CalCalendarStore Class Reference

Cocoa > Apple Applications

2009-04-08

Contents

CalCalendarStore Class Reference

Inherits from	NSObject
Conforms to	NSObject (NSObject)
Framework	/System/Library/Frameworks/CalendarStore.framework
Availability	Available in Mac OS X v10.5 and later.
Companion guide	Calendar Store Programming Guide
Declared in	CalCalendarStore.h
Related sample code	Reminders SimpleCalendar

Overview

There is only one `CalCalendarStore` object and it represents a connection to the iCal database. You retrieve all other types of calendar objects—including calendars, events, and tasks—using this shared `CalCalendarStore` object.

Use the `defaultCalendarStore` (page 7) class method to get the shared `CalCalendarStore` object. Use the `calendars` (page 11) method to fetch all the `CalCalendar` objects, and the `calendarWithUID:` (page 11) method to fetch individual calendars. Use the `eventsWithPredicate:` (page 12) and `tasksWithPredicate:` (page 17) methods to fetch events and tasks. These methods take an `NSPredicate` object as the argument. Use the `eventPredicate...` and `taskPredicate...` methods to create `NSPredicate` objects for common queries that you can pass to these methods.

Use specific `CalCalendar`, `CalEvent`, and `CalTask` properties and methods to make changes to these types of objects. If you make local changes to calendars, events, and tasks, invoke the corresponding `save...` method; otherwise, your changes do not persist. You might also need to track changes.

Observing Notifications

Calendars, events, and tasks may be added, changed, or deleted after you fetch them. Changes can occur locally or externally where local changes are made by your application and external changes are made by other applications. When these objects change, you might want to take some action, especially if you retain fetched objects—for example, update the display. You can track changes by observing notifications.

The following notifications are posted when your application changes these types of objects: `CalCalendarsChangedNotification` (page 21), `CalEventsChangedNotification` (page 21), and `CalTasksChangedNotification` (page 21).

The following notifications are posted when another application changes these types of objects: `CalCalendarsChangedExternallyNotification` (page 22), `CalEventsChangedExternallyNotification` (page 22), and `CalTasksChangedExternallyNotification` (page 22) notifications.

Tasks

Creating and Initializing

`+ defaultCalendarStore` (page 7)
Returns the shared `CalCalendarStore` object.

Accessing Calendars

`– calendars` (page 11)
Returns an array of `CalCalendar` objects representing the user's calendars in the order they appear in iCal.

`– calendarWithUID:` (page 11)
Returns a `CalCalendar` object that corresponds to the given UID.

`– saveCalendar:error:` (page 15)
Saves local changes to the specified calendar object to the iCal database.

`– removeCalendar:error:` (page 13)
Removes the specified calendar from the iCal database.

Accessing Events

`– eventsWithPredicate:` (page 12)
Returns an array of `CalEvent` objects that match the specified predicate.

`– eventWithUID:occurrence:` (page 13)
Returns an event that matches the specified unique identifier.

`– saveEvent:span:error:` (page 16)
Saves the specified event to the calendar store.

`– removeEvent:span:error:` (page 14)
Removes the specified event from the calendar store.

Accessing Tasks

`– tasksWithPredicate:` (page 17)
Returns an array of `CalTask` objects that match the specified predicate.

`- taskWithUID:` (page 18)
: Returns a task that matches the specified unique identifier.

`- saveTask:error:` (page 16)
: Saves the specified task to the calendar store.

`- removeTask:error:` (page 15)
: Removes the specified task from the calendar store.

Creating Predicates

`+ eventPredicateWithStartDate:endDate:calendars:` (page 8)
: Returns an `NSPredicate` object that specifies events within a date range that belong to the specified calendars.

`+ eventPredicateWithStartDate:endDate:UID:calendars:` (page 8)
: Returns an `NSPredicate` object that specifies events with a UID and within a date range that belong to the specified calendars.

`+ taskPredicateWithCalendars:` (page 9)
: Returns an `NSPredicate` object that specifies tasks that belong to the specified calendars.

`+ taskPredicateWithUncompletedTasks:` (page 10)
: Returns an `NSPredicate` object that specifies tasks that are incomplete and belong to the specified calendars.

`+ taskPredicateWithUncompletedTasksDueBefore:calendars:` (page 10)
: Returns an `NSPredicate` object that specifies incomplete tasks due before the specified date and that belong to the specified calendars.

`+ taskPredicateWithTasksCompletedSince:calendars:` (page 9)
: Returns an `NSPredicate` object that specifies completed tasks since the specified date and that belong to the specified calendars.

Class Methods

defaultCalendarStore

Returns the shared `CalCalendarStore` object.

```
+ (CalCalendarStore *)defaultCalendarStore
```

Return Value
The shared `CalCalendarStore` object.

The first time an application invokes this method, Calendar Store might migrate the calendar data from a previous Mac OS X file format to the current one. This method returns `nil` if the migration fails.

Discussion
Invoke this method to get the shared instance. Do not create a `CalCalendarStore` object directly.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
Reminders

SimpleCalendar

Declared In
`CalCalendarStore.h`

eventPredicateWithStartDate:endDate:calendars:

Returns an `NSPredicate` object that specifies events within a date range that belong to the specified calendars.

```
+ (NSPredicate *)eventPredicateWithStartDate:(NSDate *)startDate endDate:(NSDate
    *)endDate calendars:(NSArray *)calendars
```

Parameters

startDate
    The start date of the date range.

endDate
    The end date of the date range.

calendars
    An array of `CalCalendar` objects that the events must belong to.

Return Value
An `NSPredicate` object that specifies events within a date range that belong to the specified calendars.

Discussion
If an event is greater than or equal to *startDate* and less than or equal to *endDate*, and belongs to a calendar in *calendars*, then it is included in the predicate's result.

Availability
Available in Mac OS X v10.5 and later.

See Also
`+ eventPredicateWithStartDate:endDate:UID:calendars:` (page 8)

Related Sample Code
Reminders

Declared In
`CalCalendarStore.h`

eventPredicateWithStartDate:endDate:UID:calendars:

Returns an `NSPredicate` object that specifies events with a UID and within a date range that belong to the specified calendars.

```
+ (NSPredicate *)eventPredicateWithStartDate:(NSDate *)startDate endDate:(NSDate
    *)endDate UID:(NSString *)UID calendars:(NSArray *)calendars
```

Parameters

`startDate`
: The start date of the date range.

`endDate`
: The end date of the date range.

`UID`
: The UID for the returned events.

`calendars`
: An array of `CalCalendar` objects that the events must belong to.

Return Value
An `NSPredicate` object that specifies events with a UID and within a date range that belong to the specified calendars.

Discussion
Recurring events have the same UID. Therefore, use this method, instead of the `eventPredicateWithStartDate:endDate:calendars:` method, if you want to fetch all events belonging to the same master recurring event.

Availability
Available in Mac OS X v10.5 and later.

See Also
`+ eventPredicateWithStartDate:endDate:calendars:` (page 8)

Declared In
`CalCalendarStore.h`

taskPredicateWithCalendars:

Returns an `NSPredicate` object that specifies tasks that belong to the specified calendars.

```
+ (NSPredicate *)taskPredicateWithCalendars:(NSArray *)calendars
```

Parameters

`calendars`
: An array of `CalCalendar` objects that the tasks must belong to.

Return Value
An `NSPredicate` object that specifies tasks that belong to the specified calendars.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`CalCalendarStore.h`

taskPredicateWithTasksCompletedSince:calendars:

Returns an `NSPredicate` object that specifies completed tasks since the specified date and that belong to the specified calendars.

```
+ (NSPredicate *)taskPredicateWithTasksCompletedSince:(NSDate *)completedSince
    calendars:(NSArray *)calendars
```

Parameters

completedSince

Specifies the completion date.

calendars

An array of `CalCalendar` objects that the tasks must belong to.

Discussion

If a task is completed after *completedSince* and belongs to a calendar in *calendars*, then it is included in the returned predicate's result.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`CalCalendarStore.h`

taskPredicateWithUncompletedTasks:

Returns an `NSPredicate` object that specifies tasks that are incomplete and belong to the specified calendars.

```
+ (NSPredicate *)taskPredicateWithUncompletedTasks:(NSArray *)calendars
```

Parameters

calendars

An array of `CalCalendar` objects that the tasks must belong to.

Return Value

An `NSPredicate` object that specifies tasks that are incomplete and belong to the specified calendars.

Availability

Available in Mac OS X v10.5 and later.

Declared In

`CalCalendarStore.h`

taskPredicateWithUncompletedTasksDueBefore:calendars:

Returns an `NSPredicate` object that specifies incomplete tasks due before the specified date and that belong to the specified calendars.

```
+ (NSPredicate *)taskPredicateWithUncompletedTasksDueBefore:(NSDate *)dueDate
    calendars:(NSArray *)calendars
```

Parameters

dueDate

Specifies the due date of the tasks.

calendars

An array of `CalCalendar` objects that the tasks must belong to.

Discussion
If an incomplete task's due date is before *`dueDate`* and belongs to a calendar in *`calendars`*, then it is included in the returned predicate's result.

Availability
Available in Mac OS X v10.5 and later.

Related Sample Code
Reminders

Declared In
`CalCalendarStore.h`

Instance Methods

calendars

Returns an array of `CalCalendar` objects representing the user's calendars in the order they appear in iCal.

```
- (NSArray *)calendars
```

Return Value
An array of `CalCalendar` objects. Returns an empty array if the user hasn't launched iCal and created any calendars.

Discussion
This method returns an empty array if the user has calendar data from a previous version of Mac OS X but has not launched iCal in Mac OS X v10.5 yet. iCal needs to be launched at least once on Mac OS X v10.5 to migrate the user's calendar data from a previous version. If no calendar data from a previous version exists, then this method creates and returns the default calendars.

Availability
Available in Mac OS X v10.5 and later.

See Also
`– calendarWithUID:` (page 11)

Related Sample Code
Reminders
SimpleCalendar

Declared In
`CalCalendarStore.h`

calendarWithUID:

Returns a `CalCalendar` object that corresponds to the given UID.

```
- (CalCalendar *)calendarWithUID:(NSString *)UID
```

Parameters

`UID`

A string that uniquely identifies a calendar.

Return Value

The calendar that corresponds to the UID. Returns `nil` if no calendar with the specified UID exists.

Availability

Available in Mac OS X v10.5 and later.

See Also

`– calendars` (page 11)

Related Sample Code

Reminders

Declared In

`CalCalendarStore.h`

eventsWithPredicate:

Returns an array of `CalEvent` objects that match the specified predicate.

```
- (NSArray *)eventsWithPredicate:(NSPredicate *)predicate
```

Parameters

`predicate`

Describes the set of records that should be returned.

Return Value

An array of `CalEvent` objects that match the specified predicate. Returns `nil` if the predicate is invalid.

Discussion

The predicate passed to this method must be valid. You create valid predicates using the predicate methods described in this class. The predicate is valid if it contains two predicates: one using a `BETWEEN` operator to specify the date range of the events, and one using the `IN` operator to specify the parent calendar objects. The calendars need to be represented by an array of `CalCalendar` objects. The dates and calendar arrays added to the predicate should use variable substitution.

For example, the following code fragment creates a date and calendar predicate that is combined to create a predicate that is passed as the argument to this method. The returned array contains all events that occur today.

```
datePredicate = [NSPredicate predicateWithFormat:@"date BETWEEN { %@, %@ }"
   argumentArray:[NSArray arrayWithObjects:[NSDate
dateWithNaturalLanguageString:@"today at midnight"],
   [NSDate dateWithNaturalLanguageString:@"tomorrow at midnight"], nil]];
calendarPredicate = [NSPredicate predicateWithFormat:@"calendar IN %@))"
   argumentArray:[NSArray arrayWithObject:[CalCalendarStore calendars]]];
finalPredicate = [NSPredicate andPredicateWithSubpredicates: [NSArray
arrayWithObjects:datePredicate, calendarPredicate, nil]];
NSArray *events = [[CalCalendarStore defaultCalendarStore]
eventsWithPredicate:finalPredicate];
```

Date boundaries are exclusive, so an event that ends today at midnight is not included in the returned events array in the above sample code.

For performance reasons, this method only returns events that fall within a specific four year timespan. If the date range between the predicate's start date and end date is greater than four years, then the timespan containing events is always the first four years of date range

If you observe the `CalEventsChangedNotification` notification, you can retain the predicate and send it `evaluateWithObject:`, passing the changed event to determine whether the notification affects the event objects you previously fetched.

Availability
Available in Mac OS X v10.5 and later.

See Also
`– eventWithUID:occurrence:` (page 13)

Related Sample Code
Reminders
SimpleCalendar

Declared In
`CalCalendarStore.h`

eventWithUID:occurrence:

Returns an event that matches the specified unique identifier.

```
- (CalEvent *)eventWithUID:(NSString *)uid occurrence:(NSDate *)date
```

Parameters

`uid`
The unique identifier of an event.

`date`
The date of a recurring event. Pass `nil` if the event is not recurring.

Return Value
A `CalEvent` object that matches the specified unique identifier and date. Returns `nil` if the event is not found, or the event is recurring and *`date`* is not specified.

Availability
Available in Mac OS X v10.5 and later.

See Also
`– eventsWithPredicate:` (page 12)

Declared In
`CalCalendarStore.h`

removeCalendar:error:

Removes the specified calendar from the iCal database.

```
- (BOOL)removeCalendar:(CalCalendar *)calendar error:(NSError **)error
```

Parameters

calendar
: The calendar object to remove. Must be a local calendar not a subscribed, birthday, or CalDAV calendar.

error
: If this method returns `NO`, an `NSError` object describing the error. See *Calendar Store Constants Reference* for description of error codes.

Return Value
If successful, returns `YES`; otherwise, returns `NO` and sets the *error* argument to an `NSError` object describing the error.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- saveCalendar:error:` (page 15)

Declared In
`CalCalendarStore.h`

removeEvent:span:error:

Removes the specified event from the calendar store.

```
- (BOOL)removeEvent:(CalEvent *)event span:(CalSpan)span error:(NSError **)error
```

Parameters

event
: The event to remove.

span
: Specifies the span of a recurring event—whether the change should be applied to future occurrences, all occurrences, or just this instance. Applying changes to future occurrences or all occurrences may cause the UID or occurrence date of the event to change.

error
: If this method returns `NO`, an `NSError` object describing the error. See *Calendar Store Constants Reference* for description of error codes.

Return Value
If successful, returns `YES`; otherwise, returns `NO` and sets the *error* argument to an `NSError` object describing the error.

Availability
Available in Mac OS X v10.5 and later.

See Also
`- saveEvent:span:error:` (page 16)

Declared In
`CalCalendarStore.h`

removeTask:error:

Removes the specified task from the calendar store.

```
- (BOOL)removeTask:(CalTask *)task error:(NSError **)error
```

Parameters

task

The task to remove.

error

If this method returns `NO`, an `NSError` object describing the error. See *InstantMessage Constants Reference* for description of error codes.

Return Value

If successful, returns `YES`; otherwise, returns `NO` and sets the *error* argument to an `NSError` object describing the error.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- saveTask:error:` (page 16)

Declared In

`CalCalendarStore.h`

saveCalendar:error:

Saves local changes to the specified calendar object to the iCal database.

```
- (BOOL)saveCalendar:(CalCalendar *)calendar error:(NSError **)error
```

Parameters

calendar

The calendar object to save. Must be a local calendar not a subscribed, birthday, or CalDAV calendar.

error

If this method returns `NO`, an `NSError` object describing the error. See *InstantMessage Constants Reference* for description of error codes.

Return Value

If successful, returns `YES`; otherwise, returns `NO` and sets the *error* argument to an `NSError` object describing the error.

Discussion

Use this method to both create a new calendar and save changes to an existing calendar. Changes you make to calendars do not persist unless you invoke this method.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- removeCalendar:error:` (page 13)

Related Sample Code
Reminders

Declared In
`CalCalendarStore.h`

saveEvent:span:error:

Saves the specified event to the calendar store.

```
- (BOOL)saveEvent:(CalEvent *)event span:(CalSpan)span error:(NSError **)error
```

Parameters

`event`
The event to save.

`span`
Specifies the span of a recurring event—if the change should be applied to future occurrences, all occurrences, or just this instance. Applying changes to future occurrences or all occurrences may cause the UID or occurrence date of the event to change.

`error`
If this method returns `NO`, an `NSError` object describing the error. See *InstantMessage Constants Reference* for description of error codes.

Return Value
If successful, returns `YES`; otherwise, returns `NO` and sets the *`error`* argument to an `NSError` object describing the error.

Discussion
Use this method to save new event objects and modifications to existing event objects. Changes to event objects are not persistent until this method is invoked. The `calendar` property needs to be set before attempting to save an event.

Availability
Available in Mac OS X v10.5 and later.

See Also
`– removeEvent:span:error:` (page 14)

Related Sample Code
Reminders

Declared In
`CalCalendarStore.h`

saveTask:error:

Saves the specified task to the calendar store.

```
- (BOOL)saveTask:(CalTask *)task error:(NSError **)error
```

Parameters

`task`

The task to save.

`error`

If this method returns `NO`, an `NSError` object describing the error. See *Calendar Store Constants Reference* for description of error codes.

Return Value

If successful, returns `YES`; otherwise, returns `NO` and sets the *`error`* argument to an `NSError` object describing the error.

Discussion

Use this method to save new task objects and modifications to existing task objects. Changes to task objects are not persistent until this method is invoked. The `calendar` property needs to be set before attempting to save a task.

Availability

Available in Mac OS X v10.5 and later.

See Also

`- removeTask:error:` (page 15)

Related Sample Code

Reminders

Declared In

`CalCalendarStore.h`

tasksWithPredicate:

Returns an array of `CalTask` objects that match the specified predicate.

```
- (NSArray *)tasksWithPredicate:(NSPredicate *)predicate
```

Parameters

`predicate`

Describes the set of records that should be returned.

Return Value

An array of `CalTask` objects. Returns `nil` if the predicate is invalid.

Discussion

The predicate passed to this method must be valid. You create valid predicates using the predicate methods described in this class. The predicate must specify an array of calendars to query and can filter the tasks by due date and completion date. All subpredicates must be joined with an `AND`, and all dates and calendars must be arrays and added via variable substitution. Calendars are specified using the `IN` operator.

Date ranges may be specified using the `BETWEEN`, greater than, less than, greater than or equal, and less than or equal operators. Use a subpredicate that tests equality to `null` to return tasks that do not have a completion or due date.

For example, the following code fragment creates a predicate that returns all incomplete tasks due before March 1, 2006.

```
dueDatePredicate = [NSPredicate predicateWithFormat:@"dueDate < %@"
   argumentArray:[NSArray arrayWithObject:[NSDate
dateWithNaturalLanguageString:@"12am March 1, 2006"]]];
completionDatePredicate = [NSPredicate predicateWithFormat:@"completedDate =
<null>"];
calendarPredicate = [NSPredicate predicateWithFormat:@"calendar IN %@))"
   argumentArray:[NSArray arrayWithObject:[CalCalendarStore calendars]]];
finalPredicate = [NSPredicate andPredicateWithSubpredicates:
   [NSArray arrayWithObjects:dueDatePredicate, completionDatePredicate,
calendarPredicate, nil]];
NSArray *tasks = [[CalCalendarStore defaultCalendarStore]
tasksWithPredicate:finalPredicate];
```

If you observe the `CalTasksChangedNotification` notification, you can retain the predicate and send it `evaluateWithObject:`, passing the changed task to determine whether the notification affects the task objects you previously fetched.

Availability
Available in Mac OS X v10.5 and later.

See Also
`– taskWithUID:` (page 18)

Related Sample Code
Reminders

Declared In
`CalCalendarStore.h`

taskWithUID:

Returns a task that matches the specified unique identifier.

```
- (CalTask *)taskWithUID:(NSString *)uid
```

Parameters

uid
A unique identifier for a task.

Return Value
A task that matches the specified unique identifier.

Availability
Available in Mac OS X v10.5 and later.

See Also
`– tasksWithPredicate:` (page 17)

Declared In
`CalCalendarStore.h`

Constants

CalSpan

The range of events to apply changes to for a recurring event.

```
typedef enum {
    CalSpanThisEvent,
    CalSpanFutureEvents,
    CalSpanAllEvents
} CalSpan;
```

Constants

`CalSpanThisEvent`

Apply changes to just this instance of a recurring event.

Available in Mac OS X v10.5 and later.

Declared in `CalCalendarStore.h`.

`CalSpanFutureEvents`

Apply changes to all future events of a recurring event.

Available in Mac OS X v10.5 and later.

Declared in `CalCalendarStore.h`.

`CalSpanAllEvents`

Apply changes to all events of a recurring event.

Available in Mac OS X v10.5 and later.

Declared in `CalCalendarStore.h`.

Discussion

Use these constants when invoking the `saveEvent:span:error:` (page 16) and `removeEvent:span:error:` (page 14) methods.

Declared In

`CalendarStore/CalCalendarStore.h`

Changed Externally Notification Keys

Specifies the records that changed.

```
extern NSString * const CalInsertedRecordsKey;
extern NSString * const CalUpdatedRecordsKey;
extern NSString * const CalDeletedRecordsKey;
```

Constants

`CalInsertedRecordsKey`

An array of record UIDs that were inserted.

Available in Mac OS X v10.5 and later.

Declared in `CalCalendarStore.h`.

`CalUpdatedRecordsKey`
: An array of record UIDs whose properties were changed.

  Available in Mac OS X v10.5 and later.

  Declared in `CalCalendarStore.h`.

`CalDeletedRecordsKey`
: An array of record UIDs that were deleted.

  Available in Mac OS X v10.5 and later.

  Declared in `CalCalendarStore.h`.

Discussion
These keys are used in the user information dictionary of `CalCalendarStore` change notifications.

Declared In
`CalendarStore/CalNotificationCenter.h`

User Information Dictionary Keys

Keys in the user information dictionary of a notification sent by an instance of this class.

```
extern NSString * const CalSenderProcessIDKey;
extern NSString * const CalUserUIDKey;
```

Constants

`CalSenderProcessIDKey`
: The key for a process ID that identifies the application that changed either a calendar, event, or task object.

  This key is used in the user information dictionary for these notifications:

  `CalCalendarsChangedNotification`

  `CalEventsChangedNotification`

  `CalTasksChangedNotification`

  `CalCalendarsChangedExternallyNotification`

  `CalEventsChangedExternallyNotification`

  `CalTasksChangedExternallyNotification`

  Available in Mac OS X v10.5 and later.

  Declared in `CalCalendarStore.h`.

`CalUserUIDKey`
: The key for a user UID that identifies the user that changed either a calendar, event, or task object.

  This key is used in the user information dictionary for these notifications:

  `CalCalendarsChangedNotification`

  `CalEventsChangedNotification`

  `CalTasksChangedNotification`

  `CalCalendarsChangedExternallyNotification`

  `CalEventsChangedExternallyNotification`

  `CalTasksChangedExternallyNotification`

  Available in Mac OS X v10.5 and later.

  Declared in `CalCalendarStore.h`.

Declared In
`CalCalendarStore.h`

Notifications

CalCalendarsChangedNotification

Posted by the shared `NSDistributedNotificationCenter` object when this application process changes calendar objects. The notification object is the shared `CalCalendarStore` object. The user information dictionary contains the `CalSenderProcessIDKey` and `CalUserUIDKey` keys which identify the process and user that made the change to the calendars. In addition, the user information dictionary may contain one or more of the following keys which indicate the type of change: `CalInsertedRecordsKey`, `CalUpdatedRecordsKey`, and `CalDeletedRecordsKey`. The value for these keys is an array containing the UIDs for the events that were inserted, updated, or deleted. Use the `calendarWithUID:` (page 11) method to get the changed calendar object.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`CalCalendarStore.h`

CalEventsChangedNotification

Posted by the shared `NSDistributedNotificationCenter` object when this application process changes event objects. The notification object is the shared `CalCalendarStore` object. The user information dictionary contains the `CalSenderProcessIDKey` and `CalUserUIDKey` keys which identify the process and user that made the change to the calendars. In addition, the user information dictionary may contain one or more of the following keys which indicate the type of change: `CalInsertedRecordsKey`, `CalUpdatedRecordsKey`, and `CalDeletedRecordsKey`. The value for these keys is an array containing the UIDs for the events that were inserted, updated, or deleted. Use the `eventsWithPredicate:` (page 12) or the `eventWithUID:occurrence:` (page 13) method to get the changed event object. If the event is recurring, read Fetching Objects in *Calendar Store Programming Guide* for how to fetch recurring events within a date range.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`CalCalendarStore.h`

CalTasksChangedNotification

Posted by the shared `NSDistributedNotificationCenter` object when this application process changes task objects. The notification object is the shared `CalCalendarStore` object. The user information dictionary contains the `CalSenderProcessIDKey` and `CalUserUIDKey` keys which identify the process and user that made the change to the calendars. In addition, the user information dictionary may contain one or more of the following keys which indicate the type of change: `CalInsertedRecordsKey`, `CalUpdatedRecordsKey`, and `CalDeletedRecordsKey`. The value for these keys is an array containing the UIDS for the tasks that were inserted, updated, or deleted. Use the `tasksWithPredicate:` (page 17) or the `taskWithUID:` (page 18) method to get the changed task object.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`CalCalendarStore.h`

CalCalendarsChangedExternallyNotification

Posted by the shared `NSDistributedNotificationCenter` object when another application process changes calendar objects. The notification object is the shared `CalCalendarStore` object. The user information dictionary contains the `CalSenderProcessIDKey` and `CalUserUIDKey` keys which identify the process and user that made the change to the calendars. In addition, the user information dictionary may contain one or more of the following keys which indicate the type of change: `CalInsertedRecordsKey`, `CalUpdatedRecordsKey`, and `CalDeletedRecordsKey`. The value for these keys is an array containing the UIDs for the calendars that were inserted, updated, or deleted. Use the `calendarWithUID:` (page 11) method to get the changed calendar object.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`CalCalendarStore.h`

CalEventsChangedExternallyNotification

Posted by the shared `NSDistributedNotificationCenter` object when another application process changes event objects. This notification is identical to the `CalEventsChangedNotification` (page 21) notification except that it is triggered by an external process.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`CalCalendarStore.h`

CalTasksChangedExternallyNotification

Posted by the shared `NSDistributedNotificationCenter` object when another application process changes task objects. This notification is identical to the `CalTasksChangedNotification` (page 21) notification except that it is triggered by an external process.

Availability
Available in Mac OS X v10.5 and later.

Declared In
`CalCalendarStore.h`

Document Revision History

This table describes the changes to *CalCalendarStore Class Reference*.

Date	**Notes**
2009-04-08	Minor edits throughout.
2008-10-15	Minor wording change.
2007-07-08	New document that describes the class used to retrieve calendar objects, including calendars, events and tasks.

Index