Class

AlarmManager

An object that exposes functions to work with alarms: scheduling, snoozing, cancelling.

iOS 26.0+ | iPadOS 26.0+

```
class AlarmManager
```

Overview

Schedule your alarm alert using `AlarmManager`. The following example calls the `AlarmManager` schedule function by passing in the id and configuration.

```
Task {
let _ = try? await AlarmManager.shared.schedule(id: id, configuration: configuratior
}
```

Topics

Creating a shared instance

`static let shared: AlarmManager`

The singleton instance for interacting with the alarm system.

Updating an alarm

```
struct AlarmUpdates
```

An async sequence that publishes whenever an alarm changes.

```
var alarmUpdates: some AsyncSequence<Array<Alarm>, Never>
```

An asynchronous sequence that emits events when the set of alarms changes.

```
var alarms: [Alarm]
```

Fetches all alarms from the daemon that belong to the current client.

Scheduling an alarm

```
func schedule<Metadata>(id: Alarm.ID, configuration: AlarmManager.Alarm
Configuration<Metadata>) async throws -> Alarm
```

Schedules a new alarm.

```
struct AlarmConfiguration
```

An object that contains all the properties necessary to schedule an alarm.

Requesting authorization

```
func requestAuthorization() async throws -> AlarmManager.Authorization
State
```

Requests permission to use the alarm system if it hasn’t been requested before.

Checking authorization status

```
struct AlarmAuthorizationStateUpdates
```

An asynchronous sequence that publishes a new value when authorization for the alarms and timers system changes.

```
var authorizationUpdates: some AsyncSequence<AlarmManager.Authorization
State, Never>
```

An asynchronous sequence that emits events when authorization to use alarms changes.

```
enum AuthorizationState
```

An enumeration describing all authorization states for the client process.

```
var authorizationState: AlarmManager.AuthorizationState
```

Returns the current authorization state for this client.

Changing an alarm state

```
func cancel(id: Alarm.ID) throws
```

Cancels the alarm with the specified ID.

```
func countdown(id: Alarm.ID) throws
```

Performs a countdown for the alarm with the specified ID if it’s currently alerting.

```
func pause(id: Alarm.ID) throws
```

Pauses the alarm with the specified ID if it’s in the countdown state.

```
func resume(id: Alarm.ID) throws
```

Resumes the alarm with the specified ID if it’s in the paused state.

```
func stop(id: Alarm.ID) throws
```

Stops the alarm with the specified ID.

Throwing an error

```
enum AlarmError
```

An error that occurs when trying to schedule a timer.

See Also

Alarm management

{} Scheduling an alarm with AlarmKit

Create prominent alerts at specified dates for your iOS app.

```
struct Alarm
```

An object that describes an alarm that can alert once or on a repeating schedule.