CFRunLoopTimer Reference

Core Foundation

2006-02-07

Contents

CFRunLoopTimer Reference

Derived From:	CFType
Framework:	CoreFoundation/CoreFoundation.h
Companion guide	Run Loops
Declared in	CFRunLoop.h

Overview

A CFRunLoopTimer object represents a specialized run loop source that fires at a preset time in the future. Timers can fire either only once or repeatedly at fixed time intervals. Repeating timers can also have their next firing time manually adjusted.

A timer is not a real-time mechanism; it fires only when one of the run loop modes to which the timer has been added is running and able to check if the timer's firing time has passed. If a timer's firing time occurs while the run loop is in a mode that is not monitoring the timer or during a long callout, the timer does not fire until the next time the run loop checks the timer. Therefore, the actual time at which the timer fires potentially can be a significant period of time after the scheduled firing time.

A repeating timer reschedules itself based on the scheduled firing time, not the actual firing time. For example, if a timer is scheduled to fire at a particular time and every 5 seconds after that, the scheduled firing time will always fall on the original 5 second time intervals, even if the actual firing time gets delayed. If the firing time is delayed so far that it passes one or more of the scheduled firing times, the timer is fired only once for that time period; the timer is then rescheduled, after firing, for the next scheduled firing time in the future.

Each run loop timer can be registered in only one run loop at a time, although it can be added to multiple run loop modes within that run loop.

CFRunLoopTimer is "toll-free bridged" with its Cocoa Foundation counterpart, NSTimer. This means that the Core Foundation type is interchangeable in function or method calls with the bridged Foundation object. Therefore, in a method where you see an `NSTimer *` parameter, you can pass in a `CFRunLoopTimerRef`, and in a function where you see a `CFRunLoopTimerRef` parameter, you can pass in an `NSTimer` instance. This also applies to concrete subclasses of `NSTimer`. See Interchangeable Data Types for more information on toll-free bridging.

Functions

CFRunLoopTimerCreate

Creates a new CFRunLoopTimer object.

```
CFRunLoopTimerRef CFRunLoopTimerCreate (
   CFAllocatorRef allocator,
   CFAbsoluteTime fireDate,
   CFTimeInterval interval,
   CFOptionFlags flags,
   CFIndex order,
   CFRunLoopTimerCallBack callout,
   CFRunLoopTimerContext *context
);
```

Parameters

allocator

The allocator to use to allocate memory for the new object. Pass `NULL` or `kCFAllocatorDefault` to use the current default allocator.

fireDate

The time at which the timer should first fire. The fine precision (sub-millisecond at most) of the fire date may be adjusted slightly by the timer if there are implementation reasons to do.

interval

The firing interval of the timer. If `0` or negative, the timer fires once and then is automatically invalidated. The fine precision (sub-millisecond at most) of the interval may be adjusted slightly by the timer if implementation reasons to do so exist.

flags

Currently ignored. Pass `0` for future compatibility.

order

A priority index indicating the order in which run loop timers are processed. Run loop timers currently ignore this parameter. Pass `0`.

callout

The callback function that is called when the timer fires.

context

A structure holding contextual information for the run loop timer. The function copies the information out of the structure, so the memory pointed to by *context* does not need to persist beyond the function call. Can be `NULL` if the callback function does not need the context's `info` pointer to keep track of state.

Return Value

The new CFRunLoopTimer object. Ownership follows the Create Rule.

Discussion

A timer needs to be added to a run loop mode before it will fire. To add the timer to a run loop, use `CFRunLoopAddTimer`. A timer can be registered to only one run loop at a time, although it can be in multiple modes within that run loop.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

Watcher

Worm

Declared In

`CFRunLoop.h`

CFRunLoopTimerDoesRepeat

Returns a Boolean value that indicates whether a CFRunLoopTimer object repeats.

```
Boolean CFRunLoopTimerDoesRepeat (
   CFRunLoopTimerRef timer
);
```

Parameters

timer

The run loop timer to test.

Return Value

`true` if *timer* repeats, or has a periodicity; otherwise `false`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CFRunLoop.h`

CFRunLoopTimerGetContext

Returns the context information for a CFRunLoopTimer object.

```
void CFRunLoopTimerGetContext (
   CFRunLoopTimerRef timer,
   CFRunLoopTimerContext *context
);
```

Parameters

timer

The run loop timer to examine.

context

A pointer to the structure into which the context information for *timer* is to be copied. The information being returned is the same information passed to `CFRunLoopTimerCreate` (page 5) when creating *timer*.

Discussion

The context version number for run loop timers is currently 0. Before calling this function, you need to initialize the `version` member of *context* to `0`.

Availability

Available in Mac OS X v10.0 and later.

Declared In

`CFRunLoop.h`

CFRunLoopTimerGetInterval

Returns the firing interval of a repeating CFRunLoopTimer object.

```
CFTimeInterval CFRunLoopTimerGetInterval (
   CFRunLoopTimerRef timer
);
```

Parameters

timer

The run loop timer to examine.

Return Value

The firing interval of *timer*. Returns 0 if *timer* does not repeat.

Availability

Available in Mac OS X v10.0 and later.

Related Sample Code

Worm

Declared In

CFRunLoop.h

CFRunLoopTimerGetNextFireDate

Returns the next firing time for a CFRunLoopTimer object.

```
CFAbsoluteTime CFRunLoopTimerGetNextFireDate (
   CFRunLoopTimerRef timer
);
```

Parameters

timer

The run loop timer to examine.

Return Value

The next firing time for *timer*. This time could be a date in the past if a run loop has not been able to process the timer since the firing time arrived.

Availability

Available in Mac OS X v10.0 and later.

Declared In

CFRunLoop.h

CFRunLoopTimerGetOrder

Returns the ordering parameter for a CFRunLoopTimer object.

```
CFIndex CFRunLoopTimerGetOrder (
   CFRunLoopTimerRef timer
);
```

Parameters

timer

The run loop timer to examine.

Return Value
The ordering parameter for *`timer`*.

Discussion
The ordering parameter is currently ignored by run loop timers.

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

Declared In
`CFRunLoop.h`

CFRunLoopTimerGetTypeID

Returns the type identifier of the CFRunLoopTimer opaque type.

```
CFTypeID CFRunLoopTimerGetTypeID (
   void
);
```

Return Value
The type identifier for the CFRunLoopTimer opaque type.

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

Declared In
`CFRunLoop.h`

CFRunLoopTimerInvalidate

Invalidates a CFRunLoopTimer object, stopping it from ever firing again.

```
void CFRunLoopTimerInvalidate (
   CFRunLoopTimerRef timer
);
```

Parameters

`timer`
: The run loop timer to invalidate.

Discussion
Once invalidated, *`timer`* will never fire and call its callback function again. This function automatically removes *`timer`* from all run loop modes in which it had been added. The memory is not deallocated unless the run loop held the only reference to *`timer`*.

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

Related Sample Code
Worm

Declared In
`CFRunLoop.h`

CFRunLoopTimerIsValid

Returns a Boolean value that indicates whether a CFRunLoopTimer object is valid and able to fire.

```
Boolean CFRunLoopTimerIsValid (
   CFRunLoopTimerRef timer
);
```

Parameters

timer

The run loop timer to examine.

Return Value
`true` if *timer* is valid; otherwise `false`.

Discussion
A nonrepeating timer is automatically invalidated after it fires.

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

Declared In
`CFRunLoop.h`

CFRunLoopTimerSetNextFireDate

Sets the next firing date for a CFRunLoopTimer object .

```
void CFRunLoopTimerSetNextFireDate (
   CFRunLoopTimerRef timer,
   CFAbsoluteTime fireDate
);
```

Parameters

timer

The run loop timer to modify.

fireDate

The new firing time for *timer*.

Discussion
Resetting a timer’s next firing time is a relatively expensive operation and should not be done if it can be avoided; letting timers autorepeat is more efficient. In some cases, however, manually-adjusted, repeating timers are useful. For example, if you have an action that will be performed multiple times in the future, but at irregular time intervals, it would be very expensive to create, add to run loop modes, and then destroy a timer for each firing event. Instead, you can create a repeating timer with an initial firing time in the distant future (or the initial firing time) and a very large repeat interval—on the order of decades or more—and add it to all the necessary run loop modes. Then, when you know when the timer should fire next, you reset the firing time with `CFRunLoopTimerSetNextFireDate`, perhaps from the timer’s own callback function. This technique effectively produces a reusable, asynchronous timer.

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

Related Sample Code
Worm

Declared In
`CFRunLoop.h`

Callbacks

CFRunLoopTimerCallBack

Callback invoked when a CFRunLoopTimer object fires.

```
typedef void (*CFRunLoopTimerCallBack) (
    CFRunLoopTimerRef timer,
    void *info
);
```

If you name your function `MyCallBack`, you would declare it like this:

```
void MyCallBack (
    CFRunLoopTimerRef timer,
    void *info
);
```

Parameters

`timer`
The run loop timer that is firing.

`info`
The `info` member of the `CFRunLoopTimerContext` (page 11) structure that was used when creating the run loop timer.

Discussion
If *`timer`* repeats, the run loop automatically schedules the next firing time after calling this function, unless you manually update the firing time within this callback by calling `CFRunLoopTimerSetNextFireDate` (page 10). If *`timer`* does not repeat, the run loop invalidates *`timer`*.

You specify this callback when you create the timer with `CFRunLoopTimerCreate` (page 5).

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

Declared In
`CFRunLoop.h`

Data Types

CFRunLoopTimerContext

A structure that contains program-defined data and callbacks with which you can configure a CFRunLoopTimer's behavior.

```
struct CFRunLoopTimerContext {
    CFIndex version;
    void *info;
    CFAllocatorRetainCallBack retain;
    CFAllocatorReleaseCallBack release;
    CFAllocatorCopyDescriptionCallBack copyDescription;
};
typedef struct CFRunLoopTimerContext CFRunLoopTimerContext;
```

Fields

`version`
: Version number of the structure. Must be 0.

`info`
: An arbitrary pointer to program-defined data, which can be associated with the run loop timer at creation time. This pointer is passed to all the callbacks defined in the context.

`retain`
: A retain callback for your program-defined `info` pointer. Can be `NULL`.

`release`
: A release callback for your program-defined `info` pointer. Can be `NULL`.

`copyDescription`
: A copy description callback for your program-defined `info` pointer. Can be `NULL`.

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

Declared In
`CFRunLoop.h`

CFRunLoopTimerRef

A reference to a run loop timer object.

```
typedef struct __CFRunLoopTimer *CFRunLoopTimerRef;
```

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

Declared In
`CFRunLoop.h`

Document Revision History

This table describes the changes to *CFRunLoopTimer Reference*.

Date	Notes
2006-02-07	Made formatting changes.
2005-08-11	Cosmetic changes to conform to documentation guidelines.
2003-01-01	First version of this document.

Index