Date, Time, and Measurement Utilities Reference

Internationalization > Carbon

2006-09-29

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

Simultaneously published in the United States and Canada.

Contents

Date, Time, and Measurement Utilities Reference

Framework:	CoreServices/CoreServices.h
Declared in	DateTimeUtils.h QuickTimeComponents.k.h UTCUtils.h

Overview

You can use the Date, Time, and Measurement Utilities to manipulate the date-time information and geographic location data used by a Macintosh computer. A Macintosh computer contains a battery-operated clock chip that maintains information on the current date-time. In Mac OS 9, the date and time were retrieved from this clock chip. In Mac OS X, the date and time are retrieved through the BSD/CoreFoundation level.

You can use the routines provided by the Date, Time, and Measurement Utilities to

- get the current date and time
- set the current date and time, if necessary
- convert between internal date-time structures
- get and set the geographic location and time-zone information
- determine the number of elapsed microseconds since system startup

Please note, setting the time or geographical location requires authorization by using Authorization Services. See Authorization Concepts for more information.

To make the best use of the Date, Time, and Measurement Utilities, you should be familiar with the international resources, especially the numeric-format and long-date-format resources, and the Script Manager.

Carbon supports the majority of the Date, Time, and Measurement Utilities. However, obsolete functions that are prefixed with "iu" or "IU" (such as `IUDateString` and `IUTimeString`) are not supported.

Functions by Task

Converting Between Date-Time Formats

`DateToSeconds` (page 28) Deprecated in Mac OS X v10.3
Converts a date and time to a number of seconds elapsed since midnight, January 1, 1904. (Deprecated. Use the `CFCalendarRef` data type and the functions that operate on it instead.)

`SecondsToDate` (page 33) Deprecated in Mac OS X v10.3
Converts a number of seconds elapsed since midnight, January 1, 1904 to a date and time. (Deprecated. Use the `CFCalendarRef` data type and the functions that operate on it instead.)

Converting Between Long Date-Time Format

`LongDateToSeconds` (page 31) Deprecated in Mac OS X v10.3
Converts a date and time to the number of seconds elapsed since midnight, January 1, 1904. (Deprecated. Use the `CFCalendarRef` data type and the functions that operate on it instead.)

`LongSecondsToDate` (page 31) Deprecated in Mac OS X v10.3
Converts the number of seconds elapsed since midnight, January 1, 1904 to a date and time. (Deprecated. Use the `CFCalendarRef` data type and the functions that operate on it instead.)

Converting Date and Time Strings Into Numeric Representations

`InitDateCache` (page 29) Deprecated in Mac OS X v10.3
Initializes the date cache structure, which is used to store data for use by the `StringToDate` and `StringToTime` functions. (Deprecated. There is no replacement.)

`StringToDate` (page 35) Deprecated in Mac OS X v10.3
Parses a string for a date and converts the date information into values in a date-time structure. (Deprecated. Use `CFDateFormatterCreateDateFromString` instead.)

`StringToTime` (page 37) Deprecated in Mac OS X v10.3
Parses a string for a time specification and converts the date information into values in a date-time structure. (Deprecated. Use `CFDateFormatterCreateDateFromString` instead.)

Converting Long Date and Time Values Into Strings

`LongDateString` (page 30) Deprecated in Mac OS X v10.3
Converts a date that is specified as a `LongDateTime` value into a Pascal string, making use of the date formatting information in the specified resource. (Deprecated. Use `CFDateFormatterCreateStringWithDate` instead.)

`LongTimeString` (page 32) Deprecated in Mac OS X v10.3
Converts a time that is specified as a `LongDateTime` value into a Pascal string, making use of the time formatting information in the specified resource. (Deprecated. Use `CFDateFormatterCreateStringWithDate` instead.)

Converting Numeric Representations Into Date and Time Strings

`DateString` (page 27) Deprecated in Mac OS X v10.3
Converts a date in the standard date-time representation into a Pascal string, making use of the date formatting information in the specified resource. (Deprecated. Use `CFDateFormatterCreateStringWithDate` instead.)

`TimeString` (page 38) Deprecated in Mac OS X v10.3
Converts a time in the standard date-time representation into a string, making use of the time formatting information in the specified resource. (Deprecated. Use `CFDateFormatterCreateStringWithDate` instead.)

Converting Between CF and Carbon Time Types

`UCConvertCFAbsoluteTimeToUTCDateTime` (page 10)
Converts a value of type `CFAbsoluteTime` to `UTCDateTime`.

`UCConvertCFAbsoluteTimeToSeconds` (page 9)
Converts a value of type `CFAbsoluteTime` to seconds.

`UCConvertCFAbsoluteTimeToLongDateTime` (page 9)
Converts a value of type `CFAbsoluteTime` to `LongDateTime`.

`UCConvertLongDateTimeToCFAbsoluteTime` (page 11)
Converts a value of type `LongDateTime` to `CFAbsoluteTime`.

`UCConvertSecondsToCFAbsoluteTime` (page 11)
Converts a value from the normal seconds time representation to `CFAbsoluteTime`.

`UCConvertUTCDateTimeToCFAbsoluteTime` (page 12)
Converts a value of type `UTCDateTime` time to `CFAbsoluteTime`.

Converting Between UTC and Local Time

`ConvertLocalTimeToUTC` (page 41) Deprecated in Mac OS X v10.4
Converts local time to UTC. (Deprecated. Use `CFTimeZoneGetSecondsFromGMT` instead.)

`ConvertLocalToUTCDateTime` (page 42) Deprecated in Mac OS X v10.4
Converts local date and time to UTC date and time. (Deprecated. Use `CFTimeZoneGetSecondsFromGMT` instead.)

`ConvertUTCToLocalDateTime` (page 42) Deprecated in Mac OS X v10.4
Converts UTC date and time to local date and time. (Deprecated. Use `CFTimeZoneGetSecondsFromGMT` instead.)

`ConvertUTCToLocalTime` (page 43) Deprecated in Mac OS X v10.4
Converts UTC time to local time. (Deprecated. Use `CFTimeZoneGetSecondsFromGMT` instead.)

Getting the Current Date and Time

`GetTime` (page 8)
Obtains the current date-time information, expressed as a date and time. (Deprecated. Use `CFAbsoluteTimeGetCurrent` instead.)

`GetDateTime` (page 28) Deprecated in Mac OS X v10.3
Obtains the current date-time information, expressed as the number of seconds elapsed since midnight, January 1, 1904. (Deprecated. Use `CFAbsoluteTimeGetCurrent` instead.)

`ReadDateTime` (page 33) Deprecated in Mac OS X v10.3
Reads time information from the system. (Deprecated. Use `CFAbsoluteTimeGetCurrent` instead.)

`GetLocalDateTime` (page 43) Deprecated in Mac OS X v10.4
: Gets the local date and time. (Deprecated. Use `CFAbsoluteTimeGetCurrent` and `CFTimeZoneGetSecondsFromGMT` instead.)

`GetUTCDateTime` (page 44) Deprecated in Mac OS X v10.4
: Gets the UTC date and time. (Deprecated. Use `CFAbsoluteTimeGetCurrent` instead.)

Modifying and Verifying Long Date-Time Records

`ToggleDate` (page 38) Deprecated in Mac OS X v10.3
: Modifies a date and time, by modifying one specific component of a date and time (day, hour, minute, seconds, day of week, and so on). (Deprecated. Use the `CFCalendarRef` data type and the functions that operate on it instead.)

`ValidDate` (page 40) Deprecated in Mac OS X v10.3
: Verifies specific date and time values in a long date-time structure. (Deprecated. Use the `CFCalendarRef` data type and the functions that operate on it instead.)

Setting the Current Date and Time

`SetDateTime` (page 34) Deprecated in Mac OS X v10.3
: Changes the date-time information stored by the system to the specified value, expressed as the number of seconds elapsed since midnight, January 1, 1904. (Deprecated. There is no replacement.)

`SetTime` (page 35) Deprecated in Mac OS X v10.3
: Changes the date-time information in the system to the specified value, expressed as a date and time. (Deprecated. There is no replacement.)

`SetLocalDateTime` (page 45) Deprecated in Mac OS X v10.4
: Sets the local date and time. (Deprecated. There is no replacement.)

`SetUTCDateTime` (page 45) Deprecated in Mac OS X v10.4
: Sets the UTC date and time. (Deprecated. Use settimeofday(2) instead.)

Functions

GetTime

Obtains the current date-time information, expressed as a date and time. (**Deprecated.** Use `CFAbsoluteTimeGetCurrent` instead.)

```
void GetTime (
   DateTimeRec *d
);
```

Parameters

d
: On return, the fields of the date-time structure contain the current date and time.

Discussion
The `GetTime` function first calls the `GetDateTime` function to obtain the number of seconds elapsed since midnight, January 1, 1904. It then calls the `SecondsToDate` function to convert the number of seconds into a date and time.

As an alternative to using the `GetTime` procedure, you can pass the value of the global variable `Time` to the `SecondsToDate` (page 33) function; a `SecondsToDate(Time)` function call is identical to a `GetTime(d)` function call.

If an application disables interrupts for longer than a second, the date-time information returned by the `GetTime` function might not be exact. The `GetTime` function is intended to provide fairly accurate time information, but not scientifically precise data.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`QuickTimeComponents.k.h`

UCConvertCFAbsoluteTimeToLongDateTime

Converts a value of type `CFAbsoluteTime` to `LongDateTime`.

```
OSStatus UCConvertCFAbsoluteTimeToLongDateTime (
   CFAbsoluteTime iCFTime,
   LongDateTime *oLongDate
);
```

Parameters

iCFTime
A CFAbsoluteTime value that represents the time from which you wish to convert.

oLongDate
A pointer to a value of type `LongDateTime`. On successful return, this will contain the converted time from the `CFAbsoluteTime` input.

Return Value
A result code. See "Date, Time, and Measurement Utilities Result Codes" (page 24).

Discussion
Use `UCConvertCFAbsoluteTimeToLongDateTime` to convert from a `CFAbsoluteTime` to a `LongDateTime`. Remember that the epoch for `LongDateTime` is January 1, 1904 while the epoch for `CFAbsoluteTime` is January 1, 2001.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`DateTimeUtils.h`

UCConvertCFAbsoluteTimeToSeconds

Converts a value of type `CFAbsoluteTime` to seconds.

```
OSStatus UCConvertCFAbsoluteTimeToSeconds (
   CFAbsoluteTime iCFTime,
   UInt32 *oSeconds
);
```

Parameters

iCFTime

A `CFAbsoluteTime` value that represents the time from which you wish to convert.

oSeconds

A pointer to a value of type `UInt32`. On successful return, this contains the converted time from the `CFAbsoluteTime` input.

Return Value

A result code. See "Date, Time, and Measurement Utilities Result Codes" (page 24).

Discussion

Use `UCConvertCFAbsoluteTimeToSeconds` to convert from a `CFAbsoluteTime` to a *`UInt32`* representation of seconds. Remember that the epoch for seconds is January 1, 1904 while the epoch for CFAbsoluteTime is January 1, 2001.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`DateTimeUtils.h`

UCConvertCFAbsoluteTimeToUTCDateTime

Converts a value of type `CFAbsoluteTime` to `UTCDateTime`.

```
OSStatus UCConvertCFAbsoluteTimeToUTCDateTime (
   CFAbsoluteTime iCFTime,
   UTCDateTime *oUTCDate
);
```

Parameters

iCFTime

A `CFAbsoluteTime` value that represents the time from which you wish to convert.

oUTCDate

A pointer to a `UTCDateTime`. On successful return, this will contain the converted time from the `CFAbsoluteTime` input.

Return Value

A result code. See "Date, Time, and Measurement Utilities Result Codes" (page 24).

Discussion

Use `UCConvertCFAbsoluteTimeToUTCDateTime` to convert from a `CFAbsoluteTime` to a `UTCDateTime`. Remember that the epoch for `UTCDateTime` is January 1, 1904 while the epoch for `CFAbsoluteTime` is January 1, 2001.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`DateTimeUtils.h`

UCConvertLongDateTimeToCFAbsoluteTime

Converts a value of type `LongDateTime` to `CFAbsoluteTime`.

```
OSStatus UCConvertLongDateTimeToCFAbsoluteTime (
   LongDateTime iLongTime,
   CFAbsoluteTime *oCFTime
);
```

Parameters

iLongTime

A `LongDateTime` value that represents the time from which you wish to convert.

oCFTime

A pointer to a `CFAbsoluteTime`. On successful return, this will contain the converted time from the input time type.

Return Value

A result code. See "Date, Time, and Measurement Utilities Result Codes" (page 24).

Discussion

Use `UCConvertLongDateTimeToCFAbsoluteTime` to convert from a `LongDateTime` to a `CFAbsoluteTime`. Remember that the epoch for LongDateTime is January 1, 1904 while the epoch for `CFAbsoluteTime` is January 1, 2001.

Availability

Available in Mac OS X v10.2 and later.

Declared In

`DateTimeUtils.h`

UCConvertSecondsToCFAbsoluteTime

Converts a value from the normal seconds time representation to `CFAbsoluteTime`.

```
OSStatus UCConvertSecondsToCFAbsoluteTime (
   UInt32 iSeconds,
   CFAbsoluteTime *oCFTime
);
```

Parameters

iSeconds

A `UInt32` value that represents the time from which you wish to convert.

oCFTime

A pointer to a `CFAbsoluteTime`. On successful return, this will contain the converted time from the input time type.

Return Value

A result code. See "Date, Time, and Measurement Utilities Result Codes" (page 24).

Discussion

Use `UCConvertSecondsToCFAbsoluteTime` to convert from the normal seconds representation of time to a `CFAbsoluteTime`. Remember that the epoch for seconds is January 1, 1904 while the epoch for `CFAbsoluteTime` is January 1, 2001.

Keep in mind that this function converts local time (that is, the time in the local time zone) to GMT/UTC.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`DateTimeUtils.h`

UCConvertUTCDateTimeToCFAbsoluteTime

Converts a value of type `UTCDateTime` time to `CFAbsoluteTime`.

```
OSStatus UCConvertUTCDateTimeToCFAbsoluteTime (
   const UTCDateTime *iUTCDate,
   CFAbsoluteTime *oCFTime
);
```

Parameters

iUTCDate

A pointer to a `UTCDateTime` structure that represents the time from which you wish to convert.

oCFTime

A pointer to a `CFAbsoluteTime`. On successful return, this contains the converted time from the input time type.

Return Value
A result code. See "Date, Time, and Measurement Utilities Result Codes" (page 24).

Discussion
Use `UCConvertUTCDateTimeToCFAbsoluteTime` to convert from a `UTCDDateTime` to a `CFAbsoluteTime`. Remember that the epoch for `UTCDateTime` is January 1, 1904 while the epoch for `CFAbsoluteTime` is January 1, 2001.

Availability
Available in Mac OS X v10.2 and later.

Declared In
`DateTimeUtils.h`

Data Types

DateCacheRecord

```
struct DateCacheRecord {
    short hidden[256];
};
typedef struct DateCacheRecord DateCacheRecord;
typedef DateCacheRecord * DateCachePtr;
```

Fields

`hidden`

The storage used for converting dates and times.

Discussion
The `StringToDate` and `StringToTime` functions use the date cache, defined by the `DateCacheStructure` data type, as an area to store date conversion data that is used by the date conversion functions. This structure must be initialized by a call to the `InitDateCache` (page 29) function. The data in this structure is private—you should not attempt to access it.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`DateTimeUtils.h`

DateDelta

```
typedef SInt8 DateDelta;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`DateTimeUtils.h`

DateTimeRec

```
struct DateTimeRec {
    short year;
    short month;
    short day;
    short hour;
    short minute;
    short second;
    short dayOfWeek;
};
typedef struct DateTimeRec DateTimeRec;
```

Fields

`year`

The year, ranging from 1904 to 2040. Note that to indicate the year 1984, this field would store the integer 1984, not just 84. This field accepts input of 0 or negative values, but these values produce unpredictable results in the `year`, `month`, and `day` fields when you use the `SecondsToDate` and `DateToSeconds` functions. In addition, using `SecondsToDate` and `DateToSeconds` with year values greater than 2040 causes a wraparound to 1904 plus the number of years over 2040. For example, setting the year to 2045 returns a value of 1909, and the other fields in this record return unpredictable results.

`month`

The month of the year, where 1 represents January, and 12 represents December. Values greater than 12 cause a wraparound to a future year and month. This field accepts input of 0 or negative values, but these values produce unpredictable results in the `year`, `month`, and `day` fields when you use the `SecondsToDate` and `DateToSeconds` functions.

`day`

The day of the month, ranging from 1 to 31. Values greater than the number of days in a given month cause a wraparound to a future month and day. This feature is useful for working with leap years. For example, the 366th day of January in 1992 (1992 was a leap year) evaluates as December 31, 1992, and the 367th day of that year evaluates as January 1, 1993.

This field accepts 0 or negative values, but when you use the `SecondsToDate` and `DateToSeconds` procedures, a value of 0 in this field returns the last day of the previous month. For example, a month value of 2 and a day value of 0 return 1 and 31, respectively.

Using `SecondsToDate` and `DateToSeconds` with a negative number in this field subtracts that number of days from the last day in the previous month. For example, a month value of 5 and a day value of –1 return 4 for the month and 29 for the day a month value of 2 and a day value of –15 return 1 and 16, respectively.

`hour`

The hour of the day, ranging from 0 to 23, where 0 represents midnight and 23 represents 11:00 P.M. Values greater than 23 cause a wraparound to a future day and hour. This field accepts input of negative values, but these values produce unpredictable results in the `month`, `day`, `hour`, and `minute` fields you use the `SecondsToDate` and `DateToSeconds` procedures.

`minute`

The minute of the hour, ranging from 0 to 59. Values greater than 59 cause a wraparound to a future hour and minute. When you use the `SecondsToDate` and `DateToSeconds` procedures, a negative value in this field has the effect of subtracting that number from the beginning of the given hour. For example, an `hour` value of 1 and a `minute` value of –10 return 0 hours and 50 minutes. However, if the negative value causes the `hour` value to be less than 0, for example `hour` = 0, `minute` = –61, unpredictable results occur.

`second`

The second of the minute, ranging from 0 to 59. Values greater than 59 cause a wraparound to a future minute and second. When you use the `SecondsToDate` and `DateToSeconds` procedures, a negative value in this field has the effect of subtracting that number from the beginning of the given minute. For example, a `minute` value of 1 and a `second` value of –10 returns 0 minutes and 50 seconds. However, if the negative value causes the `hour` value to be less than 0, for example `hour` = 0, `minute` = 0, and `second` = –61, unpredictable results occur.

`dayOfWeek`

The day of the week, where 1 indicates Sunday and 7 indicates Saturday. This field accepts 0, negative values, or values greater than 7. When you use the `SecondsToDate` and `DateToSeconds` procedures, you get correct values because this field is automatically calculated from the values in the `year`, `month`, and `day` fields.

Discussion

The date-time record describes the date-time information as a date and time. The Date, Time, and Measurement Utilities use a date-time record to read and write date-time information to and from the system.

The date-time record can be used to hold date and time values only for a Gregorian calendar. The long date-time record, `LongDateRec` (page 16), can be used for a Gregorian calendar as well as other calendar systems.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`DateTimeUtils.h`

LocalDateTime

```
struct LocalDateTime {
     UInt16 highSeconds;
     UInt32 lowSeconds;
     UInt16 fraction;
};
typedef struct LocalDateTime LocalDateTime;
typedef LocalDateTime * LocalDateTimePtr;
typedef LocalDateTimePtr * LocalDateTimeHandle;
```

Discussion
`UTCDateTime` and `LocalDateTime` are both 64 bits wide. The first 48 bits represent the number of seconds since 1904. The remaining 16 bits are used to indicate a fractional seconds value, which has no inherent precision. Each unit of this 16-bit value represents 1/65535 of a second. Developers may apply the appropriate arithmetic to derive milliseconds or microseconds.

Note that the decision to have the `lowSeconds` field divided between the high and low 32 bits of the 64 bit structure was intentional. The structure above is perfect for performing 64 bit math and logical comparisons. Having the `lowSeconds` field in the low or high 32 bits would have been easier for the compilers to handle and probably execute faster, however it would have rendered the structure unusable for 64 bit math and logical comparisons.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`UTCUtils.h`

LongDateCvt

```
union LongDateCvt {
     SInt64 c
     struct {
         UInt32 lHigh;
         UInt32 lLow;
     } hl;
};
typedef union LongDateCvt LongDateCvt;
```

Fields

`c`

The date and time, specified in seconds relative to midnight, January 1, 1904, as a signed, 64-bit integer in SANE `comp` format. The high-order bit of this field represents the sign of the 64-bit integer. Negative values allow you to indicate dates and times prior to midnight, January 1, 1904.

`hl`

The high-order 32 bits when converting from a standard date-time value. Set this field to 0.

Discussion
The Date, Time, and Measurement Utilities provide the `LongDateCvt` structure to help in setting up `LongDateTime` values.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`DateTimeUtils.h`

LongDateRec

```
union LongDateRec {
     struct {
         short era;
         short year;
         short month;
         short day;
         short hour;
         short minute;
         short second;
         short dayOfWeek;
         short dayOfYear;
         short weekOfYear;
         short pm;
         short res1;
         short res2;
         short res3;
     } ld;
     short list[14]
     struct {
         short eraAlt;
         DateTimeRec oldDate;
     } od;
};
typedef union LongDateRec LongDateRec;
```

Fields

`era`
The value `0` represents A.D. and `-1` represents B.C

`year`
The year, from 30081 B.C. to 29940 A.D.

`month`
The month (1 = January and 12 = December).

`day`
The day of the month, from 1 to 31.

`hour`
The hour, from 0 to 23.

`minute`
The minute, from 0 to 59.

`second`
The second., from 0 to 59

`dayOfWeek`
The day of the week (1 through 7).

`dayOfYear`
The day of the year, from 1 to 365.

`weekOfYear`
: The week of the year. from 1 through 52.

`pm`
: The value `0` represents AM and the value `1` represents PM.

`res1`
: Reserved.

`res2`
: Reserved.

`res3`
: Reserved.

`list`
: An array `[0 . . 13]` whose values indicate which of the fields in a long date-time record need to be verified.

`eraAlt`
: Indicates the era, used only for conversion from a date-time record to a long date-time record.

`oldDate`
: Used only for conversion from a date-time record to a long date-time record.

Discussion
In addition to the date-time record, system software provides the long date-time record, which extends the date-time record format by adding several more fields. This format lets you use dates and times with a much longer span (30,000 B.C. to 30,000 A.D.). In addition, the long date-time record allows conversions to different calendar systems, such as a lunar calendar.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`DateTimeUtils.h`

LongDateTime

```
typedef SInt64 LongDateTime;
```

Discussion
The long date-time value specifies the date and time as seconds relative to midnight, January 1, 1904. But where the standard date-time value is an unsigned, 32-bit long integer, the long date-time value is a signed, 64-bit integer in SANE `comp` format. This format lets you use dates and times with a much longer span—roughly 500 billion years. You can use this value to represent dates and times prior to midnight, January 1, 1904. The `LongDateTime` data type defines the long date-time value.

When storing a long date-time value in files, you can use a 5-byte or 6-byte format for a range of roughly 35,000 years. You should sign extend this value to restore it to a `comp` format. Use the `LongDateCvt` (page 15) structure to help you in setting up a `LongDateTime` value.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`DateTimeUtils.h`

String2DateStatus

```
typedef StringToDateStatus String2DateStatus;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`DateTimeUtils.h`

StringToDateStatus

```
typedef short StringToDateStatus;
```

Availability
Available in Mac OS X v10.0 and later.

Declared In
`DateTimeUtils.h`

TogglePB

```
struct TogglePB {
     long togFlags;
     ResType amChars;
     ResType pmChars;
     long reserved[4];
};
typedef struct TogglePB TogglePB;
```

Fields

`togFlags`

The high-order word of this field contains flags that specify special conditions for the `ToggleDate` function.

The low-order word of this field contains masks representing fields to be checked by the `ValidDate` function. Each mask corresponds to a value in the enumerated type `LongDateField`. See Long Date Mask Constants (page 21) for a description of the values which you can use in this field. You can set this field to check the `era` through `second` fields by using the predeclared constant `dateStdMask`.

`amChars`

The trailing string to display for morning (for example, A.M.). This string is read from the numeric-format resource (resource type `'itl0'`) of the current script system.

`pmChars`

The trailing to display for evening (for example, P.M.). This string is read from the numeric-format resource (resource type `'itl0'`) of the current script system.

`reserved`

Reserved. Set each of the three elements of this field to 0.

Discussion
The `ToggleDate` function exchanges information with your application using the toggle parameter block, defined by the `TogglePB` data type.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`DateTimeUtils.h`

UTCDateTime

```
struct UTCDateTime {
     UInt16 highSeconds;
     UInt32 lowSeconds;
     UInt16 fraction;
};
typedef struct UTCDateTime UTCDateTime;
typedef UTCDateTime * UTCDateTimePtr;
typedef UTCDateTimePtr * UTCDateTimeHandle;
```

Discussion
`UTCDateTime` and `LocalDateTime` are both 64 bits wide. The first 48 bits represent the number of seconds since 1904. The remaining 16 bits are used to indicate a fractional seconds value, which has no inherent precision. Each unit of this 16-bit value represents 1/65535 of a second. Developers may apply the appropriate arithmetic to derive milliseconds or microseconds.

Note that the decision to divide the `lowSeconds` field between the high and low 32 bits of the 64 bit structure was intentional. You can use the structure to perform 64 bit math and logical comparisons. Having the `lowSeconds` field in the low or high 32 bits would have been easier for the compilers to handle and probably execute faster, however it would have rendered the structure unusable for 64 bit math and logical comparisons.

Important: You cannot access this structure as a `UInt64` data type. Doing so on systems that use little-endian byte odering may produce the wrong result.

Availability
Available in Mac OS X v10.0 and later.

Declared In
`UTCUtils.h`

Constants

Date Form Constants

```
typedef SInt8 DateForm;
enum {
     shortDate = 0,
     longDate = 1,
     abbrevDate = 2
};
```

Default Options

Options for use with the functions `SetDateTime` and `GetDateTime`.

```
enum {
     kUTCDefaultOptions = 0
};
```

Error Codes

```
enum {
     fatalDateTime = 0x8000,
     longDateFound = 1,
     leftOverChars = 2,
     sepNotIntlSep = 4,
     fieldOrderNotIntl = 8,
     extraneousStrings = 16,
     tooManySeps = 32,
     sepNotConsistent = 64,
     tokenErr = 0x8100,
     cantReadUtilities = 0x8200,
     dateTimeNotFound = 0x8400,
     dateTimeInvalid = 0x8800
};
```

Long Date Field Constants

```
typedef SInt8 LongDateField;
enum {
     eraField = 0,
     yearField = 1,
     monthField = 2,
     dayField = 3,
     hourField = 4,
     minuteField = 5,
     secondField = 6,
     dayOfWeekField = 7,
     dayOfYearField = 8,
     weekOfYearField = 9,
     pmField = 10,
     res1Field = 11,
     res2Field = 12,
     res3Field = 13
};
```

Long Date Mask Constants

```
enum {
     eraMask = 0x0001,
     yearMask = 0x0002,
     monthMask = 0x0004,
     dayMask = 0x0008,
     hourMask = 0x0010,
     minuteMask = 0x0020,
     secondMask = 0x0040,
     dayOfWeekMask = 0x0080,
     dayOfYearMask = 0x0100,
```

```
    weekOfYearMask = 0x0200,
    pmMask = 0x0400,
    dateStdMask = 0x007F
};
```

Constants

`eraMask`
Verify the era.
Available in Mac OS X v10.0 and later.
Declared in `DateTimeUtils.h`.

`yearMask`
Verify the year.
Available in Mac OS X v10.0 and later.
Declared in `DateTimeUtils.h`.

`monthMask`
Verify the month.
Available in Mac OS X v10.0 and later.
Declared in `DateTimeUtils.h`.

`dayMask`
Verify the day
Available in Mac OS X v10.0 and later.
Declared in `DateTimeUtils.h`.

`hourMask`
Verify the hour.
Available in Mac OS X v10.0 and later.
Declared in `DateTimeUtils.h`.

`minuteMask`
Verify the minute.
Available in Mac OS X v10.0 and later.
Declared in `DateTimeUtils.h`.

`secondMask`
Verify the second.
Available in Mac OS X v10.0 and later.
Declared in `DateTimeUtils.h`.

`dayOfWeekMask`
Verify the day of the week.
Available in Mac OS X v10.0 and later.
Declared in `DateTimeUtils.h`.

`dayOfYearMask`
Verify the day of the year.
Available in Mac OS X v10.0 and later.
Declared in `DateTimeUtils.h`.

`weekOfYearMask`
: Verify the week of the year.

  Available in Mac OS X v10.0 and later.

  Declared in `DateTimeUtils.h`.

`pmMask`
: Verify the evening (P.M.).

  Available in Mac OS X v10.0 and later.

  Declared in `DateTimeUtils.h`.

`dateStdMask`
: Verify the era through the second.

  Available in Mac OS X v10.0 and later.

  Declared in `DateTimeUtils.h`.

Discussion
These constants are used in the `field` parameter of the `ToggleDate` (page 38) function to specify the `LongDateRec` fields for the `ValidDate` function to check.

Flags

```
enum {
     smallDateBit = 31,
     togChar12HourBit = 30,
     togCharZCycleBit = 29,
     togDelta12HourBit = 28,
     genCdevRangeBit = 27,
     validDateFields = -1,
     maxDateField = 10
};
```

Constants
`smallDateBit`
: If this bit is set, the valid date and time are restricted to the range of the system global variable `Time`—that is, between midnight on January 1, 1904 and 6:28:15 A.M. on February 6, 2040.

  Available in Mac OS X v10.0 and later.

  Declared in `DateTimeUtils.h`.

`togChar12HourBit`
: If this bit is set, modifying the hour by character is limited to the 12-hour range defined by `togCharZCycleBit`, mapped to the appropriate half of the 24-hour range, as determined by the `pm` field. This bit works with system software version 6.0.4 and later.

  Available in Mac OS X v10.0 and later.

  Declared in `DateTimeUtils.h`.

`togCharZCycleBit`
: If this bit is set, the input character is treated as if it modifies an hour whose value is in the range 0–11. If this bit is not set, the input character is treated as if it modifies an hour whose value is in the range 12, 1–11. This bit works with system software version 6.0.4 and later.

  Available in Mac OS X v10.0 and later.

  Declared in `DateTimeUtils.h`.

`togDelta12HourBit`
: If this bit is set, modifying the hour up or down is limited to a 12-hour range. For example, increasing by one from 11 produces 0, increasing by one from 23 produces 12, and so on. This bit works with system software version 6.0.4 and later.

  Available in Mac OS X v10.0 and later.

  Declared in `DateTimeUtils.h`.

`genCdevRangeBit`
: If this bit is set in addition to `smallDateBit`, then the date range is restricted to that used by the General Controls control panel—January 1, 1920 to December 31, 2019 in the Gregorian calendar (the routine works correctly for other calendars as well). For dates outside this range but within the range specified by the system global variable `Time`—January 1, 1904 to February 6, 2040 in the Gregorian calendar—`ToggleDate` adds or subtracts 100 years to bring the dates into the range of the General Controls control panel if these bits are set. The `ToggleDate` function returns an error if the `smallDateBit` is set and the date is outside the range specified by the system global variable `Time`. This bit works with system software version 6.0.4 and later.

  Available in Mac OS X v10.0 and later.

  Declared in `DateTimeUtils.h`.

`validDateFields`
: Available in Mac OS X v10.0 and later.

  Declared in `DateTimeUtils.h`.

`maxDateField`
: Available in Mac OS X v10.0 and later.

  Declared in `DateTimeUtils.h`.

Toggle Results

```
typedef SInt16 ToggleResults;
enum {
     toggleUndefined = 0,
     toggleOK = 1,
     toggleBadField = 2,
     toggleBadDelta = 3,
     toggleBadChar = 4,
     toggleUnknown = 5,
     toggleBadNum = 6,
     toggleOutOfRange = 7,
     toggleErr3 = 7,
     toggleErr4 = 8,
     toggleErr5 = 9
};
```

Result Codes

The most common result codes returned by Date, Time, and Measurement Utilities are listed below.

Result Code	Value	Description
`clkRdErr`	-85	Unable to read the same clock value twice. Available in Mac OS X v10.0 and later.
`clkWrErr`	-86	The time written did not verify. Available in Mac OS X v10.0 and later.
`kUTCUnderflowErr`	-8850	An underflow error occurred. Available in Mac OS X v10.0 and later.
`kUTCOverflowErr`	-8851	An overflow error occurred. Available in Mac OS X v10.0 and later.
`kIllegalClockValueErr`	-8852	An illegal clock value was encountered. Available in Mac OS X v10.0 and later.

Deprecated Date, Time, and Measurement Utilities Functions

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

Deprecated in Mac OS X v10.3

DateString

Converts a date in the standard date-time representation into a Pascal string, making use of the date formatting information in the specified resource. (Deprecated in Mac OS X v10.3. Use `CFDateFormatterCreateStringWithDate` instead.)

```
void DateString (
   SInt32 dateTime,
   DateForm longFlag,
   Str255 result,
   Handle intlHandle
);
```

Parameters

dateTime

The date-time value in the representation returned by the `GetDateTime` function. The numeric representation used in these functions is the standard date-time representation: a 32-bit integer value that is returned by the `GetDateTime` function. This is a long integer value that represents the number of seconds between midnight, January 1, 1904, and the time at which `GetDateTime` was called.

longFlag

A flag that indicates the desired format for the date string. This is one of the three values defined as the `DateForm` type.

The string produced by `DateString` is in one of three standard date formats used on the Macintosh, depending on which of the three `DateForm` values that you specify for the `longFlag` parameter: `shortDate`, `abbrevDate`, or `longDate`. The information in the supplied resource defines how month and day names are written and provides for calendars with more than 7 days and more than 12 months.

For the Roman script system's resource, the date January 31, 1992, produces the following three strings: "1/31/92", "Fri, Jan 31, 1992", and "Friday, January 31, 1992"(for `DateForm` values `shortDate`, `abbrevDate`, and `longDate`, respectively).

result

On output, contains the string representation of the date in the format indicated by the `longFlag` parameter.

intlHandle

A handle to a numeric-format or a long-date-format resource that specifies date formatting information for use in the conversion. If you specify `NULL` as the value of the resource handle parameter, `DateString` uses information from the current script. The numeric-format (`'itl0'`) resource specifies the short date formats and the long-date-format (`'itl1'`) resource specifies the long date formats.

`DateString` formats its data according to the information in the specified numeric-format resource (for short date formats) or long-date-format resource (for long date formats). If you specify `shortDate`, the `intlHandle` value should be the handle to a numeric-format resource; if you specify `abbrevDate` or `longDate`, it should be the handle to a long-date-format resource.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.3.
Not available to 64-bit applications.

Declared In
`DateTimeUtils.h`

DateToSeconds

Converts a date and time to a number of seconds elapsed since midnight, January 1, 1904. (Deprecated in Mac OS X v10.3. Use the `CFCalendarRef` data type and the functions that operate on it instead.)

```
void DateToSeconds (
   const DateTimeRec *d,
   unsigned long *secs
);
```

Parameters

d

The date-time structure containing the date and time to convert.

secs

On return, the number of seconds elapsed between midnight, January 1, 1904, and the time specified in the `d` parameter. For example, specifying a date and time of 11:33 A.M. on January 1, 1904 results in 41580 being returned in this parameter.

Special Considerations

For information on using the `CFCalendarRef` data type, see *Data Formatting Guide for Core Foundation* and *CFCalendar Reference*.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.3.
Not available to 64-bit applications.

Declared In
`DateTimeUtils.h`

GetDateTime

Obtains the current date-time information, expressed as the number of seconds elapsed since midnight, January 1, 1904. (Deprecated in Mac OS X v10.3. Use `CFAbsoluteTimeGetCurrent` instead.)

```
void GetDateTime (
   unsigned long *secs
);
```

Parameters

secs

On return, the number of seconds elapsed since midnight, January 1, 1904.

Discussion

The low-memory copy of the date and time information is also accessible through the global variable `Time`.

If an application disables interrupts for longer than a second, the date-time information returned by the `GetDateTime` function might not be exact. The `GetDateTime` function is intended to provide fairly accurate time information, but not scientifically precise data.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.3.

Not available to 64-bit applications.

Declared In

`DateTimeUtils.h`

InitDateCache

Initializes the date cache structure, which is used to store data for use by the `StringToDate` and `StringToTime` functions. (Deprecated in Mac OS X v10.3. There is no replacement.)

```
OSErr InitDateCache (
   DateCachePtr theCache
);
```

Parameters

theCache

A pointer to a date cache structure. This parameter can be a local variable, a pointer, or a locked handle.

Return Value

A result code. See "Date, Time, and Measurement Utilities Result Codes" (page 24).

Discussion

You must call `InitDateCache` to initialize the date cache structure before using either the `StringToDate` (page 35) or `StringToTime` (page 37) functions. You must pass a pointer to a date cache structure. You have to declare the structure as a variable or allocate it in the heap.

If you are writing an application that allows the use of global variables, you can make your date cache structure a global variable and initialize it once, when you perform other global initialization.

`InitDateCache` calls the `GetResource` and `LoadResource` functions and it can also return the error codes they produce.

Special Considerations

You no longer need to initialize the data cache in Mac OS X.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.3.
Not available to 64-bit applications.

Declared In
`DateTimeUtils.h`

LongDateString

Converts a date that is specified as a `LongDateTime` value into a Pascal string, making use of the date formatting information in the specified resource. (Deprecated in Mac OS X v10.3. Use `CFDateFormatterCreateStringWithDate` instead.)

```
void LongDateString (
   const LongDateTime *dateTime,
   DateForm longFlag,
   Str255 result,
   Handle intlHandle
);
```

Parameters

dateTime

A pointer to a 64-bit, signed representation of the number of seconds since Jan. 1, 1904. This allows coverage of a much longer span of time (plus or minus approximately 30,000 years) than the standard, 32-bit representation.

longFlag

A flag that indicates the desired format for the date string. This is one of the three values defined as the `DateForm` type.

The string produced by `LongDateString` is in one of three standard date formats used on the Macintosh, depending on which of the three `DateForm` values that you specify for the `longFlag` parameter: `shortDate`, `abbrevDate`, or `longDate`. The information in the supplied resource defines how month and day names are written and provides for calendars with more than 7 days and more than 12 months.

For the U.S. resource, the date January 31, 1992, produces the following three strings: “1/31/92“, “Fri, Jan 31, 1992“, and “Friday, January 31, 1992“(for `DateForm` values `shortDate`, `abbrevDate`, and `longDate`, respectively).

result

On output, contains the string representation of the date in the format indicated by the `longFlag` parameter.

intlHandle

A handle to a numeric-format or long-date-format resource that specifies date formatting information for use in the conversion. If you specify `NULL` as the value of the resource handle parameter, `LongDateString` uses information from the current script. The numeric-format (`'itl0'`) resource specifies the short date formats and the long-date-format (`'itl1'`) resource specifies the long date formats.

If you specify `shortDate` in the `longFlag` parameter, the `intlHandle` value should be the handle to a numeric-format resource; if you specify `abbrevDate` or `longDate`, it should be the handle to a long-date-format resource.

Discussion
You can use the `LongSecondsToDate` and `LongDateToSeconds` functions to convert between the `LongDateRec` (as produced by the `StringToDate` function) and `LongDateTime` data types.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.3.
Not available to 64-bit applications.

Declared In
`DateTimeUtils.h`

LongDateToSeconds

Converts a date and time to the number of seconds elapsed since midnight, January 1, 1904. (Deprecated in Mac OS X v10.3. Use the `CFCalendarRef` data type and the functions that operate on it instead.)

```
void LongDateToSeconds (
   const LongDateRec *lDate,
   LongDateTime *lSecs
);
```

Parameters

`lDate`

The long date-time structure containing the date and time to convert.

`lSecs`

On return, the number of seconds elapsed since midnight, January 1, 1904, and the time specified in the `lDate` parameter. The number of seconds are returned as a long date-time value.

Special Considerations

For information on using the `CFCalendarRef` data type, see *Data Formatting Guide for Core Foundation* and *CFCalendar Reference*.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.3.
Not available to 64-bit applications.

Declared In
`DateTimeUtils.h`

LongSecondsToDate

Converts the number of seconds elapsed since midnight, January 1, 1904 to a date and time. (Deprecated in Mac OS X v10.3. Use the `CFCalendarRef` data type and the functions that operate on it instead.)

```
void LongSecondsToDate (
   const LongDateTime *lSecs,
   LongDateRec *lDate
);
```

Parameters

lSecs

The number of seconds elapsed since midnight, January 1, 1904.

lDate

On return, the fields of the long date-time structure that contain the date and time corresponding to the value indicated in the `lSecs` parameter. For example, specifying the number of seconds 41580 results in the date and time 11:33 A.M. on January 1, 1904 being returned in this parameter.

Special Considerations

For information on using the `CFCalendarRef` data type, see *Data Formatting Guide for Core Foundation* and *CFCalendar Reference*.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.3.

Not available to 64-bit applications.

Declared In

`DateTimeUtils.h`

LongTimeString

Converts a time that is specified as a `LongDateTime` value into a Pascal string, making use of the time formatting information in the specified resource. (Deprecated in Mac OS X v10.3. Use `CFDateFormatterCreateStringWithDate` instead.)

```
void LongTimeString (
   const LongDateTime *dateTime,
   Boolean wantSeconds,
   Str255 result,
   Handle intlHandle
);
```

Parameters

dateTime

A pointer to a 64-bit, signed representation of the number of seconds since Jan. 1, 1904. This allows coverage of a much longer span of time (plus or minus approximately 30,000 years) than the standard, 32-bit representation.

wantSeconds

A flag that indicates whether the seconds are to be included in the resulting string. `LongTimeString` produces a string that includes the seconds if you set this parameter to `TRUE`.

result

On output, contains the string representation of the time.

intlHandle

A handle to a numeric-format (`'itl0'`) resource that specifies time formatting information for use in the conversion. If you specify `NULL` as the value of the resource handle parameter, `LongTimeString` uses information from the current script.

The numeric-format resource specifies whether or not to use leading zeros for the time values, whether to use a 12- or 24-hour time cycle, and how to specify morning or evening if a 12-hour time cycle is used.

Discussion

You can use the `LongSecondsToDate` and `LongDateToSeconds` functions to convert between the `LongDateRec` (as produced by the `StringToTime` (page 37) function) and `LongDateTime` data types.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.3.

Not available to 64-bit applications.

Declared In

`DateTimeUtils.h`

ReadDateTime

Reads time information from the system. (Deprecated in Mac OS X v10.3. Use `CFAbsoluteTimeGetCurrent` instead.)

```
OSErr ReadDateTime (
   unsigned long *time
);
```

Parameters

time

On return, the current time expressed as the number of seconds elapsed since midnight, January 1, 1904.

Return Value

A result code. See "Date, Time, and Measurement Utilities Result Codes" (page 24). If the clock chip cannot be read, `ReadDateTime` returns the `clkRdErr` result code. The operation might fail if the clock chip is damaged. Otherwise, the function returns the `noErr` result code.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.3.

Not available to 64-bit applications.

Declared In

`DateTimeUtils.h`

SecondsToDate

Converts a number of seconds elapsed since midnight, January 1, 1904 to a date and time. (Deprecated in Mac OS X v10.3. Use the `CFCalendarRef` data type and the functions that operate on it instead.)

```
void SecondsToDate (
   unsigned long secs,
   DateTimeRec *d
);
```

Parameters

secs

The number of seconds elapsed since midnight, January 1, 1904.

d

On return, the fields of the date-time structure that contain the date and time corresponding to the value indicated in the `s` parameter.

Special Considerations

For information on using the `CFCalendarRef` data type, see *Data Formatting Guide for Core Foundation* and *CFCalendar Reference*.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.3.

Not available to 64-bit applications.

Declared In

`DateTimeUtils.h`

SetDateTime

Changes the date-time information stored by the system to the specified value, expressed as the number of seconds elapsed since midnight, January 1, 1904. (Deprecated in Mac OS X v10.3. There is no replacement.)

```
OSErr SetDateTime (
   unsigned long time
);
```

Parameters

time

The number of seconds elapsed since midnight, January 1, 1904; this value is written to the system.

Return Value

A result code. See "Date, Time, and Measurement Utilities Result Codes" (page 24). The `SetDateTime` function attempts to verify the value written by reading it back in and comparing it to the value in the low-memory copy. If a problem occurs, the `SetDateTime` function returns either the `clkRdErr` result code, because the clock chip could not be read, or the `clkWrErr` result code, because the time written to the clock chip could not be verified. Otherwise, the function returns the `noErr` result code.

Special Considerations

Only the root user can set the time in Mac OS X.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.3.

Not available to 64-bit applications.

Declared In
`DateTimeUtils.h`

SetTime

Changes the date-time information in the system to the specified value, expressed as a date and time. (Deprecated in Mac OS X v10.3. There is no replacement.)

```
void SetTime (
   const DateTimeRec *d
);
```

Parameters

d

The date and time to which to set in the system.

Discussion
The `SetTime` function first converts the date and time to the number of seconds elapsed since midnight, January 1, 1904 by calling the `DateToSeconds` function. It then writes these seconds to the system and to the system global variable `Time` by calling the `SetDateTime` function.

The `SetTime` function does not return a result code. If you need to know whether an attempt to change the date and time information in the system is successful, you must use the `SetDateTime` function.

As an alternative to using the `SetTime` procedure, you can use the `DateToSeconds` (page 28) and `SetDateTime` (page 34) functions.

Special Considerations

Only the root user can set the time in Mac OS X.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.3.
Not available to 64-bit applications.

Declared In
`DateTimeUtils.h`

StringToDate

Parses a string for a date and converts the date information into values in a date-time structure. (Deprecated in Mac OS X v10.3. Use `CFDateFormatterCreateDateFromString` instead.)

```
StringToDateStatus StringToDate (
   Ptr textPtr,
   SInt32 textLen,
   DateCachePtr theCache,
   SInt32 *lengthUsed,
   LongDateRec *dateTime
);
```

Parameters

textPtr

A pointer to the text string to be parsed. `StringToDate` expects a date specification, in a format defined by the current script, at the beginning of the string.

textLen

The number of bytes in the text string.

theCache

A pointer to the date cache structure initialized by the `InitDateCache` (page 29) function with data that is used during the conversion process.

lengthUsed

On output, contains a pointer to the number of bytes of the string that were parsed for the date. Use this value to compute the starting location of the text that you can pass to `StringToTime` (page 37). Alternatively, you can use them in reverse order.

dateTime

On output, a pointer to the `LongDateRec` structure, which contains the year, month, day, and day of the week parsed for the date.

Return Value

A set of bit values that indicate confidence levels, with higher numbers indicating low confidence in how closely the input string matched what the function expected. For example, specifying a date with nonstandard separators may work, but it returns a message indicating that the separator was not standard. See the description of the `StringToDateStatus` data type.

Discussion

`StringToDate` parses the text string until it has finished finding all date information or until it has examined the number of bytes specified by `textLen`.

Note that `StringToDate` fills in only the year, month, day, and day of the week; `StringToTime` fills in the hour, minute, and second. You can use these two functions sequentially to fill in all of the values in a `LongDateRec` structure.

When one of the date components is missing, such as the year, the current date value is used as a default.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.3.

Not available to 64-bit applications.

Declared In

`DateTimeUtils.h`

StringToTime

Parses a string for a time specification and converts the date information into values in a date-time structure. (Deprecated in Mac OS X v10.3. Use `CFDateFormatterCreateDateFromString` instead.)

```
StringToDateStatus StringToTime (
   Ptr textPtr,
   SInt32 textLen,
   DateCachePtr theCache,
   SInt32 *lengthUsed,
   LongDateRec *dateTime
);
```

Parameters

textPtr

A pointer to the text string to be parsed. At the beginning of the string, `StringToTime` expects a time specification in a format defined by the current script.

textLen

The number of bytes in the text string.

theCache

A pointer to the date cache structure initialized by the `InitDateCache` function with data that is used during the conversion process.

lengthUsed

On output, contains a pointer to the length, in bytes, of the string that was parsed for the time.

dateTime

On output, a pointer to the `LongDateRec` structure, which contains the hour, minute, and second values that were parsed for the time.

Return Value

`StringToTime` returns a status value that indicates the confidence level for the success of the conversion. This is the same status value indicator type as does `StringToDate`: a set of bit values that indicate confidence levels, with higher numbers indicating low confidence in how closely the input string matched what the function expected. See the description of the `StringToDateStatus` data type.

Discussion

`StringToTime` parses the string until it has finished finding all time information or until it has examined the number of bytes specified by `textLen`.

Note that `StringToTime` fills in only the hour, minute, and second; `StringToDate` (page 35) fills in the year, month, day, and day of the week. You can use these two functions sequentially to fill in all of the values in a `LongDateRec` structure.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.3.

Not available to 64-bit applications.

Declared In

`DateTimeUtils.h`

TimeString

Converts a time in the standard date-time representation into a string, making use of the time formatting information in the specified resource. (Deprecated in Mac OS X v10.3. Use `CFDateFormatterCreateStringWithDate` instead.)

```
void TimeString (
   SInt32 dateTime,
   Boolean wantSeconds,
   Str255 result,
   Handle intlHandle
);
```

Parameters

dateTime

The date-time value in the representation returned by the Operating System function `GetDateTime`. The numeric representation used in these functions is the standard date-time representation: a 32-bit integer value that is returned by the `GetDateTime` function. This is a long integer value that represents the number of seconds between midnight, January 1, 1904, and the time at which `GetDateTime` was called.

wantSeconds

A flag that indicates whether the seconds are to be included in the resulting string.

result

On output, contains the string representation of the time.

intlHandle

A handle to a numeric-format (`'itl0'`) resource that specifies time formatting information for use in the conversion. If you specify `NULL` as the value of the resource handle parameter, `TimeString` uses information from the current script.

The numeric-format resource specifies whether or not to use leading zeros for the time values, whether to use a 12- or 24-hour time cycle, and how to specify morning or evening if a 12-hour time cycle is used.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.3.

Not available to 64-bit applications.

Declared In

`DateTimeUtils.h`

ToggleDate

Modifies a date and time, by modifying one specific component of a date and time (day, hour, minute, seconds, day of week, and so on). (Deprecated in Mac OS X v10.3. Use the `CFCalendarRef` data type and the functions that operate on it instead.)

```
ToggleResults ToggleDate (
   LongDateTime *lSecs,
   LongDateField field,
   DateDelta delta,
   short ch,
   const TogglePB *params
);
```

Parameters

lSecs

The date-time information to modify, expressed as the number of seconds elapsed since midnight, January 1, 1904.

field

The name of the field in the date-time structure you want modify. Use one of the Long Date Mask Constants (page 21) for the value of this parameter.

delta

A signed byte specifying the action you want to perform on the value specified in the `field` parameter. Set `delta` to 1, to increase the value in the field by 1. Set `delta` to -1, to decrease the value of the field by 1. Set `delta` to 0. If you want to set the value of the field explicitly; pass the new value through the `ch` field.

ch

If the value in the `delta` field is 0, the value of the field in the date-time structure (specified by the `field` parameter) is set to the value in the `ch` parameter. If the value in the `delta` field is not equal to 0, the value in the `ch` parameter is ignored.

params

The user-defined settings of the toggle parameter block settings.

Return Value

See the description of the `ToggleResults` data type.

Discussion

The relevant fields of the toggle parameter block are:

- `togFlags` A value of type `SInt32`. On input, the fields to be checked by the `ValidDate` function.
- `amChars` A value of type `ResType`. On input, A.M. characters from `'itl0'` resource.
- `pmChars` A value of type `ResType`. On input, P.M. characters from `'itl0'` resource.
- `reserved` An array of `SInt32` values. Reserved; on input, set each element to 0.

You must supply values for all input parameters.

The `ToggleDate` function first converts the number of seconds and makes each component of the date and time available through a long date-time structure. The `ToggleDate` function then modifies the value of the field, specified by the `field` parameter. If the value in the `delta` field is greater than 0, the value of the field increases by 1; if the value in the `delta` field is less than 0, the value of the field decreases by 1; and if the value of `delta` is 0, the value of the field is explicitly set to the value specified in the `ch` field. After the `ToggleDate` function modifies the field, it calls the `ValidDate` function. The `ValidDate` function checks the long date-time structure for correctness. If any of the structure fields are invalid, the `ValidDate` function returns a `LongDateField` value corresponding to the field in error. Otherwise, it returns the result code for `validDateFields`. Note that `ValidDate` reports only the least significant erroneous field.

After the `ToggleDate` function checks the validity of the modified field, it converts the modified date and time back into a number of seconds and returns these seconds in the `lSecs` parameter.

The `ToggleDate` function was previously available with the Script Manager.

For more information on the `LongDateRec` structure, see `LongDateRec` (page 16). The toggle parameter block structure is described in `TogglePB` (page 18).

For more information about the `GetIntlResource` function, see the Script Manager. For details on the `UppercaseText` function, see Text Utilities.

Special Considerations

For information on using the `CFCalendarRef` data type, see *Data Formatting Guide for Core Foundation* and *CFCalendar Reference*.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.3.

Not available to 64-bit applications.

Declared In

`DateTimeUtils.h`

ValidDate

Verifies specific date and time values in a long date-time structure. (Deprecated in Mac OS X v10.3. Use the `CFCalendarRef` data type and the functions that operate on it instead.)

```
short ValidDate (
   const LongDateRec *vDate,
   long flags,
   LongDateTime *newSecs
);
```

Parameters

vDate

The long date-time structure whose fields you want to verify.

flags

The fields that you want to verify in the long date-time structure. For a description of the values you can use in this parameter, see Long Date Mask Constants (page 21).

newSecs

The date-time information, passed by the `ToggleDate` function, that you want to verify.

Return Value

If any of the specified fields contain invalid values, the `ValidDate` function returns a `LongDateField` value indicating the field in error. Otherwise, it returns the constant `validDateFields`. `ValidDate` reports only the least significant erroneous field.

Discussion

For more information on the `LongDateRec` structure, see `LongDateRec` (page 16). The toggle parameter block structure is described in `TogglePB` (page 18).

Special Considerations

For information on using the `CFCalendarRef` data type, see *Data Formatting Guide for Core Foundation* and *CFCalendar Reference*.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.3.
Not available to 64-bit applications.

Declared In
`DateTimeUtils.h`

Deprecated in Mac OS X v10.4

ConvertLocalTimeToUTC

Converts local time to UTC. (Deprecated in Mac OS X v10.4. Use `CFTimeZoneGetSecondsFromGMT` instead.)

```
OSStatus ConvertLocalTimeToUTC (
   UInt32 localSeconds,
   UInt32 *utcSeconds
);
```

Parameters

`localSeconds`
A value of type `UInt32` containing the local time.

`utcSeconds`
A pointer to a value of type `UInt32`. On return, this points to the UTC value corresponding to the given time in `localSeconds`.

Return Value
A result code. See "Date, Time, and Measurement Utilities Result Codes" (page 24).

Discussion
Given a local time in `localSeconds`, the function will place the corresponding UTC value in `utcSeconds`. This function returns `noErr` if the conversion is successful. Otherwise, it may return `kUTCUnderflowErr` or `kUTCOverflowErr`.

Special Considerations

For information on using `CFTimeZoneGetSecondsFromGMT`, see *Dates and Times Programming Guide for Core Foundation* and*CFTimeZone Reference*.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`UTCUtils.h`

ConvertLocalToUTCDateTime

Converts local date and time to UTC date and time. (Deprecated in Mac OS X v10.4. Use `CFTimeZoneGetSecondsFromGMT` instead.)

```
OSStatus ConvertLocalToUTCDateTime (
   const LocalDateTime *localDateTime,
   UTCDateTime *utcDateTime
);
```

Parameters

`localDateTime`

A value of type `LocalDateTime` containing the local date and time.

`utcDateTime`

A pointer to a value of type `UTCDateTime`. On return, this points to the UTC value corresponding to the given date and time in `localDateTime`.

Return Value

A result code. See "Date, Time, and Measurement Utilities Result Codes" (page 24).

Discussion

Given a local date and time in the `localDateTime` parameter, this function places the corresponding UTC value in `utcDateTime`. This function returns `noErr` if the conversion is successful. Otherwise, it may return `kUTCUnderflowErr`, `kUTCOverflowErr`, or `paramErr` if `utcDateTime` is `NULL`.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`UTCUtils.h`

ConvertUTCToLocalDateTime

Converts UTC date and time to local date and time. (Deprecated in Mac OS X v10.4. Use `CFTimeZoneGetSecondsFromGMT` instead.)

```
OSStatus ConvertUTCToLocalDateTime (
   const UTCDateTime *utcDateTime,
   LocalDateTime *localDateTime
);
```

Parameters

`utcDateTime`

A value of type `UTCDateTime` specifying the UTC date and time.

`localDateTime`

A pointer to a value of type `LocalDateTime`. On return, this points to the local value corresponding to the given date and time in `utcDateTime`.

Return Value

A result code. See "Date, Time, and Measurement Utilities Result Codes" (page 24).

Discussion
Given a UTC date and time in `utcDateTime`, this function places the corresponding local value in `localDateTime`. This function returns `noErr` if the conversion is successful. Otherwise, it may return `kUTCUnderflowErr`, `kUTCOverflowErr`, or `paramErr` if `localDateTime` is `NULL`.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`UTCUtils.h`

ConvertUTCToLocalTime

Converts UTC time to local time. (Deprecated in Mac OS X v10.4. Use `CFTimeZoneGetSecondsFromGMT` instead.)

```
OSStatus ConvertUTCToLocalTime (
   UInt32 utcSeconds,
   UInt32 *localSeconds
);
```

Parameters
`utcSeconds`
A value of type `UInt32` specifying UTC time in seconds.

`localSeconds`
A pointer to a value of type `UInt32`. On return, this points to the local time corresponding to the UTC time specified in `utcSeconds`.

Return Value
A result code. See "Date, Time, and Measurement Utilities Result Codes" (page 24).

Discussion
Given a UTC time in `utcSeconds` this function places the corresponding local value in `localSeconds`. This function returns `noErr` if the conversion is successful. Otherwise, it may return `kUTCUnderflowErr` or `kUTCOverflowErr`.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`UTCUtils.h`

GetLocalDateTime

Gets the local date and time. (Deprecated in Mac OS X v10.4. Use `CFAbsoluteTimeGetCurrent` and `CFTimeZoneGetSecondsFromGMT` instead.)

```
OSStatus GetLocalDateTime (
   LocalDateTime *localDateTime,
   OptionBits options
);
```

Parameters

localDateTime

A pointer to a value of type `LocalDateTime`. On return, the value this parameter points to is the current local date and time.

options

A value of type `OptionBits`. Pass `kUTCDefaultOptions` for the default behavior.

Return Value
A result code. See "Date, Time, and Measurement Utilities Result Codes" (page 24).

Discussion
This API returns the current date and time in `localTime`. Otherwise, it is set to 0. Use `kUTCDefaultOptions` in the options parameter for default behavior. Different behavior may be specified through this parameter in the future. If the operation is successful `noErr` is returned. If a `NULL` pointer is passed in the `localDateTime` parameter, `paramErr` is returned.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`UTCUtils.h`

GetUTCDateTime

Gets the UTC date and time. (Deprecated in Mac OS X v10.4. Use `CFAbsoluteTimeGetCurrent` instead.)

```
OSStatus GetUTCDateTime (
   UTCDateTime *utcDateTime,
   OptionBits options
);
```

Parameters

utcDateTime

A pointer to a value of type `UTCDateTime`. On return, the value this parameter points to is the current UTC date and time.

options

A value of type `OptionBits`. Pass `kUTCDefaultOptions` for the default behavior.

Return Value
A result code. See "Date, Time, and Measurement Utilities Result Codes" (page 24).

Discussion
This API returns the current date and time as UTC in `utcDateTime`. Otherwise, it is set to 0. Use `kUTCDefaultOptions` in the options for default behavior. Different behavior may be specified through this parameter in the future. If the operation is successful `noErr` is returned. If a `NULL` pointer is passed in `utcDateTime`, `paramErr` is returned.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`UTCUtils.h`

SetLocalDateTime

Sets the local date and time. (Deprecated in Mac OS X v10.4. There is no replacement.)

```
OSStatus SetLocalDateTime (
   const LocalDateTime *localDateTime,
   OptionBits options
);
```

Parameters

`localDateTime`

A pointer to a value of type `LocalDateTime` specifying the current local date and time.

`options`

A value of type `OptionBits`. Pass `kUTCDefaultOptions` for the default behavior.

Return Value
A result code. See "Date, Time, and Measurement Utilities Result Codes" (page 24).

Discussion
Use this call to set the clock to the date and time passed in the `localDateTime` parameter. Use `kUTCDefaultOptions` in the options for default behavior. Different behavior may be specified through this parameter in the future. If successful `noErr` is returned. Other errors include `kIllegalClockValueErr`, `paramErr` if `localDateTime` is `NULL`, or `clkWrErr` due to a failed attempt to write the value to the system.

Special Considerations

Only the root user can set the time in Mac OS X.

Availability
Available in Mac OS X v10.0 and later.
Deprecated in Mac OS X v10.4.
Not available to 64-bit applications.

Declared In
`UTCUtils.h`

SetUTCDateTime

Sets the UTC date and time. (Deprecated in Mac OS X v10.4. Use settimeofday(2) instead.)

```
OSStatus SetUTCDateTime (
   const UTCDateTime *utcDateTime,
   OptionBits options
);
```

Parameters

utcDateTime

A pointer to a value of type `UTCDateTime` specifying the current UTC date and time.

options

A value of type `OptionBits`. Pass `kUTCDefaultOptions` for the default behavior.

Return Value

A result code. See "Date, Time, and Measurement Utilities Result Codes" (page 24).

Discussion

Use this call to set the clock to the date and time passed in the `utcDateTime` parameter. Use `kUTCDefaultOptions` in the options for default behavior. Different behavior may be specified through this parameter in the future. If successful `noErr` is returned. Other errors include `kIllegalClockValueErr`, `kUTCUnderflowErr`, `kUTCOverflowErr`, and `paramErr` if `NULL` is passed for `utcDateTime`. It may also return `clkWrErr` due to a failed attempt to write the value to the system.

Special Considerations

Only the root user can set the time in Mac OS X.

Availability

Available in Mac OS X v10.0 and later.

Deprecated in Mac OS X v10.4.

Not available to 64-bit applications.

Declared In

`UTCUtils.h`

Document Revision History

This table describes the changes to *Date, Time, and Measurement Utilities Reference*.

Date	Notes
2006-09-29	Made minor formatting changes.
2006-07-24	Added information on deprecated functions.
2005-07-07	Added documentation for the fields of the LongDateRec data structure.
2003-05-01	Changed “January 31, 1991” to “January 31, 1992” in the function parameters for the `DateString` and `LongDateString` functions.
2003-02-01	Updated the EDD.
	Removed references to clock chips.
	Added descriptions of some new functions.
	Regrouped functions.
	Removed unsupported functions. These can be found in the Carbon Specification.
2002-02-01	Last version of this document.

Document Revision History

Index

M

P

R

S

T

U

V

W

Y