat:

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

Sets the default calendar format for the receiver. A calendar format is a string formatted with date-conversion specifiers listed in the class description. If you do not specify a calendar format for an object, NSCalendarDate substitutes its own default. This is the international format of “%Y-%m-%d %H:%M:%S %z” (for example, 1994-01-14 16:45:12 +0900).

See also: – **calendarFormat**, – **description**, – **descriptionWithLocale:**

setTimeZone:

– (void)**setTimeZone:**(NSTimeZone *)*aTimeZone*

Sets the time zone for the receiver. If you do not specify a time zone for an object at initialization time, NSCalendarDate uses the default time zone for the locale. Use this method to set it to another time zone.

See also: – **timeZoneDetail**

timeZoneDetail

– (NSTimeZoneDetail *)**timeZoneDetail**

Returns the time-zone detail object that is associated with the receiver. You can set the time zone when you create the NSCalendarDate using the class methods **dateWithString:calendarFormat:** or **dateWithString:calendarFormat:locale:** by including the time zone in the description and format arguments. Or you can explicitly set the time zone to an NSTimeZone object using **dateWithYear:month:day:hour:minute:second:timeZone:**. If you do not specify a time zone for an object at initialization time, NSCalendarDate uses the default time zone for the locale.

See also: – **setTimeZone:**

yearOfCommonEra

– (int)**yearOfCommonEra**

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

See also: – **dayOfCommonEra**, – **dayOfMonth**, – **dayOfWeek**, – **dayOfYear**, – **hourOfDay**, – **minuteOfHour**, – **monthOfYear**, – **secondOfMinute**

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

– (void)**years:**(int *)*yearsPointer*
months:(int *)*monthsPointer*
days:(int *)*daysPointer*
hours:(int *)*hoursPointer*
minutes:(int *)*minutesPointer*
seconds:(int *)*secondsPointer*
sinceDate:(NSDate *)*date*

Computes the calendrical time difference between the receiver and *date* and returns it in *yearsPointer*, *monthsPointer*, *daysPointer*, *hoursPointer*, *minutesPointer*, and *secondsPointer*. You can choose any representation you wish for the time difference by passing NULL for the arguments you want to ignore. For example, the following code fragment computes the difference in months, days, and years between two dates:

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

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

This message returns 29 years, 10 months, and 29 days. If you want to express the years in terms of months, you pass NULL for the years argument:

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

This message returns 358 months and 29 days.

See also: – **dateByAddingYears:months:days:hours:minutes:seconds:**