Class

EKEventStore

An object that accesses a person's calendar events and reminders and supports the scheduling of new events.

iOS 4.0+ | iPadOS 4.0+ | Mac Catalyst 13.1+ | macOS 10.8+ | visionOS 1.0+ | watchOS 2.0+

```
class EKEventStore
```

Mentioned in

- Retrieving events and reminders
- Accessing the event store
- Creating events and reminders

Overview

The `EKEventStore` class is an app's point of contact for accessing calendar and reminder data.

After initializing the event store, you must request access to events or reminders before attempting to fetch or create data. To request access to reminders, call `requestFullAccessToReminders(completion:)`. To request access to events, call `requestWriteOnlyAccessToEvents(completion:)` or `requestFullAccessToEvents(completion:)`.

Important

To request access to events and reminders, your app needs to include permission strings in its `Info.plist` file that explain to someone why the app needs access. For more information, see Accessing the event store.

A typical workflow for using an event store is:

1. Create a predicate, or a search query for events, with `predicateForEvents(withStart:end:calendars:)`.
2. Fetch and process events that match the predicate with the `events(matching:)` and `enumerateEvents(matching:using:)` methods.
3. Save and delete events from the event store with the `save(_:span:commit:)` and `remove(_:span:commit:)` methods.

Use similar methods to access and manipulate reminders.

After receiving an object from an event store, don't use that object with a different event store. This restriction applies to `EKObject` subclasses such as `EKEvent`, `EKReminder`, `EKCalendar`, and `EKSource`, as well as predicates that the event store creates. For example, don't fetch an event from one event store, modify the event, and then pass it to `save(_:span:)` in a different store.

Topics

Creating event stores

`init()`

Creates a new event store.

`init(sources: [EKSource])`

Creates an event store that contains data for the specified sources.

`var eventStoreIdentifier: String`

The unique identifier for the event store.

Requesting access to events and reminders

`func requestWriteOnlyAccessToEvents(completion: (Bool, (any Error)?) -> Void)`

Prompts the person using your app to grant or deny write access to event data.

`func requestFullAccessToEvents(completion: (Bool, (any Error)?) -> Void)`

Prompts people to grant or deny read and write access to event data.

`func requestFullAccessToReminders(completion: (Bool, (any Error)?) -> Void)`

Prompts people to grant or deny read and write access to reminders.

`class func authorizationStatus(for: EKEntityType) -> EKAuthorizationStatus`

Determines the authorization status for the given entity type.

`enum EKAuthorizationStatus`

The current authorization status for a specific entity type.

`typealias EKEventStoreRequestAccessCompletionHandler`

The signature for a closure that EventKit calls when requesting access to event and reminder data.

`NSCalendarsFullAccessUsageDescription`

A message that tells people why the app is requesting access to read and write their calendar data.

`NSCalendarsWriteOnlyAccessUsageDescription`

A message that tells people why the app is requesting access to create calendar events.

`NSRemindersFullAccessUsageDescription`

A message that tells people why the app is requesting access to read and write their reminders data.

Accessing account sources

`var sources: [EKSource]`

An unordered array of objects that represent accounts that contain calendars.

`var delegateSources: [EKSource]`

The event sources delegated to the person using your app.

`func source(withIdentifier: String) -> EKSource?`

Locates an event source with the specified identifier.

Saving and restoring state

`func commit() throws`

Commits all unsaved changes to the event store.

`func reset()`

Reverts the event store to its saved state.

`func refreshSourcesIfNecessary()`

Pulls new data from remote sources, if necessary.

Accessing calendars

`var defaultCalendarForNewEvents: EKCalendar?`

The calendar that events are added to by default, as specified by user settings.

`func defaultCalendarForNewReminders() -> EKCalendar?`

Identifies the default calendar for adding reminders to, as specified by user settings.

`func calendars(for: EKEntityType) -> [EKCalendar]`

Identifies the calendars that support a given entity type, such as reminders or events.

`func calendar(withIdentifier: String) -> EKCalendar?`

Locates a calendar with the specified identifier.

`func saveCalendar(EKCalendar, commit: Bool) throws`

Saves a calendar to the event store by either committing or batching the changes.

`func removeCalendar(EKCalendar, commit: Bool) throws`

Removes a calendar from the event store by either committing or batching the changes.

~~`var calendars: [EKCalendar]`~~

The calendars associated with the event store.

Deprecated

Accessing calendar events

`func event(withIdentifier: String) -> EKEvent?`

Locates the first occurrence of an event with a given identifier.

`func calendarItem(withIdentifier: String) -> EKCalendarItem?`

Locates a reminder or the first occurrence of an event with the specified identifier.

`func calendarItems(withExternalIdentifier: String) -> [EKCalendarItem]`

Locates all reminders or the first occurrences of all events with the specified external identifier.

`func remove(EKEvent, span: EKSpan) throws`

Removes an event from the event store.

`func remove(EKEvent, span: EKSpan, commit: Bool) throws`

Removes an event or recurring events from the event store by either committing or batching the changes.

`func remove(EKReminder, commit: Bool) throws`

Removes a reminder from the event store by either committing or batching the changes.

`func save(EKEvent, span: EKSpan) throws`

Saves changes to an event permanently.

`func save(EKEvent, span: EKSpan, commit: Bool) throws`

Saves an event or recurring events to the event store by either committing or batching the changes.

`func save(EKReminder, commit: Bool) throws`

Saves changes to a reminder by either committing or batching the changes.

Searching calendars

`func enumerateEvents(matching: NSPredicate, using: EKEventSearchCallback)`

Finds all events that match a given predicate and calls a given callback for each event found.

`func events(matching: NSPredicate) -> [EKEvent]`

Finds all events that match a given predicate.

`func fetchReminders(matching: NSPredicate, completion: ([EKReminder]?) -> Void) -> Any`

Fetches reminders that match a given predicate.

`func cancelFetchRequest(Any)`

Cancels the request to fetch reminders.

`func predicateForEvents(withStart: Date, end: Date, calendars: [EKCalendar]?) -> NSPredicate`

Creates a predicate to identify events that occur within a given date range.

`func predicateForReminders(in: [EKCalendar]?) -> NSPredicate`

Creates a predicate to identify all reminders in a collection of calendars.

`func predicateForCompletedReminders(withCompletionDateStarting: Date?, ending: Date?, calendars: [EKCalendar]?) -> NSPredicate`

Creates a predicate to identify all completed reminders that occur within a given date range.

`func predicateForIncompleteReminders(withDueDateStarting: Date?, ending: Date?, calendars: [EKCalendar]?) -> NSPredicate`

Creates a predicate to identify all incomplete reminders that occur within a given date range.

`typealias EKEventSearchCallback`

The signature for a closure that operates on events when enumerating them.

Deprecated methods

~~`func requestAccess(to: EKEntityType, completion: (Bool, (any Error)?) -> Void)`~~

Prompts the person using your app to grant or deny access to event or reminder data.

Deprecated

Structures

`struct EventStoreChanged`

A notification posted when changes are made to the Calendar or Reminders database.

Relationships

Inherits From

`NSObject`

Conforms To

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

`NSObjectProtocol`

See Also

Essentials

Accessing the event store

Request access to a person's calendar data through the event store.

Accessing Calendar using EventKit and EventKitUI

Choose and implement the appropriate Calendar access level in your app.