Class

NSBackgroundActivityScheduler

A task scheduler suitable for low priority operations that can run in the background.

macOS 10.10+

```
class NSBackgroundActivityScheduler
```

Overview

Use an `NSBackgroundActivityScheduler` object to schedule an arbitrary maintenance or background task. It's similar to an `Timer` object, in that it lets you schedule a repeating or non-repeating task. However, `NSBackgroundActivityScheduler` gives the system flexibility to determine the most efficient time to execute based on energy usage, thermal conditions, and CPU use.

For example, use an `NSBackgroundActivityScheduler` object to schedule:

- Automatic saves
- Backups
- Data maintenance
- Periodic content fetches
- Installation of updates
- Activities occurring in intervals of 10 minutes or more
- Any other deferrable task

For information about performing non-deferrable tasks efficiently, see Specify Nondeferrable Background Activities in Energy Efficiency Guide for Mac Apps.

> **Note**
>
> The `NSBackgroundActivityScheduler` class interfaces with the XPC Activity API. However, your app doesn't need to be an XPC service in order to use `NSBackgroundActivityScheduler`.

Create a Scheduler

To initialize a scheduler, call `init(identifier:)` for `NSBackgroundActivityScheduler`, and pass it a unique identifier string in reverse DNS notation (`nil` and zero-length strings are not allowed) that remains constant across launches of your application.

Swift | Objective-C

```
let activity = NSBackgroundActivityScheduler(identifier: "com.example.MyApp.updatech
```

> **Note**
>
> The system uses this unique identifier to track the number of times the activity has run and to improve the heuristics for deciding when to run it again in the future.

Configure Scheduler Properties

Configure the scheduler with any of the following scheduling properties:

- `repeats`—If set to `true`, the activity is rescheduled at the specified interval after finishing.
- `interval`—For repeating schedulers, the average interval between invocations of the activity. For nonrepeating schedulers, `interval` is the suggested interval of time between scheduling the activity and the invocation of the activity.
- `tolerance`—The amount of time before or after the nominal fire date when the activity should be invoked. The nominal fire date is calculated by using the interval combined with the previous fire date or the time when the activity is started. These two properties create a window in time, during which the activity may be scheduled. The system will more aggressively schedule the activity as it nears the end of the grace period after the nominal fire date. The default value is half the interval.
- `qualityOfService`—The default value is `NSQualityOfServiceBackground`. If you upgrade the quality of service above this level, the system schedules the activity more

aggressively. The default value is the recommended value for most activities. For information on quality of service, see Prioritize Work at the Task Level in Energy Efficiency Guide for Mac Apps.

The next three code examples demonstrate different scheduling scenarios.

Scheduling an activity to fire in the next 10 minutes

Swift Objective-C

```
activity.tolerance = 10 * 60
```

Scheduling an activity to fire between 15 and 45 minutes from now

Swift Objective-C

```
activity.interval = 30 * 60
activity.tolerance = 15 * 60
```

Scheduling an activity to fire once each hour

Swift Objective-C

```
activity.repeats = true
activity.interval = 60 * 60
```

Schedule Activity with scheduleWithBlock:

When you're ready to schedule the activity, call `scheduleWithBlock:` and provide a block of code to execute when the scheduler runs, as shown in the following example. The block will be called on a serial background queue appropriate for the level of quality of service specified. The system automatically uses the `beginActivity(options:reason:)` method (of `ProcessInfo`) while invoking the block, choosing appropriate options based on the specified quality of service.

When your block is called, it's passed a completion handler as an argument. Configure the block to invoke this handler, passing it a result of type `NSBackgroundActivityScheduler.Result` to indicate whether the activity finished (`NSBackgroundActivityScheduler.Result.finished`) or should be deferred (`NSBackgroundActivityScheduler.Result.deferred`) and rescheduled for a later time. Failure to invoke the completion handler results in the activity not being rescheduled. For work that will be deferred and rescheduled, the block may

optionally adjust scheduler properties, such as `interval` or `tolerance`, before calling the completion handler.

Scheduling background activity

Swift Objective-C

```
activity.scheduleWithBlock() { (completion: NSBackgroundActivityCompletionHandler) i
    // Perform the activity
    self.completion(NSBackgroundActivityResult.Finished)
}
```

Detect Whether to Defer Activity

It's conceivable that while a lengthy activity is running, conditions may change, resulting in the activity now requiring deferral. For example, perhaps the user has unplugged the Mac and it's now running on battery power. Your activity can call `shouldDefer` to determine whether this has occurred. A value of `true` indicates that the block should finish what it's currently doing and invoke its completion handler with a value of `NSBackgroundActivityScheduler.Result.deferred`. See the following example.

Detecting deferred background activity

Swift Objective-C

```
if activity.shouldDefer {
    // Wrap up processing and prepare to defer activity
    self.completion(NSBackgroundActivityResult.Deferred)
} else {
    // Continue processing
    self.completion(NSBackgroundActivityResult.Finished)
}
```

Stop Activity

Call `invalidate()` to stop scheduling an activity, as shown in the following example.

Stopping background activity

Swift Objective-C

```
activity.invalidate()
```

> **Note**
>
> When an activity is stopped, a block that's currently executing will still finish executing.

Topics

Background Scheduler Attributes

`var identifier: String`

A unique reverse DNS notation string, such as `com.example.MyApp.updatecheck`, that identifies the activity.

`var repeats: Bool`

A Boolean value indicating whether the activity should be rescheduled after it completes.

`var interval: TimeInterval`

An integer providing a suggested interval between scheduling and invoking the activity.

`var qualityOfService: QualityOfService`

A value of type `NSQualityOfService`, which controls how aggressively the system schedules the activity.

`var shouldDefer: Bool`

A Boolean value indicating whether your app should stop performing background activity and resume at a more optimal time.

`var tolerance: TimeInterval`

A value of type `TimeInterval`, which specifies a range of time during which the background activity may occur.

Initializing Schedulers

`init(identifier: String)`

Initializes a background activity scheduler object with a specified unique identifier.

Scheduling Activity

`func schedule((NSBackgroundActivityScheduler.CompletionHandler) -> Void)`

Begins scheduling the background activity.

`typealias CompletionHandler`

Stopping Scheduled Activity

`func invalidate()`

Prevents the background activity from being scheduled again.

Constants

`enum Result`

These constants indicate whether background activity has been completed successfully or whether additional processing should be deferred until a more optimal time.

`enum QualityOfService`

Constants that indicate the nature and importance of work to the system.

Relationships

Inherits From

`NSObject`

Conforms To

`CVarArg`
`CustomDebugStringConvertible`
`CustomStringConvertible`
`Equatable`
`Hashable`
`NSObjectProtocol`

See Also

System Interaction

`class ProcessInfo`

A collection of information about the current process.