Date and Time Programming Guide for Core Foundation

Developer

Contents

Listings

Introduction to Dates and Times Programming Guide for Core Foundation

Important: This is a preliminary document for an API or technology in development. Apple is supplying this information to help you plan for the adoption of the technologies and programming interfaces described herein for use on Apple-branded products. This information is subject to change, and software implemented according to this document should be tested with final operating system software and final documentation. Newer versions of this document may be provided with future betas of the API or technology.

Core Foundation provides date and time objects you can use without depending on operating-system internals.

Who Should Read This Document

You should read this document to learn about representations of date and time in the Core Foundation framework, and how to use them.

Organization of This Document

This topic discusses the different date and time representations, issues with time zones, and contains examples on how to use date and time objects:

Date Representations

Core Foundation allows you to work with five different representations of time:

- CFAbsoluteTime, a specific point in time relative to 1 January 2001 00:00:00 GMT
- CFTimeInterval, an interval of time in seconds
- CFGregorianDate, a specific point in time represented using the Gregorian calender
- CFGregorianUnits, an interval of time in one or more of the units used in the Gregorian calendar
- CFDate, an absolute time in the format of a Core Foundation opaque type

CFAbsoluteTime

CFAbsoluteTime is useful when you need to refer to a specific point in time. A CFAbsoluteTime value represents time as a number of seconds relative to the reference date of 1 January, 2001 00:00:00 GMT. A positive value represents a date after the reference date, a negative value represents a date before it.

Absolute time can be confusing at first because an absolute time value is literally a time interval (the number of seconds since the reference date), but it is interpreted as a specific instant in time. For example, the absolute time -32940326 indicates both a date and time—December 16th, 1999 at 17:54:34. An absolute time value cannot be used to refer to a date or a clock time independently, it always includes both.

CFAbsoluteTime is implemented as a `double` and can be compared with another absolute time using the standard C comparison operators.

CFTimeInterval

CFTimeInterval is appropriate when you need to measure duration. A CFTimeInterval represents elapsed time in seconds. As with CFAbsoluteTime, a CFTimeInterval is implemented using the C type `double` and so you can compare two CFTimeInterval values using the standard C comparison operators.

CFGregorianDate

CFGregorianDate represents time using the Gregorian calendar that has been in general use in Europe and the Western Hemisphere since 1582. A CFGregorianDate is implemented as a C structure with separate fields for years, months, days, hours, minutes, and seconds. You can check any or all of the fields of a CFGregorianDate for validity. A Gregorian date can also be converted to and from an absolute time.

CFGregorianUnits

CFGregorianUnits is analogous to a CFTimeInterval in that it represents a duration rather than a specific point in time. Like CFGregorianDate, CFGregorianUnits is implemented as a C structure, but the data types of the fields are different to allow for larger values. For example, a CFGregorianDate will never have more than 52 weeks, or 24 hours, so the fields of the CFGregorianDate structure are implemented using the smallest data type appropriate for its maximum value. Because CFGregorianUnits is intended to represent arbitrary time intervals, it is implemented with 32 bit integers (except for seconds, which is of type `double` to allow for fractional values).

CFDate

If you need to place a date in a Core Foundation property list, it must be of type CFDate. A CFDate object is simply an absolute time "wrapped" as a Core Foundation opaque type. A Gregorian date must first be converted to an absolute time, and then it can become a CFDate object. A CFDate object can be compared with another CFDate using a standard Core Foundation comparison function. Note that a CFDate can only be created with an absolute time, CFTimeInterval values are not supported. Use a CFNumber to wrap ordinary floating point values like a CFTimeInterval. CFDate objects are immutable.

Time Zones

CFDate objects are all expressed as Greenwich Mean Time, or GMT. In order to convert a GMT date to your local time you must use a CFTimeZone object. A CFTimeZone represents a geopolitical region that has some temporal offset, either plus or minus, from GMT as well as an abbreviation—such as "PST". In addition to familiar abbreviations, time zones are also named by country and region. For example the United States spans these times zones:

USA Eastern: -5 hours GMT

USA Indiana East: -5 hours GMT

USA Central: -6 hours GMT

USA Mountain: -7 hours GMT

USA Arizona: -7 hours GMT

USA Pacific: -8 hours GMT

USA Alaska: -9 hours GMT

USA Aleutian: -10 hours GMT

USA Hawaii: -10 hours GMT

To make matters even more complex, any region may or may not be on Daylight Savings Time (DST).

In order to properly convert GMT to local time, you have to know which time zone you are in and if DST is in effect. Core Foundation uses time zone names, abbreviations, GMT offset, and DST information for a particular time zone obtained from a public-domain database maintained at `ftp://elsie.nci.nih.gov//pub/`. This database contains information representing the history of local time for many representative locations around the globe. The database is updated periodically to reflect changes made to GMT offsets and daylight-saving rules by political entities.

For examples of how to use CFDate and CFTimeZone, see the Using Dates (page 8).

Using Dates

This task contains examples on creating, comparing, and converting dates. Listing 1 (page 8) shows you how to get the current absolute time and convert it into a CFDate object.

Listing 1 Creating a CFDate object

```
CFAbsoluteTime      absTime;
CFDateRef           aCFDate;

absTime = CFAbsoluteTimeGetCurrent();
aCFDate = CFDateCreate(kCFAllocatorDefault, absTime);
```

To compare two dates, use the compare function `CFDateCompare` as shown in Listing 2 (page 8).

Listing 2 Comparing two CFDate objects

```
// Standard Core Foundation comparison result.
CFComparisonResult result;

// Create two CFDates from absolute time.
date1 = CFDateCreate(kCFAllocatorDefault, CFAbsoluteTimeGetCurrent());
date2 = CFDateCreate(kCFAllocatorDefault, CFAbsoluteTimeGetCurrent());

// Pass NULL for the context param.
result = CFDateCompare(date1, date2, NULL);

switch (result) {
    case kCFCompareLessThan:
        printf("date1 is before date2!\n");
        break;
    case kCFCompareEqualTo:
        printf("date1 is the same as date2!\n");
```

```
        break;
    case kCFCompareGreaterThan:
        printf("date1 is after date2!\n");
        break;
    }
```

The `CFDateCompare` function performs exact comparisons, which means it detects sub-second differences between dates. You might want to compare dates with a less fine granularity. For example, you might want to consider two dates equal if they are within one minute of each other. This can be accomplished by simply converting the CFDates to absolute time and comparing the two floating-point values using your fuzziness factor. To compare Gregorian units like month or week, you can convert both CFDates to CFGregorianDate and compare the appropriate fields. Converting absolute time to and from Gregorian dates is quite simple. Listing 3 (page 9) demonstrates how to do this.

Listing 3 Working with Gregorian dates

```
Boolean             status;
CFGregorianDate     gregDate;
CFAbsoluteTime      absTime;

// Construct a Gregorian date.
gregDate.year = 1999;
gregDate.month = 11;
gregDate.day = 23;
gregDate.hour = 17;
gregDate.minute = 33;
gregDate.second = 22.7;

// Check the validity of the date.
status = CFGregorianDateIsValid(gregDate, kCFGregorianAllUnits);
printf("Is my Gregorian date valid? %d\n", status);

// Convert the Gregorian date to absolute time.
absTime = CFGregorianDateGetAbsoluteTime(gregDate, NULL);
printf("The Absolute Time from a Gregorian date is: %d\n", absTime);
```

Document Revision History

This table describes the changes to *Date and Time Programming Guide for Core Foundation* .

Date	Notes
2005-08-11	Changed title from "Dates and Times." Added links to formatting references.
2003-01-17	Converted existing Core Foundation documentation into topic format. Added revision history.

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