What’s New in HomeKit

Session 705

Matt Lucas, HomeKit Engineering
Praveen Chegondi, HomeKit Engineering
Overview
Overview
Framework updates
Overview

Framework updates

Accessory updates
Hey Siri, good morning

Your house is waking up! It might need some coffee, though.
Hey Siri, good morning

Your house is waking up! It might need some coffee, though.
Remote Access
Choose when you want this automation to occur.

- People Arrive Home
- People Leave Home
- A Time of Day Occurs
  Ex. “At 8:00 AM” or “At Sunset”
- An Accessory is Controlled
  Ex. “Light Turns On” or “Door Opened”
- A Sensor Detects Something
  Ex. “Motion Detected” or “Smoke Detected”
Framework Updates
Event Triggers
Event Triggers

New Events
Event Triggers

New Events

New Conditions
Event Triggers

New Events

New Conditions

End Events
Event Triggers

New Events

New Conditions

End Events

Recurrence
Event Triggers

New Events
New Conditions
End Events
Recurrence
Mutable Events
Overview
Event Triggers

Events
Event Triggers

Events activate a trigger
Event Triggers

Events

Events activate a trigger

• State of an accessory
Event Triggers

Events

Events activate a trigger

- State of an accessory
- Geofence
Event Triggers

Scenes
Event Triggers

Scenes:
- Arrive Home
  - Light on
  - Unlock doors
  - Temp up
  - Raise shades
  - Switch on
  - Open garage
  - Open gates
Event Triggers

Scenes

- Arrive Home
  - Light on
  - Unlock doors
  - Temp up
  - Raise shades
  - Switch on
  - Open garage
  - Open gates
  - Light on
  - Unlock doors
  - Temp up
  - Raise shades
  - Switch on
  - Open garage
  - Open gates
  - Light on
  - Unlock doors
  - Temp up
  - Raise shades
  - Switch on
  - Open garage
  - Open gates
  - Light on
  - Unlock doors
  - Temp up
  - Raise shades
  - Switch on
  - Open garage
  - Open gates
Event Triggers

Scenes

Arrive Home

- Light on
- Unlock doors
- Temp up
- Raise shades
- Switch on
- Open garage
- Open gates
- Light on
- Unlock doors
- Temp up
- Raise shades
- Switch on
- Open garage
- Open gates
Event Triggers
Conditions
Event Triggers

Conditions

Gates execution of scenes
Event Triggers

Conditions

Gates execution of scenes

• Time-based
Event Triggers

Conditions

Gates execution of scenes

- Time-based
- State of an accessory
Event Triggers

Conditions

Gates execution of scenes
- Time-based
- State of an accessory
- Significant events
Event Triggers
Conditions

Gates execution of scenes
- Time-based
- State of an accessory
- Significant events
  - Sunrise
Event Triggers

Conditions

Gates execution of scenes

- Time-based
- State of an accessory
- Significant events
  - Sunrise
  - Sunset
Event Triggers
Event Triggers
Event Triggers
Event Triggers

Arrive Home:
- Light on
- Unlock doors
- Temp up
- Raise shades
- Switch on
- Open garage
- Open gates
- Light on
- Unlock doors

Door opens, sun sets, arrive home, triggers:
- Open gates
- Open garage
- Switch on
- Temp up
- Raise shades
- Unlock doors
- Light on
Event Triggers

- Arrive Home
- Open gates
- Open garage
- Switch on
- Raise shades
- Temp up
- Unlock doors
- Open garage
- Light on
Event Triggers
Event Triggers
HMCalendarEvent
Event Triggers
HMCalendarEvent

Supports absolute dates, including:
• Year
• Month
• Day
• Hour Minute
// Example - Creating calendar time event

import HomeKit

var dateComponents = DateComponents()
dateComponents.hour = 17
dateComponents.minute = 30

let calendarEvent = HMCalendarEvent(fire: dateComponents)

let eventTrigger = HMEventTrigger(name: "Every day at 5:30PM",
                                   events: [calendarEvent],
                                   predicate: nil)
import HomeKit

var dateComponents = DateComponents()
dateComponents.hour = 17
dateComponents.minute = 30

let calendarEvent = HMCalendarEvent(fire: dateComponents)
let eventTrigger = HMEventTrigger(name: "Every day at 5:30PM",
                                 events: [calendarEvent],
                                 predicate: nil)
// Example - Creating calendar time event

import HomeKit

var dateComponents = DateComponents()
dateComponents.hour = 17
dateComponents.minute = 30

let calendarEvent = HMCalendarEvent(fire: dateComponents)
let eventTrigger = HMEventTrigger(name: "Every day at 5:30PM",
    events: [calendarEvent],
    predicate: nil)
import HomeKit

var dateComponents = DateComponents()
dateComponents.hour = 17
dateComponents.minute = 30

let calendarEvent = HMCalendarEvent(fire: dateComponents)
let eventTrigger = HMEventTrigger(name: "Every day at 5:30PM",
    events: [calendarEvent],
    predicate: nil)
// Example - Creating calendar time event

import HomeKit

var dateComponents = DateComponents()
dateComponents.hour = 17
dateComponents.minute = 30

let calendarEvent = HMCalendarEvent(fire: dateComponents)
let eventTrigger = HMEventTrigger(name: "Every day at 5:30PM",
                                 events: [calendarEvent],
                                 predicate: nil)
// Example - Creating calendar time event

import HomeKit

var dateComponents = DateComponents()
dateComponents.hour = 17
dateComponents.minute = 30

let calendarEvent = HMCalendarEvent(fire: dateComponents)
let eventTrigger = HMEventTrigger(name: "Every day at 5:30PM",
                                 events: [calendarEvent],
                                 predicate: nil)
// Example - Creating calendar time event

import HomeKit

var dateComponents = DateComponents()
dateComponents.hour = 17
dateComponents.minute = 30

let calendarEvent = HMCalendarEvent(fire: dateComponents)
let eventTrigger = HMEventTrigger(name: "Every day at 5:30PM",
  events: [calendarEvent],
  predicate: nil)
Event Triggers
HMSignificantTimeEvent
Event Triggers
HMSignificantTimeEvent

Activates on significant events:
Event Triggers
HMSignificantTimeEvent

Activates on significant events:
• Sunrise
Event Triggers
HMSignificantTimeEvent

Activates on significant events:

• Sunrise
• Sunset
Event Triggers
HMSignificantTimeEvent

Activates on significant events:
• Sunrise
• Sunset

Supports relative time offset
import HomeKit

let sunriseEvent = HMSignificantTimeEvent(significantEvent: HMSignificantEvent.sunrise, offset: nil)

let eventTrigger = HMEventTrigger(name: "Sunrise, Daily", events: [sunriseEvent], predicate: nil)
// Example - Creating significant time events

import HomeKit

let sunriseEvent = HMSignificantTimeEvent(significantEvent: HMSignificantEvent.sunrise,
                                          offset: nil)

let eventTrigger = HMEventTrigger(name: "Sunrise, Daily",
                                        events: [sunriseEvent],
                                        predicate: nil)
// Example - Creating significant time events

import HomeKit

let sunriseEvent = HMSignificantTimeEvent(significantEvent: HMSignificantEvent.sunrise,
                                           offset: nil)

let eventTrigger = HMEventTrigger(name: "Sunrise, Daily",
                                   events: [sunriseEvent],
                                   predicate: nil)
// Example - Creating significant time events

import HomeKit

let sunriseEvent = HMSignificantTimeEvent(significantEvent: HMSignificantEvent.sunrise,
                                          offset: nil)

let eventTrigger = HMEventTrigger(name: "Sunrise, Daily",
                                     events: [sunriseEvent],
                                     predicate: nil)
// Example - Creating significant time events

import HomeKit

let sunriseEvent = HMSignificantTimeEvent(significantEvent: HMSignificantEvent.sunrise,
                                          offset: nil)

let eventTrigger = HMEventTrigger(name: "Sunrise, Daily",
                                   events: [sunriseEvent],
                                   predicate: nil)
// Example – Creating significant time events

import HomeKit

let sunriseEvent = HMSignificantTimeEvent(significantEvent: HMSignificantEvent.sunrise,
                                          offset: nil)

let eventTrigger = HMEventTrigger(name: "Sunrise, Daily",
                                   events: [sunriseEvent],
                                   predicate: nil)
Event Triggers

Significant event conditions

```swift
extension HMEventTrigger {
    open class func predicate(forEvaluatingTriggerOccurringBetweenSignificantEvent
    -> NSPredicate
}
```
Event Triggers
Significant event conditions

```swift
extension HMEventTrigger {
    open class func predicate(forEvaluatingTriggerOccurringBetweenSignificantEvent
    -> NSPredicate
}
```
Event Triggers

HMCharacteristicThresholdRangeEvent
Event Triggers
HMCharacteristicThresholdRangeEvent

Activates on crossing threshold
Event Triggers
HMCharacteristicThresholdRangeEvent

Activates on crossing threshold

Supports:
• Minimum threshold
Event Triggers
HMCharacteristicThresholdRangeEvent

Activates on crossing threshold

Supports:
• Minimum threshold
• Maximum threshold
Event Triggers
HMCharacteristicThresholdRangeEvent

Activates on crossing threshold

Supports:
• Minimum threshold
• Maximum threshold
• Range threshold
// Example - Characteristic threshold range events

import HomeKit

let characteristic = ...

let numberRange = HMNumberRange(minValue: 76)

let event = HMCharacteristicThresholdRangeEvent(characteristic: characteristic,
    thresholdRange: numberRange)

let eventTrigger = HMEventTrigger(name: "Temperature over 76",
    events: [event],
    predicate: nil)
// Example - Characteristic threshold range events

import HomeKit

let characteristic = ...

let numberRange = HMNumberRange(minValue: 76)

let event = HMCharacteristicThresholdRangeEvent(characteristic: characteristic,
                                                thresholdRange: numberRange)

let eventTrigger = HMEventTrigger(name: "Temperature over 76",
                                   events: [event],
                                   predicate: nil)
// Example - Characteristic threshold range events

import HomeKit

let characteristic = ...

let numberRange = HMNumberRange(minValue: 76)
let event = HMCharacteristicThresholdRangeEvent(characteristic: characteristic, thresholdRange: numberRange)
let eventTrigger = HMEventTrigger(name: "Temperature over 76", events: [event], predicate: nil)
import HomeKit

let characteristic = …

let numberRange = HMNumberRange(minValue: 76)
let event = HMCharacteristicThresholdRangeEvent(characteristic: characteristic,
                                              thresholdRange: numberRange)
let eventTrigger = HMEventTrigger(name: "Temperature over 76",
                           events: [event],
                           predicate: nil)
// Example - Characteristic threshold range events

import HomeKit

let characteristic = …

let numberRange = HMNumberRange(minValue: 76)
let event = HMCharacteristicThresholdRangeEvent(characteristic: characteristic, thresholdRange: numberRange)

let eventTrigger = HMEventTrigger(name: "Temperature over 76", events: [event], predicate: nil)
import HomeKit

let characteristic = ...

let numberRange = HMNumberRange(minValue: 76)
let event = HMCharacteristicThresholdRangeEvent(characteristic: characteristic, thresholdRange: numberRange)
let eventTrigger = HMEventTrigger(name: "Temperature over 76", events: [event], predicate: nil)
// Example - Characteristic threshold range events

import HomeKit

let characteristic = ...

let numberRange = HMNumberRange(minValue: 76)
let event = HMCharacteristicThresholdRangeEvent(characteristic: characteristic,
                                            thresholdRange: numberRange)

let eventTrigger =HEMEventTrigger(name: "Temperature over 76",
                                events: [event],
                                predicate: nil)
// Example – Characteristic threshold range events

import HomeKit

let characteristic = …

let numberRange = HMNumberRange(minValue: 76)
let event = HMCharacteristicThresholdRangeEvent(characteristic: characteristic,
    thresholdRange: numberRange)

let eventTrigger = HMEventTrigger(name: "Temperature over 76",
    events: [event],
    predicate: nil)
Event Triggers
HMPresenceEvent
Event Triggers
HMPresenceEvent
Event Triggers
HMPresenceEvent

Current user arrives home
Event Triggers
HMPresenceEvent

Current user arrives home

Current user leaves home
Event Triggers
HMPresenceEvent

Current user arrives home

Current user leaves home
Event Triggers
HMPresenceEvent

Current user arrives home
Current user leaves home
Last user leaves home
Event Triggers

HMPresenceEvent

Current user arrives home
Current user leaves home
Last user leaves home
First user arrives home
// Example - Creating Presence Event

import HomeKit

let event = HMPresenceEvent(presenceType: .anyUserAtHome)
let eventTrigger = HMEventTrigger(name: "Anyone Arrives Home",
                                events: [event],
                                predicate: nil)
// Example - Creating Presence Event

import HomeKit

let event = HMPresenceEvent(presenceType: HMPresenceType.anyUserAtHome)
let eventTrigger = HMEventTrigger(name: "Anyone Arrives Home",
            events: [event],
            predicate: nil)
// Example - Creating Presence Event

import HomeKit

let event = HMPresenceEvent(presenceType: HMPresenceType.anyUserAtHome)
let eventTrigger = HMEventTrigger(name: "Anyone Arrives Home",
    events: [event],
    predicate: nil)
// Example - Creating Presence Event

import HomeKit

let event = HMPresenceEvent(presenceType: HMPresenceType.anyUserAtHome)
let eventTrigger = HMEventTrigger(name: "Anyone Arrives Home",
                                  events: [event],
                                  predicate: nil)
// Example - Creating Presence Event

import HomeKit

let event = HMPresenceEvent(presenceType: HMPresenceType.anyUserAtHome)
let eventTrigger = HMEventTrigger(name: "Anyone Arrives Home",
    events: [event],
    predicate: nil)
Event Triggers
User presence conditions
Event Triggers
User presence conditions

Good Morning
- Raise shades
- Temp up
- Switch on
- Open garage
- Open gates
- Unlock doors
- Light on
Event Triggers
User presence conditions

- Good Morning
- Open gates
- Open garage
- Switch on
- Temp up
- Raise shades
- Unlock doors
- Light on

NEW
Event Triggers
User presence conditions

Good Morning
- Open gates
- Raise shades
- Temp up
- Switch on
- Open garage
- Unlock doors
- Light on
Event Triggers
User presence conditions

Good Morning

- Open gates
- Open garage
- Switch on
- Temp up
- Raise shades
- Unlock doors
- Light on
Event Triggers
User presence conditions

- Good Morning
- Open gates
- Open garage
- Switch on
- Light on
- Unlock doors
- Raise shades
- Temp up
- Open gates
Event Triggers
User presence conditions

extension HMEventTrigger {
    open class func predicateForEvaluatingTrigger(withPresence presenceEvent: HMPresenceEvent) -> NSPredicate
}

Event Triggers
User presence conditions

extension HMEventTrigger {
    open class func predicateForEvaluatingTrigger(withPresence presenceEvent: HMPresenceEvent) -> NSPredicate
}

NEW
// Example - User presence condition

import HomeKit

let sunriseEvent = HMSignificantTimeEvent(significantEvent: HMSignificantEvent.sunrise,
                                           offset: nil)

let presenceEvent = HMPresenceEvent(presenceType: HMPresenceType.anyUserAtHome)
let predicate = HMEventTrigger.predicateForEvaluatingTrigger(withPresence: presenceEvent)

let eventTrigger = HMEventTrigger(name: "Sunrise, Daily",
                                  events: [sunriseEvent],
                                  predicate: predicate)
// Example - User presence condition

import HomeKit

let sunriseEvent = HMSignificantTimeEvent(significantEvent: HMSignificantEvent.sunrise,
                                          offset: nil)

let presenceEvent = HMPresenceEvent(presenceType: HMPresenceType.anyUserAtHome)
let predicate = HMEventTrigger.predicateForEvaluatingTrigger(withPresence: presenceEvent)

let eventTrigger = HMEventTrigger(name: "Sunrise, Daily",
                                   events: [sunriseEvent],
                                   predicate: predicate)
// Example - User presence condition

import HomeKit

let sunriseEvent = HMSignificantTimeEvent(significantEvent: HMSignificantEvent.sunrise, offset: nil)

let presenceEvent = HMPresenceEvent(presenceType: HMPresenceType.anyUserAtHome)
let predicate = HMEventTrigger.predicateForEvaluatingTrigger(withPresence: presenceEvent)

let eventTrigger = HMEventTrigger(name: "Sunrise, Daily", events: [sunriseEvent], predicate: predicate)
// Example - User presence condition

import HomeKit

let sunriseEvent = HMSignificantTimeEvent(significantEvent: HMSignificantEvent.sunrise, offset: nil)

let presenceEvent = HMPresenceEvent(presenceType: HMPresenceType.anyUserAtHome)
let predicate = HMEventTrigger.predicateForEvaluatingTrigger(withPresence: presenceEvent)

let eventTrigger = HMEventTrigger(name: "Sunrise, Daily", events: [sunriseEvent], predicate: predicate)
// Example – User presence condition

import HomeKit

let sunriseEvent = HMSignificantTimeEvent(significantEvent: HMSignificantEvent.sunrise,
                                          offset: nil)

let presenceEvent = HMPresenceEvent(presenceType: HMPresenceType.anyUserAtHome)
let predicate = HMEventTrigger.predicateForEvaluatingTrigger(withPresence: presenceEvent)

let eventTrigger = HMEventTrigger(name: "Sunrise, Daily",
                                     events: [sunriseEvent],
                                     predicate: predicate)
// Example - User presence condition

import HomeKit

let sunriseEvent = HMSignificantTimeEvent(significantEvent: HMSignificantEvent.sunrise, offset: nil)

let presenceEvent = HMPresenceEvent(presenceType: HMPresenceType.anyUserAtHome)
let predicate = HMEventTrigger.predicateForEvaluatingTrigger(withPresence: presenceEvent)

let eventTrigger = HMEventTrigger(name: "Sunrise, Daily",
events: [sunriseEvent],
predicate: predicate)
// Example - User presence condition

import HomeKit

let sunriseEvent = HMSignificantTimeEvent(significantEvent: HMSignificantEvent.sunrise,
                                           offset: nil)

let presenceEvent = HMPresenceEvent(presenceType: HMPresenceType.anyUserAtHome)
let predicate = HMEventTrigger.predicateForEvaluatingTrigger(withPresence: presenceEvent)

let eventTrigger = HMEventTrigger(name: "Sunrise, Daily",
                                   events: [sunriseEvent],
                                   predicate: predicate)
Event Triggers
End events
Event Triggers

End events
Event Triggers
End events
Event Triggers
End events
Event Triggers
End events

```swift
open class HMEventTrigger : HMTrigger {
    open var endEvents: [HMEvent] { get }
    open func updateEndEvents(_ endEvents: [HMEvent],
        completionHandler completion: @escaping (Error?) -> Swift.Void)
}
```
Event Triggers
End events

```swift
open class HMEventTrigger : HMTrigger {
    open var endEvents: [HMEvent] { get }
    open func updateEndEvents(_ endEvents: [HMEvent],
                                completionHandler completion: @escaping (Error?) -> Swift.Void)
}
```
open class HMEventTrigger : HMTrigger {
    open var endEvents: [HMEvent] { get }
    open func updateEndEvents(_: endEvents: [HMEvent], completionHandler completion: @escaping (Error?) -> Swift.Void)
}
open class HMEventTrigger : HMTrigger {
    open var endEvents: [HMEvent] { get }
    open func updateEndEvents(_ endEvents: [HMEvent],
                                completionHandler completion: @escaping (Error?) -> Swift.Void)
}
Event Triggers
End events

```swift
open class HMDurationEvent : NSCopying, NSMutableCopying {
    public init(duration: TimeInterval)
}
```
Event Triggers
Recurrence

Sun  Mon  Tue  Wed  Thu  Fri  Sat
Event Triggers
Recurrence
open class HMEventTrigger : HMTrigger {
    open var recurrences: [DateComponents]? { get }
    open func updateRecurrences(_ recurrences: [DateComponents]?,
        completionHandler completion: @escaping (Error?) -> Swift.Void)
}
open class HMEventTrigger : HMTrigger {
    open var recurrences: [DateComponents]? { get }
    open func updateRecurrences(_ recurrences: [DateComponents]?,
        completionHandler completion: @escaping (Error?) -> Swift.Void)
}
open class HMEventTrigger : HMTrigger {
    open var recurrences: [DateComponents]? { get }
    open func updateRecurrences(_ recurrences: [DateComponents]?,
                          completionHandler completion: @escaping (Error?) -> Swift.Void)
}
import HomeKit

let sunriseEvent = HMSignificantTimeEvent(signedEnt: HMSignificantEvent.sunrise,
                                          offset: nil)

var weekdays = [DateComponents]()
for weekday in 2...5 {
  var recurrence = DateComponents()
  recurrence.weekday = weekday
  weekdays.append(recurrence)
}

let eventTrigger = HMEventTrigger(name: "Sunrise, Daily",
                                   events: [sunriseEvent],
                                   end: nil,
                                   recurrences: weekdays,
                                   predicate: nil)
// Example - Recurrence

import HomeKit

let sunriseEvent = HMSignificantTimeEvent(significantEvent: HMSignificantEvent.sunrise, offset: nil)

var weekdays = [DateComponents]()
for weekday in 2...5 {
    var recurrence = DateComponents()
    recurrence.weekday = weekday
    weekdays.append(recurrence)
}

let eventTrigger = HMEventTrigger(name: "Sunrise, Daily", events: [sunriseEvent], end: nil, recurrences: weekdays, predicate: nil)
// Example - Recurrence

import HomeKit

let sunriseEvent = HMSignificantTimeEvent(significantEvent: HMSignificantEvent.sunrise, offset: nil)

var weekdays = [DateComponents]()
for weekday in 2...5 {
    var recurrence = DateComponents()
    recurrence.weekday = weekday
    weekdays.append(recurrence)
}

let eventTrigger = HMEventTrigger(name: "Sunrise, Daily", events: [sunriseEvent], end: nil, recurrences: weekdays, predicate: nil)
import HomeKit

let sunriseEvent = HMSignificantTimeEvent(significantEvent: HMSignificantEvent.sunrise, offset: nil)

var weekdays = [DateComponents]()
for weekday in 2...5 {
  var recurrence = DateComponents()
  recurrence.weekday = weekday
  weekdays.append(recurrence)
}

let eventTrigger = HMEventTrigger(name: "Sunrise, Daily", events: [sunriseEvent], end: nil, recurrences: weekdays, predicate: nil)
// Example - Recurrence

import HomeKit

let sunriseEvent = HMSignificantTimeEvent(significantEvent: HMSignificantEvent.sunrise,
                                          offset: nil)

var weekdays = [DateComponents]()
for weekday in 2...5 {
    var recurrence = DateComponents()
    recurrence.weekday = weekday
    weekdays.append(recurrence)
}

let eventTrigger = HMEventTrigger(name: "Sunrise, Daily",
                                   events: [sunriseEvent],
                                   end: nil,
                                   recurrences: weekdays,
                                   predicate: nil)
```swift
// Example - Recurrence

import HomeKit

let sunriseEvent = HMSignificantTimeEvent(significantEvent: HMSignificantEvent.sunrise,
                                          offset: nil)

var weekdays = [DateComponents]()
for weekday in 2...5 {
    var recurrence = DateComponents()
    recurrence.weekday = weekday
    weekdays.append(recurrence)
}

let eventTrigger = HMEventTrigger(name: "Sunrise, Daily",
                                   events: [sunriseEvent],
                                   end: nil,
                                   recurrences: weekdays,
                                   predicate: nil)
```
Event Triggers

Execute once
Event Triggers
Execute once

```swift
open class HMEventTrigger : HMTrigger {
    open var executeOnce: Bool { get }
    open func updateExecuteOnce(_ executeOnce: Bool,
        completionHandler completion: @escaping (Error?) -> Swift.Void)
}
```
Event Triggers
Execute once

```swift
open class HMEventTrigger : HMTrigger {
    open var executeOnce: Bool { get }
    open func updateExecuteOnce(_ executeOnce: Bool,
        completionHandler completion: @escaping (Error?) -> Swift.Void)
}
```
open class HMEventTrigger : HMTrigger {
    open var executeOnce: Bool { get }
    open func updateExecuteOnce(_ executeOnce: Bool,
        completionHandler completion: @escaping (Error?) -> Swift.Void)
}
open class HMEventTrigger : HMTrigger {

    open func addEvent(_ event: HMEvent, completionHandler
    completion: @escaping (Error?) -> Swift.Void)

    open func removeEvent(_ event: HMEvent, completionHandler
    completion: @escaping (Error?) -> Swift.Void)
}

Event Triggers
Mutable events
Event Triggers
Mutable events

open class HMEventTrigger : HMTrigger {
    open func updateEvents(_ events: [HMEvent], completionHandler
        completion: @escaping (Error?) -> Swift.Void)
}
import HomeKit

if let sunriseEvent = eventTrigger.events.first as? HMSignificantTimeEvent {
    var mutableSunriseEvent = sunriseEvent.mutableCopy() as HMMutableSignificantTimeEvent

    // Update the offset of the event

    var offset = DateComponents()
    offset.minute = 30
    mutableSunriseEvent.offset = offset

    // Update the events of the trigger

    eventTrigger.updateEvents([mutableSunriseEvent]) { (e: Error?) in
        // Handle error
    }
}
import HomeKit

if let sunriseEvent = eventTrigger.events.first as? HMSignificantTimeEvent {
    var mutableSunriseEvent = sunriseEvent.mutableCopy() as HMMutableSignificantTimeEvent

    // Update the offset of the event

    var offset = DateComponents()
    offset.minute = 30
    mutableSunriseEvent.offset = offset

    // Update the events of the trigger

    eventTrigger.updateEvents([mutableSunriseEvent]) { (e: Error?) in
        // Handle error
    }
}
import HomeKit

if let sunriseEvent = eventTrigger.events.first as? HMSignificantTimeEvent {
    var mutableSunriseEvent = sunriseEvent.mutableCopy() as! HMMutableSignificantTimeEvent

    // Update the offset of the event

    var offset = DateComponents()
    offset.minute = 30
    mutableSunriseEvent.offset = offset

    // Update the events of the trigger

    eventTrigger.updateEvents([mutableSunriseEvent]) { (e: Error?) in
        // Handle error
    }
}
import HomeKit

if let sunriseEvent = eventTrigger.events.first as? HMSignificantTimeEvent {
    var mutableSunriseEvent = sunriseEvent.mutableCopy() as HMMutableSignificantTimeEvent

    // Update the offset of the event
    var offset = DateComponents()
    offset.minute = 30
    mutableSunriseEvent.offset = offset

    // Update the events of the trigger
    eventTrigger.updateEvents([mutableSunriseEvent]) { (e: Error?) in
        // Handle error
    }
}
import HomeKit

if let sunriseEvent = eventTrigger.events.first as? HMSignificantTimeEvent {
    var mutableSunriseEvent = sunriseEvent.mutableCopy() as HMMutableSignificantTimeEvent

    // Update the offset of the event
    var offset = DateComponents()
    offset.minute = 30
    mutableSunriseEvent.offset = offset

    // Update the events of the trigger

    eventTrigger.updateEvents([mutableSunriseEvent]) { (e: Error?) in
        // Handle error
    }
}
Event Triggers
Event Triggers

New Events
Event Triggers

New Events

New Conditions
Event Triggers

New Events

New Conditions

End Events
Event Triggers

- New Events
- New Conditions
- End Events
- Recurrence
Event Triggers

New Events

New Conditions

End Events

Recurrence

Mutable Events
Accessory Updates

Praveen Chegondi, HomeKit Engineering
Specification
Specification
Protocol Enhancements
Categories
Specification
Protocol Enhancements
Categories
Authentication
Specification
Protocol Enhancements
Categories
Authentication
Self-Certification
Specification
Protocol Enhancements
Categories
Authentication
Self-Certification
HomeKit Accessory Protocol Specification
HomeKit Accessory Protocol Specification

Communication between iOS and Accessory
HomeKit Accessory Protocol Specification

Communication between iOS and Accessory

Security
HomeKit Accessory Protocol Specification

Communication between iOS and Accessory

Security

Transports

• IP
• Bluetooth LE
HomeKit Accessory Protocol Specification

Communication between iOS and Accessory

Security

Transports

- IP
- Bluetooth LE

Accessory categories
HomeKit Accessory Protocol Specification
HomeKit Accessory Protocol Specification

Available to all developers today
http://developer.apple.com/homekit
HomeKit Accessory Protocol Specification

Available to all developers today
http://developer.apple.com/homekit

Commercialize HomeKit accessory

Become MFi Licensee
http://developer.apple.com/mfi/
Specification
Protocol Enhancements
Categories
Authentication
Self-Certification
Accessory Setup
Accessory Setup

Power on accessory
Accessory Setup

Power on accessory

Choose accessory
Accessory Setup

Power on accessory

Choose accessory

HomeKit pairing
Select an accessory to add to Home.

Make sure your accessory is powered on and nearby.

Lightbulb
Lightbulb D7
Lightbulb B2
Lightbulb F9
Lightbulb H6
Lightbulb K4

My Accessory Isn’t Shown Here
Learn About Home Accessories
Enhanced Setup Code
Enhanced Setup Code

Scan first
Enhanced Setup Code

Scan first

Automatic selection
Enhanced Setup Code

Scan first

Automatic selection

Enhanced setup code
• Setup ID
• Setup code (8-digit)
Enhanced Setup Code
Enhanced Setup Code

QR Code
Enhanced Setup Code

QR Code

• Standard-based
Enhanced Setup Code

QR Code
- Standard-based
- Small-form factor

NEW
10mm x 10mm
Enhanced Setup Code

Tap-to-pair
- NFC tags
Pair with HomeKit Code
Position the HomeKit code in the frame.
Look for the HomeKit code in the packaging or on the accessory.

Pair with NFC
Place your iPhone near the NFC logo.
If the accessory is not recognized, make sure that it is powered on or try resetting the accessory.

Don’t Have or Can’t Scan Code
HomeKit Bluetooth LE Accessories
HomeKit Bluetooth LE Accessories

Form-factor friendly
HomeKit Bluetooth LE Accessories

Form-factor friendly
Low-power
HomeKit Bluetooth LE Accessories

Form-factor friendly
Low-power
Battery-operated
HomeKit Bluetooth LE Accessories

Broadcast notifications
HomeKit Bluetooth LE Accessories

Broadcast notifications
HomeKit Bluetooth LE Accessories
Broadcast notifications
HomeKit Bluetooth LE Accessories

Broadcast notifications
HomeKit Bluetooth LE Accessories

Broadcast notifications
HomeKit Bluetooth LE Accessories

Broadcast notifications

Diagram showing network connections between HomeKit devices, including a lamp, a door, apple tv, and another lamp.
HomeKit Bluetooth LE Accessories

Broadcast notifications
HomeKit Bluetooth LE Accessories
Broadcast notifications

Secure-broadcast notifications
Secure-broadcast notifications

Sub-second latencies
Specification
Protocol Enhancements
Categories
Authentication
Self-Certification
Accessory Categories
Specification
Protocol Enhancements
Categories
Authentication
Self-Certification
Accessory Authentication
Accessory Authentication

Quality
Accessory Authentication

Quality

Trusted
Accessory Authentication

Quality

Trusted

Hardware-based authentication
Accessory Authentication
Accessory Authentication

Software-based authentication
Accessory Authentication

Software-based authentication

Enables HomeKit on shipping accessories
Specification
Protocol Enhancements
Categories
Authentication
Self-Certification
Self-Certification

Improved work flow
• HomeKit Certification Assistant
• Increase audit capacity
Self-Certification

Improved work flow
• HomeKit Certification Assistant
• Increase audit capacity

Apple Authorized Labs
• United States
• China
• UK
Summary

Event Triggers

HomeKit Specifications

Protocol Enhancements

New Accessory Categories

Self-Certification Process
More Information

## Related Sessions

<table>
<thead>
<tr>
<th>Session Title</th>
<th>Location</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>What’s New in Core Bluetooth</td>
<td>Grand Ballroom B</td>
<td>Thursday 11:00AM</td>
</tr>
<tr>
<td>Labs</td>
<td>Technology Lab J</td>
<td>Time</td>
</tr>
<tr>
<td>--------------</td>
<td>------------------</td>
<td>---------------</td>
</tr>
<tr>
<td>HomeKit Lab 1</td>
<td>Technology Lab J</td>
<td>Wed 10:00AM–11:30AM</td>
</tr>
<tr>
<td>HomeKit Lab 2</td>
<td>Technology Lab J</td>
<td>Thu 4:10PM–5:40PM</td>
</tr>
</tbody>
</table>