

# MBS Xojo Event Kit

Version 1.2, ©2018 by Christian Schmitz

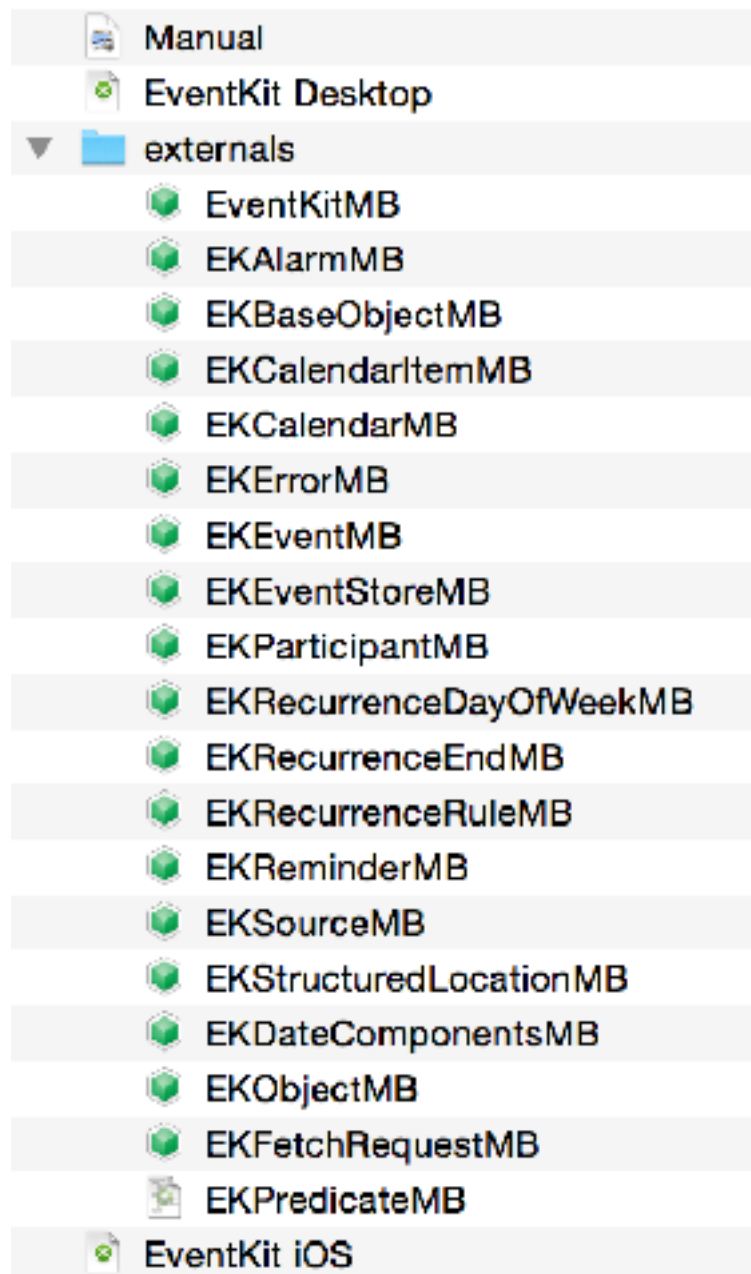
|                               |    |
|-------------------------------|----|
| About the MBS Xojo Event Kit  | 2  |
| Interfaces                    | 4  |
| EKAlarmMB class               | 4  |
| EKBaseObjectMB class          | 6  |
| EKCalendarItemMB class        | 7  |
| EKCalendarMB class            | 9  |
| EKDateComponentsMB class      | 11 |
| EKErrorMB class               | 12 |
| EKEventMB class               | 13 |
| EKEventStoreMB class          | 15 |
| EKFetchRequestMB class        | 20 |
| EKObjectMB class              | 21 |
| EKParticipantMB class         | 22 |
| EKPredicateMB class           | 24 |
| EKRecurrenceDayOfWeekMB class | 25 |
| EKRecurrenceEndMB class       | 26 |
| EKRecurrenceRuleMB class      | 27 |
| EKReminderMB class            | 32 |
| EKSourceMB class              | 34 |
| EKStructuredLocationMB class  | 35 |
| EventKitMB class              | 36 |
| Version History               | 38 |
| Known issues                  | 38 |
| Installation                  | 39 |
| Requirements                  | 40 |
| License                       | 41 |
| Contact                       | 42 |

## About the MBS Xojo Event Kit

The MBS Xojo Event Kit provides you with a few useful classes and modules to easily use calendar and reminder functions in iOS.

### The Kit contains:

- Wrapper for OS X and iOS for EventKit framework.



The Event Kit contains a wrapper for EventKit, Apple's latest framework for calendar and reminder functions on OS X and iOS:

## **Features**

- Query/Get access to the internal calendar and reminder database
- Create, modify and delete reminders and events
- Handle multiple sources of calendar data
- Query participants for events
- asynchronously fetch events or reminders
- build search parameters using predicates
- Handle recurrence rules in various aspects

## **Wrapper Features**

- For Xojo 2015r1
- Using exception handling to track error
- Test code included
- All classes with MB postfix to avoid name conflicts.
- Compiles for all targets
- All module definitions are protected to avoid conflicts
- Inline documentation
- Full Source code, no encryption
- Works for 32bit and 64bit targets.
- Example projects for Desktop, iOS and Console using our classes.

# Interfaces

## EKAlarmMB class

**Class** EKAlarmMB **Inherits** EKObjectMB

**ComputedProperty** Description **As** text

**Sub Get()**

Description of this object.

**ComputedProperty** absoluteDate **As** Date

**Sub Set()**

Represents an alarm that fires at a specific date.

**Sub Get()**

**ComputedProperty** proximity **As** EKAlarmProximity

**Sub Set()**

Defines whether this alarm triggers via entering/exiting a geofence as defined by structuredLocation.

**Sub Get()**

**ComputedProperty** relativeOffset **As Double**

**Sub Set()**

Specifies a relative offset from an event start date to fire an alarm.

Set this property to an appropriate negative value to establish an alarm trigger relative to the start date/time of an event. Setting this clears any existing date trigger.

**Sub Get()**

Specifies a relative offset from an event start date to fire an alarm.

Set this property to an appropriate negative value to establish an alarm trigger relative to the start date/time of an event. Setting this clears any existing date trigger.

**ComputedProperty** structuredLocation **As** EKStructuredLocationMB

**Sub Set()**

Allows you to set a structured location (a location with a potential geo-coordinate) on an alarm. This is used in conjunction with proximity to do geofence-based triggering of reminders.

**Sub Get()**

**Enum** EKAlarmProximity

None = 0

Enter = 1

Leave = 2

**End Enum**

**Sub** Constructor(offset **as Double**)

Creates a new alarm with a relative trigger time.

**Sub** Constructor(Handle **as Integer**, Retain **as Boolean**)

Create from Handle

**Sub** Constructor(da **as date**)

Creates a new alarm with an absolute trigger time.

**Shared Function** alarmWithAbsoluteDate(da **as date**) **As** EKAlarmMB

Creates a new alarm with an absolute trigger time.

**Shared Function** alarmWithRelativeOffset(offset **as Double**) **As** EKAlarmMB

Creates a new alarm with a relative trigger time.

**Function** copy() **As** EKAlarmMB

Copy function for NSCopying protocol

**Note** "About"

**Note** "Copyright"

© 2015 by Christian Schmitz, Monkeybread Software

This is part of the MBS Xojo Event Kit for iOS

<http://www.monkeybreadsoftware.de/xojo/>

**Note** "Note"

The EKAlarm class provides an interface for accessing and manipulating calendar event alarms.

The EKAlarm class represents alarms on an event. An alarm can be relative (e.g. 15 mins before) or absolute (specific time).

**End Class**

## EKBaseObjectMB class

**Class** EKBaseObjectMB

**Sub** Destructor()

Cleanup

**Note** "About"

This is our base class for EventKit class.

**Note** "Copyright"

© 2015 by Christian Schmitz, Monkeybread Software

This is part of the MBS Xojo Event Kit for iOS

<http://www.monkeybreadsoftware.de/xojo/>

**Property** Handle **As Integer**

**End Class**

## EKCalendarItemMB class

**Class** EKCalendarItemMB **Inherits** EKObjectMB

**ComputedProperty** URL **As** text

**Sub Set()**

set the URL

**Sub Get()**

The URL

**ComputedProperty** calendar **As** EKCalendarMB

**Sub Set()**

The calendar

**Sub Get()**

**ComputedProperty** calendarItemExternalIdentifier **As** text

**Sub Get()**

A server-provided identifier for this calendar item

This identifier, provided by the server, allows you to reference the same event or reminder

across

multiple devices. For calendars stored locally on the device, including the birthday calendar, it simply passes through to calendarItemIdentifier.

This identifier is unique as of creation for every calendar item. However, there are some cases where duplicate copies of a calendar item can exist in the same database, including:

- A calendar item was imported from an ICS file into multiple calendars
- An event was created in a calendar shared with the user and the user was also invited to the

event

- The user is a delegate of a calendar that also has this event
- A subscribed calendar was added to multiple accounts

In such cases, you should choose between calendar items based on other factors, such as the calendar or source.

This identifier is the same for all occurrences of a recurring event. If you wish to differentiate between occurrences, you may want to use the start date.

In addition, there are two caveats for Exchange-based calendars:

- This identifier will be different between EventKit on iOS versus OS X
- This identifier will be different between devices for EKReminders

**ComputedProperty** calendarItemIdentifier **As** text

**Sub Get()**

A unique identifier for a calendar item.

Item identifiers are not sync-proof in that a full sync will lose

this identifier, so you should always have a back up plan for dealing

with a reminder that is no longer fetchable by this property, e.g. by title, etc.

Use `EKEventStoreMB.calendarItemWithIdentifier` to look up the item by this value.

**ComputedProperty** creationDate **As** Date

**Sub Get()**

The creation date, can be nil!

**ComputedProperty** hasAlarms **As Boolean**

**Sub Get()**

this item has alarms?

**ComputedProperty** hasAttendees **As Boolean**

**Sub Get()**

this item has attendees?

**ComputedProperty** hasNotes **As Boolean**

**Sub Get()**

this item has notes?

**ComputedProperty** hasRecurrenceRules **As Boolean**

**Sub Get()**

this item has Recurrence Rules?

**ComputedProperty** lastModifiedDate **As Date**

**Sub Get()**

The last modification date

**ComputedProperty** location **As text**

**Sub Set()**

set the location

**Sub Get()**

the location

**ComputedProperty** notes **As text**

**Sub Set()**

the notes

**Sub Get()**

**ComputedProperty** title **As text**

**Sub Set()**

set the title

**Sub Get()**

the title

**Sub** addAlarm(alarm **as** EKAlarmMB)

Adds an alarm to this item.

This method add an alarm to an item. Be warned that some calendars can only allow a certain maximum number of alarms. When this item is saved, it will truncate any extra alarms from the array.

**Sub** addRecurrenceRule(rule **as** EKRecurrenceRuleMB)

Adds a recurrence rule

**Function** alarms() **As** EKAlarmMB()

An array of EKAlarm objects for alarms

**Function** attendees() **As** EKParticipantMB()

An array of EKParticipant objects for attendees

**Function** recurrenceRules() **As** EKRecurrenceRuleMB()

An array of EKRecurrenceRules, or nil if none.

**Sub** removeAlarm(alarm **as** EKAlarmMB)

Removes an alarm from this item.

**Sub** removeRecurrenceRule(rule **as** EKRecurrenceRuleMB)

Removes a recurrence rule

**Note** "Copyright"

© 2015 by Christian Schmitz, Monkeybread Software

This is part of the MBS Xojo Event Kit for iOS

<http://www.monkeybreadsoftware.de/xojo/>

**End Class**



## EKCalendarMB class

**Class** EKCalendarMB **Inherits** EKObjectMB

**ComputedProperty** CGColorRef **As Integer**

**Sub Set()**

The calendar color as a CGColorRef.

**Sub Get()**

**ComputedProperty** ColorValue **As color**

**Sub Set()**

The calendar color as a CGColorRef.

**Sub Get()**

**ComputedProperty** Immutable **As Boolean**

**Sub Get()**

If this is set to YES, it means you cannot modify any attributes of the calendar or delete it. It does NOT imply that you cannot add events or reminders to the calendar.

**ComputedProperty** allowedEntityTypes **As Integer**

**Sub Get()**

Returns the entity types this calendar can contain. While our API only allows creation of single-entity calendars, other servers might allow mixed-entity calendars.

**ComputedProperty** allowsContentModifications **As Boolean**

**Sub Get()**

Represents whether you can this add, remove, or modify items in this calendar.

**ComputedProperty** calendarIdentifier **As text**

**Sub Get()**

A unique identifier for the calendar. It is not sync-proof in that a full sync will lose this identifier, so you should always have a back up plan for dealing with a calendar that is no longer fetchable by this property, e.g. by title, type, color, etc. Use EKEventStoreMB.calendarWithIdentifier to look up the calendar by this value.

**ComputedProperty** source **As EKSourceMB**

**Sub Set()**

The source representing the 'account' this calendar belongs to. This is only settable when initially creating a calendar and then effectively read-only after that. That is, you can create a calendar, but you cannot move it to another source.

**Sub Get()**

**ComputedProperty** subscribed **As Boolean**

**Sub Get()**

True if this calendar is a subscribed calendar.

**ComputedProperty** supportedEventAvailabilities **As Integer**

**Sub Get()**

Returns a bitfield of supported event availabilities, or EKCalendarEventAvailabilityNone if this calendar does not support setting availability on an event.

**ComputedProperty** timeZone **As xojo.Core.TimeZone**

**Sub Set()**

The time zone

**Sub Get()**

**ComputedProperty** title **As text**

**Sub Set()**

The title of the calendar.

**Sub Get()**

**ComputedProperty** type **As EKCalendarType**

**Sub Get()**

The type of the calendar as an EKCalendarType. This is actually based on what source the calendar is in, as well as whether it is a subscribed calendar.

CalDAV subscribed calendars have type EKCalendarTypeCalDAV with isSubscribed = YES.

**Const** EKCalendarEventAvailabilityBusy = 1  
**Const** EKCalendarEventAvailabilityFree = 2  
**Const** EKCalendarEventAvailabilityNone = 0  
**Const** EKCalendarEventAvailabilityTentative = 4  
**Const** EKCalendarEventAvailabilityUnavailable = 8

**Enum** EKCalendarType  
Local = 0  
CalDAV = 1  
Exchange = 2  
Subscription = 3  
Birthday = 4

**End Enum**

**Sub** Constructor(entityType **as** EKEEventStoreMB.EKEntityType, Store **as** EKEEventStoreMB)

Creates a new calendar that may contain the given entity type.

You can only create calendars that accept either reminders or events via our API.

However, other servers might allow mixing the two (though it is not common).

**Sub** Constructor(Handle **as Integer**, Retain **as Boolean**)

Create from Handle

**Shared Function** calendarForEntityType(entityType **as** EKEEventStoreMB.EKEntityType, Store **as** EKEEventStoreMB) **As** EKCalendarMB

Creates a new calendar that may contain the given entity type.

You can only create calendars that accept either reminders or events via our API.

However, other servers might allow mixing the two (though it is not common).

**Note** "About"

The EKCalendar class represents a calendar for events.

**Note** "Copyright"

© 2015 by Christian Schmitz, Monkeybread Software

This is part of the MBS Xojo Event Kit for iOS

<http://www.monkeybreadsoftware.de/xojo/>

**End Class**

## EKDateComponentsMB class

**Class** EKDateComponentsMB **Inherits** EKBaseObjectMB

**ComputedProperty** Description **As** text

**Sub Get()**

Description of this object.

**ComputedProperty** date **As** date

**Sub Get()**

**ComputedProperty** day **As** integer

**Sub Set()**

**Sub Get()**

**ComputedProperty** hour **As** integer

**Sub Set()**

**Sub Get()**

**ComputedProperty** minute **As** integer

**Sub Set()**

**Sub Get()**

**ComputedProperty** month **As** integer

**Sub Set()**

**Sub Get()**

**ComputedProperty** second **As** integer

**Sub Set()**

**Sub Get()**

**ComputedProperty** timeZone **As** xojo.Core.TimeZone

**Sub Set()**

The time zone

**Sub Get()**

**ComputedProperty** validDate **As** Boolean

**Sub Get()**

Reports whether or not the combination of properties which have been set in the receiver is a date which exists in the calendar.

This method is not appropriate for use on NSDateComponents objects which are specifying relative quantities of calendar components.

Except for some trivial cases (e.g., 'seconds' should be 0 - 59 in any calendar), this method is not necessarily cheap.

If the time zone property is set in the NSDateComponents object, it is used.

The calendar property must be set, or false is returned.

**ComputedProperty** weekday **As** integer

**Sub Set()**

**Sub Get()**

**ComputedProperty** year **As** integer

**Sub Set()**

**Sub Get()**

**Sub** Constructor()

**Sub** Constructor(Handle **as** Integer, Retain **as** Boolean)

Create from Handle

**Note** "About"

Wrapper for NSDateComponents.

Needed for EKReminderMB

**Note** "Copyright"

© 2015 by Christian Schmitz, Monkeybread Software

This is part of the MBS Xojo Event Kit for iOS

<http://www.monkeybreadsoftware.de/xojo/>

**End Class**

## EKErrorMB class

```
Class EKErrorMB Inherits EKbaseObjectMB
  ComputedProperty code As integer
    Sub Get()
      get error code
    End Sub
  ComputedProperty domain As text
    Sub Get()
      get error domain
    End Sub
  ComputedProperty localizedDescription As text
    Sub Get()
      get localized error description
    End Sub
  Sub Constructor(Handle as Integer, Retain as Boolean)
    Create from Handle
    Note "Copyright"
    © 2015 by Christian Schmitz, Monkeybread Software

    This is part of the MBS Xojo Event Kit for iOS

    http://www.monkeybreadsoftware.de/xojo/
  End Sub
End Class
```

## EKEventMB class

**Class** EKEventMB **Inherits** EKCalendarItemMB

**ComputedProperty** AllDay **As Boolean**

**Sub Set()**

Indicates this event is an 'all day' event.

**Sub Get()**

**ComputedProperty** Detached **As Boolean**

**Sub Get()**

Represents whether this event is detached from a recurring series.

If this EKEvent is an instance of a repeating event, and an attribute of this EKEvent has been changed from the default value generated by the repeating event, isDetached will return true. If the EKEvent is unchanged from its default state, or is not a repeating event, isDetached returns false.

**ComputedProperty** Name **As text**

**Sub Get()**

A unique identifier for this event.

This identifier can be used to look the event up using EKEventStoreMB.eventWithIdentifier. You can use this not only to simply fetch the event, but also to validate the event has not been deleted out from under you when you get an external change notification via the EKEventStore database changed notification. If eventWithIdentifier returns nil, the event was deleted.

Please note that if you change the calendar of an event, this ID will likely change. It is currently also possible for the ID to change due to a sync operation. For example, if a user moved an event on a different client to another calendar, we'd see it as a completely new event here.

**ComputedProperty** availability **As** EKEventMB.EKEventAvailability

**Sub Set()**

The availability setting for this event.

The availability setting is used by CalDAV and Exchange servers to indicate how the time should be treated for scheduling. If the calendar the event is currently in does not support event availability, EKEventAvailabilityNotSupported is returned.

**Sub Get()**

**ComputedProperty** birthdayPersonID **As Integer**

**Sub Get()**

Specifies the address book ID of the person this event was created for.

This property is only valid for events in the built-in Birthdays calendar. It specifies the Address Book ID of the person this event was created for. For any other type of event, this property returns -1.

**ComputedProperty** endDate **As Date**

**Sub Set()**

The end date for the event.

**Sub Get()**

**ComputedProperty** organizer **As** EKParticipantMB

**Sub Get()**

The organizer of this event, or nil.

**ComputedProperty** startDate **As Date**

**Sub Set()**

The start date for the event.

This property represents the start date for this event. Floating events (such as all-day events) are currently always returned in the default time zone. ([NSTimeZone defaultTimeZone])

**Sub Get()**

**ComputedProperty** status **As** EKEEventMB.EKEEventStatus

**Sub Set()**

The status of the event.

While the status offers four different values in the EKEEventStatus enumeration, in practice, the only actionable and reliable status is canceled. Any other status should be considered informational at best. You cannot set this property. If you wish to cancel an event, you should simply remove it using removeEvent.

**Sub Get()**

**Const** NSOrderedAscending = -1

**Const** NSOrderedDescending = 1

**Const** NSOrderedSame = 0

**Enum** EKEEventAvailability

NotSupported = -1

Busy = 0

Free = 1

Tentative = 2

Unavailable = 3

**End Enum**

**Enum** EKEEventStatus

None = 0

Confirmed = 1

Tentative = 2

Canceled = 3

**End Enum**

**Sub** Constructor(Store **as** EKEEventStoreMB)

Creates a new event object.

**Sub** Constructor(Handle **as Integer**, Retain **as Boolean**)

Create from Handle

**Function** Refresh() **As Boolean**

Refreshes an event object to ensure it's still valid.

**Function** compareStartDateWithEvent(other **as** EKEEventMB) **As Integer**

Comparison function you can pass to sort NSArray's of EKEEvents by start date.  
see NSOrdered constants

**Shared Function** eventWithEventStore(Store **as** EKEEventStoreMB) **As** EKEEventMB

Creates a new event object.

**Note** "About"

The EKEEvent class represents an occurrence of an event.

**Note** "Copyright"

© 2015 by Christian Schmitz, Monkeybread Software

This is part of the MBS Xojo Event Kit for iOS

<http://www.monkeybreadsoftware.de/xojo/>

**End Class**

## EKEventStoreMB class

**Class** EKEventStoreMB **Inherits** EKBaseObjectMB

**ComputedProperty** eventStoreIdentifier **As** text

**Sub** Get()

Returns a unique identifier string representing this calendar store.

**Enum** EKAuthorizationStatus

NotDetermined = 0

Restricted = 1

Denied = 2

Authorized = 3

**End Enum**

**Enum** EKEntityMask

Unknown = 0

Events = 1

Reminders = 2

Both = 3

**End Enum**

**Enum** EKEntityType

Events = 0

Reminders = 1

**End Enum**

**Enum** EKSpan

ThisEvent = 0

FutureEvents = 1

**End Enum**

**Sub** Constructor()

Creates a event store

**Sub** Constructor(Handle **as** Integer, Retain **as** Boolean)

Create from Handle

**Sub** Destructor()

**Shared Function** authorizationStatusForEntityType(entityType **as** EKEntityType) **As** EKAuthorizationStatus

Returns the authorization status for the given entity type

**Function** calendarItemWithIdentifier(identifier **as** text) **As** EKCalendarItemMB

Returns either a reminder or the first occurrence of an event.

**Function** calendarItemsWithExternalIdentifier(identifier **as** text) **As** EKCalendarItemMB()

Returns either matching reminders or the first occurrences of any events matching the given external identifier.

This method returns a set of EKEvents or EKReminders with the given external identifier.

Due to reasons discussed in EKCalendarItem.calendarItemExternalIdentifier, there may be more than one matching calendar item.

externalIdentifier The value obtained from EKCalendarItem's

calendarItemExternalIdentifier property

An unsorted array of EKCalendarItem instances

**Function** calendarWithIdentifier(identifier **as** text) **As** EKCalendarMB

Returns a calendar with a specified identifier.

**Function** calendars() **As** EKCalendarMB()

**Function** calendarsForEntityType(EntityType **as** EKEntityType) **As** EKCalendarMB()

Returns calendars that support a given entity type (reminders, events)

**Sub** cancelFetchRequest(r **as** EKFetchRequestMB)

Given a value returned from fetchRemindersMatchingPredicate, this method can be used to cancel the request. Once called, the completion block specified in fetchReminders... will not be called.

**Function** commit(**byref** error **as** EKErrorMB) **As** Boolean

Commits pending changes to the database.

If you use `saveCalendar/saveEvent/removeCalendar/removeEvent`, etc. and you pass `NO` to their parameter, you are batching changes for a later commit. This method does that commit. This allows you to save the database only once for many additions or changes. If you pass `true` to methods' `commit` parameter, then you don't need to call this method.

This method will return `true` as long as nothing went awry, even if nothing was actually committed. If it returns `false`, error should contain the reason it became unhappy.

**Function** `defaultCalendarForNewEvents()` **As** `EKCalendarMB`

Returns the calendar that events should be added to by default, as set in the Settings application.

**Function** `defaultCalendarForNewReminders()` **As** `EKCalendarMB`

Returns the calendar that reminders should be added to by default, as set in the Settings application.

**Function** `eventWithIdentifier(identifier as text)` **As** `EKEventMB`

Returns the first occurrence of an event matching the given event identifier.

An `EKEvent` object, or `nil` if not found.

**Function** `eventsMatchingPredicate(predicate as EKPredicateMB)` **As** `EKEventMB()`

Searches for events that match the given predicate.

This call executes a search for the events indicated by the predicate passed to it.

It only includes events which have been committed (e.g. those saved using `saveEvent:commit:NO` are not included until `commit` is called.)

It is synchronous. If you want `async` behavior, you should either use `dispatch_async` or `NSOperation` to run the query someplace other than the main thread, and then funnel the array back to the main thread.

`predicate` The predicate to invoke. If this predicate was not created with the predicate creation functions in this class, an exception is raised.

An array of `EKEvent` objects, or `nil`. There is no guaranteed order to the events.

**Function** `fetchRemindersMatchingPredicate(predicate as EKPredicateMB)` **As** `EKFetchRequestMB`

Fetches reminders asynchronously.

This method fetches reminders asynchronously and returns a value which can be used in `cancelFetchRequest` to cancel the request later if desired. The completion block is called with an array of reminders that match the given predicate (or potentially `nil`).

This only includes reminders which have been committed (e.g. those saved using `saveReminder:commit:NO` are not included until `commit` is called.)

**Function** `predicateForCompletedReminders(startDate as Date, endDate as Date, calendars() as EKCalendarMB = nil)` **As** `EKPredicateMB`

Fetch completed reminders in a set of calendars.

You can use this method to search for reminders completed between a range of dates.

You can pass `nil` for start date to find all reminders completed before `endDate`.

You can pass `nil` for both start and end date to get all completed reminders in the specified calendars.

**Function** `predicateForCompletedReminders(startDate as Date, endDate as Date, calendar as EKCalendarMB)` **As** `EKPredicateMB`

Fetch completed reminders in a set of calendars.

**Function** `predicateForEvents(startDate as Date, endDate as Date, calendars() as EKCalendarMB = nil)` **As** `EKPredicateMB`

Creates a predicate for use with `eventsMatchingPredicate` or `enumerateEventsMatchingPredicate:usingBlock:`.

Creates a simple query predicate to search for events within a certain date range. At present, this will return events in the default time zone (`[NSTimeZone defaultTimeZone]`).

`startDate` The start date.

`endDate` The end date.

`calendars` The calendars to search for events in, or `nil` to search all calendars.



**Function** predicateForEvents(startDate as Date, endDate as Date, calendar as EKCalendarMB) As EKPredicateMB

Creates a predicate for use with eventsMatchingPredicate

**Function** predicateForIncompleteReminders(startDate as Date, endDate as Date, calendars() as EKCalendarMB = nil) As EKPredicateMB

Fetch incomplete reminders in a set of calendars.

You can use this method to search for incomplete reminders due in a range.

You can pass nil for start date to find all reminders due before endDate.

You can pass nil for both start and end date to get all incomplete reminders in the specified calendars.

**Function** predicateForIncompleteReminders(startDate as Date, endDate as Date, calendar as EKCalendarMB) As EKPredicateMB

Fetch incomplete reminders in a set of calendars.

**Function** predicateForReminders(calendars() as EKCalendarMB = nil) As EKPredicateMB

Fetch all reminders in a set of calendars.

**Function** predicateForReminders(calendar as EKCalendarMB) As EKPredicateMB

Fetch all reminders in one calendar.

**Sub** refreshSourcesIfNecessary()

Cause a sync to potentially occur taking into account the necessity of it.

You can call this method to pull new data from remote sources.

This only updates the event store's data. If you want to update your objects after refreshing the sources, you should call refresh on each of them afterwards.

On iOS, this sync only occurs if deemed necessary.

On OS X, this will occur regardless of necessity, but may change in a future release to match the iOS

behavior.

**Function** removeCalendar(calendar as EKCalendarMB, commit as Boolean, byref error as EKErrorMB) As Boolean

Removes a calendar from the database.

This method attempts to delete the given calendar from the calendar database. It returns true if successful and false otherwise. Passing a calendar fetched from another EKEventStore instance into this function will raise an exception.

calendar The calendar to delete.

commit Pass true to cause the database to save. You can pass false to batch multiple changes and then call commit to save them all at once.

error If an error occurs, this will contain a valid NSError object on exit.

**Function** removeEvent(theEvent as EKErrorMB, span as EKSpan, byref error as EKErrorMB) As Boolean

Removes an event from the calendar store.

This method attempts to remove the event from the calendar database. It returns YES if successful and false otherwise. It's possible for this method to return false, and error will be set to nil. This occurs if the event wasn't ever added and didn't need removing. This means the correct way to detect failure is a result of NO and a non-nil error parameter.

Passing an event from another CalendarStore into this function will raise an exception. After an event is removed, it is no longer tied to this calendar store, and all data in the event is cleared except for the eventIdentifier.

event The event to save.

span The span to use (this event, or this and future events).

error If an error occurs, this will contain a valid NSError object on exit.

**Function** removeEvent(theEvent as EKEventMB, span as EKSpan, commit as Boolean, byref error as EKErrorMB) As Boolean

Remove event

**Function** removeReminder(reminder as EKReminderMB, commit as Boolean, byref error as EKErrorMB) As Boolean

Removes a reminder from the event store.

This method attempts to remove the reminder from the event store database. It returns true if

successful and false otherwise. Passing a reminder from another EKEventStore into this function will raise an exception. After a reminder is removed, it is no longer tied to this event store.

**Sub** requestAccessToEntityType(entityType as EKEntityType)

Users are able to grant or deny access to event and reminder data on a per-app basis. To request access to event and/or reminder data, call requestAccessToEntityType. This will not block the app while the user is being asked to grant or deny access.

Until access has been granted for an entity type, the event store will not contain any calendars for that entity type, and any attempt to save will fail. The user will only be prompted the first time access is requested; any subsequent instantiations of EKEventStore will use the existing permissions. When the

user taps to grant or deny access, the completion handler will be called on an arbitrary queue.

**Sub** reset()

Resets the event store.

You can use this method to forget ALL changes made to the event store (all additions, all fetched objects, etc.). It essentially is as if you released the store and then created a new one. It brings it back to its initial state. All objects ever created/fetched, etc. using this store are no longer connected to it and are considered invalid.

**Function** saveCalendar(calendar as EKCalendarMB, commit as Boolean, byref error as EKErrorMB) As

Boolean

Saves changes to a calendar, or adds a new calendar to the database.

This method attempts to save the given calendar to the calendar database. It returns true if successful and NO otherwise. Passing a calendar fetched from another EKEventStore instance into this function will raise an exception.

calendar The calendar to save.

commit Pass true to cause the database to save. You can pass false to save multiple calendars and then call commit to save them all at once.

error If an error occurs, this will contain a valid NSError object on exit.

**Function** saveEvent(theEvent as EKEventMB, span as EKSpan, byref error as EKErrorMB) As Boolean

Saves changes to an event permanently.

This method attempts to save the event to the calendar database. It returns YES if successful and false otherwise. It's possible for this method to return NO, and error will be set to nil. This occurs if the event wasn't dirty and didn't need saving. This means the correct way to detect failure is a result of false and a non-nil error parameter. Passing an event fetched from another EKEventStore instance into this function will raise an exception.

After an event is successfully saved, it is also put into sync with the database, meaning that all fields you did not change will be updated to the latest values. If you save the event, but it was deleted by a different store/process, you will effectively recreate the event as a new event.

event The event to save.

span The span to use (this event, or this and future events).

error If an error occurs, this will contain a valid NSError object on exit.

**Function** saveEvent(theEvent as EKEventMB, span as EKSpan, commit as Boolean, byref error as

EKErrorMB) As Boolean

Saves event

**Function** saveReminder(reminder as EKReminderMB, commit as Boolean, byref error as EKErrorMB) As

Boolean

Saves changes to a reminder.

This method attempts to save the reminder to the event store database. It returns true if successful and false otherwise. Passing a reminder fetched from another EKEventStore instance into this function will raise an exception.

After a reminder is successfully saved, its fields are updated to the latest values in the database.

`reminder` The reminder to save.

`commit` Whether to save to the database or not. Pass `NO` to batch changes together and commit with `EKEventStore.commit`.

`error` If an error occurs, this will contain a valid `NSError` object on exit.

**Function** `sourceWithIdentifier(identifier as text) As EKSourceMB`

Returns a source with a specified identifier.

**Function** `sources() As EKSourceMB()`

Returns an unordered array of sources.

**Note "About"**

The `EKEventStore` class provides an interface for accessing and manipulating calendar events and

reminders.

The `EKEventStore` class is the main point of contact for accessing Calendar data. You must create a `EKEventStore` object in order to retrieve/add/delete events or reminders from the Calendar

database.

Events, Reminders, and Calendar objects retrieved from an event store cannot be used with any other

event

store. It is generally best to hold onto a long-lived instance of an event store, most likely as a singleton instance in your application.

**Note "Copyright"**

© 2015 by Christian Schmitz, Monkeybread Software

This is part of the MBS Xojo Event Kit for iOS

<http://www.monkeybreadsoftware.de/xojo/>

**End Class**

## EKFetchRequestMB class

**Class** EKFetchRequestMB

**Sub** Constructor(Handle **as Integer**)

Create from Handle

**Sub** Destructor()

**Note** "Copyright"

© 2015 by Christian Schmitz, Monkeybread Software

This is part of the MBS Xojo Event Kit for iOS

<http://www.monkeybreadsoftware.de/xojo/>

**Property** Cancelled **As Boolean**

**Property** Handle **As Integer**

**End Class**

## EKObjectMB class

**Class** EKObjectMB **Inherits** EKBaseObjectMB

**ComputedProperty** hasChanges **As Boolean**

**Sub Get()**

Returns true if this object or any sub-object (alarm, etc.) has uncommitted changes.

**ComputedProperty** isNew **As Boolean**

**Sub Get()**

Returns YES if this object has never been saved.

**Function** Refresh() **As Boolean**

Determines if the object is still valid (i.e. it still exists in the database), and unloads all properties that have not been modified. If this ever returns false, it indicates the record was deleted and this instance should be discarded and no longer referenced.

**Sub** reset()

If this object is not new, this method will unload all loaded properties and clear any dirty state

**Sub** rollback()

If this object is not new, this method will unload dirty state only.

**Note** "About"

**Note** "Copyright"

© 2015 by Christian Schmitz, Monkeybread Software

This is part of the MBS Xojo Event Kit for iOS

<http://www.monkeybreadsoftware.de/xojo/>

**End Class**

## EKParticipantMB class

**Class** EKParticipantMB **Inherits** EKObjectMB

**ComputedProperty** Description **As** text

**Sub Get()**

Description of this participant.

**ComputedProperty** Name **As** text

**Sub Get()**

Name of this participant.

**ComputedProperty** URL **As** text

**Sub Get()**

URL representing this participant.

**ComputedProperty** isCurrentUser **As Boolean**

**Sub Get()**

A boolean indicating whether this participant represents the owner of this account.

**ComputedProperty** participantRole **As** EKParticipantMB.EKParticipantRole

**Sub Get()**

Returns the role of the attendee as a EKParticipantRole value.

**ComputedProperty** participantStatus **As** EKParticipantMB.EKParticipantStatus

**Sub Get()**

Returns the status of the attendee as a EKParticipantStatus value.

**ComputedProperty** participantType **As** EKParticipantMB.EKParticipantType

**Sub Get()**

Returns the type of the attendee as a EKParticipantType value.

**Enum** EKParticipantRole

Unknown = 0

Required = 1

Optional\_ = 2

Chair = 3

NonParticipant = 4

**End Enum**

**Enum** EKParticipantStatus

Unknown = 0

Pending = 1

Accepted = 2

Declined = 3

Tentative = 4

Delegated = 5

Completed = 6

InProcess = 7

**End Enum**

**Enum** EKParticipantType

Unknown = 0

Person = 1

Room = 2

Resource = 3

Group = 4

**End Enum**

**Function** ABRecordWithAddressBook(ABAddressBookRef **as Integer**) **As Integer**

Returns the ABRecordRef that represents this participant.

This method returns the ABRecordRef that represents this participant, if a match can be found based on email address in the address book passed. If we cannot find the participant, nil is returned.

**Sub** Constructor(Handle **as Integer**, Retain **as Boolean**)

Create from Handle

**Function** copy() **As** EKParticipantMB  
Copy function for NSCopying protocol

**Note** "About"

**Note** "Copyright"

© 2015 by Christian Schmitz, Monkeybread Software

This is part of the MBS Xojo Event Kit for iOS

<http://www.monkeybreadsoftware.de/xojo/>

**End Class**

## **EKPredicateMB class**

**Class** EKPredicateMB **Inherits** EKbaseObjectMB

**Sub** Constructor(Handle **as Integer**, Retain **as Boolean**)

Create from Handle

**Note** "Copyright"

© 2015 by Christian Schmitz, Monkeybread Software

This is part of the MBS Xojo Event Kit for iOS

<http://www.monkeybreadsoftware.de/xojo/>

**End Class**



## EKRecurrenceDayOfWeekMB class

**Class** EKRecurrenceDayOfWeekMB **Inherits** EKBaseObjectMB

**ComputedProperty** dayOfTheWeek **As integer**

**Sub Get()**

The day of the week.

**ComputedProperty** weekNumber **As integer**

**Sub Get()**

The week number.

**Sub** Constructor(Handle **as Integer**, Retain **as Boolean**)

Create from Handle

**Sub** Constructor(dayOfWeek **as Integer**, weekNumber **as Integer** = 0)

Creates an object with a day of the week and week number.

**Function** copy() **As** EKRecurrenceDayOfWeekMB

Copy function for NSCopying protocol

**Shared Function** dayOfWeek(dayOfWeek **as integer**) **As** EKRecurrenceDayOfWeekMB

Creates an object with a day of the week and week number of zero.

**Shared Function** dayOfWeek(dayOfWeek **as integer**, weekNumber **as Integer**) **As**

EKRecurrenceDayOfWeekMB

Creates an object with a day of the week and week number of zero.

**Note** "About"

Class which represents a day of the week this recurrence will occur.

EKRecurrenceDayOfWeek specifies either a simple day of the week, or the nth instance of a particular day of the week, such as the third Tuesday of every month. The week number is only valid when used with monthly or yearly recurrences, since it would be otherwise meaningless.

Valid values for dayOfTheWeek are integers 1-7, which correspond to days of the week with Sunday = 1. Valid values for weekNumber portion are (+/-)1-53, where a negative value indicates a value from the end of the range. For example, in a yearly event -1 means last week of the year. -1 in a Monthly recurrence indicates the last week of the month.

The value 0 also indicates the weekNumber is irrelevant (every Sunday, etc.).

Day-of-week weekNumber values that are out of bounds for the recurrence type will result in an exception when trying to initialize the recurrence. In particular, weekNumber must be zero when passing EKRecurrenceDayOfWeek objects to initialize a weekly recurrence.

**Note** "Copyright"

© 2015 by Christian Schmitz, Monkeybread Software

This is part of the MBS Xojo Event Kit for iOS

<http://www.monkeybreadsoftware.de/xojo/>

**End Class**

## EKRecurrenceEndMB class

**Class** EKRecurrenceEndMB **Inherits** EKBaseObjectMB

**ComputedProperty** completionDate **As** Date

**Sub Get()**

The end date of this recurrence, or nil if it's count-based.

**ComputedProperty** occurrenceCount **As integer**

**Sub Get()**

The maximum occurrence count, or 0 if it's date-based.

**Sub Constructor**(Count **as Integer**)

Creates a recurrence end with a maximum occurrence count.

**Sub Constructor**(Handle **as Integer**, Retain **as Boolean**)

Create from Handle

**Sub Constructor**(EndDate **as date**)

Creates a recurrence end with a maximum occurrence count.

**Function** copy() **As** EKRecurrenceEndMB

Copy function for NSCopying protocol

**Shared Function** recurrenceEndWithEndDate(EndDate **as date**) **As** EKRecurrenceEndMB

Creates an autoreleased recurrence end with a specific end date.

**Shared Function** recurrenceEndWithOccurrenceCount(count **as integer**) **As** EKRecurrenceEndMB

Creates a recurrence end with a maximum occurrence count.

**Note "About"**

Class which represents when a recurrence should end.

EKRecurrenceEnd is an attribute of EKRecurrenceRule that defines how long the recurrence is scheduled to repeat. The recurrence can be defined either with an Integer that indicates the total number times it repeats, or with an NSDate, after which it no longer repeats. An event which is set to never end should have its EKRecurrenceEnd set to nil.

If the end of the pattern is defined with an Date, the client must pass a valid NSDate, nil cannot be passed. If the end of the pattern is defined as terms of a number of occurrences, the occurrenceCount passed to the initializer must be positive, it cannot be 0. If the client attempts to initialize a EKRecurrenceEnd with a nil Date or OccurrenceCount of 0, an exception is raised.

A EKRecurrenceEnd initialized with an end date will return 0 for occurrenceCount.

One initialized with a number of occurrences will return nil for its endDate.

**Note "Copyright"**

© 2015 by Christian Schmitz, Monkeybread Software

This is part of the MBS Xojo Event Kit for iOS

<http://www.monkeybreadsoftware.de/xojo/>

**End Class**

## EKRecurrenceRuleMB class

**Class** EKRecurrenceRuleMB **Inherits** EKObjectMB

**ComputedProperty** Description **As** text

**Sub Get()**

Description of this object.

**ComputedProperty** calendarIdentifier **As** text

**Sub Get()**

Calendar used by this recurrence rule.

**ComputedProperty** firstDayOfTheWeek **As integer**

**Sub Get()**

Recurrence patterns can specify which day of the week should be treated as the first day.

Possible values for this

property are integers 0 and 1-7, which correspond to days of the week with Sunday = 1. Zero

indicates that the

property is not set for this recurrence. The first day of the week only affects the way the

recurrence is expanded

for weekly recurrence patterns with an interval greater than 1. For those types of recurrence

patterns, the

Calendar framework will set firstDayOfTheWeek to be 2 (Monday). In all other cases, this

property will be set

to zero. The iCalendar spec stipulates that the default value is Monday if this property is not

set.

**ComputedProperty** frequency **As** EKRecurrenceRuleMB.EKRecurrenceFrequency

**Sub Get()**

This property designates the unit of time used to describe the recurrence pattern.

**ComputedProperty** interval **As integer**

**Sub Get()**

The interval of a EKRecurrenceRule is an integer value which specifies how often the

recurrence rule repeats

over the unit of time described by the EKRecurrenceFrequency. For example, if the

EKRecurrenceFrequency is

EKRecurrenceWeekly, then an interval of 1 means the pattern is repeated every week. A

value of 2

indicates it is repeated every other week, 3 means every third week, and so on. The value

must be a

positive integer; 0 is not a valid value, and nil will be returned if the client attempts to

initialize a

rule with a negative or zero interval.

**ComputedProperty** recurrenceEnd **As** EKRecurrenceEndMB

**Sub Set()**

This property defines when the the repeating event is scheduled to end. The end date can be

specified by a number of

occurrences, or with an end date.

**Sub Get()**

**Enum** EKRecurrenceFrequency

Daily = 0

Weekly = 1

Monthly = 2

Yearly = 3

**End Enum**

**Sub** Constructor(type **as** EKRecurrenceRuleMB.EKRecurrenceFrequency, interval **as integer**, daysOfTheWeek() **as** EKRecurrenceDayOfWeekMB, daysOfTheMonth() **as Integer**, monthsOfTheYear() **as Integer**, weekOfTheYear() **as Integer**, daysOfTheYear() **as integer**, setPositions() **as Integer**, ende **as** EKRecurrenceEndMB = **nil**)

The designated initializer.

This can be used to build any kind of recurrence rule. But be aware that certain combinations make no sense and will be ignored. For example, if you pass `daysOfTheWeek` for a daily recurrence, they will be ignored.

`type` The type of recurrence

`interval` The interval. Passing zero will raise an exception.

`daysOfTheWeek` An array of `EKRecurrenceDayOfWeek` objects. Valid for all recurrence types except daily. Ignored otherwise.

Corresponds to the `BYDAY` value in the iCalendar specification.

`daysOfTheMonth` An array of Integers ([+/-] 1 to 31). Negative numbers infer counting from the end of the month.

For example, -1 means the last day of the month. Valid only for monthly recurrences. Ignored otherwise.

Corresponds to the `BYMONTHDAY` value in the iCalendar specification.

`monthsOfTheYear` An array of Integers (1 to 12). Valid only for yearly recurrences. Ignored otherwise. Corresponds to

the `BYMONTH` value in the iCalendar specification.

`weeksOfTheYear` An array of Integers ([+/-] 1 to 53). Negative numbers infer counting from the end of the year.

For example, -1 means the last week of the year. Valid only for yearly recurrences. Ignored otherwise.

Corresponds to the `BYWEEKNO` value in the iCalendar specification.

`daysOfTheYear` An array of Integers ([+/-] 1 to 366). Negative numbers infer counting from the end of the year.

For example, -1 means the last day of the year. Valid only for yearly recurrences. Ignored otherwise.

Corresponds to the `BYYEARDAY` value in the iCalendar specification.

`setPositions` An array of Integers ([+/-] 1 to 366). Used at the end of recurrence computation to filter the list

to the positions specified. Negative numbers indicate starting at the end, i.e. -1 indicates taking the last result of the set. Valid when `daysOfTheWeek`, `daysOfTheMonth`, `monthsOfTheYear`,

`weeksOfTheYear`, or

`daysOfTheYear` is passed. Ignored otherwise. Corresponds to the `BYSETPOS` value in the iCalendar specification.

`end` The recurrence end, or nil.

**Sub Constructor**(`type` as `EKRecurrenceRuleMB.EKRecurrenceFrequency`, `interval` as `integer`, `ende` as `EKRecurrenceEndMB = nil`)

Simple initializer to create a recurrence.

This is used to create a simple recurrence with a specific type, interval and end.

If interval is 0, an exception is raised. The end parameter can be nil.

**Sub Constructor**(`Handle` as `Integer`, `Retain` as `Boolean`)

Create from Handle

**Shared Function** `RecurrenceRule`(`type` as `EKRecurrenceRuleMB.EKRecurrenceFrequency`, `interval` as `integer`, `daysOfTheWeek()` as `EKRecurrenceDayOfWeekMB`, `daysOfTheMonth()` as `Integer`, `monthsOfTheYear()` as `Integer`, `weekOfTheYear()` as `Integer`, `daysOfTheYear()` as `integer`, `setPositions()` as `Integer`, `ende` as `EKRecurrenceEndMB = nil`) **As** `EKRecurrenceRuleMB`

Creates a recurrence.

This can be used to build any kind of recurrence rule. But be aware that certain combinations make no sense and will be ignored. For example, if you pass `daysOfTheWeek` for a daily recurrence, they will be ignored.

`type` The type of recurrence

`interval` The interval. Passing zero will raise an exception.

`daysOfTheWeek` An array of `EKRecurrenceDayOfWeek` objects. Valid for all recurrence types except daily. Ignored otherwise.

Corresponds to the `BYDAY` value in the iCalendar specification.

`daysOfTheMonth` An array of Integers ([+/-] 1 to 31). Negative numbers infer counting from the end of the month.

For example, -1 means the last day of the month. Valid only for monthly recurrences. Ignored otherwise.

Corresponds to the BYMONTHDAY value in the iCalendar specification.

monthsOfTheYear An array of Integers (1 to 12). Valid only for yearly recurrences. Ignored otherwise. Corresponds to

the BYMONTH value in the iCalendar specification.

weeksOfTheYear An array of Integers ([+/-] 1 to 53). Negative numbers infer counting from the end of the year.

For example, -1 means the last week of the year. Valid only for yearly recurrences. Ignored otherwise.

Corresponds to the BYWEEKNO value in the iCalendar specification.

daysOfTheYear An array of Integers ([+/-] 1 to 366). Negative numbers infer counting from the end of the year.

For example, -1 means the last day of the year. Valid only for yearly recurrences. Ignored otherwise.

Corresponds to the BYYEARDAY value in the iCalendar specification.

setPositions An array of Integers ([+/-] 1 to 366). Used at the end of recurrence computation to filter the list

to the positions specified. Negative numbers indicate starting at the end, i.e. -1 indicates taking the last result of the set. Valid when daysOfTheWeek, daysOfTheMonth, monthsOfTheYear,

weeksOfTheYear, or

daysOfTheYear is passed. Ignored otherwise. Corresponds to the BYSETPOS value in the iCalendar

specification.

end The recurrence end, or nil.

**Shared Function** RecurrenceRule(type as EKRecurrenceRuleMB.EKRecurrenceFrequency, interval as integer, ende as EKRecurrenceEndMB = nil) As EKRecurrenceRuleMB

Creates a recurrence.

This is used to create a simple recurrence with a specific type, interval and end.

If interval is 0, an exception is raised. The end parameter can be nil.

**Function** copy() As EKRecurrenceRuleMB

Copy function for NSCopying protocol

**Function** daysOfTheMonth() As Integer()

This property is valid for rules whose EKRecurrenceFrequency is EKRecurrenceFrequencyMonthly, and that were initialized

with one or more specific days of the month (not with a day of the week and week of the month). This property can be

accessed as an array containing one or more NSNumbers corresponding to the days of the month the event recurs.

For all other EKRecurrenceRules, this property is nil. This property corresponds to BYMONTHDAY in the iCalendar

specification.

**Function** daysOfTheWeek() As Integer()

This property is valid for rules whose EKRecurrenceFrequency is EKRecurrenceFrequencyWeekly, EKRecurrenceFrequencyMonthly, or

EKRecurrenceFrequencyYearly. This property can be accessed as an array containing one or more EKRecurrenceDayOfWeek objects

corresponding to the days of the week the event recurs. For all other EKRecurrenceRules, this property is nil.

This property corresponds to BYDAY in the iCalendar specification.

**Function** daysOfTheYear() As Integer()

This property is valid for rules whose EKRecurrenceFrequency is EKRecurrenceFrequencyYearly. This property can be accessed

as an array containing one or more NSNumbers corresponding to the days of the year the event recurs. For all other

EKRecurrenceRules, this property is nil. This property corresponds to BYYEARDAY in the iCalendar specification. It should

contain values between 1 to 366 or -366 to -1.

**Function** monthsOfTheYear() **As Integer**()

This property is valid for rules whose EKRecurrenceFrequency is EKRecurrenceFrequencyYearly. This property can be accessed as an array containing one or more NSNumbers corresponding to the months of the year the event recurs. For all other EKRecurrenceRules, this property is nil. This property corresponds to BYMONTH in the iCalendar specification.

**Function** setPositions() **As Integer**()

This property is valid for rules which have a valid daysOfTheWeek, daysOfTheMonth, weeksOfTheYear, or monthsOfTheYear property. It allows you to specify a set of ordinal numbers to help choose which objects out of the set of selected events should be included. For example, setting the daysOfTheWeek to Monday-Friday and including a value of -1 in the array would indicate the last weekday in the recurrence range (month, year, etc). This value corresponds to the iCalendar BYSETPOS property.

**Function** weeksOfTheYear() **As Integer**()

This property is valid for rules whose EKRecurrenceFrequency is EKRecurrenceFrequencyYearly. This property can be accessed as an array containing one or more NSNumbers corresponding to the weeks of the year the event recurs. For all other EKRecurrenceRules, this property is nil. This property corresponds to BYWEEK in the iCalendar specification. It should contain integers from 1 to 53 or -1 to -53.

**Note "About"**

Represents how an event repeats.

This class describes the recurrence pattern for a repeating event. The recurrence rules that can be expressed are not restricted to the recurrence patterns that can be set in Calendar's UI.

It is currently not possible to directly modify a EKRecurrenceRule or any of its properties. This functionality is achieved by creating a new EKRecurrenceRule, and setting an event to use the new rule.

When a new recurrence rule is set on an EKEvent, that change is not saved until the client has passed the modified event to EKEventStore's saveEvent: method.

**Note "Copyright"**

© 2015 by Christian Schmitz, Monkeybread Software

This is part of the MBS Xojo Event Kit for iOS

<http://www.monkeybreadsoftware.de/xojo/>

**Note "Properties"**

Properties that are only valid for certain EKRecurrenceRules

```
// The properties that follow are only valid for certain recurrence rules, and are always arrays. For
recurrence rules
// that can be expressed with one of the simple initializers, the arrays will contain a single object,
corresponding
// to the day of the week, the day of the month, the "NthWeekDay" (for example, the fourth
Thursday), or the month of
// the year the event recurs. The objects will be NSNumbers, except in the "NthWeekDay" case just
mentioned, when
// the array will contain a CalNthWeekDay instead of an NSNumber.
//
// Repeating events using one of the advanced initializers may recur multiple times in the specified
time period, for
```

```
        // example, the first and sixteenth days of a month. When this is true, the arrays may contain more
than one entry.
        //
        // These properties will only be valid for certain EKRecurrenceRules, depending on how the rule's
recurrence is
        // defined. The constraints on when these properties is valid are described below. When these
constraints are not met,
        // the property's value will be nil.
```

**End Class**

## EKReminderMB class

**Class** EKReminderMB **Inherits** EKCalendarItemMB

**ComputedProperty** Completed **As Boolean**

**Sub Set()**

Whether or not the reminder is completed.

Setting it to true will set the completed date to the current date.

Setting it to false will set the completed date to nil.

**Sub Get()**

**ComputedProperty** completionDate **As Date**

**Sub Set()**

The date on which this reminder was completed.

**Sub Get()**

**ComputedProperty** dueDateComponents **As EKDateComponentsMB**

**Sub Set()**

The date by which this reminder should be completed.

The use of date components allows the due date and its time zone to be represented in a

single property.

A nil time zone represents a floating date. Setting a date component without a hour, minute and second component will set allDay to YES.

If you set this property, the calendar must be set to NSGregorianCalendar. An exception is raised otherwise.

On iOS, if you set the due date for a reminder, you must also set a start date, otherwise you

will receive

an error (EKErrorNoStartDate) when attempting to save this reminder. This is not a

requirement on OS X.

**Sub Get()**

**ComputedProperty** priority **As integer**

**Sub Set()**

The priority of the reminder.

Priorities run from 1 (highest) to 9 (lowest). A priority of 0 means no priority.

Saving a reminder with any other priority will fail.

Per RFC 5545, priorities of 1-4 are considered "high," a priority of 5 is "medium," and

priorities of 6-9 are "low."

**Sub Get()**

**ComputedProperty** startDateComponents **As EKDateComponentsMB**

**Sub Set()**

The start date of the task, as date components.

The use of date components allows the start date and its time zone to be represented in a

single property.

A nil time zone represents a floating date. Setting a date component without a hour, minute and second component will set allDay to YES.

If you set this property, the calendar must be set to NSGregorianCalendar. An exception is raised otherwise.

**Sub Get()**

**Sub** Constructor(Store **as** EKEventStoreMB)

Creates a new reminder in the given event store.

**Sub** Constructor(Handle **as Integer**, Retain **as Boolean**)

Create from Handle

**Shared Function** reminderWithEventStore(Store **as** EKEventStoreMB) **As** EKReminderMB

Creates a new reminder in the given event store.

**Note** "About"



**Note "Completed"**

```
// These two properties Completed and CompletionDate are inextricably linked.  
// Setting completed to be true, will set the completedDate to be now,  
// and setting any completedDate will change completed to be true.  
// Similarly, setting completed to be false will set  
// the completedDate to be nil, and setting the completedDate changes completed to NO.  
// Note, you may encounter the case where isCompleted is true, but completionDate is nil,  
// if the reminder was completed using a different client.
```

**Note "Copyright"**

© 2015 by Christian Schmitz, Monkeybread Software

This is part of the MBS Xojo Event Kit for iOS

<http://www.monkeybreadsoftware.de/xojo/>

**End Class**

## EKSourceMB class

**Class** EKSourceMB **Inherits** EKObjectMB

**ComputedProperty** sourceIdentifier **As** text

**Sub Get()**

the source identifier

**ComputedProperty** sourceType **As** EKSourceMB.EKSourceType

**Sub Get()**

The source type

**ComputedProperty** title **As** text

**Sub Get()**

the title for this source

**Enum** EKEntityType

Events = 0

Reminder = 1

**End Enum**

**Enum** EKSourceType

Local = 0

Exchange = 1

CalDAV = 2

MobileMe = 3

Subscribed = 4

Birthdays = 5

**End Enum**

**Sub** Constructor(Handle **as Integer**, Retain **as Boolean**)

Create from Handle

**Function** calendars() **As** EKCalendarMB()

the calendars for this source

**Function** calendarsForEntityType(Type **as** EKSourceMB.EKEntityType) **As** EKCalendarMB()

Returns the calendars that belong to this source that support a given entity type (reminders, events)

**Function** eventCalendars() **As** EKCalendarMB()

convenience function

**Function** reminderCalendars() **As** EKCalendarMB()

convenience function

**Note** "About"

**Note** "Copyright"

© 2015 by Christian Schmitz, Monkeybread Software

This is part of the MBS Xojo Event Kit for iOS

<http://www.monkeybreadsoftware.de/xojo/>

**End Class**

## EKStructuredLocationMB class

**Class** EKStructuredLocationMB **Inherits** EKBaseObjectMB

**ComputedProperty** geoLocation **As Integer**

**Sub Set()**

the location as CLLocation reference

**Sub Get()**

**ComputedProperty** radius **As double**

**Sub Set()**

0 = use default, unit is meters

**Sub Get()**

**ComputedProperty** title **As text**

**Sub Set()**

set the title

**Sub Get()**

the title

**Sub Constructor**(Handle **as Integer**, Retain **as Boolean**)

Create from Handle

**Sub Constructor**(title **as text**)

Creates a new location

**Function** copy() **As** EKStructuredLocationMB

Copy function for NSCopying protocol

**Shared Function** reminderWithEventStore(title **as text**) **As** EKStructuredLocationMB

Creates a new location

**Note** "About"

**Note** "Copyright"

© 2015 by Christian Schmitz, Monkeybread Software

This is part of the MBS Xojo Event Kit for iOS

<http://www.monkeybreadsoftware.de/xojo/>

**End Class**

## EventKitMB class

**Module** EventKitMB

**ComputedProperty** Available **As Boolean**

**Sub Get()**

*Check and see if EventKit is Available*

**Const** EKErrorAlarmGreaterThanRecurrence = 8

**Const** EKErrorAlarmProximityNotSupported = 21

**Const** EKErrorCalendarDoesNotAllowEvents = 22

**Const** EKErrorCalendarDoesNotAllowReminders = 23

**Const** EKErrorCalendarHasNoSource = 14

**Const** EKErrorCalendarIsImmutable = 16

**Const** EKErrorCalendarReadOnly = 6

**Const** EKErrorCalendarSourceCannotBeModified = 15

**Const** EKErrorDatesInverted = 4

**Const** EKErrorDomain = EKErrorDomain

**Const** EKErrorDurationGreaterThanRecurrence = 7

**Const** EKErrorEventNotMutable = 0

**Const** EKErrorInternalFailure = 5

**Const** EKErrorInvalidEntityType = 26

**Const** EKErrorInvalidSpan = 13

**Const** EKErrorInvitesCannotBeMoved = 12

**Const** EKErrorNoCalendar = 1

**Const** EKErrorNoEndDate = 3

**Const** EKErrorNoStartDate = 2

**Const** EKErrorObjectBelongsToDifferentStore = 11

**Const** EKErrorPriorityIsInvalid = 25

**Const** EKErrorRecurringReminderRequiresDueDate = 18

**Const** EKErrorReminderLocationsNotSupported = 20

**Const** EKErrorSourceDoesNotAllowCalendarAddDelete = 17

**Const** EKErrorSourceDoesNotAllowReminders = 24

**Const** EKErrorStartDateCollidesWithOtherOccurrence = 10

**Const** EKErrorStartDateTooFarInFuture = 9

**Const** EKErrorStructuredLocationsNotSupported = 19

**Const** EKFriday = 6

**Const** EKMonday = 2

**Const** EKSaturday = 7

**Const** EKSunday = 1

**Const** EKThursday = 5

**Const** EKTuesday = 3

**Const** EKWednesday = 4

**Protected Function** CGColorToColor(cgColorRef **as Integer**) **As color**

**Protected Function** ClassName(ref **as integer**) **As text**

*class name of object*

**Protected Function** ColorToCGColor(co **as color**) **As Integer**

**Protected Function** DateToNSDate(d **as date**) **As Integer**

**Protected Function** Description(ref **as integer**) **As text**

*class description of object*

**Protected Sub** LoadConstants()

*optionally called to load constants*

**Protected Function** NSArrayCount(NSArrayRef **as integer**) **As Integer**

**Protected Function** NSArrayOfAlarms(r **as integer**) **As** EKAlarmMB()

**Protected Function** NSArrayOfCalendarItems(r **as integer**) **As** EKCalendarItemMB()

**Protected Function** NSArrayOfCalendars(r **as integer**) **As** EKCalendarMB()

**Protected Function** NSArrayOfEvents(r **as integer**) **As** EKEventMB()

**Protected Function** NSArrayOfIntegers(r **as integer**) **As Integer**()  
**Protected Function** NSArrayOfParticipants(r **as integer**) **As** EKParticipantMB()  
**Protected Function** NSArrayOfRecurrenceRules(r **as integer**) **As** EKRecurrenceRuleMB()  
**Protected Function** NSArrayOfReminders(r **as integer**) **As** EKReminderMB()  
**Protected Function** NSArrayOfSources(r **as integer**) **As** EKSourceMB()  
**Protected Function** NSClassFromText(Name **as Text**) **As Integer**  
**Protected Function** NSDateToDate(Ref **as Integer**) **As** date  
**Protected Sub** NSLog(s **as** text, h **as integer**)  
    log some object  
**Protected Function** NSSetToNSArray(NSSetRef **as integer**) **As integer**  
**Protected Function** NSStringToText(ref **as integer**) **As** text  
**Protected Function** NSTimeZoneToTimeZone(t **as integer**) **As** xojo.Core.TimeZone  
**Protected Function** NSURLFromText(URL **as String**) **As Integer**  
**Protected Function** NSURLFromText(URL **as Text**) **As Integer**  
**Protected Function** NSURLToText(NSURLHandle **as integer**) **As** text  
**Protected Function** NewEKCalendarNSArray(values() **as** EKCalendarMB) **As Integer**  
**Protected Function** NewEKRecurrenceDayOfWeekNSArray(values() **as** EKRecurrenceDayOfWeekMB) **As**

## Integer

**Protected Function** NewIntegerNSArray(values() **as Integer**) **As Integer**  
**Protected Function** TextToNSString(t **as** text) **As integer**  
**Protected Function** TimeZoneToNSTimeZone(t **as** xojo.Core.TimeZone) **As integer**  
**Protected Function** defaultCenter() **As Integer**  
**Protected Function** mainQueue() **As Integer**  
**Protected Function** newEKCalendarItemMB(Ref **as Integer**, Retain **as Boolean**) **As** EKCalendarItemMB

**Note** "Copyright"

© 2015 by Christian Schmitz, Monkeybread Software

This is part of the MBS Xojo Event Kit for iOS

<http://www.monkeybreadsoftware.de/xojo/>

**Property Protected** EKEventStoreChangedNotification **As** text

**End Module**

## Version History

Tip: If you want to update your existing code with new release, you'd best compare projects with Arbed (<http://www.tempel.org/Arbed>) and copy modifications to your project.

1.2, 23rd April 2018

- Added privacy entries for info.plist via build script step.
- Updated for Xojo 2018r1
- Fixed bugs with color for Mac and we now use NSColor there.
- Fixed bug with EKCalendarItemMB.URL setter

1.1, 10th November 2016

- Updated for Xojo 2016r3.
- Added retain/release for arrays in methods to not get them freed too early.
- Worked around Feedback case #44874 by using CFDate functions now.

1.0, first release, 29th September 2015

## Known issues

- Change event doesn't fire currently.
- If you don't use NSCalendarsUsageDescription in info.plist of your app to declare calendar usage, you will see a crash with the function `__CRASHING_DUE_TO_PRIVACY_VIOLATION__` in stack trace.

## Installation

To get your projects working with this Event Kit, you need to follow a few steps.

Drop the folder „externals“ into your project and access all the modules and classes. Or copy from existing example projects what you need.

## Requirements

You need Xojo 2015r1 or newer.

We did not test with older versions, but you can if you need.

If you need similar functions for Mac OS X cross platform, please check our MBS Plugins, especially the Cocoa and Leopard plugin parts.



# License

## Summary:

- You may use Event Kit only with one licensed Xojo installation.
- You agree not to share the Event Kit or use someone else's Event Kit copy.

Christian Schmitz Software GmbH, of Nickenich Germany is the owner, developer and sole copyright holder of this product, which is licensed -not sold- to you on a non-exclusive basis.

You agree not to share your MBS Xojo Event Kit with anyone.

You may transfer your license to another person only after receiving written authorization from Christian Schmitz Software GmbH and only if the recipient agrees to be bound by the terms of this agreement.

Christian Schmitz Software GmbH reserves the right to cancel the license key(s) of any user who Christian Schmitz Software GmbH determines is in violation of this agreement.

THE WARRANTIES IN THIS AGREEMENT REPLACE ALL OTHER WARRANTIES, EXPRESS OR IMPLIED, INCLUDING ANY WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE. THE SOFTWARE IS PROVIDED "AS IS" AND Christian Schmitz Software GmbH DISCLAIMS AND EXCLUDES ALL OTHER WARRANTIES. IN NO EVENT WILL Christian Schmitz Software GmbH BE LIABLE FOR ANY SPECIAL, INCIDENTAL OR CONSEQUENTIAL DAMAGES, INCLUDING LOST PROFITS, EVEN IF WE HAVE KNOWLEDGE OF THE POTENTIAL LOSS OR DAMAGE.

If you are located in Germany this agreement is subject to the laws of Germany. If you are located outside Germany local law may apply. Some states do not allow the exclusion of warranties, so the above exclusion may not apply to you.

Christian Schmitz Software GmbH does not charge royalties or deployment fees for Xojo applications.

Access to updates is included for one year. After that time you can order an update or keep using the old version you have.

# Contact

Monkeybread Software  
Christian Schmitz Software GmbH  
Eckertshohl 22  
56645 Nickenich  
Germany

Email: [support@monkeybreadsoftware.de](mailto:support@monkeybreadsoftware.de)

Phone: +49 26 32 95 89 55 (Office) or +49 17 58 36 37 10 (Mobile)