What’s New in HomeKit

Session 210

Anush Nadathur HomeKit Engineer
Naveen Kommareddi HomeKit Engineer
HomeKit
Simplifying home automation
HomeKit
Simplifying home automation
HomeKit
Simplifying home automation
HomeKit
Simplifying home automation
HomeKit in iOS 9
HomeKit in iOS 9

Accessory Updates
HomeKit in iOS 9

Accessory Updates
HomeKit in iOS 9
Maintain Existing Objects

Living Room

Family Room

Kitchen

Bedroom
Maintain Existing Objects

Living Room

Family Room

Kitchen

Bedroom
Maintain Existing Objects
Targeted change notifications

```swift
protocol HMHomeManagerDelegate : NSObjectProtocol {
    func homeManagerDidUpdateHomes(manager: HMHomeManager)
}
```
Maintain Existing Objects

Targeted change notifications

```swift
protocol HMHomeManagerDelegate : NSObjectProtocol {
    func homeManagerDidUpdateHomes(manager: HMHomeManager)
}
```
Maintain Existing Objects
Targeted change notifications

```swift
protocol HMHomeManagerDelegate : NSObjectProtocol {
    func homeManagerDidUpdateHomes(manager: HMHomeManager)
}
```

```swift
protocol HMHomeDelegate : NSObjectProtocol {
    func home(home: HMHome, didUpdateRoom: HMRoom, forAccessory: HMAccessory)
}
```
class HMRoom : NSObject {
    var name: String { get }
}

class HMRoom : NSObject {
    var name: String { get }
}

Persistent Identifiers
class HMRoom : NSObject {
    var name: String { get }
    var uniqueIdentifier: NSUUID { get }
}

Available on all relevant HomeKit classes
extension HMHome {
    func addUserWithCompletionHandler(completion: (HMUser?, NSError?) -> Void)

    func removeUser(user: HMUser, completionHandler: (NSError?) -> Void)

    var users: [HMUser] { get }
}
extension HMHome {

    func addUserWithCompletionHandler(completion: (HMUser?, NSError?) -> Void)

    func removeUser(user: HMUser, completionHandler: (NSError?) -> Void)

    var users: [HMUser] { get }

}
extension HMHome {

    func addUserWithCompletionHandler(completion: (HMUser?, NSError?) -> Void)

    func removeUser(user: HMUser, completionHandler: (NSError?) -> Void)

    var users: [HMUser] { get }
}

User Management
extension HMHome {
    func manageUsersWithCompletionHandler(completion: (NSError?) -> Void)
}
extension HMHome {
    func manageUsersWithCompletionHandler(completion: (NSError?) -> Void)
}

User Management
PEOPLE

Emily Parker

Meg Parker Invited

Invite People

Users can control accessories in your home. You can share your home with anybody with an iCloud account.
User Capabilities
User Capabilities

Display relevant capabilities
User Capabilities

Display relevant capabilities
Determine privileges
extension HMHome {
    var currentUser: HMUser { get }
}
User Capabilities
HMHomeAccessControl

extension HMHome {
  var currentUser: HMUser { get }
}

User Capabilities

HMHomeAccessControl

extension HMHome {
    var currentUser: HMUser { get }
    func homeAccessControlForUser(user: HMUser) -> HMHomeAccessControl
}

User Capabilities
HMHomeAccessControl

extension HMHome {
    var currentUser: HMUser { get }
    func homeAccessControlForUser(user: HMUser) -> HMHomeAccessControl
}
extension HMHome {
    var currentUser: HMUser { get }
    func homeAccessControlForUser(user: HMUser) -> HMHomeAccessControl
}

class HMHomeAccessControl : NSObject {
    var administrator: Bool { get }
}
extension HMHome {
    var currentUser: HMUser { get }
    func homeAccessControlForUser(user: HMUser) -> HMHomeAccessControl
}

class HMHomeAccessControl: NSObject {
    var administrator: Bool { get }
}
Good night

- Close gates
- Close garage
- Lower shades
- Temp down
- Switch off
- Light off
- Lock doors

Lock doors
Light off
Predefined Scenes
Predefined Scenes

Four common events
Predefined Scenes

Four common events

• Get up
Predefined Scenes

Four common events

• Get up
• Leave
Predefined Scenes

Four common events

• Get up
• Leave
• Return
Predefined Scenes

Four common events

• Get up
• Leave
• Return
• Go to bed
Predefined Scenes

Four common events

• Get up
• Leave
• Return
• Go to bed

Suggest actions
Predefined Scenes

Four common events

• Get up
• Leave
• Return
• Go to bed

Suggest actions

Cannot be deleted
class HMActionSet : NSObject {
    var actionSetType: String { get }
}
class HMAActionSet : NSObject {
    var actionSetType: String { get }
}

Predefined Scenes
HMAActionSet
class HMAActionSet: NSObject {
    var actionSetType: String { get }
}

let HMAActionSetTypeWakeUp: String
let HMAActionSetTypeSleep: String
let HMAActionSetTypeHomeDeparture: String
let HMAActionSetTypeHomeArrival: String
Predefined Scenes

class HMActionSet : NSObject {
    var actionSetType: String { get }
}

let HMActionSetTypeWakeUp: String
let HMActionSetTypeSleep: String
let HMActionSetTypeHomeDeparture: String
let HMActionSetTypeHomeArrival: String
class HMActionSet : NSObject {
    var actionSetType: String { get }
}

let HMActionSetTypeWakeUp: String
let HMActionSetTypeSleep: String
let HMActionSetTypeHomeDeparture: String
let HMActionSetTypeHomeArrival: String
let HMActionSetTypeUserDefined: String
class HMAActionSet : NSObject {
    var actionSetType: String { get }
}

let HMAActionSetTypeWakeUp: String
let HMAActionSetTypeSleep: String
let HMAActionSetTypeHomeDeparture: String
let HMAActionSetTypeHomeArrival: String
let HMAActionSetTypeUserDefined: String
extension HMHome {
    var actionSets: [HMActionSet] { get }
}
extension HMHome {
    var actionSets: [HMActionSet] { get }
}
extension HMHome {
    var actionSets: [HMActionSet] { get }

    func builtinActionSetOfType(actionSetType: String) -> HMActionSet?
}
extension HMHome {
    var actionSets: [HMActionSet] { get }

    func builtinActionSetOfType(actionSetType: String) -> HMActionSet?
}
Scenes and Siri
Scenes and Siri

Siri understands the names of scenes
Scenes and Siri

Siri understands the names of scenes
Speaking the name of scene executes it
Acme Ultra
A984CCD90714

Acme Deluxe
4C6B8F191C84
Accessory Category
Accessory Category

Available during setup
Accessory Category

Available during setup

Specifies primary category of the accessory
class HMAccessoryCategory : NSObject {
    var categoryType: String { get }
}
class HMAccessoryCategory : NSObject {
    var categoryType: String { get }
}

Accessory Category
HMAccessoryCategory
class HMAccessoryCategory : NSObject {
    var categoryType: String { get }
}
let HMAccessoryCategoryTypeLightbulb: String
class HMAccessoryCategory : NSObject {
    var categoryType: String { get }
}

let HMAccessoryCategoryTypeLightbulb: String
class HMAccessoryCategory : NSObject {
    var categoryType: String { get }
}

let HMAccessoryCategoryTypeLightbulb: String
let HMAccessoryCategoryTypeFan: String
class HMAccessoryCategory : NSObject {
    var categoryType: String { get }
}

let HMAccessoryCategoryTypeLightbulb: String
let HMAccessoryCategoryTypeFan: String
class HMAccessoryCategory : NSObject {
    var categoryType: String { get }
}

let HMAccessoryCategoryTypeLightbulb: String
let HMAccessoryCategoryTypeFan: String

class HMAccessory : NSObject {
    var category: HMAccessoryCategory { get }
}
class HMAccessoryCategory : NSObject {
    var categoryType: String { get }
}

let HMAccessoryCategoryTypeLightbulb: String
let HMAccessoryCategoryTypeFan: String

class HMAccessory : NSObject {
    var category: HMAccessoryCategory { get }
}

HomeKit and Apple Watch
HomeKit and Apple Watch

HomeKit available on watchOS
HomeKit and Apple Watch

HomeKit available on watchOS
Home data is mirrored on Apple Watch
HomeKit and Apple Watch

HomeKit available on watchOS
Home data is mirrored on Apple Watch

Capabilities
• View homes
HomeKit available on watchOS
Home data is mirrored on Apple Watch
Capabilities
- View homes
- Control accessories
HomeKit and Apple Watch

HomeKit available on watchOS
Home data is mirrored on Apple Watch

Capabilities
- View homes
- Control accessories
- Execute scenes
Triggers
Triggers
Triggers
Triggers
Triggers

Arrive Home Scene

- Close gates
- Close garage
- Raise shades
- Outlet on
- Temp up
- Switch on
- Light on

Outlet on
Triggers

- Arrive Home
- Scene
- Close gates
- Close garage
- Raise shades
- Outlet on
- Switch on
- Temp up
- Light on
- Outlet on

Arrive Home Scene
Event Triggers
Event Triggers

Events

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
class HMCharacteristicEvent : HMEvent {
    init(characteristic: HMCharacteristic, triggerValue: NSCopying)
}
class HMCharacteristicEvent : HMEvent {
    init(characteristic: HMCharacteristic, triggerValue: NSCopying)
}
class HMCharacteristicEvent : HMEvent {
    init(characteristic: HMCharacteristic, triggerValue: NSCopying)
}

class HMLocationEvent : HMEvent {
    init(region: CLRegion)
}
Event Triggers

HMEvent

class HMCharacteristicEvent : HMEvent {
    init(characteristic: HMCharacteristic, triggerValue: NSCopying)
}

class HMLocationEvent : HMEvent {
    init(region: CLRegion)
}
let frontDoorUnlockedEvent = HMCharacteristicEvent(characteristic: frontDoorCurrentStateCharacteristic, triggerValue: HMCharacteristicValue.DoorStateOpen)
Event Triggers

HMCharacteristicEvent

```swift
let frontDoorUnlockedEvent = HMCharacteristicEvent(characteristic:
    frontDoorCurrentStateCharacteristic,
    triggerValue: HMCharacteristicValue.DoorStateOpen)
```
Event Triggers

Conditions
Event Triggers

Conditions

Gate execution of scenes
Event Triggers

Conditions

Gate execution of scenes

• Time-based
Event Triggers

Conditions

Gate execution of scenes
- Time-based
- State of an accessory
Event Triggers

Conditions

Gate execution of scenes

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

Conditions—time

class HMEventTrigger : HMTigger {
    class func predicateForEvaluatingTriggerOccurringBeforeDateWithComponents (dateComponents: NSDateComponents) -> NSPredicate
}

Event Triggers

Conditions–time

class HMEventTrigger : HMTrigger {
    class func predicateForEvaluatingTriggerOccurringBeforeDateWithComponents
do(ateComponents: NSDateComponents) -> NSPredicate
}
Event Triggers
Conditions–time

class HMEventTrigger : HMTrigger {
    class func predicateForEvaluatingTriggerOccurringBeforeDateWithComponents(
        dateComponents: NSDateComponents) -> NSPredicate

    class func predicateForEvaluatingTriggerOccurringAfterDateWithComponents(
        dateComponents: NSDateComponents) -> NSPredicate
}

class **HMEventTrigger** : HMTrigger {
    class func predicateForEvaluatingTriggerOccurringBeforeDateWithComponents(
        dateComponents: NSDateComponents) -> NSPredicate

    class func predicateForEvaluatingTriggerOccurringAfterDateWithComponents(
        dateComponents: NSDateComponents) -> NSPredicate
}

---

**Event Triggers**

**Conditions–time**

```swift
class HMEventTrigger : HMTrigger {
    class func predicateForEvaluatingTriggerOccurringBeforeDateWithComponents(
        dateComponents: NSDateComponents) -> NSPredicate

    class func predicateForEvaluatingTriggerOccurringAfterDateWithComponents(
        dateComponents: NSDateComponents) -> NSPredicate
}
```
Event Triggers
Conditions–time

class HMEventTrigger : HMTrigger {
    class func predicateForEvaluatingTriggerOccurringBeforeDateWithComponents(
        dateComponents: NSDateComponents) -> NSPredicate

    class func predicateForEvaluatingTriggerOccurringAfterDateWithComponents(
        dateComponents: NSDateComponents) -> NSPredicate

    class func predicateForEvaluatingTriggerOccurringOnDateWithComponents(
        dateComponents: NSDateComponents) -> NSPredicate
}

class **HMEventTrigger** : HMTrigger {
    class func predicateForEvaluatingTriggerOccurringBeforeDateWithComponents(
        dateComponents: NSDateComponents) -> NSPredicate

    class func predicateForEvaluatingTriggerOccurringAfterDateWithComponents(
        dateComponents: NSDateComponents) -> NSPredicate

    class func predicateForEvaluatingTriggerOccurringOnDateWithComponents(
        dateComponents: NSDateComponents) -> NSPredicate
}

---

**Event Triggers**

**Conditions—time**
Event Triggers

Conditions—time
Event Triggers

Conditions–time

```swift
let before6PM = NSDateComponents()
before6PM.hour = 18
```
Event Triggers

Conditions—time

let before6PM = NSDateComponents()
before6PM.hour = 18

let before6PMPredicate = HMEventTrigger.
predicateForEvaluatingTriggerOccurringBeforeDateWithComponents(before6PM)
let before6PM = NSDateComponents()
before6PM.hour = 18

let before6PMPredicate = HMEventTrigger.predicateForEvaluatingTriggerOccurringBeforeDateWithComponents(before6PM)
class HMEventTrigger : HMTrigger {
    class func predicateForEvaluatingTriggerWithCharacteristic(
        characteristic: HMCharacteristic,
        matchingValue: AnyObject) -> NSPredicate
}
Event Triggers

Conditions–accessory state

class HMEventTrigger : HMTrigger {
    class func predicateForEvaluatingTriggerWithCharacteristic(
        characteristic: HMCharacteristic,
        matchingValue: AnyObject) -> NSPredicate
}
Event Triggers

Conditions—accessory state

let motionDetectedPredicate = HMEventTrigger.
predicateForEvaluatingTriggerWithCharacteristic(characteristic:
    frontDoorMotionSensorCharacteristic,
    matchingValue:true)
let motionDetectedPredicate = HMEventTrigger.
predicateForEvaluatingTriggerWithCharacteristic(characteristic:
    frontDoorMotionSensorCharacteristic,
    matchingValue:true)
Event Triggers

Conditions—significant events in a day

class HMEventTrigger : HMTrigger {
    class func predicateForEvaluatingTriggerOccurringBeforeSignificantEvent(
        significantEvent: String,
        applyingOffset: NSDateComponents?) -> NSPredicate
}

Event Triggers

Conditions–significant events in a day

class HMEventTrigger: HMTrigger {
    class func predicateForEvaluatingTriggerOccurringBeforeSignificantEvent(
        significantEvent: String,
        applyingOffset: NSDateComponents?) -> NSPredicate
}
Event Triggers

Conditions–significant events in a day

class HMEventTrigger : HMTrigger {
    class func predicateForEvaluatingTriggerOccurringBeforeSignificantEvent (significantEvent: String, applyingOffset: NSDateComponents?) -> NSPredicate

    class func predicateForEvaluatingTriggerOccurringAfterSignificantEvent (significantEvent: String, applyingOffset: NSDateComponents?) -> NSPredicate
}
Event Triggers

Conditions—significant events in a day

class HMEventTrigger : HMTrigger {
    class func predicateForEvaluatingTriggerOccurringBeforeSignificantEvent
        (significantEvent: String,
         applyingOffset: NSDateComponents?) -> NSPredicate

class func predicateForEvaluatingTriggerOccurringAfterSignificantEvent
    (significantEvent: String,
     applyingOffset: NSDateComponents?) -> NSPredicate
}
Event Triggers

Conditions—significant events in a day

let HMSignificantEventSunrise: String
let HMSignificantEventSunset: String
Event Triggers

Conditions—significant events in a day

let HMSignificantEventSunrise: String
let HMSignificantEventSunset: String
Event Triggers

Conditions—significant events in a day

let HMSignificantEventSunrise: String
let HMSignificantEventSunset: String

let offset = NSDateComponents()
offset.minute = -30
Event Triggers

Conditions—significant events in a day

let **HMSignificantEventSunrise**: String
let **HMSignificantEventSunset**: String

```swift
let offset = NSDateComponents()
offset.minute = -30
```
Event Triggers

Conditions—significant events in a day

```swift
let HMSignificantEventSunrise: String
let HMSignificantEventSunset: String

let offset = NSDateComponents()
offset.minute = -30

let thirtyMinutesBeforeSunsetPredicate =
    HMEventTrigger.predicateForEvaluatingTriggerOccurringBeforeSignificantEvent
    (HMSignificantEventSunset, applyingOffset: offset)
```
Event Triggers

Conditions—significant events in a day

```swift
let HMSignificantEventSunrise: String
let HMSignificantEventSunset: String

let offset = NSDateComponents()
offset.minute = -30

let thirtyMinutesBeforeSunsetPredicate = 
    HMEventTrigger.predicateForEvaluatingTriggerOccurringBeforeSignificantEvent
    (HMSignificantEventSunset, applyingOffset:offset)
```
class HMEventTrigger : HMTrigger {
    init(name: String,
         events:[HMEvent],
         predicate: NSPredicate?)
}
Event Triggers

HMEventTrigger

class HMEventTrigger : HMTrigger {

    init(name: String,
            events:[HMEvent],
            predicate: NSPredicate?)

}

Event Triggers

HMEventTrigger

let predicate = NSCompoundPredicate.andPredicateWithSubpredicates([before6PMPredicate, motionDetectedPredicate])
Event Triggers

HMEventTrigger

```swift
let predicate = NSCompoundPredicate.andPredicateWithSubpredicates(
    ([before6PMPredicate, motionDetectedPredicate])
)```
let predicate = NSCompoundPredicate.andPredicateWithSubpredicates(
    ([before6PMPredicate, motionDetectedPredicate])

let eventTrigger = HMEventTrigger("Arrived Home",
    events: [frontDoorUnlockedEvent], predicate: predicate)
Event Triggers

let predicate = NSCompoundPredicate.andPredicateWithSubpredicates([before6PMPredicate, motionDetectedPredicate])

let eventTrigger = HMEventTrigger("Arrived Home", events:[frontDoorUnlockedEvent], predicate:predicate)
let predicate = NSCompoundPredicate.andPredicateWithSubpredicates(
  ([before6PMPredicate, motionDetectedPredicate])
)

let eventTrigger = HMEventTrigger("Arrived Home",
  events:[frontDoorUnlockedEvent], predicate:predicate)

let arrivedHome = home.builtinActionSetOfType(HMActionSetTypeHomeArrival)

eventTrigger.addActionSet(arrivedHome,
  completionHandler:(NSError?) -> Void { error in /* */ })
let predicate = NSCompoundPredicate.andPredicateWithSubpredicates(
    ([before6PMPredicate, motionDetectedPredicate]))

let eventTrigger = HMEventTrigger("Arrived Home",
    events:[frontDoorUnlockedEvent], predicate:predicate)

let arrivedHome = home.builtinActionSetOfType(HMActionSetTypeHomeArrival)

eventTrigger.addActionSet(arrivedHome,
    completionHandler:(NSError?) -> Void { error in /* */ })
Demo

EventTriggers using HomeKitCatalog
Accessories

Naveen Kommareddi HomeKit Engineer
Remote Access
Remote Access

Bluetooth Low Energy
Remote Access

Bluetooth Low Energy

Accessory Categories
Remote Access

Bluetooth Low Energy

Accessory Categories
Remote Access
Remote Access

Control when away from home
Remote Access

Control when away from home
Remote Access

Control when away from home
Remote Access

Control when away from home
Apple TV
• 3rd generation
Remote Access

Control when away from home
Apple TV
- 3rd generation
- Sign in with Apple ID
Remote Access
Remote Access

HomeKit Accessory Protocol (HAP) over iCloud
Remote Access

HomeKit Accessory Protocol (HAP) over iCloud

- Control and notifications
Remote Access

HomeKit Accessory Protocol (HAP) over iCloud
- Control and notifications
- Dedicated iCloud service
Remote Access

HomeKit Accessory Protocol (HAP) over iCloud

- Control and notifications
- Dedicated iCloud service
- Free
Remote Access

HomeKit Accessory Protocol (HAP) over iCloud
- Control and notifications
- Dedicated iCloud service
- Free
- Private and secure
Bluetooth Low Energy
Bluetooth Low Energy
Bluetooth Low Energy
Bluetooth Low Energy
Bluetooth Low Energy
Bluetooth Low Energy

HAP secure tunneling
Bluetooth Low Energy

HAP secure tunneling
Bluetooth Low Energy

HAP secure tunneling

Range extender
Bluetooth Low Energy

HAP secure tunneling

Range extender

Device range extension
Bluetooth Low Energy

HAP secure tunneling

Range extender
Bluetooth Low Energy
Bluetooth Low Energy

Notifications
Bluetooth Low Energy

Notifications

Metadata for custom characteristics
Bluetooth Low Energy

Notifications
Metadata for custom characteristics
Support for multiple transports
Accessory Categories
Accessory Categories

Awnings
Blinds
Shades
Window coverings
Accessory Categories

Window coverings
- Awnings
- Blinds
- Shades

Doors and windows
Accessory Categories

- Alarm systems
- Doors and windows
- Awnings
- Blinds
- Shades
- Window coverings
Accessory Categories

- Window coverings
  - Awnings
  - Blinds
  - Shades

- Doors and windows

- Alarm systems

- Sensors
  - Motion
  - Air quality
  - Smoke
Accessory Categories

- Window coverings
  - Awnings
  - Blinds
  - Shades
- Doors and windows
- Alarm systems
- Sensors
  - Motion
  - Air quality
  - Smoke
- Programmable switches
Accessory Categories
Programmable switch
Accessory Categories
Programmable switch
Accessory Categories
Programmable switch

Event
Accessory Categories

Programmable switch
Accessory Categories

Programmable switch

Event => Trigger

Good night

Light off
Lock doors
Temp down
Lower shades
Switch off
Close garage
Close gates
Accessory Categories
Programmable switch

Event → Trigger

- Good night
- Light off
- Temp down
- Lock doors
- Lower shades
- Close garage
- Close gates
- Switch off
Resources
Resources

HomeKit Accessory Simulator (HAS)
Resources

HomeKit Accessory Simulator (HAS)
HomeKit Accessory Tester (HAT)
Resources

HomeKit Accessory Simulator (HAS)
HomeKit Accessory Tester (HAT)
Updated specifications and tools at MFi Portal
Resources

HomeKit Accessory Simulator (HAS)
HomeKit Accessory Tester (HAT)
Updated specifications and tools at MFi Portal
MFi Program
Resources

HomeKit Accessory Simulator (HAS)

HomeKit Accessory Tester (HAT)

Updated specifications and tools at MFi Portal

MFi Program

- developer.apple.com/programs/mfi/
Summary

Enhancements
Predefined scenes
HomeKit on Apple Watch
Event triggers
Remote access
New features for Bluetooth accessories
New accessory categories
More Information

Documentation
HomeKit
developer.apple.com/homekit

Technical Support
Apple Developer Forum
Developer Technical Support
developer.apple.com/forums

General Inquiries
homekit@apple.com
Labs

HomeKit Lab

App Frameworks Lab  Wednesday 2:30PM