#WWDC18

HomeKit

Session 231

Keith Rauenbuehler, HomeKit Engineering

© 2018 Apple Inc. All rights reserved. Redistribution or public display not permitted without written permission from Apple.

HomeKit overview

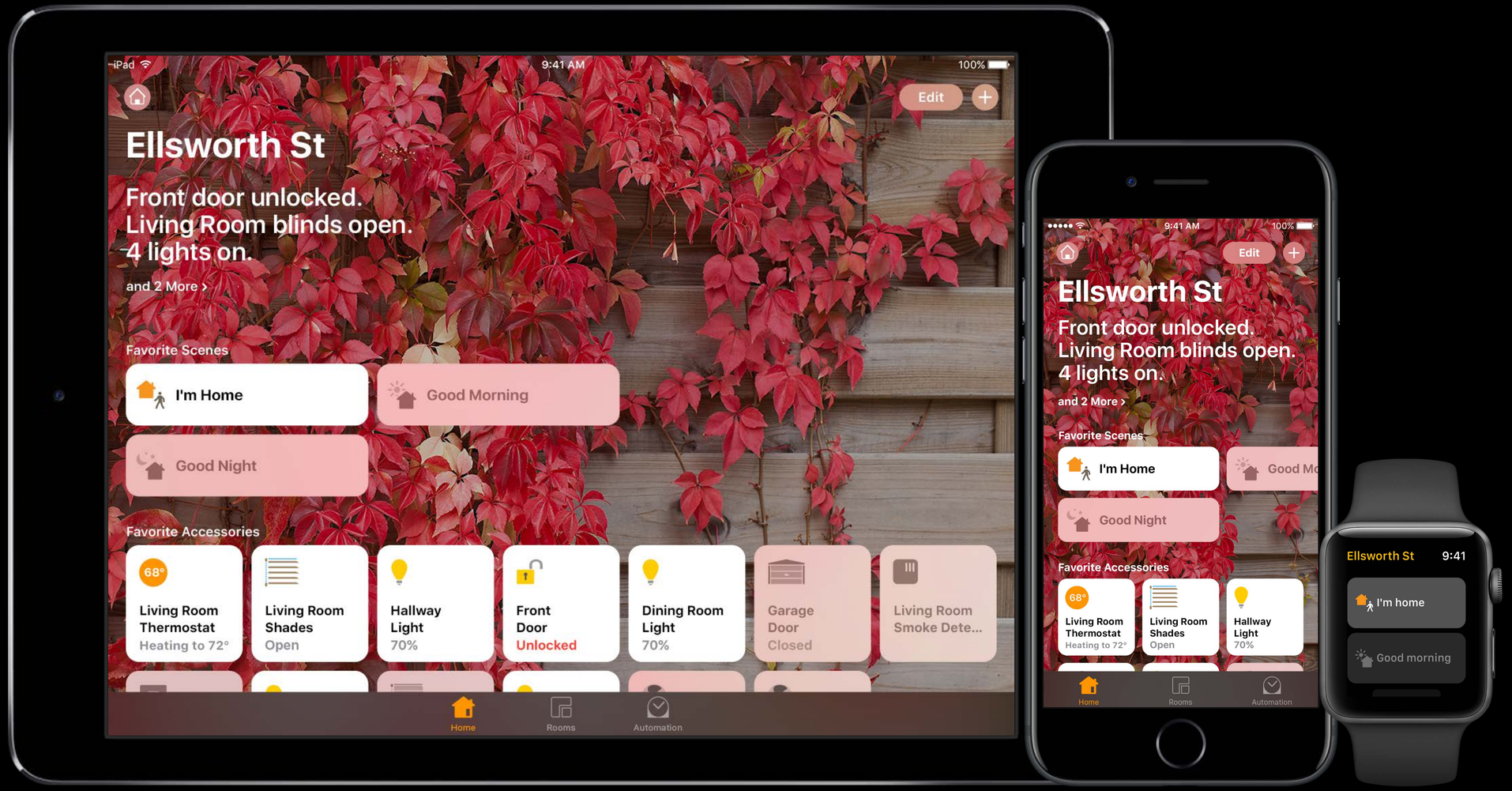
Building an accessory

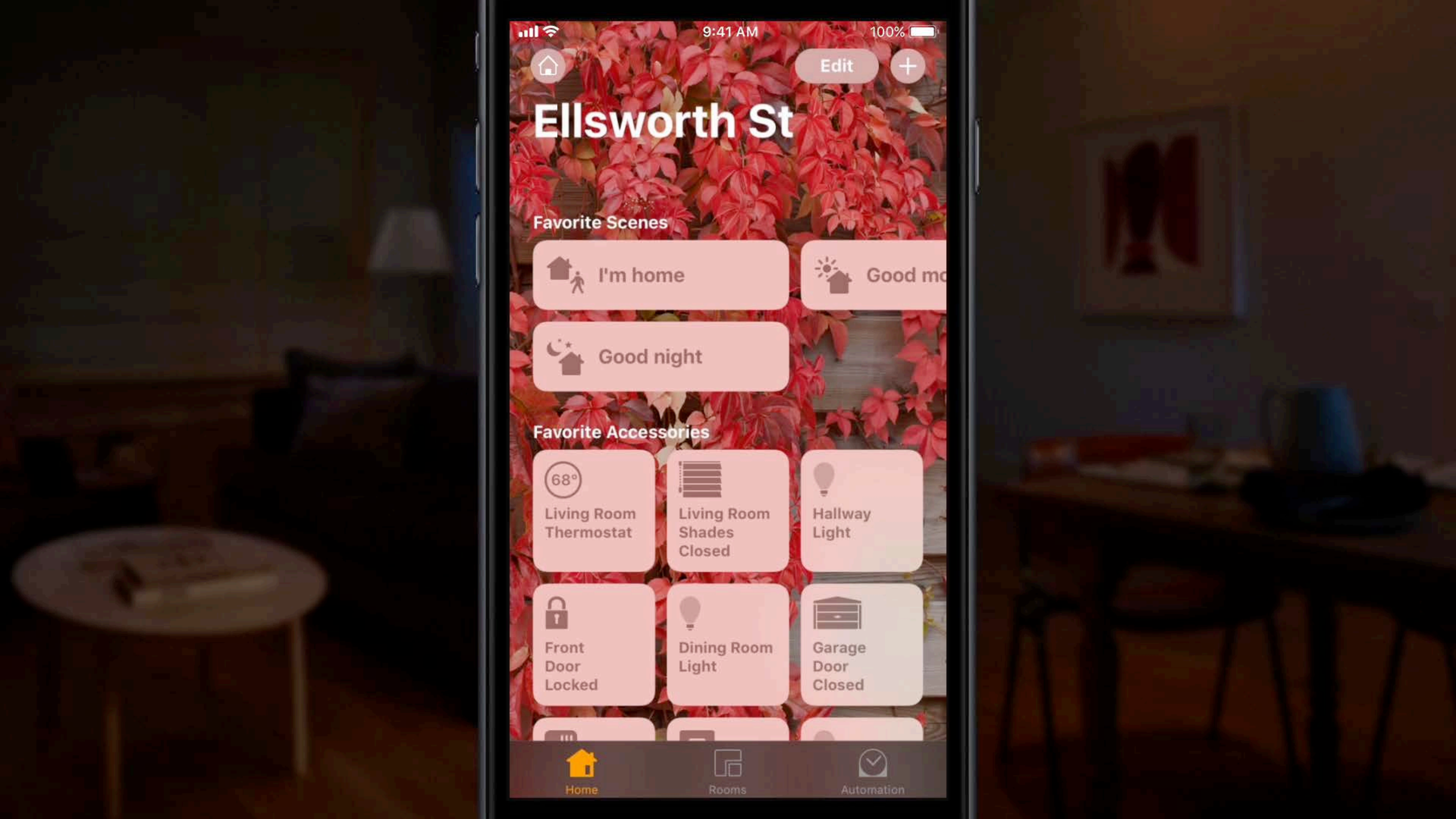
Creating a HomeKit app

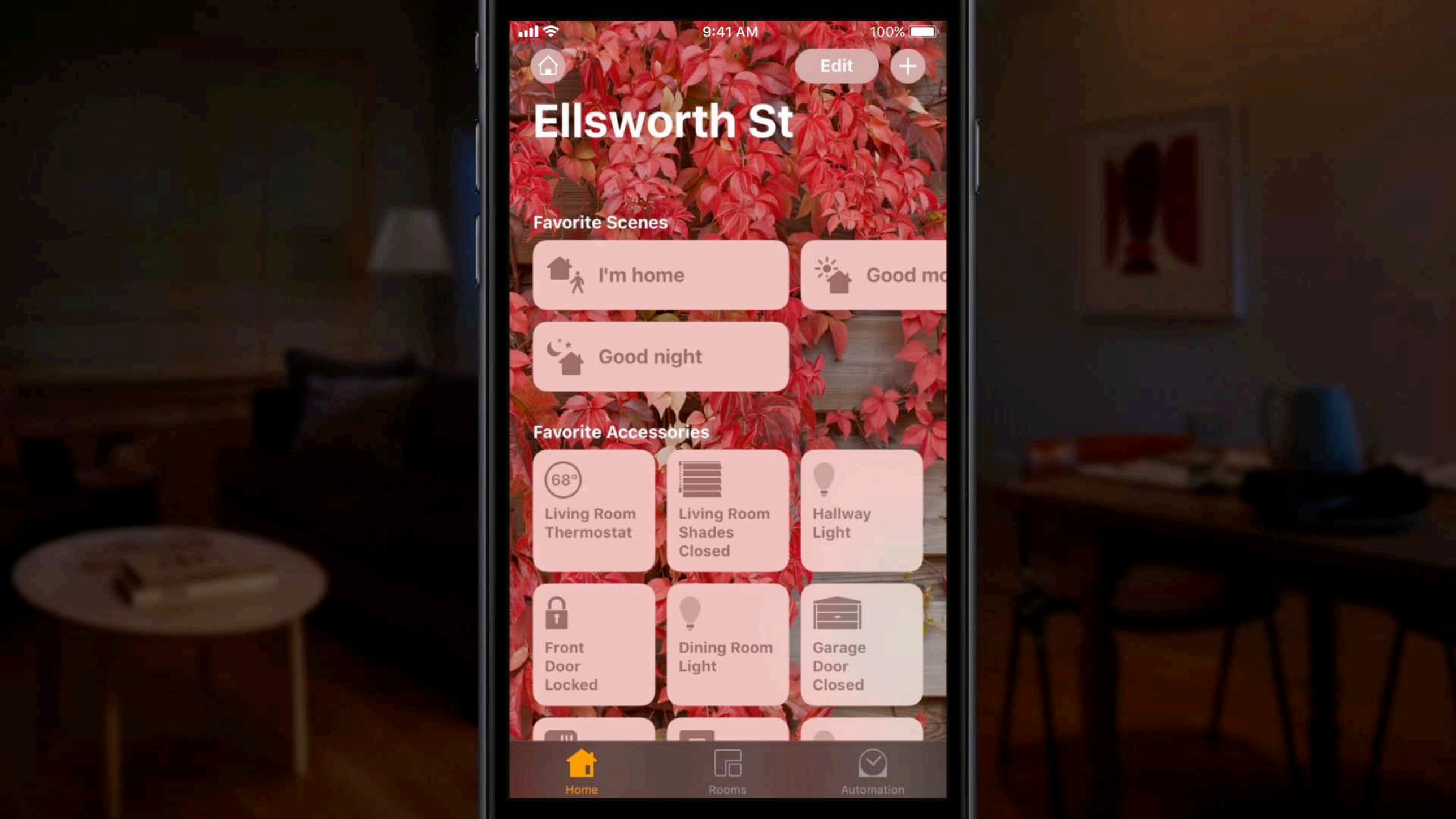
HomeKit overview

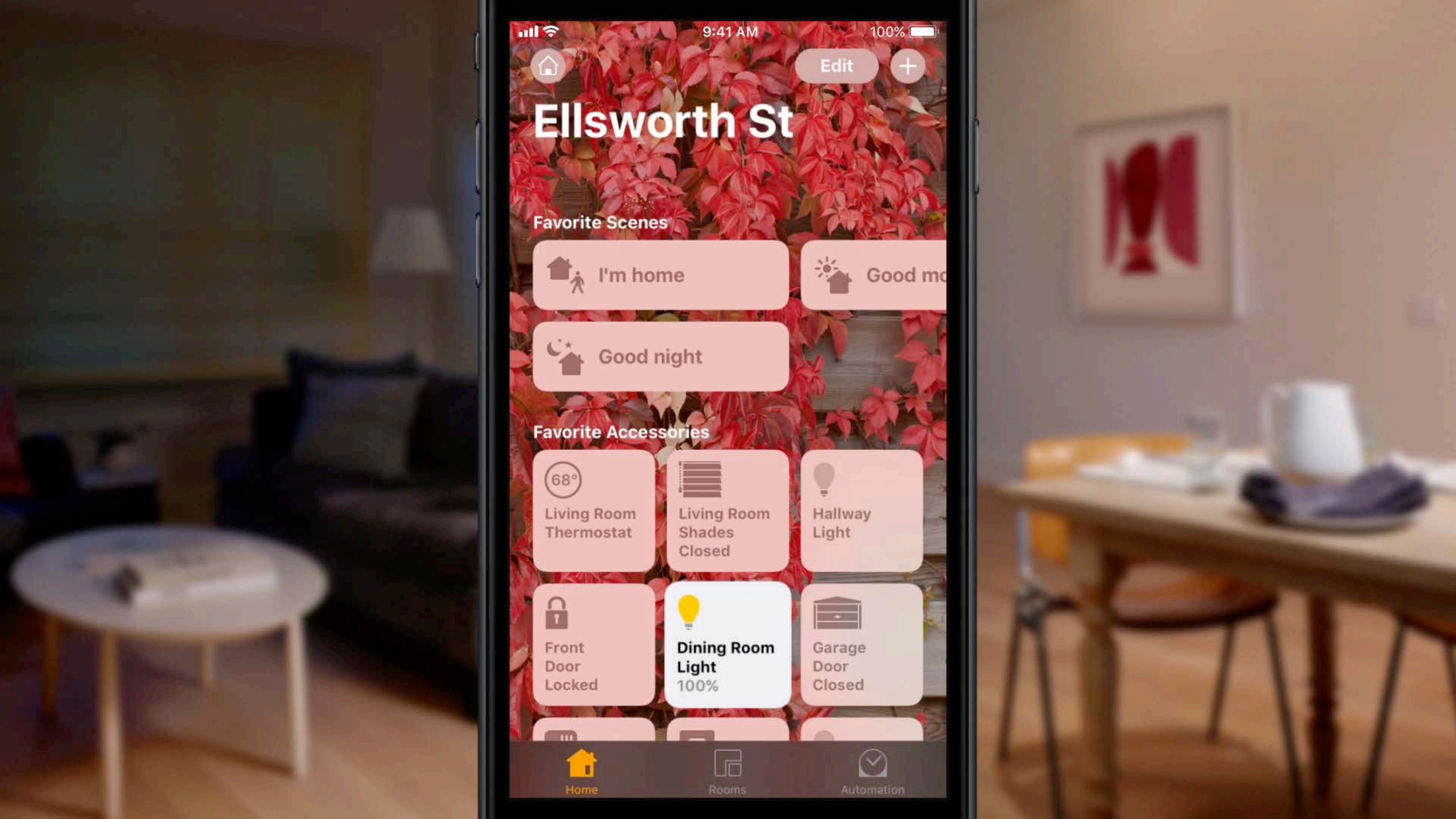
Building an accessory

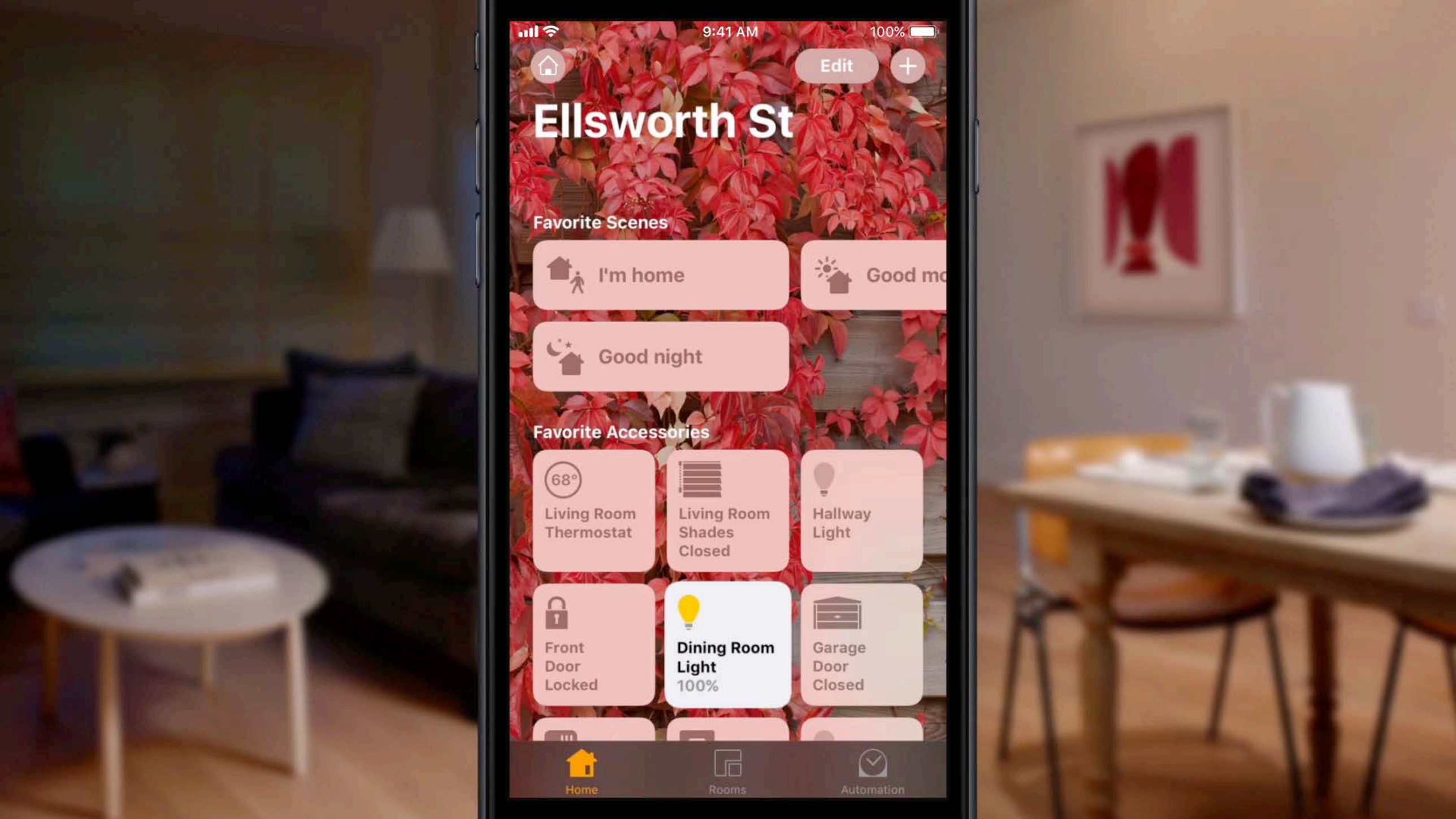
Creating a HomeKit app

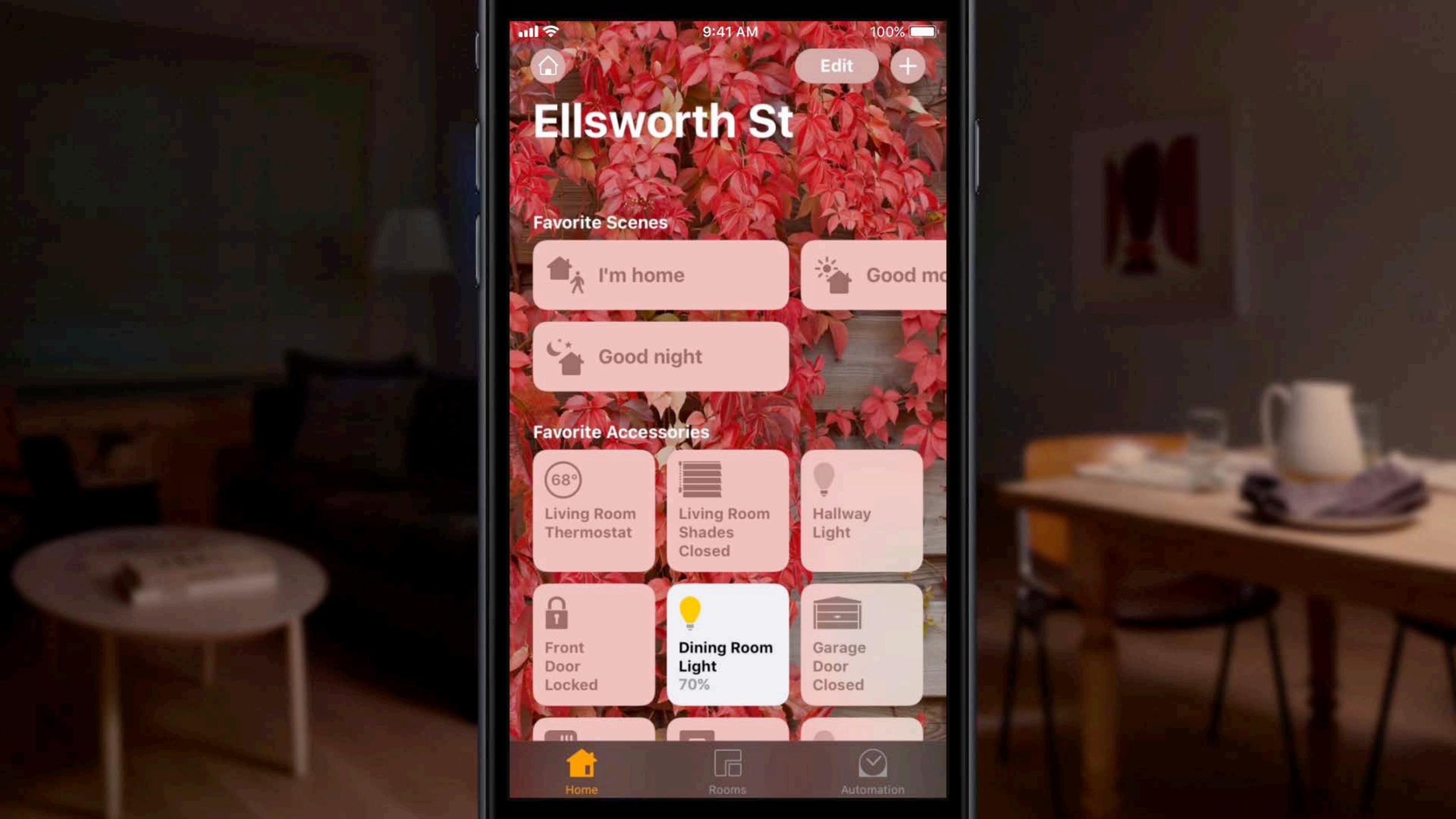

iPad
9:41 AM
100%
Edit
Ellsworth St
Front door unlocked.
Living Room blinds open.
4 lights on.
and 2 More ›
Favorite Scenes
I'm Home
Good Morning
Good Night
Favorite Accessories
68°
Living Room Thermostat
Heating to 72°
Living Room Shades
Open
Hallway Light
70%
Front Door
Unlocked
Dining Room Light
70%
Garage Door
Closed
Living Room Smoke Dete...
Home
Rooms
Automation
9:41 AM
100%
Edit
Ellsworth St
Front door unlocked.
Living Room blinds open.
4 lights on.
and 2 More ›
Favorite Scenes
I'm Home
Good Night
Favorite Accessories
68°
Living Room Thermostat
Heating to 72°
Living Room Shades
Open
Hallway Light
70%
Home
Rooms
Automation
Ellsworth St
9:41
I'm home
Good morning

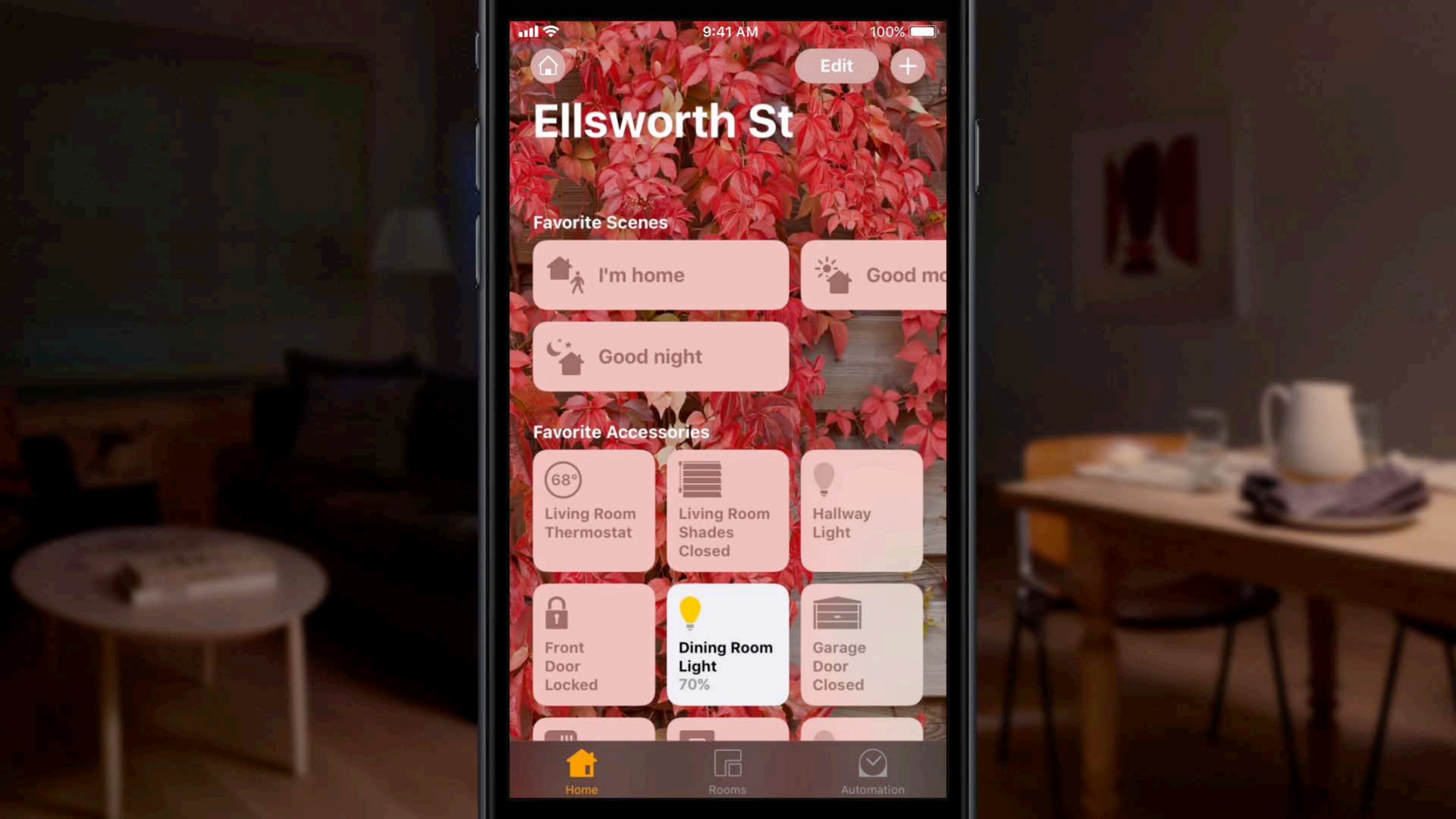

9:41 AM
100%
Edit
Ellsworth St
Favorite Scenes
I'm home
Good night
Favorite Accessories
68°
Living Room Thermostat
Living Room Shades Closed
Hallway Light
Front Door Locked
Dining Room Light
Garage Door Closed
Home
Rooms
Automation

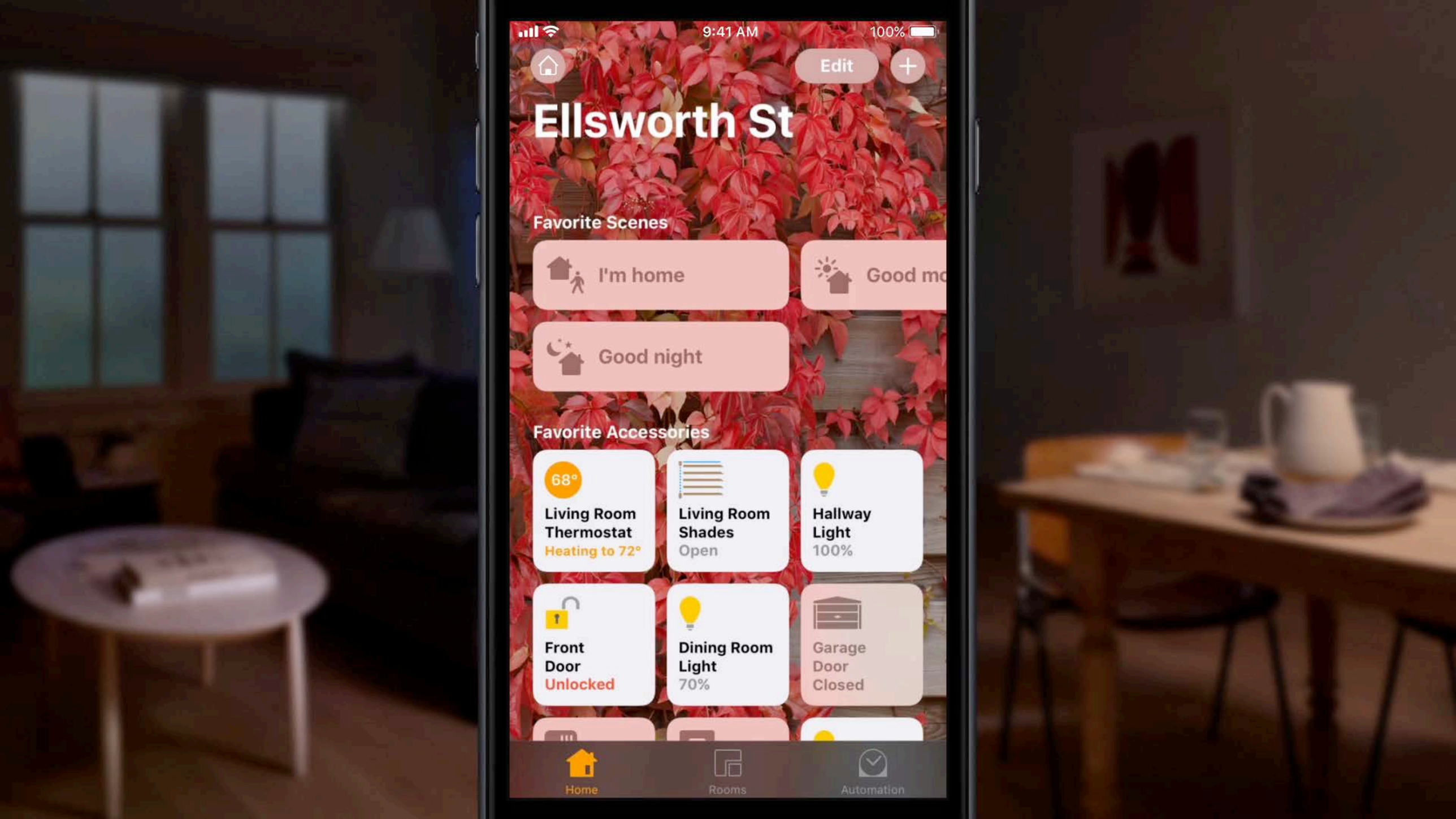

9:41 AM
100%
Edit
Ellsworth St
Favorite Scenes
I'm home
Good night
Favorite Accessories
68°
Living Room Thermostat
Living Room Shades Closed
Hallway Light
Front Door Locked
Dining Room Light
Garage Door Closed
Home
Rooms
Automation

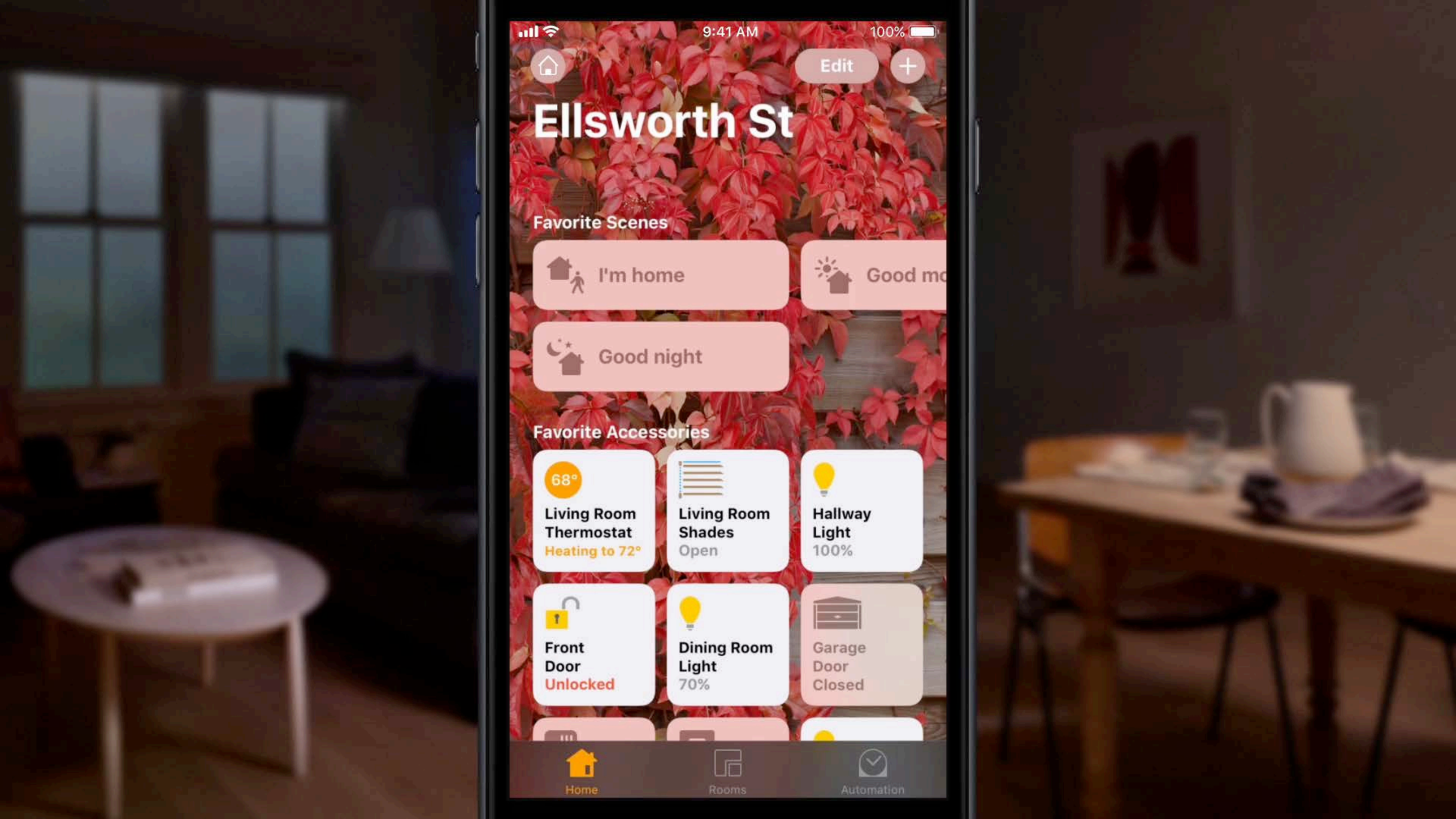

9:41 AM
100%
Edit
Ellsworth St
Favorite Scenes
I'm home
Good night
Favorite Accessories
68°
Living Room Thermostat
Living Room Shades Closed
Hallway Light
Front Door Locked
Dining Room Light
100%
Garage Door Closed
Home
Rooms
Automation

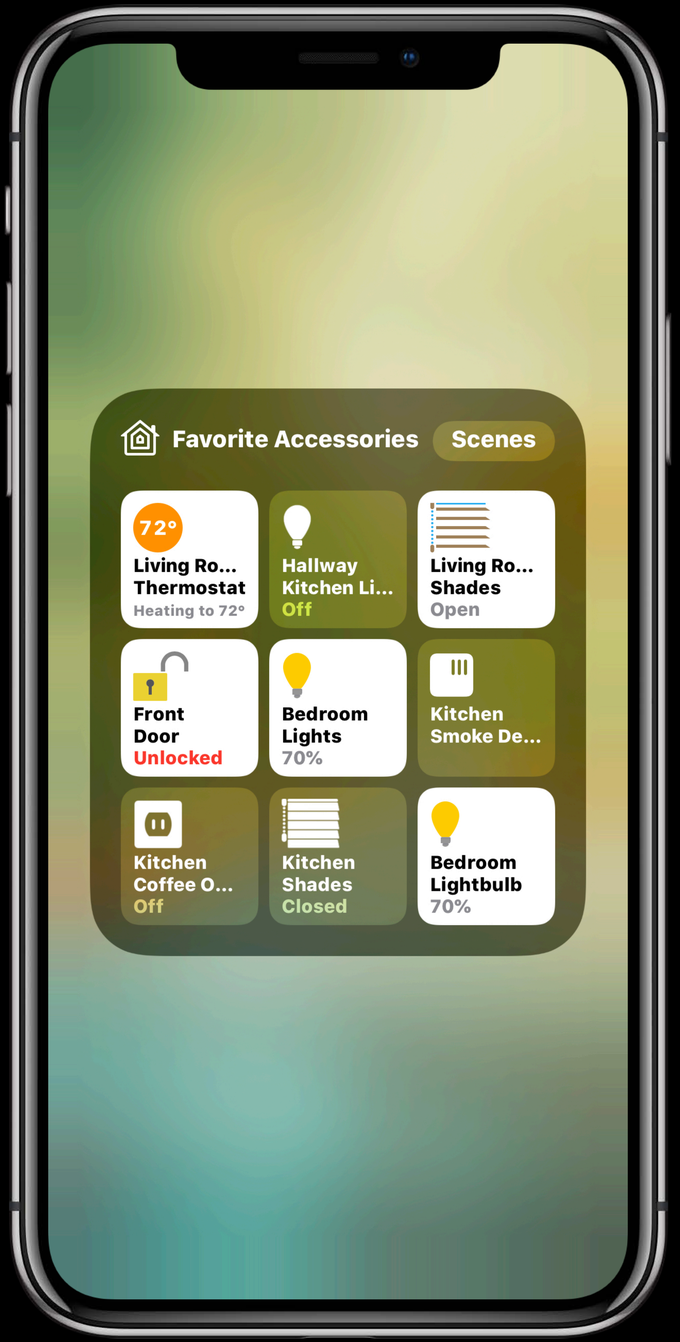

9:41 AM
100%
Edit
Ellsworth St
Favorite Scenes
I'm home
Good night
Favorite Accessories
68°
Living Room Thermostat
Living Room Shades Closed
Hallway Light
Front Door Locked
Dining Room Light
100%
Garage Door Closed
Home
Rooms
Automation


9:41 AM
100%
Edit
Ellsworth St
Favorite Scenes
I'm home
Good night
Favorite Accessories
68°
Living Room Thermostat
Living Room Shades Closed
Hallway Light
Front Door Locked
Dining Room Light
70%
Garage Door Closed
Home
Rooms
Automation

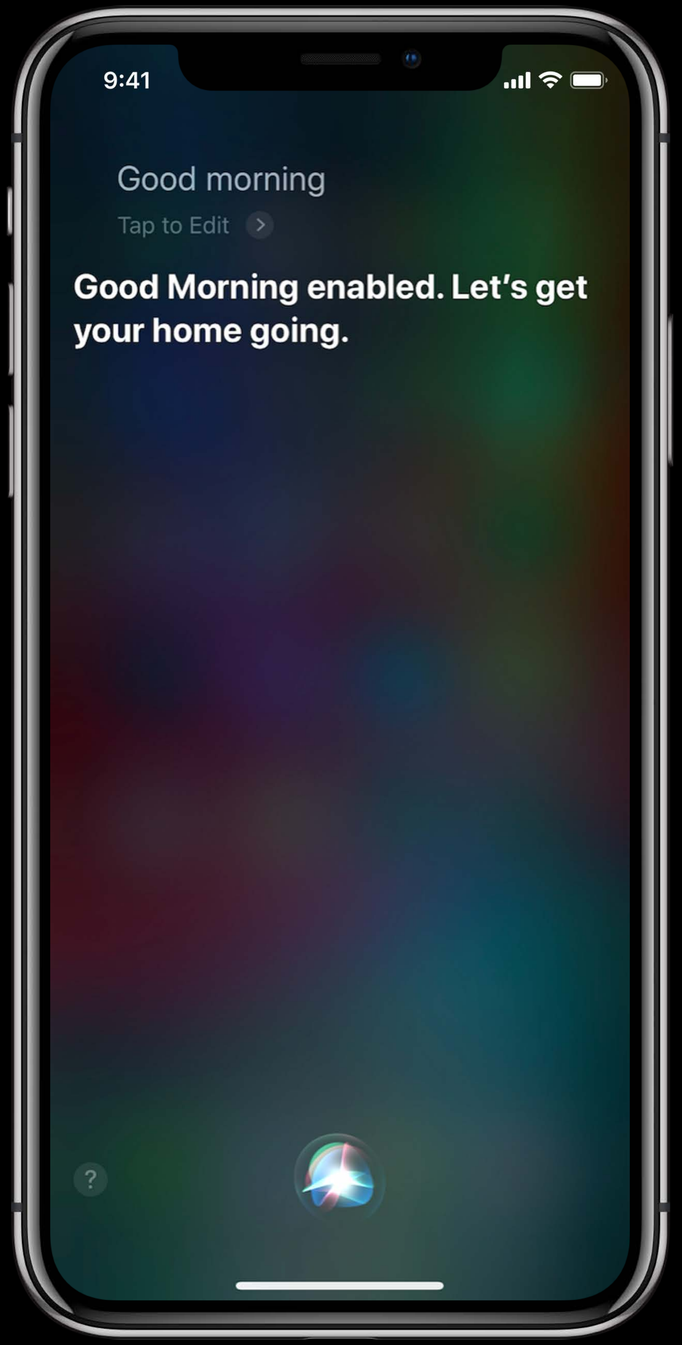

9:41 AM
100%
Edit
Ellsworth St
Favorite Scenes
I'm home
Good night
Favorite Accessories
68°
Living Room Thermostat
Living Room Shades Closed
Hallway Light
Front Door Locked
Dining Room Light
70%
Garage Door Closed
Home
Rooms
Automation


9:41 AM
100%
Edit
Ellsworth St
Favorite Scenes
I'm home
Good night
Favorite Accessories
68°
Living Room Thermostat
Heating to 72°
Living Room Shades
Open
Hallway Light
100%
Front Door
Unlocked
Dining Room Light
70%
Garage Door
Closed
Home
Rooms
Automation

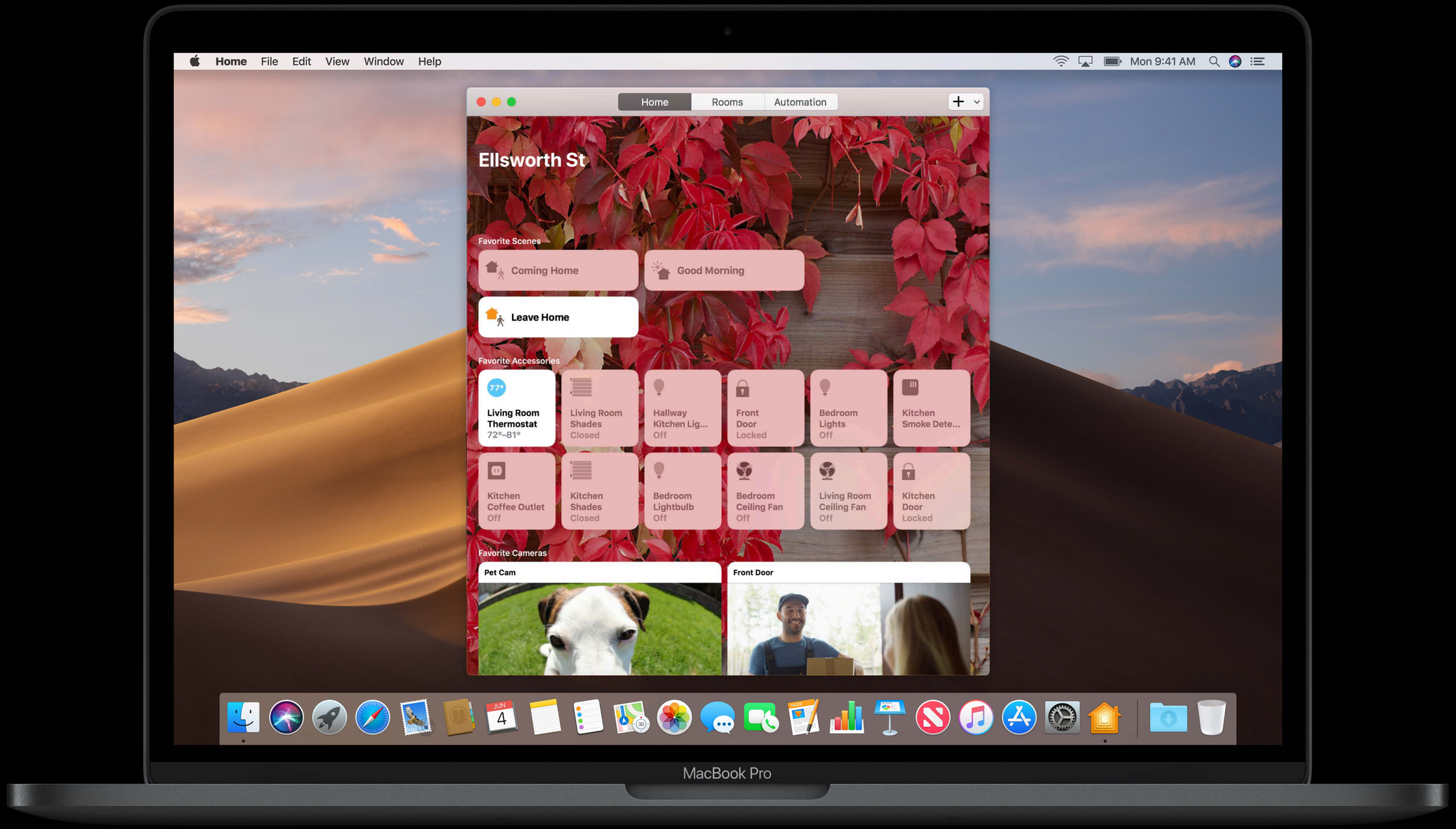

9:41 AM
100%
Edit
Ellsworth St
Favorite Scenes
I'm home
Good night
Favorite Accessories
68°
Living Room Thermostat
Heating to 72°
Living Room Shades
Open
Hallway Light
100%
Front Door
Unlocked
Dining Room Light
70%
Garage Door
Closed
Home
Rooms
Automation

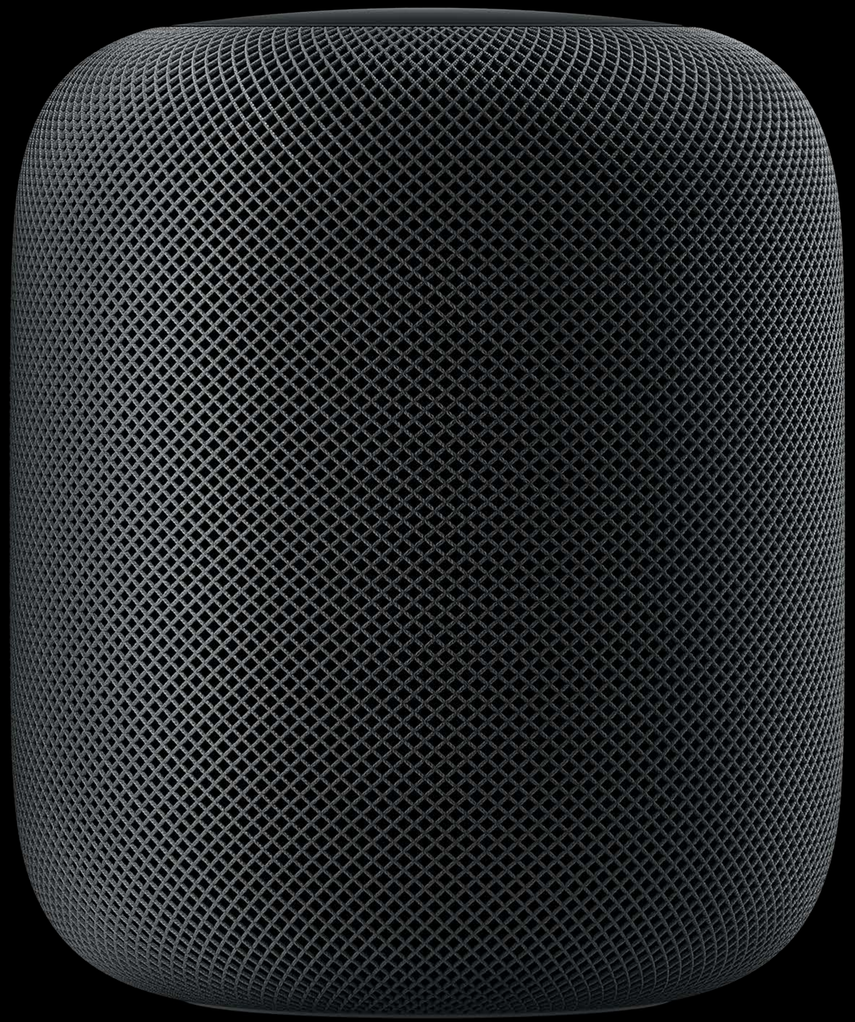


Quick controls



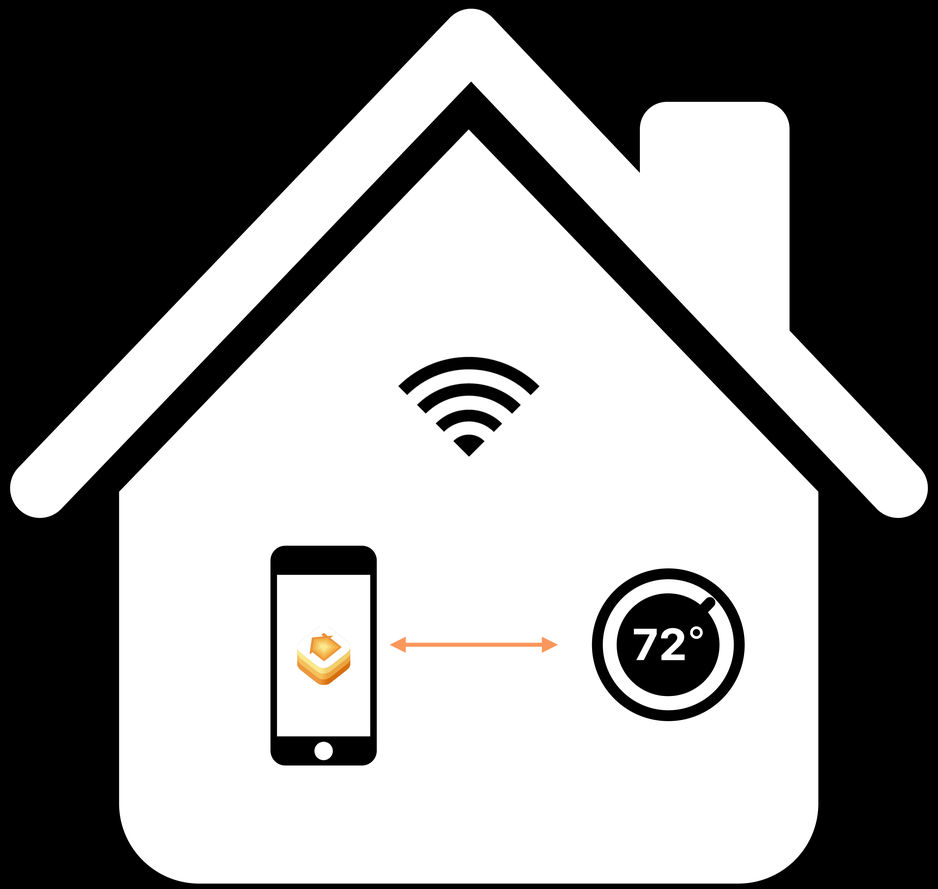
Siri voice control


Home File Edit View Window Help
Mon 9:41 AM
Home
Rooms
Automation
Ellsworth St
Favorite Scenes
Coming Home
Good Morning
Leave Home
Favorite Accessories
77°
Living Room Thermostat
72°–81°
Living Room Shades
Closed
Hallway Kitchen Lig...
Off
Front Door
Locked
Bedroom Lights
Off
Kitchen Smoke Dete...
Kitchen Coffee Outlet
Off
Kitchen Shades
Closed
Bedroom Lightbulb
Off
Bedroom Ceiling Fan
Off
Living Room Ceiling Fan
Off
Kitchen Door
Locked
Favorite Cameras
Pet Cam
Front Door
MacBook Pro



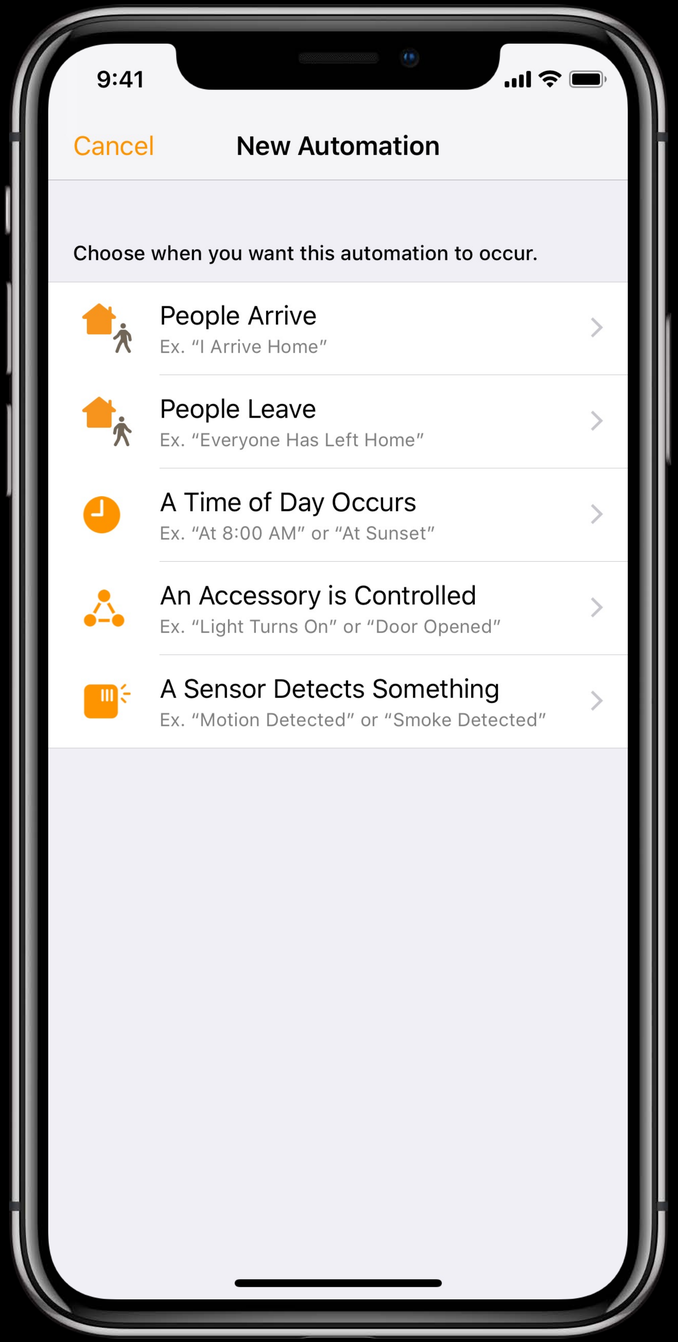
Remote access and automation

Remote Access



Remote Access

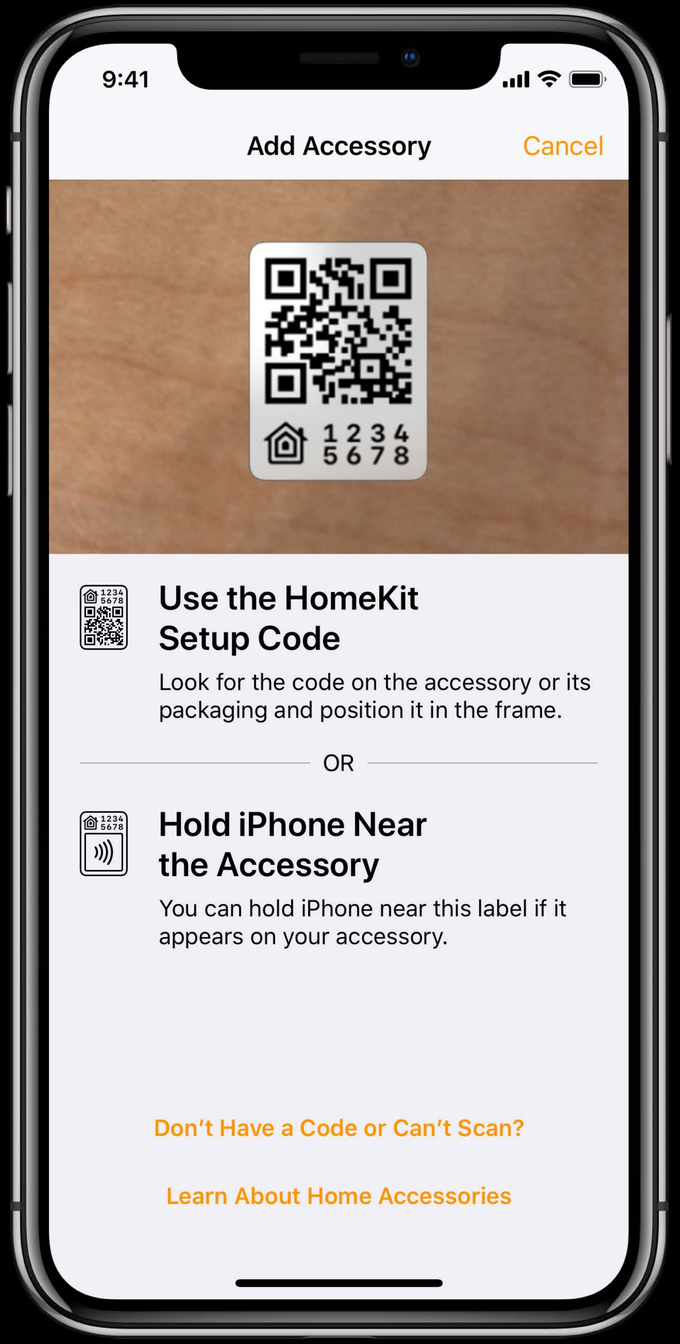
Remote Access



Automations

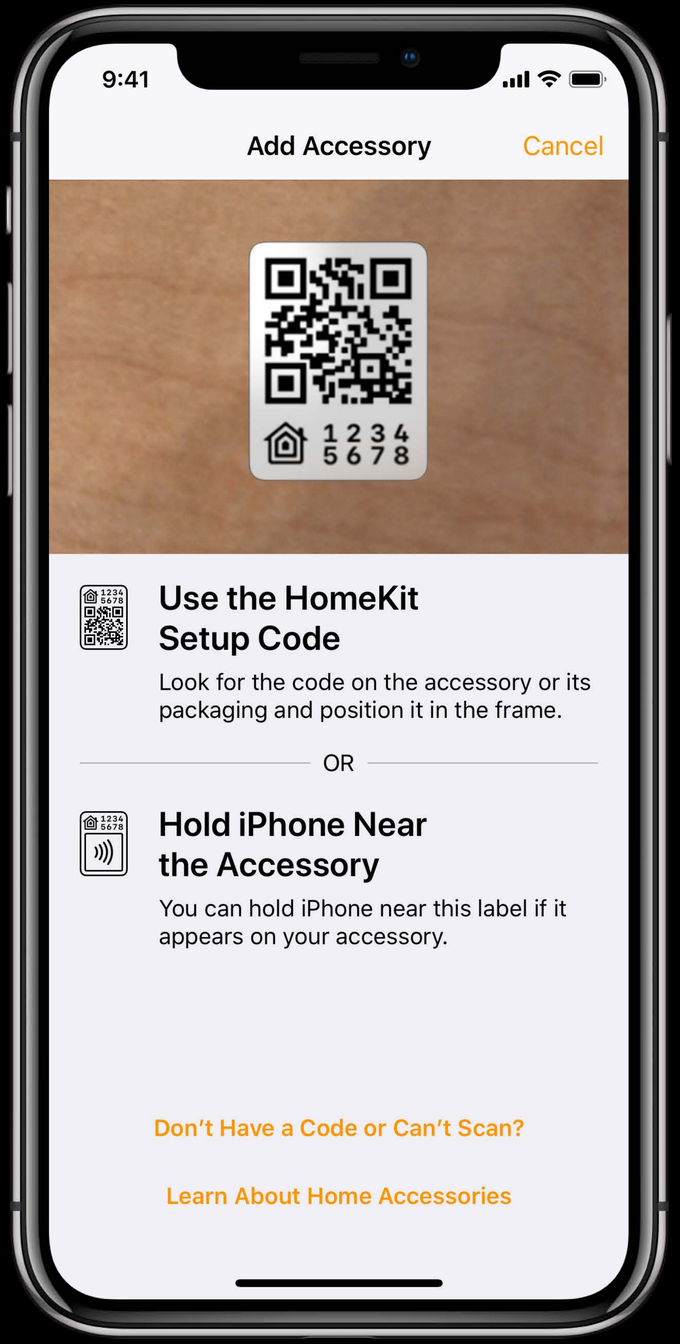

APR
5VCV195

Accessory Setup



Accessory Setup

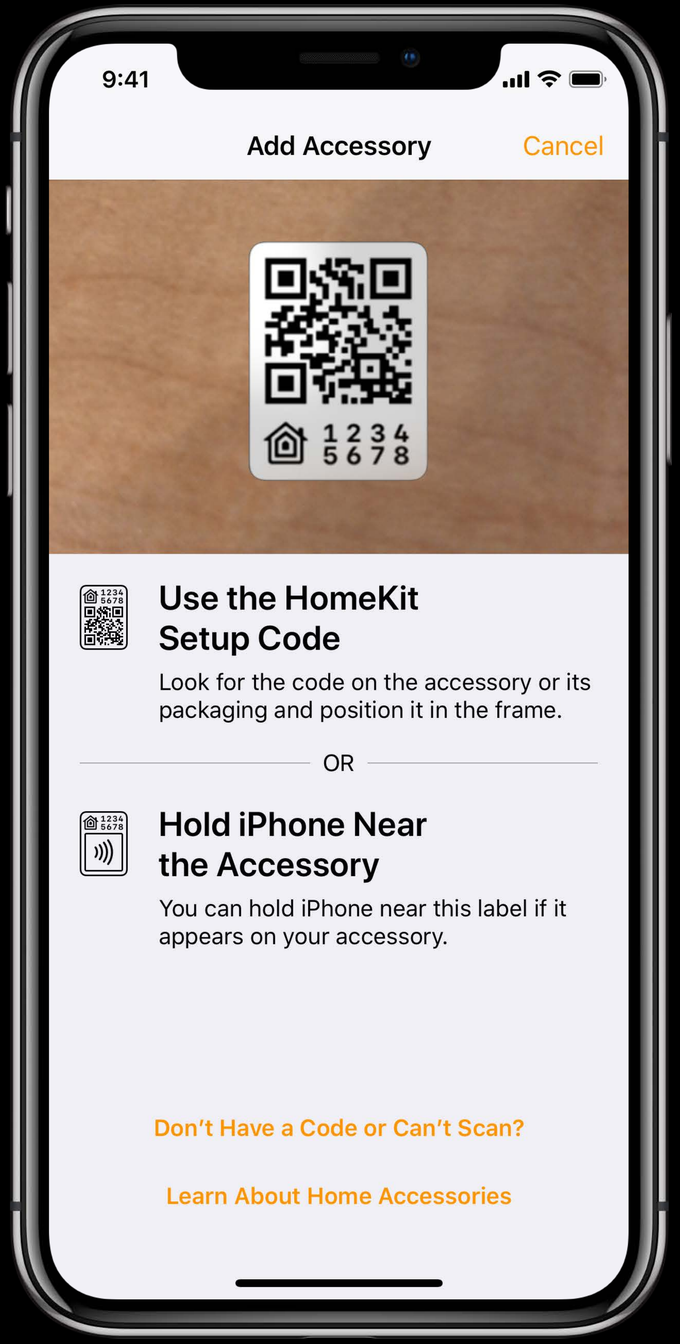
Seamless network joining



Accessory Setup

Seamless network joining

QR code scanning

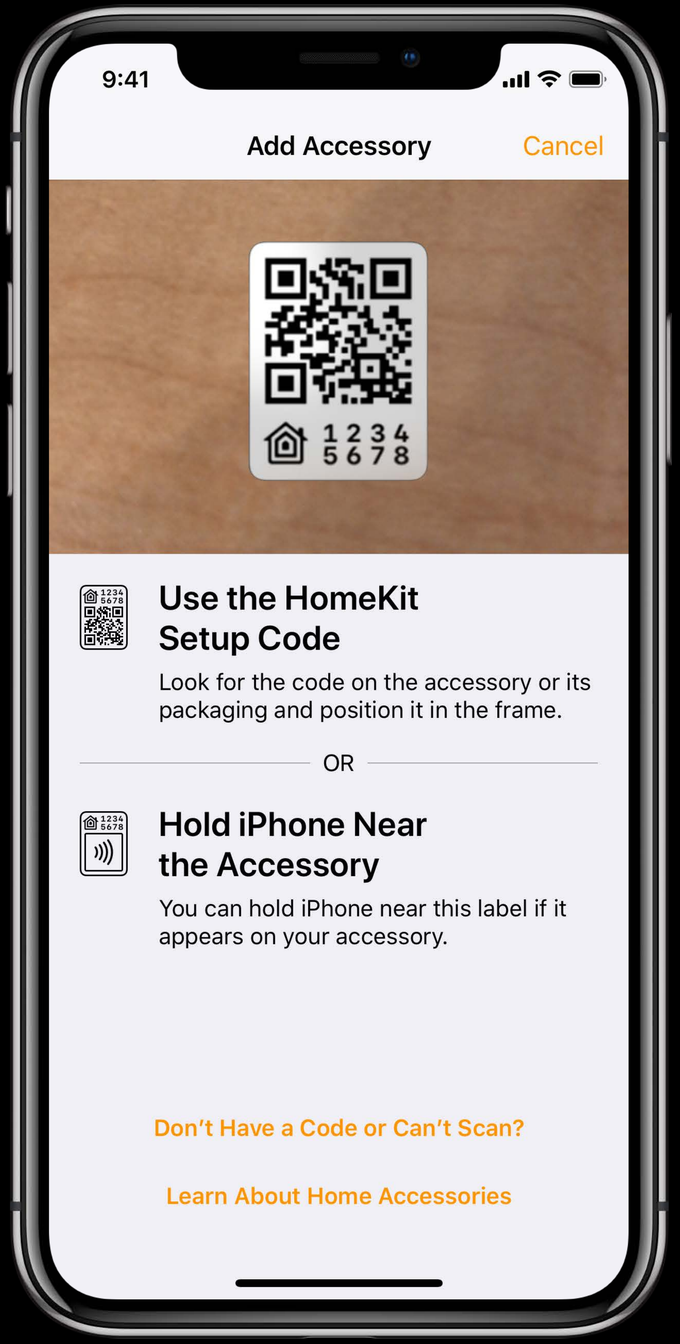


Accessory Setup

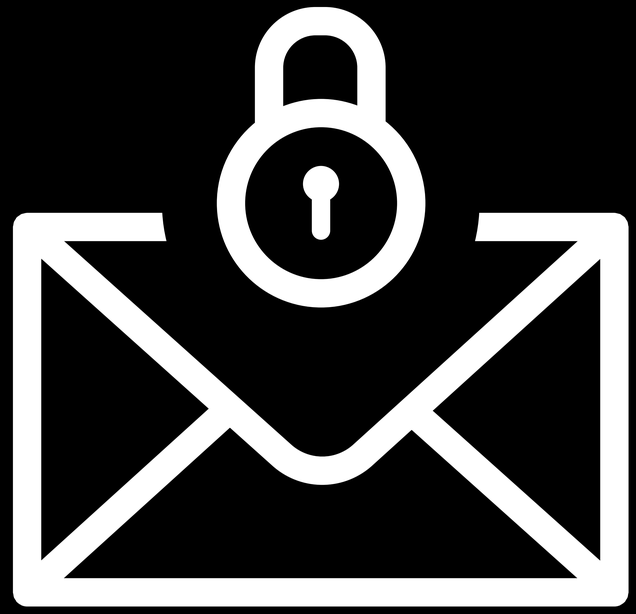
Seamless network joining

QR code scanning

NFC

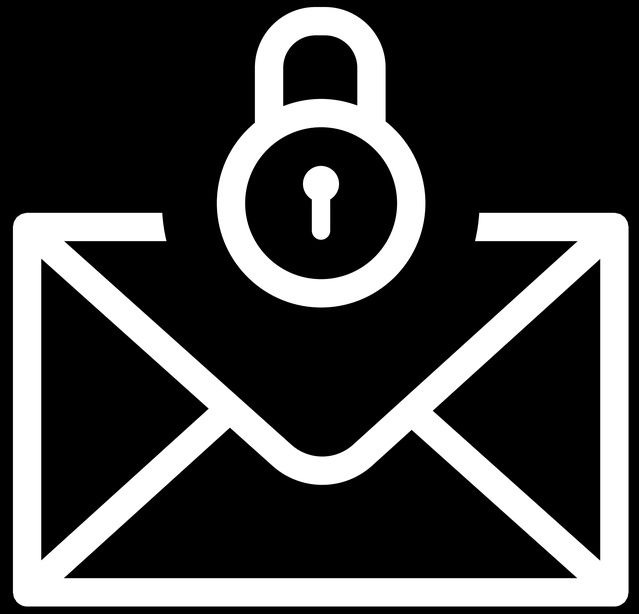


Security and Privacy



Security and Privacy

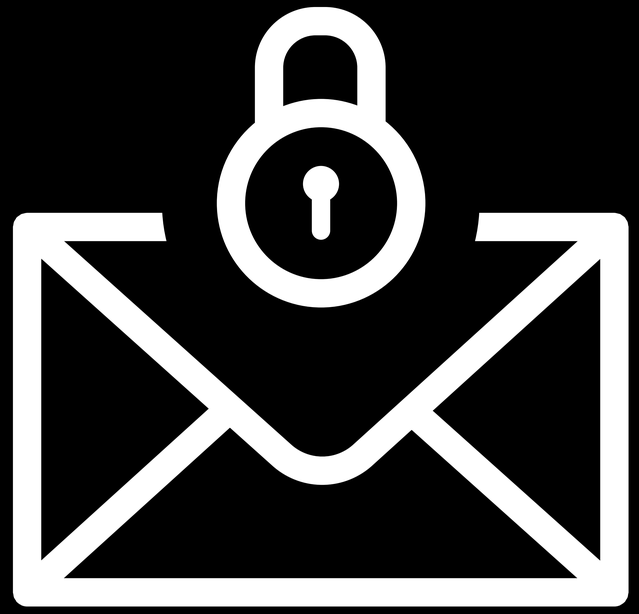
Encrypted communication



Security and Privacy

Encrypted communication

Perfect forward secrecy

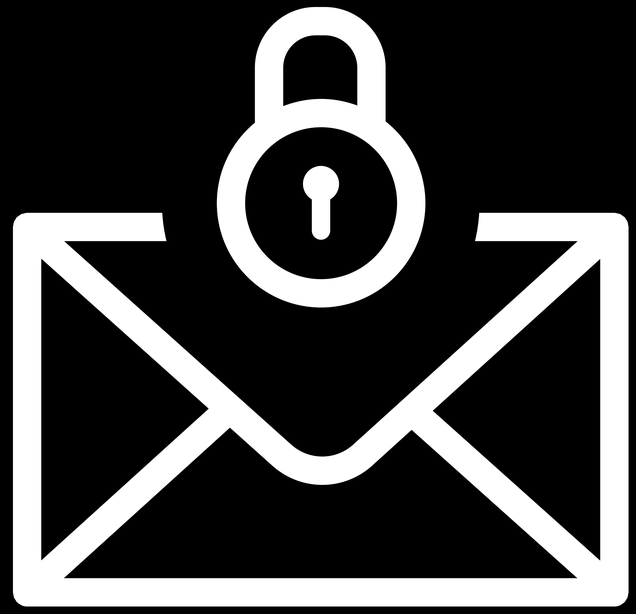


Security and Privacy

Encrypted communication

Perfect forward secrecy

All data is private



HomeKit Accessory Protocol

HomeKit Accessory Protocol

Communication channel to accessory

HomeKit Accessory Protocol

Communication channel to accessory

Secure messaging

HomeKit Accessory Protocol

Communication channel to accessory

Secure messaging

IP and Bluetooth LE transport

HomeKit Accessory Protocol

Communication channel to accessory

Secure messaging

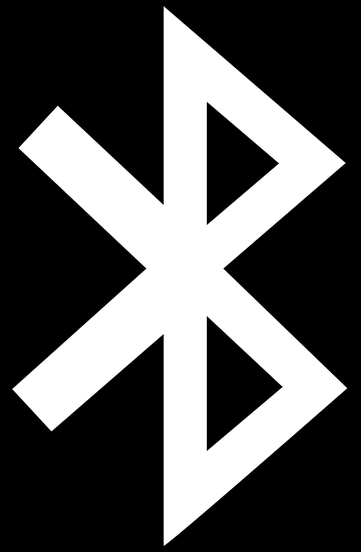
IP and Bluetooth LE transport

HomeKit Accessory Protocol

Communication channel to accessory

Secure messaging

IP and Bluetooth LE transport



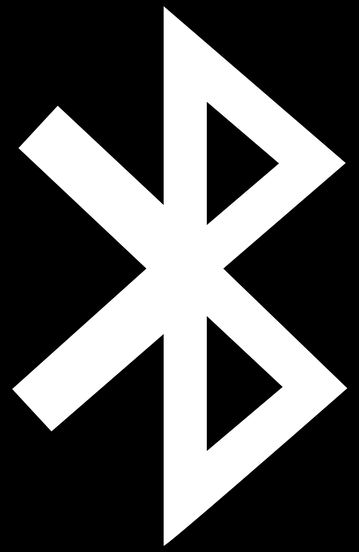
HomeKit Accessory Protocol

Communication channel to accessory

Secure messaging

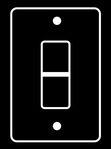
IP and Bluetooth LE transport

Accessory categories



Accessory Categories

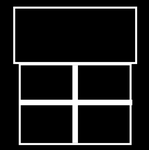
Lights



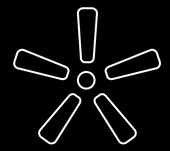
Switches

Outlets

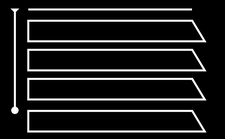
Thermostats



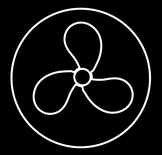
Windows



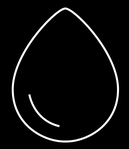
Fans



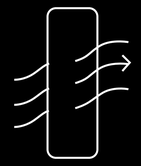
Blinds



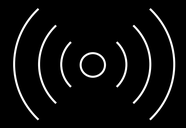
Air Conditioners



Humidifiers

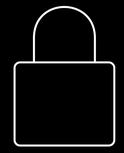


Air Purifiers



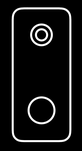
Sensors

Security



Locks

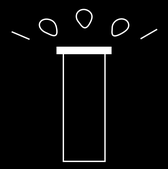
Cameras



Doorbells

Garage Doors

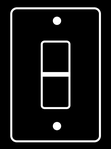
Bridges



Water Valves

Accessory Categories

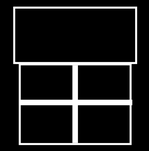
Lights



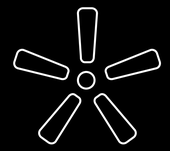
Switches

Outlets

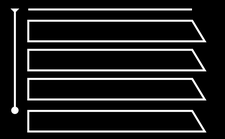
Thermostats



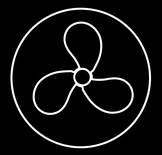
Windows



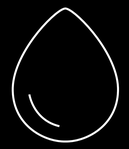
Fans



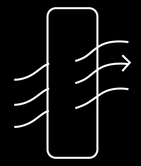
Blinds



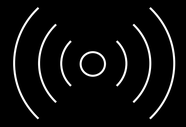
Air Conditioners



Humidifiers

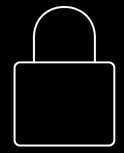


Air Purifiers



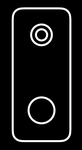
Sensors

Security



Locks

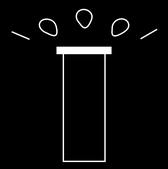
Cameras



Doorbells

Garage Doors

Bridges

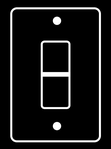


Water Valves

Speakers

Accessory Categories

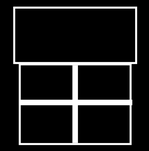
Lights



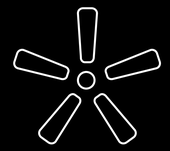
Switches

Outlets

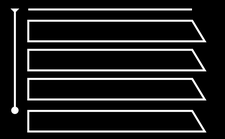
Thermostats



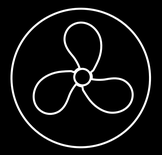
Windows



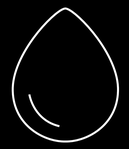
Fans



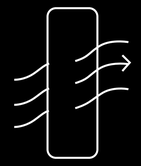
Blinds



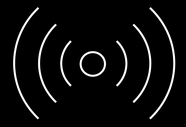
Air Conditioners



Humidifiers

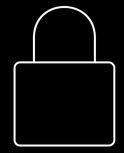


Air Purifiers



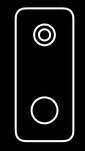
Sensors

Security



Locks

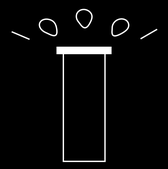
Cameras



Doorbells

Garage Doors

Bridges



Water Valves

Speakers

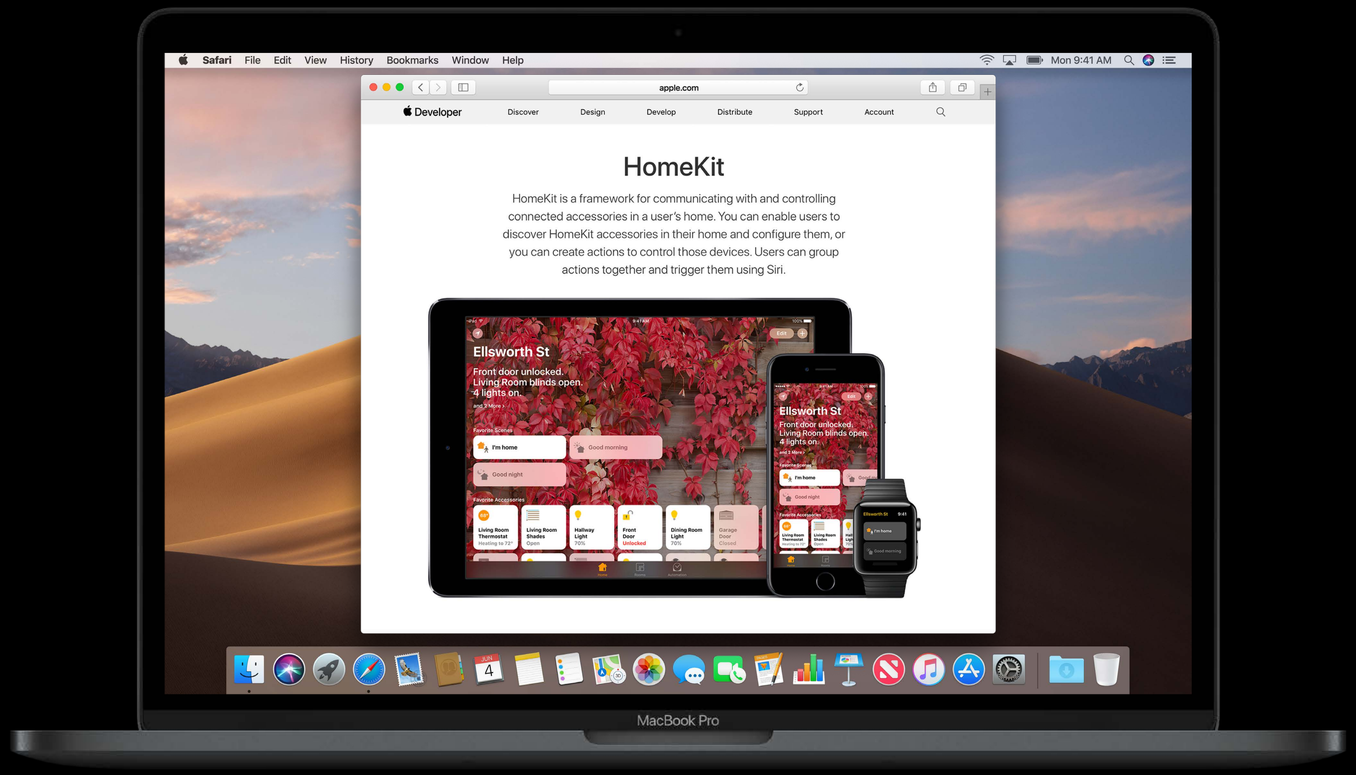
Remotes

HomeKit overview

Building an accessory

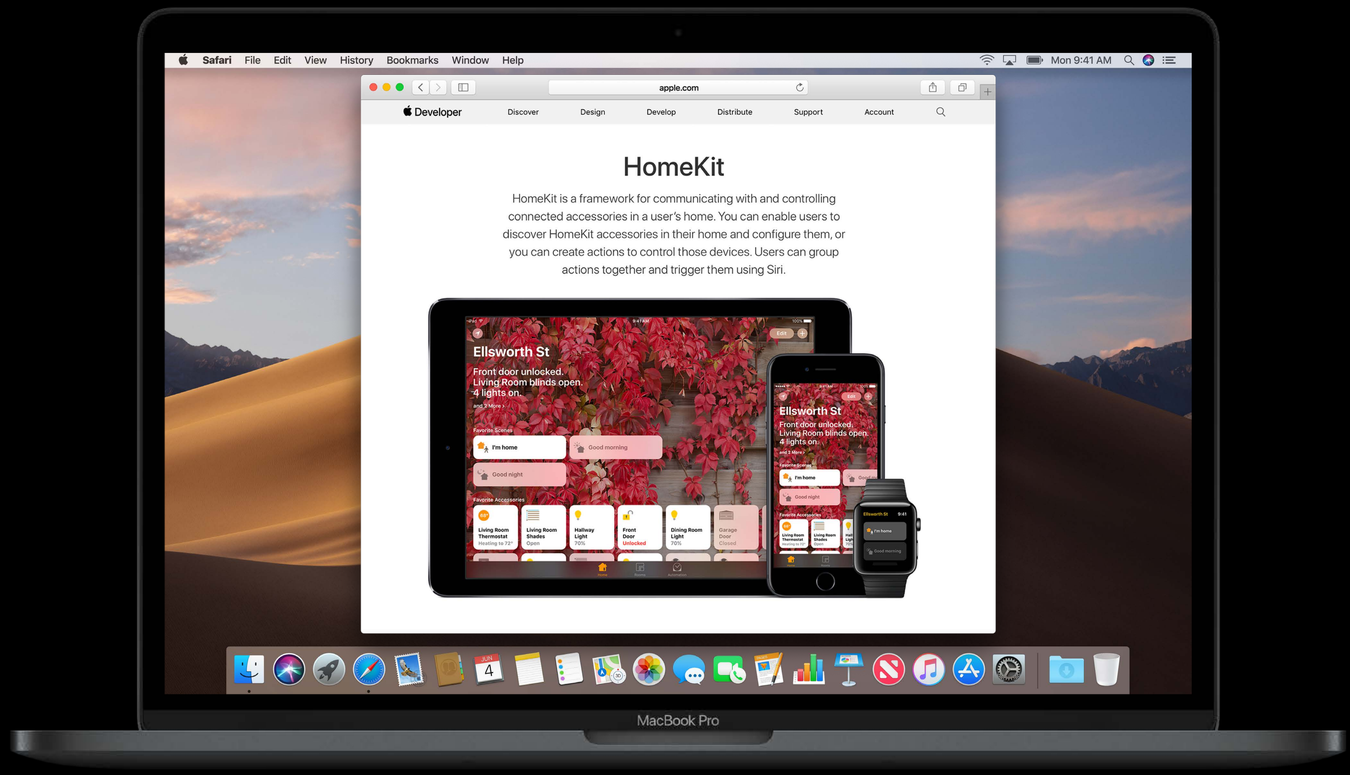
Creating a HomeKit app

HomeKit Accessory Protocol Specification



HomeKit Accessory Protocol Specification

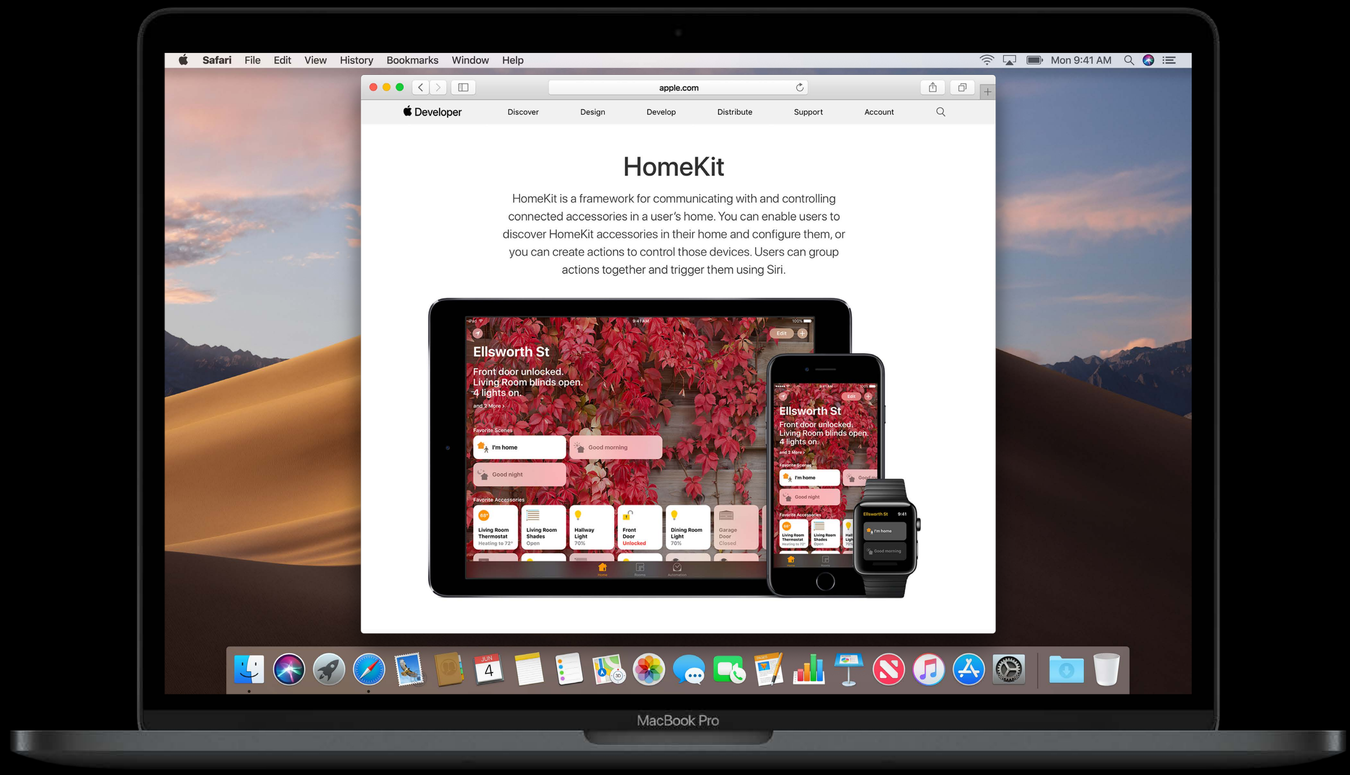
Become MFi Licensee
http://developer.apple.com/mfi



HomeKit Accessory Protocol Specification

Become MFi Licensee
http://developer.apple.com/mfi

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



Resources for MFi Partners

Resources for MFi Partners

HomeKit Accessory Development Kit

Resources for MFi Partners

HomeKit Accessory Development Kit

Certification tools

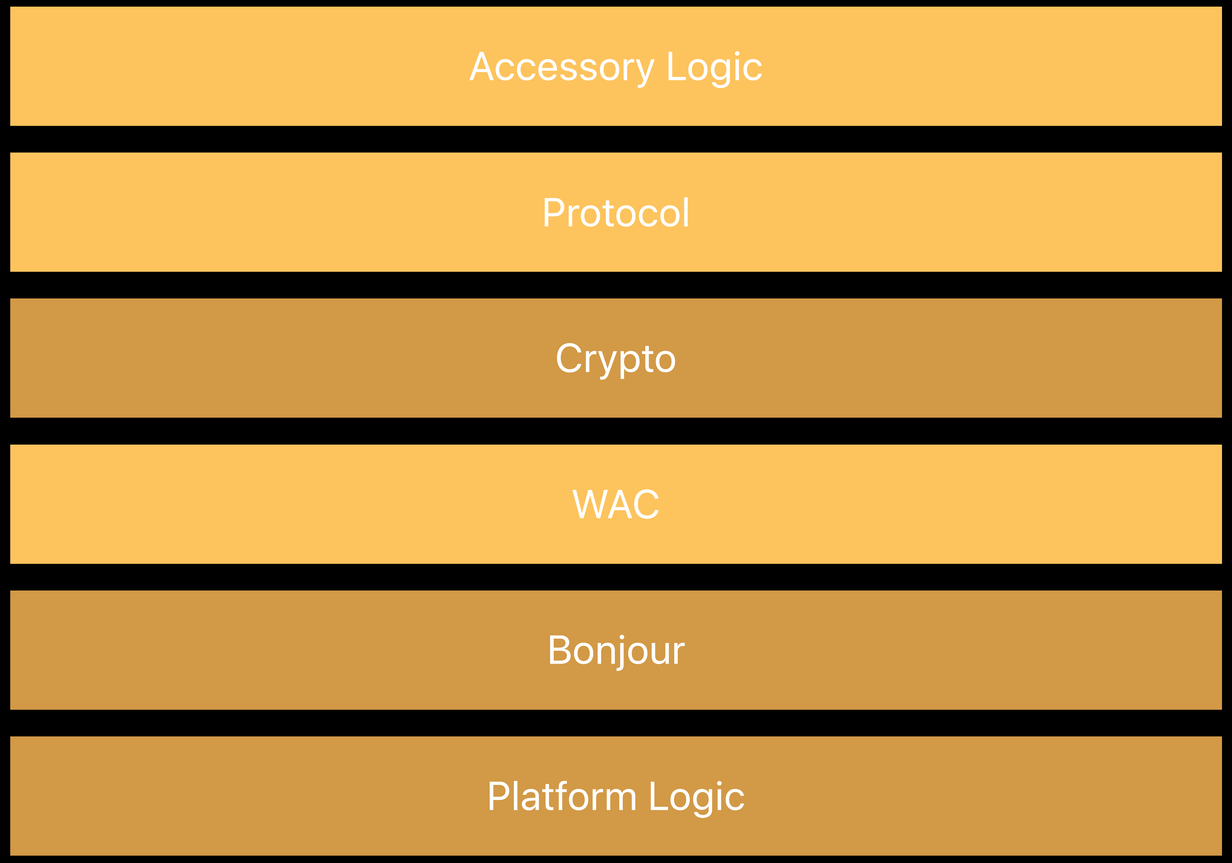
Resources for MFi Partners

HomeKit Accessory Development Kit

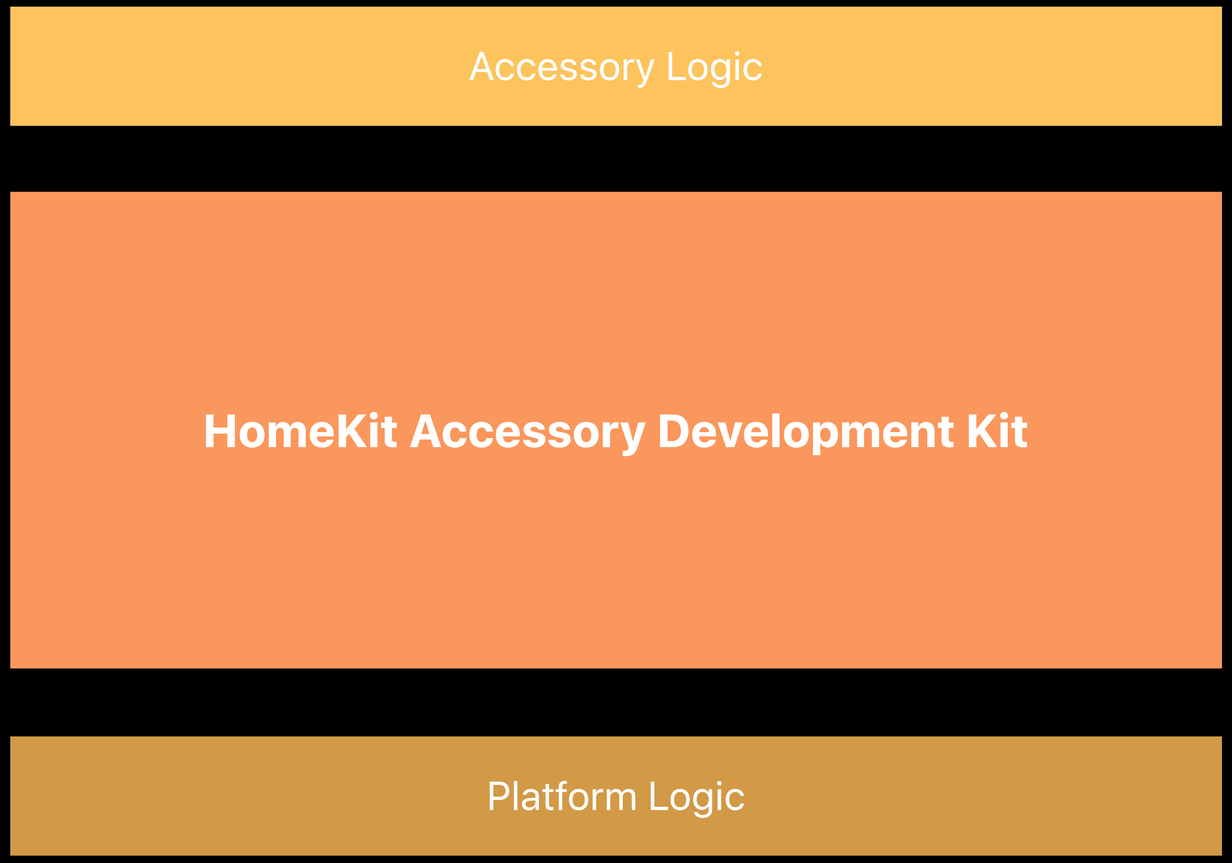
Certification tools

Authentication

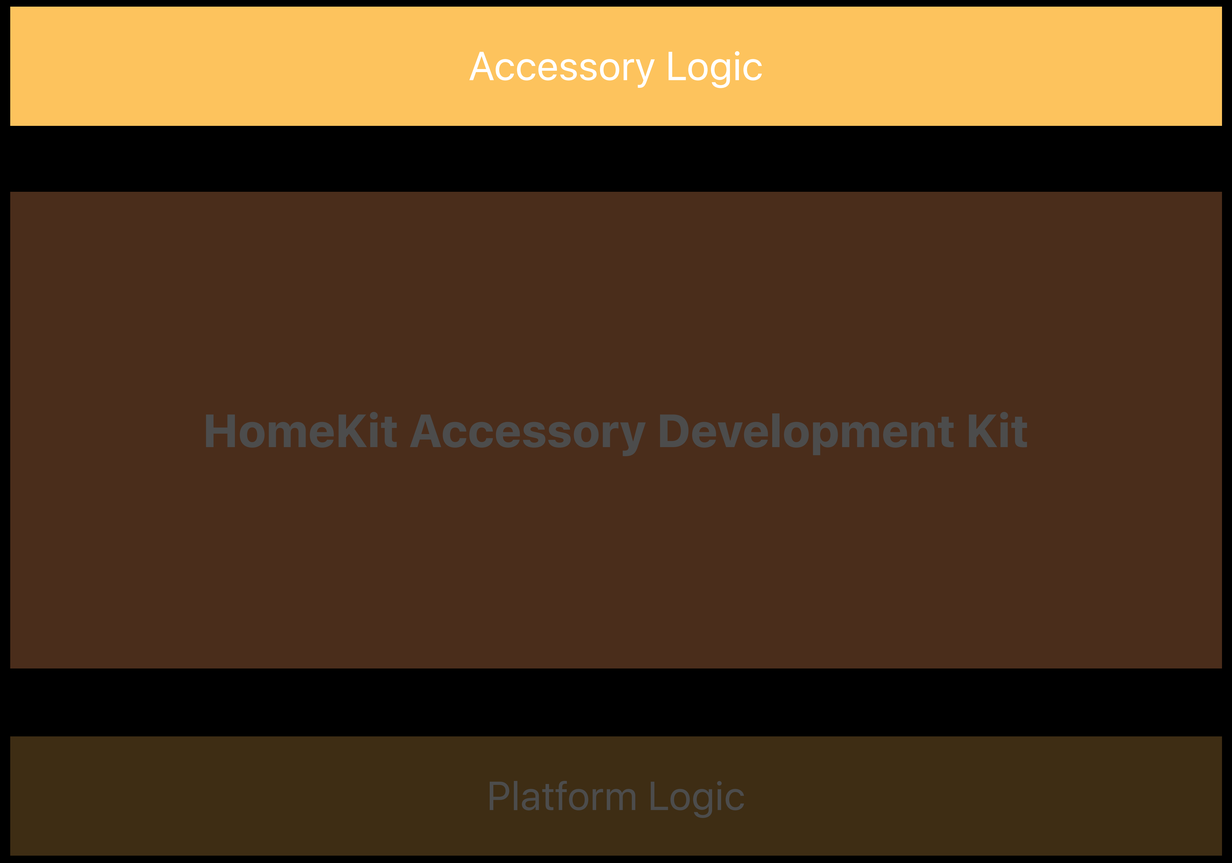
HomeKit Accessory Development Kit



HomeKit Accessory Development Kit



HomeKit Accessory Development Kit



HomeKit Accessory Development Kit

Why use the ADK?

HomeKit Accessory Development Kit

Why use the ADK?

Easier to get started

HomeKit Accessory Development Kit

Why use the ADK?

Easier to get started

Faster to integrate

HomeKit Accessory Development Kit

Why use the ADK?

Easier to get started

Faster to integrate

Reliable and secure

HomeKit Accessory Development Kit

Why use the ADK?

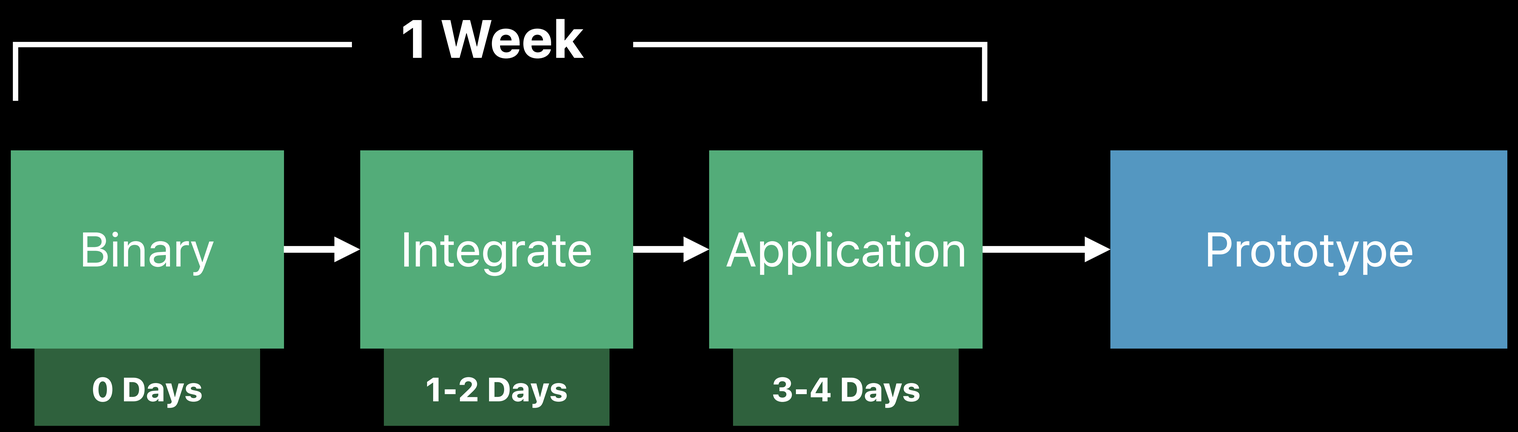
Easier to get started

Faster to integrate

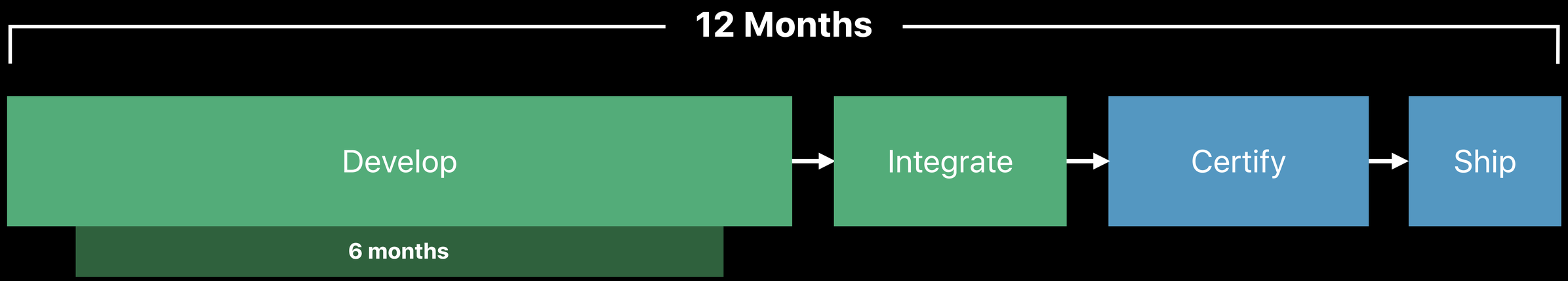
Reliable and secure

Stay in sync with iOS

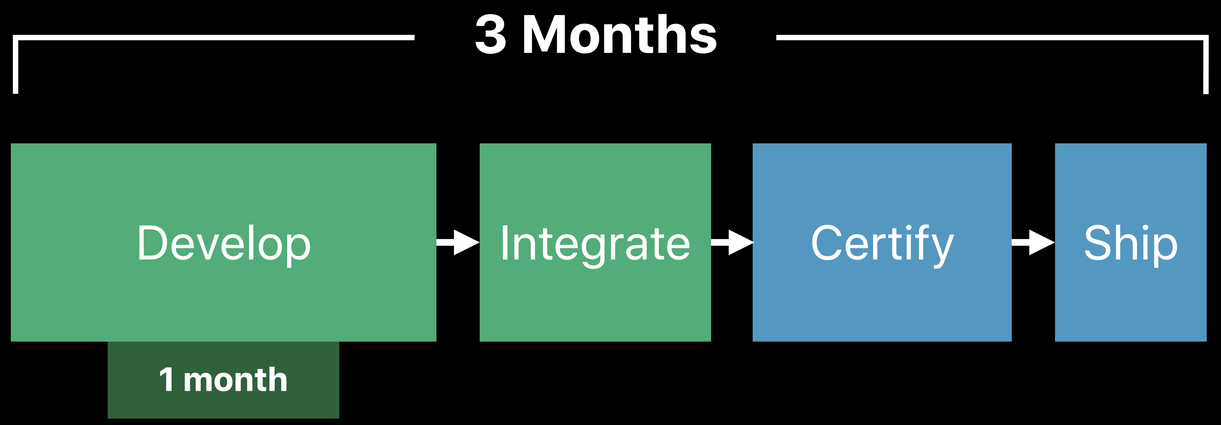
HomeKit Accessory Development Kit



HomeKit Accessory Development Kit



HomeKit Accessory Development Kit



Accessory Authentication

Accessory Authentication

For MFi partners

Accessory Authentication

For MFi partners

Consumer trusted

Accessory Authentication

For MFi partners

Consumer trusted

Accessory Authentication

For MFi partners

Consumer trusted

Quality certified

Accessory Authentication

For MFi partners

Consumer trusted

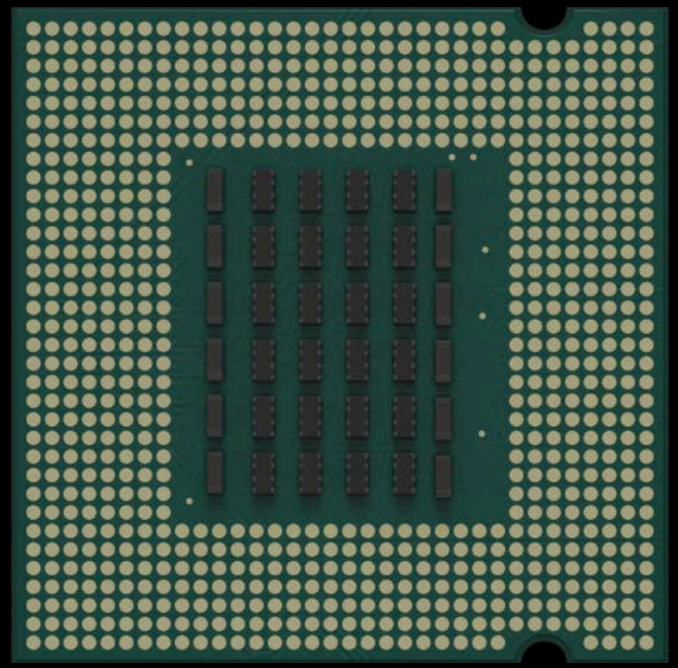
Quality certified

Hardware and software based

Authentication

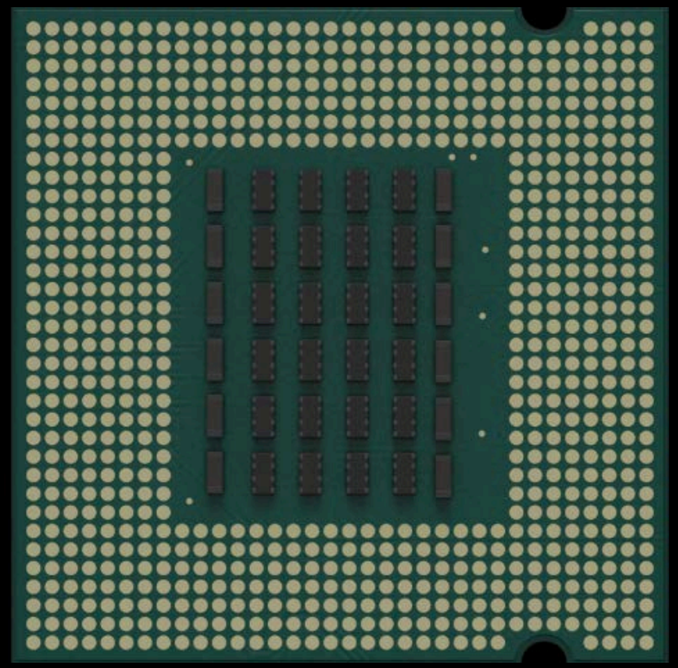
Authentication

Hardware

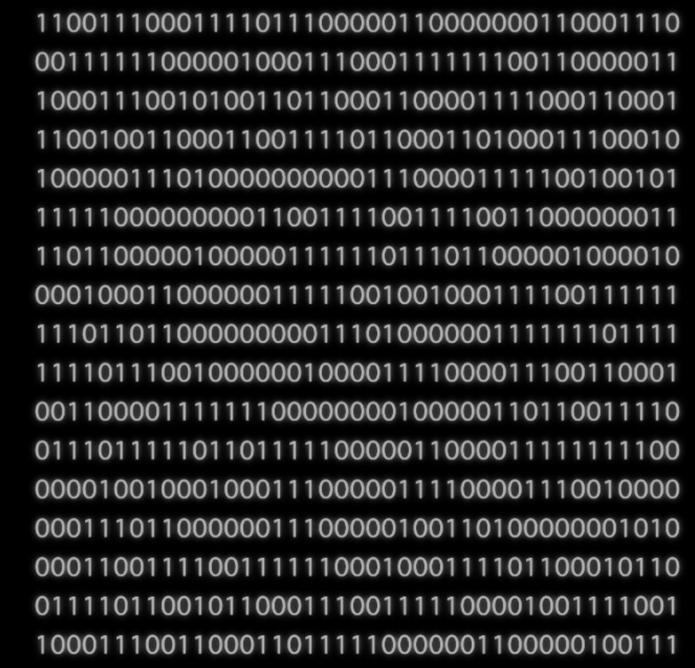


Authentication

Hardware



Software



HomeKit Accessory Tester

HomeKit Accessory Tester

Available for MFi licensees

HomeKit Accessory Tester

Available for MFi licensees

Application interface

HomeKit Accessory Tester

Available for MFi licensees

Application interface

Great for testing your accessory

HomeKit Certification Assistant

HomeKit Certification Assistant

Available for MFi licensees

HomeKit Certification Assistant

Available for MFi licensees

Self-certification tool

HomeKit Certification Assistant

Available for MFi licensees

Self-certification tool

Automated test execution

HomeKit Accessory Simulator

HomeKit Accessory Simulator

Simulate any HomeKit accessory

HomeKit Accessory Simulator

Simulate any HomeKit accessory

Great for testing your app

HomeKit Accessory Simulator

Simulate any HomeKit accessory

Great for testing your app

No physical accessories required

HomeKit Accessory Simulator

Simulate any HomeKit accessory

Great for testing your app

No physical accessories required

Available via Hardware IO Tools for Xcode

HomeKit overview

Building an accessory

Creating a HomeKit app

Making Your App Unique

Making Your App Unique

Custom functionality

Making Your App Unique

Custom functionality

Region-specific use cases

Making Your App Unique

Custom functionality

Region-specific use cases

Custom services

Home Object Hierarchy

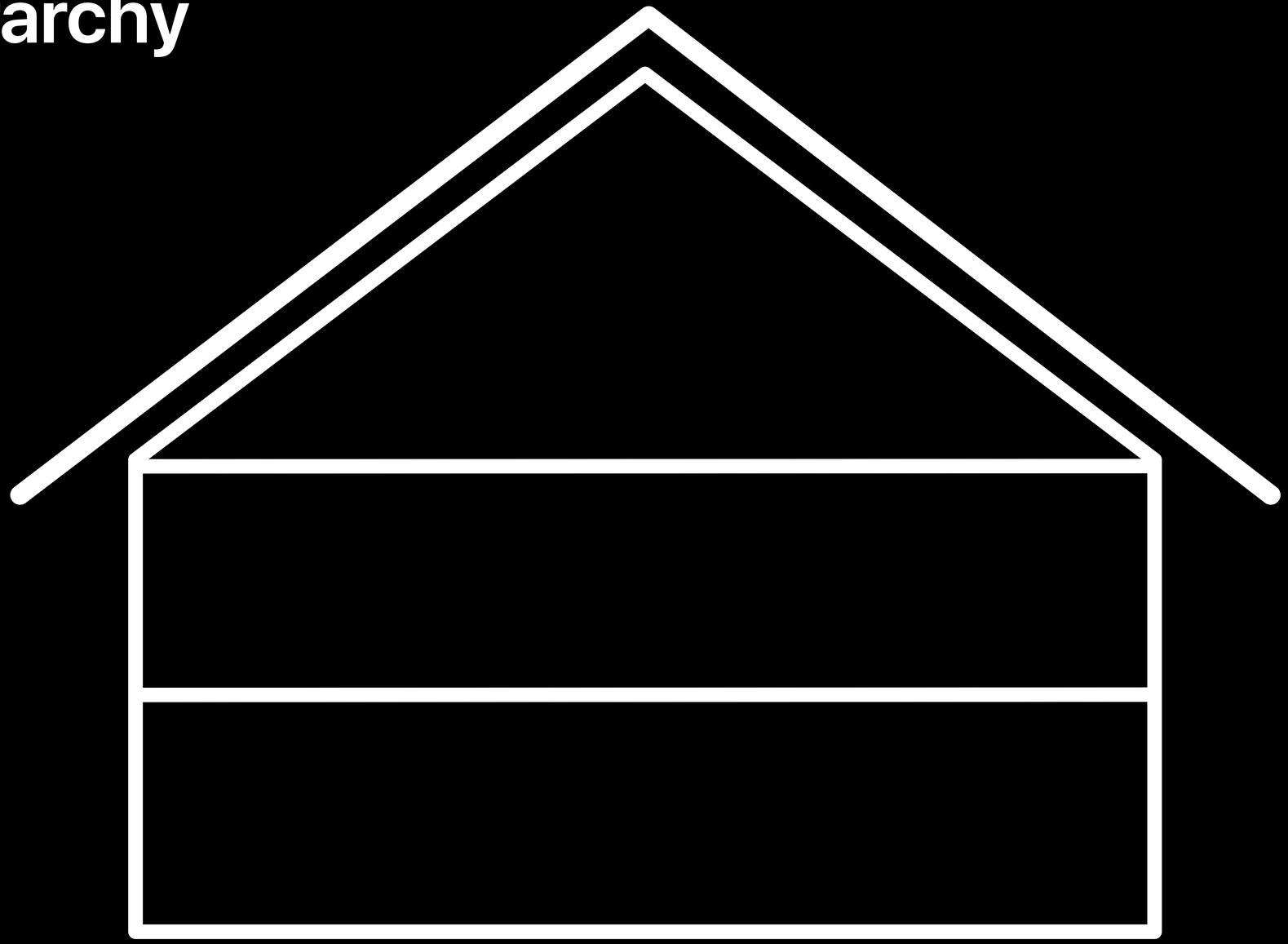
Home Object Hierarchy

HMHome

Home Object Hierarchy

HMHome

HMZone

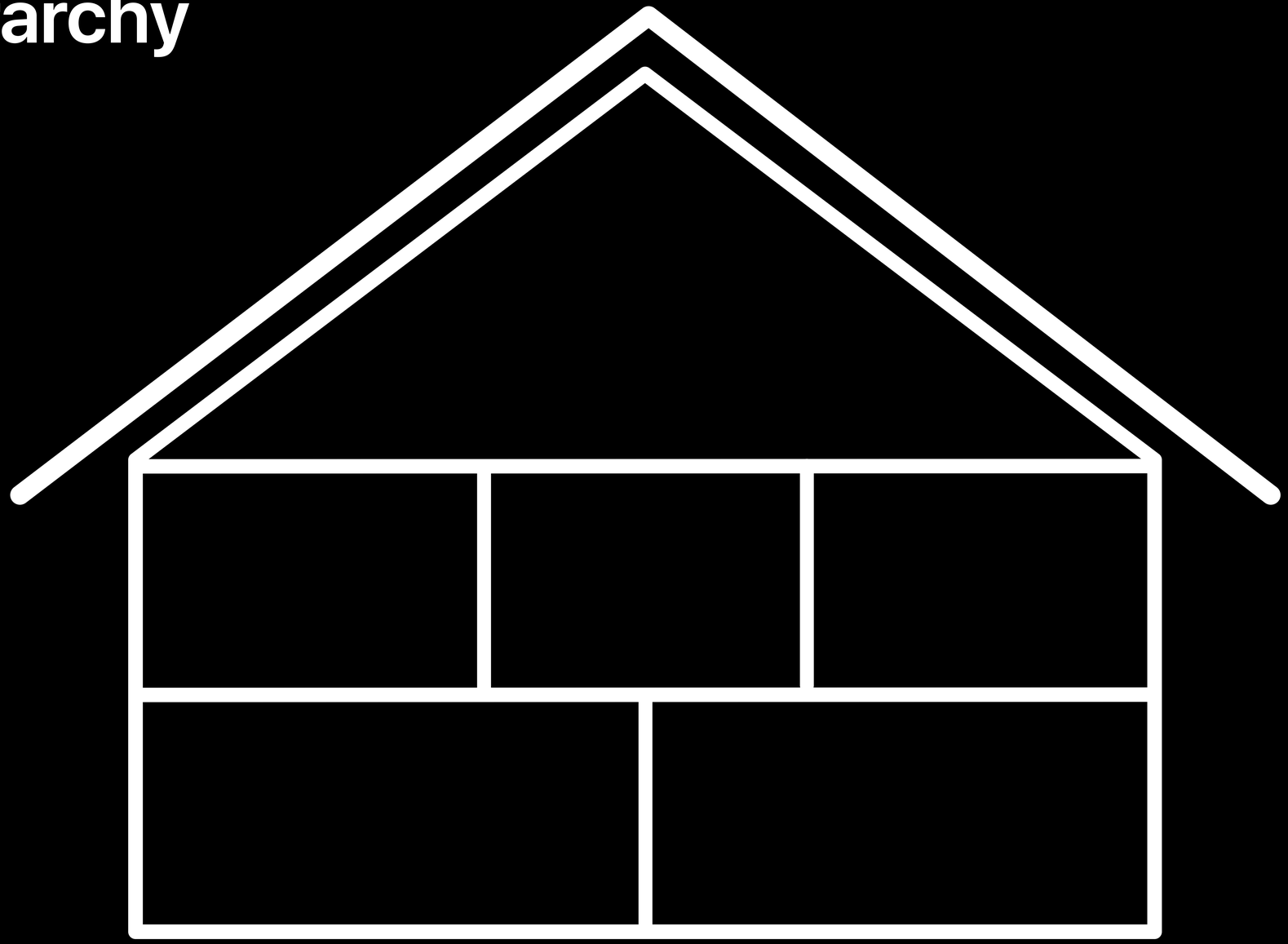


Home Object Hierarchy

HMHome

HMZone

HMRoom



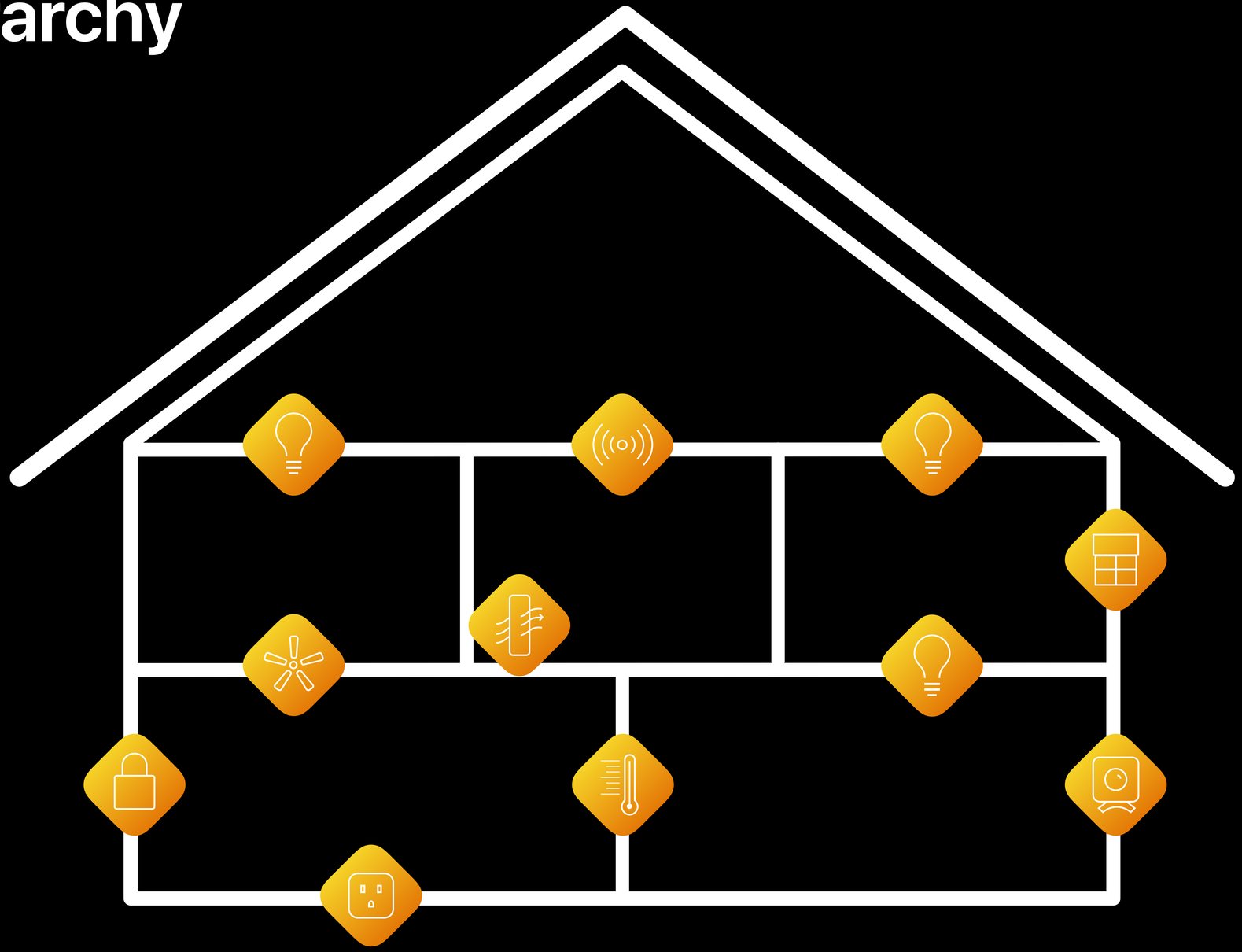
Home Object Hierarchy

HMHome

HMZone

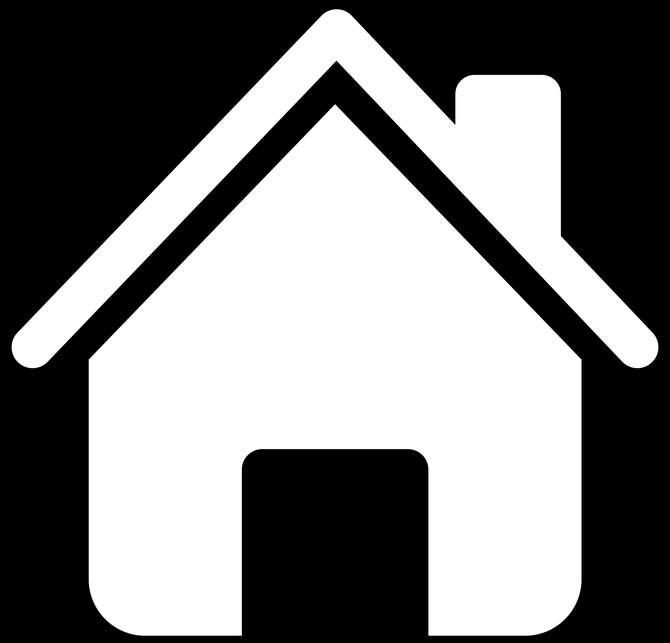
HMRoom

HMAccessory



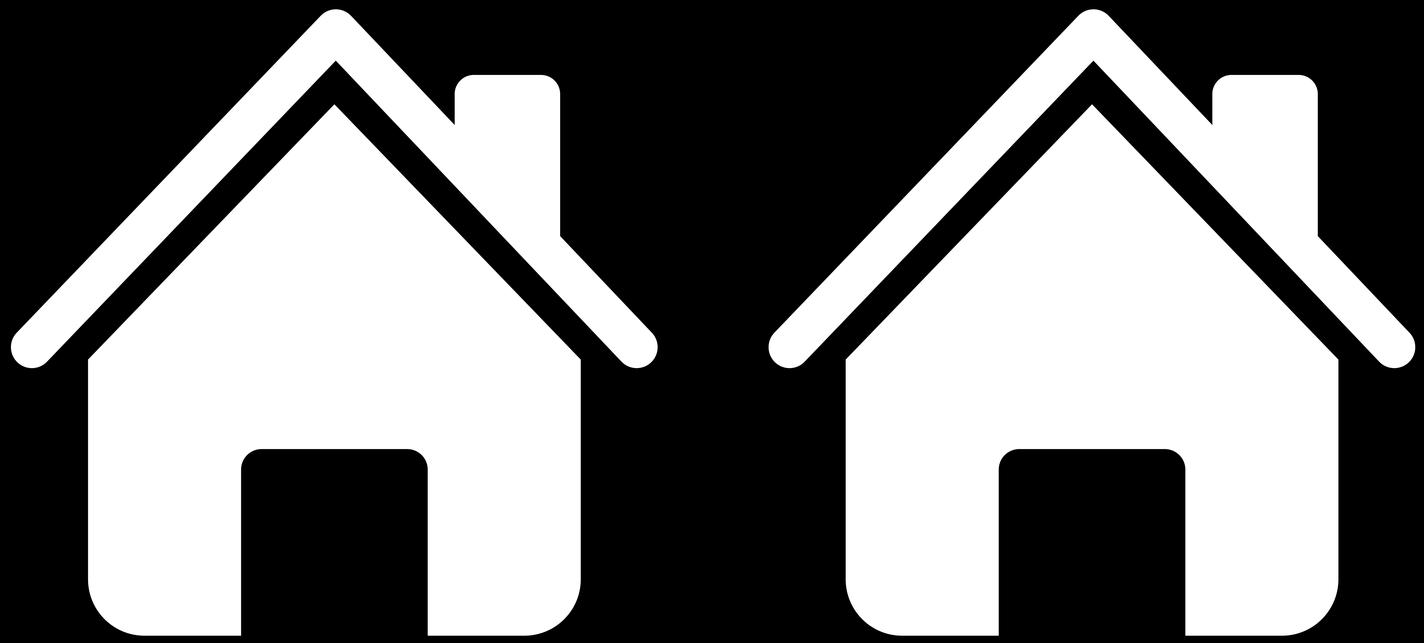
Home Management

HMHome, HMHomeManager



Home Management

HMHome, HMHomeManager



User Management

HMHome

```
extension HMHome {
  var currentUser: HMUser { get }

  func manageUsers(completionHandler: @escaping (Error?) -> Void)
  func homeAccessControl(for user: HMUser) -> HMHomeAccessControl
}
```

User Management

HMHome

```
extension HMHome {
  var currentUser: HMUser { get }

  func manageUsers(completionHandler: @escaping (Error?) -> Void)
  func homeAccessControl(for user: HMUser) -> HMHomeAccessControl
}
```

User Management

HMHome

```
extension HMHome {
  var currentUser: HMUser { get }

  func manageUsers(completionHandler: @escaping (Error?) -> Void)
  func homeAccessControl(for user: HMUser) -> HMHomeAccessControl
}
```

User Management

HMHome

```
extension HMHome {
  var currentUser: HMUser { get }

  func manageUsers(completionHandler: @escaping (Error?) -> Void)
  func homeAccessControl(for user: HMUser) -> HMHomeAccessControl
}
```

```
class HMHomeAccessControl : NSObject {
  var administrator: Bool { get }
}
```

```swift
// Create Home & Manage Users
import HomeKit
class MyClass: NSObject, HMHomeDelegate {
    let homeManager = HMHomeManager()
    var myHome: HMHome?

    func configureHome(withName name: String) {
        homeManager.addHome(withName: name) { home, error in
            guard let home = home else { /* Handle error */ return }

            self.myHome = home
            home.delegate = self
            home.manageUsers { error in
                guard error == nil else { /* Handle error */ return }
            }
        }
    }
}
```

```swift
// Create Home & Manage Users
import HomeKit
class MyClass: NSObject, HMHomeDelegate {
    let homeManager = HMHomeManager()
    var myHome: HMHome?

    func configureHome(withName name: String) {
        homeManager.addHome(withName: name) { home, error in
            guard let home = home else { /* Handle error */ return }

            self.myHome = home
            home.delegate = self
            home.manageUsers { error in
                guard error == nil else { /* Handle error */ return }
            }
        }
    }
}
```

```swift
// Create Home & Manage Users
import HomeKit
class MyClass: NSObject, HMHomeDelegate {
    let homeManager = HMHomeManager()
    var myHome: HMHome?

    func configureHome(withName name: String) {
        homeManager.addHome(withName: name) { home, error in
            guard let home = home else { /* Handle error */ return }

            self.myHome = home
            home.delegate = self
            home.manageUsers { error in
                guard error == nil else { /* Handle error */ return }
            }
        }
    }
}
```

```swift
// Create Home & Manage Users
import HomeKit
class MyClass: NSObject, HMHomeDelegate {
    let homeManager = HMHomeManager()
    var myHome: HMHome?

    func configureHome(withName name: String) {
        homeManager.addHome(withName: name) { home, error in
            guard let home = home else { /* Handle error */ return }

            self.myHome = home
            home.delegate = self
            home.manageUsers { error in
                guard error == nil else { /* Handle error */ return }
            }
        }
    }
}
```

```swift
// Create Home & Manage Users
import HomeKit
class MyClass: NSObject, HMHomeDelegate {
    let homeManager = HMHomeManager()
    var myHome: HMHome?

    func configureHome(withName name: String) {
        homeManager.addHome(withName: name) { home, error in
            guard let home = home else { /* Handle error */ return }

            self.myHome = home
            home.delegate = self
            home.manageUsers { error in
                guard error == nil else { /* Handle error */ return }
            }
        }
    }
}
```

Rooms and Zones

HMRoom, HMZone

```
class HMZone : NSObject {
  var name: String { get }
  var uniqueIdentifier: UUID { get }
  var rooms: [HMRoom] { get }
}
```

```
class HMRoom : NSObject {
  var name: String { get }
  var uniqueIdentifier: UUID { get }
  var accessories: [HMAccessory] { get }
}
```

Rooms and Zones

HMRoom, HMZone

```
class HMZone : NSObject {
  var name: String { get }
  var uniqueIdentifier: UUID { get }
  var rooms: [HMRoom] { get }
}
```

```
class HMRoom : NSObject {
  var name: String { get }
  var uniqueIdentifier: UUID { get }
  var accessories: [HMAccessory] { get }
}
```

Rooms and Zones

HMRoom, HMZone

```
class HMZone : NSObject {
  var name: String { get }
  var uniqueIdentifier: UUID { get }
  var rooms: [HMRoom] { get }
}
```

```
class HMRoom : NSObject {
  var name: String { get }
  var uniqueIdentifier: UUID { get }
  var accessories: [HMAccessory] { get }
}
```

Rooms and Zones

HMRoom, HMZone

```
class HMZone : NSObject {
  var name: String { get }
  var uniqueIdentifier: UUID { get }
  var rooms: [HMRoom] { get }
}
```

```
class HMRoom : NSObject {
  var name: String { get }
  var uniqueIdentifier: UUID { get }
  var accessories: [HMAccessory] { get }
}
```

Rooms and Zones

HMHome integration

```
extension HMHome {
  var rooms: [HMRoom] { get }
  func addRoom(withName name: String, completionHandler: @escaping (HMRoom?, Error?) -> Void)
  func removeRoom(_ room: HMRoom, completionHandler: @escaping (Error?) -> Void)
}
```

```
extension HMHome {
  var zones: [HMZone] { get }
  func addZone(withName name: String, completionHandler: @escaping (HMZone?, Error?) -> Void)
  func removeZone(_ zone: HMZone, completionHandler: @escaping (Error?) -> Void)
}
```

Rooms and Zones

HMHome integration

```
extension HMHome {
  var rooms: [HMRoom] { get }
  func addRoom(withName name: String, completionHandler: @escaping (HMRoom?, Error?) -> Void)
  func removeRoom(_ room: HMRoom, completionHandler: @escaping (Error?) -> Void)
}
```

```
extension HMHome {
  var zones: [HMZone] { get }
  func addZone(withName name: String, completionHandler: @escaping (HMZone?, Error?) -> Void)
  func removeZone(_ zone: HMZone, completionHandler: @escaping (Error?) -> Void)
}
```

Rooms and Zones

HMHome integration

```
extension HMHome {
  var rooms: [HMRoom] { get }
  func addRoom(withName name: String, completionHandler: @escaping (HMRoom?, Error?) -> Void)
  func removeRoom(_ room: HMRoom, completionHandler: @escaping (Error?) -> Void)
}
```

```
extension HMHome {
  var zones: [HMZone] { get }
  func addZone(withName name: String, completionHandler: @escaping (HMZone?, Error?) -> Void)
  func removeZone(_ zone: HMZone, completionHandler: @escaping (Error?) -> Void)
}
```

```swift
// Create Room & Zone
func addRoom(withName roomName: String, zoneName: String, home: HMHome) {
    home.addRoom(withName: roomName) { room, error in
        guard let room = room else { /* Handle error */ return }

        home.addZone(withName: zoneName) { zone, error in
            guard let zone = zone else { /* Handle error */ return }

            zone.addRoom(room) { error in
                guard error == nil else { /* Handle error */ return }
            }
        }
    }
}
```

```swift
// Create Room & Zone
func addRoom(withName roomName: String, zoneName: String, home: HMHome) {
    home.addRoom(withName: roomName) { room, error in
        guard let room = room else { /* Handle error */ return }

        home.addZone(withName: zoneName) { zone, error in
            guard let zone = zone else { /* Handle error */ return }

            zone.addRoom(room) { error in
                guard error == nil else { /* Handle error */ return }
            }
        }
    }
}
```

```
// Create Room & Zone
func addRoom(withName roomName: String, zoneName: String, home: HMHome) {
    home.addRoom(withName: roomName) { room, error in
        guard let room = room else { /* Handle error */ return }

        home.addZone(withName: zoneName) { zone, error in
            guard let zone = zone else { /* Handle error */ return }

            zone.addRoom(room) { error in
                guard error == nil else { /* Handle error */ return }
            }
        }
    }
}
```

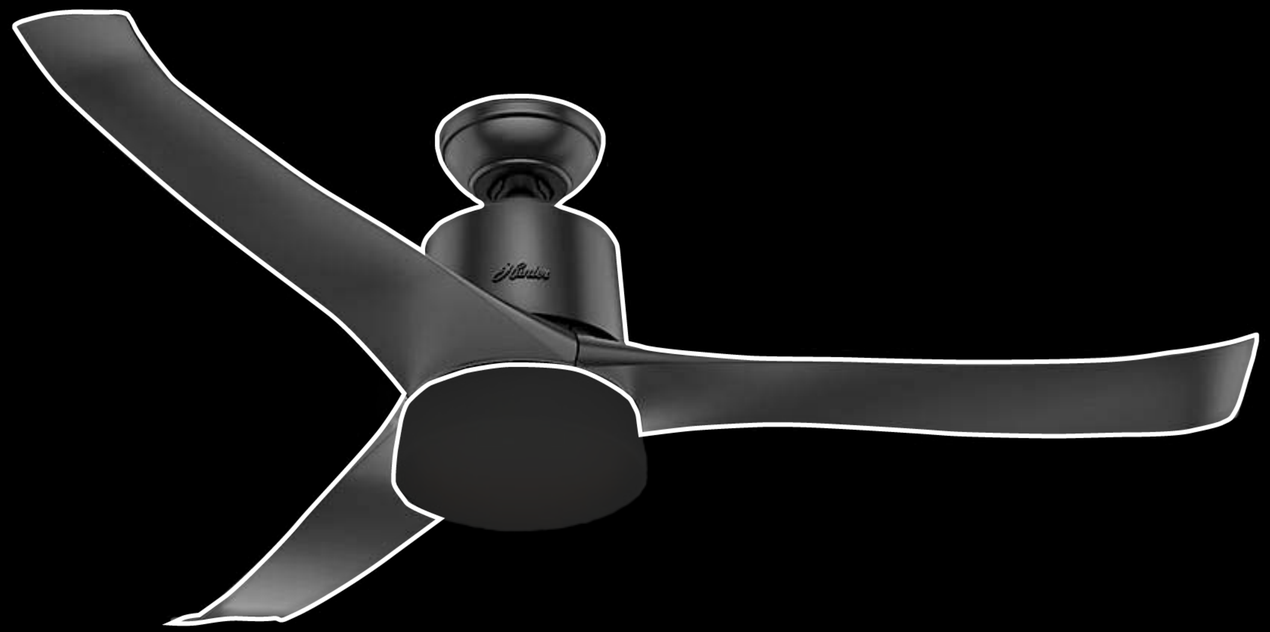
```swift
// Create Room & Zone
func addRoom(withName roomName: String, zoneName: String, home: HMHome) {
    home.addRoom(withName: roomName) { room, error in
        guard let room = room else { /* Handle error */ return }

        home.addZone(withName: zoneName) { zone, error in
            guard let zone = zone else { /* Handle error */ return }

            zone.addRoom(room) { error in
                guard error == nil else { /* Handle error */ return }
            }
        }
    }
}
```

Accessories, Services, and Characteristics

Accessories, Services, and Characteristics

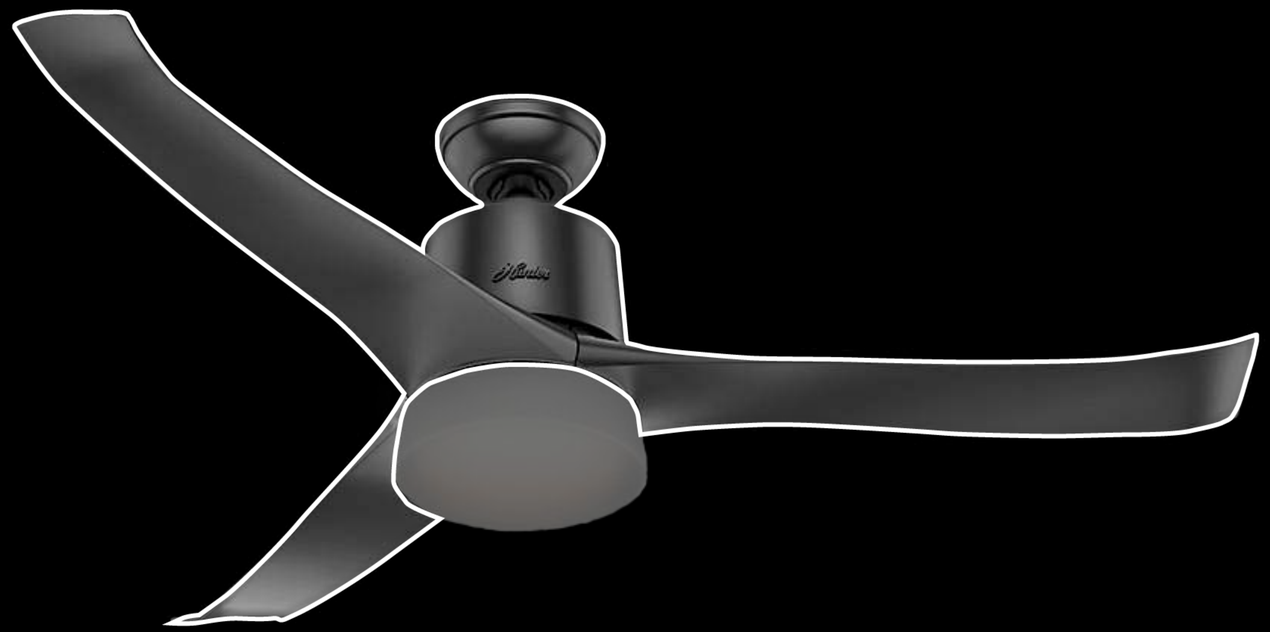
Accessories, Services, and Characteristics

Accessories, Services, and Characteristics



Accessories, Services, and Characteristics

Accessories, Services, and Characteristics



Accessories, Services, and Characteristics

Light

Power

Brightness

Saturation

Fan

Accessories, Services, and Characteristics

Light

- Power
- Brightness
- Saturation

Fan

- Power
- Rotation Speed
- Rotation Direction

Accessories, Services, and Characteristics

HMAccessory

Light

- Power
- Brightness
- Saturation

Fan

- Power
- Rotation Speed
- Rotation Direction

Accessories, Services, and Characteristics

HMAccessory

HMService

Light

- Power
- Brightness
- Saturation

Fan

- Power
- Rotation Speed
- Rotation Direction

Accessories, Services, and Characteristics

HMAccessory

HMService

Light

Fan

HMCharacteristic

Power

Brightness

Saturation

Power

Rotation Speed

Rotation Direction

Accessories

HMAccessory

```
class HMAccessory : NSObject {
  var name: String { get }
  var category: HMAccessoryCategory { get }
  var services: [HMService] { get }
  var model: String? { get }
  var manufacturer: String? { get }
  var firmwareVersion: String? { get }
}
```

Accessories

HMAccessory

```
class HMAccessory : NSObject {
  var name: String { get }
  var category: HMAccessoryCategory { get }
  var services: [HMService] { get }
  var model: String? { get }
  var manufacturer: String? { get }
  var firmwareVersion: String? { get }
}
```

Accessories

HMAccessory

```
class HMAccessory : NSObject {
  var name: String { get }
  var category: HMAccessoryCategory { get }
  var services: [HMService] { get }
  var model: String? { get }
  var manufacturer: String? { get }
  var firmwareVersion: String? { get }
}
```

Accessories

HMAccessory

```
class HMAccessory : NSObject {
  var name: String { get }
  var category: HMAccessoryCategory { get }
  var services: [HMService] { get }
  var model: String? { get }
  var manufacturer: String? { get }
  var firmwareVersion: String? { get }
}
```

Services

HMService

```
class HMService : NSObject {
  var serviceType: String { get }
  var name: String { get }
  var characteristics: [HMCharacteristic] { get }
  var isUserInteractive: Bool { get }

  func updateName(_ name: String, completionHandler: @escaping (Error?) -> Void)
}
```

Services

HMService

```
class HMService : NSObject {
  var serviceType: String { get }
  var name: String { get }
  var characteristics: [HMCharacteristic] { get }
  var isUserInteractive: Bool { get }

  func updateName(_ name: String, completionHandler: @escaping (Error?) -> Void)
}
```

Services

HMService

```
class HMService : NSObject {
  var serviceType: String { get }
  var name: String { get }
  var characteristics: [HMCharacteristic] { get }
  var isUserInteractive: Bool { get }

  func updateName(_ name: String, completionHandler: @escaping (Error?) -> Void)
}
```

Services

Naming best practices

Services

Naming best practices

Used by Home app and Siri

Services

Naming best practices

Used by Home app and Siri

Choose good default names

Services

Naming best practices

Used by Home app and Siri

Choose good default names

Services

Naming best practices

Used by Home app and Siri

Choose good default names

No special characters or numbers

Services

Naming best practices

Used by Home app and Siri

Choose good default names

No special characters or numbers

Service Types

Predefined service types

```
public let HMServiceTypeLightbulb: String
public let HMServiceTypeSwitch: String
public let HMServiceTypeThermostat: String
public let HMServiceTypeGarageDoorOpener: String
public let HMServiceTypeFan: String
public let HMServiceTypeOutlet: String
public let HMServiceTypeContactSensor: String
…
…
public let HMServiceTypeMotionSensor: String
public let HMServiceTypeSmokeSensor: String
public let HMServiceTypeWindowCovering: String
public let HMServiceTypeValve: String
```

Accessories

HMHome integration

```
extension HMHome {
  var accessories: [HMAccessory] { get }

  func addAndSetupAccessories(completionHandler: @escaping (Error?) -> Void)
  func addAndSetupAccessories(with payload: HMAccessorySetupPayload, completionHandler:
@escaping ([HMAccessory]?, Error?) -> Void)
}
```

Accessories

HMHome integration

```
extension HMHome {
  var accessories: [HMAccessory] { get }

  func addAndSetupAccessories(completionHandler: @escaping (Error?) -> Void)
  func addAndSetupAccessories(with payload: HMAccessorySetupPayload, completionHandler:
@escaping ([HMAccessory]?, Error?) -> Void)
}
```

Accessories

HMHome integration

```
extension HMHome {
  var accessories: [HMAccessory] { get }

  func addAndSetupAccessories(completionHandler: @escaping (Error?) -> Void)
  func addAndSetupAccessories(with payload: HMAccessorySetupPayload, completionHandler:
@escaping ([HMAccessory]?, Error?) -> Void)
}
```

```swift
// Add an accessory
func addAccessory(to home: HMHome) {
    home.addAndSetupAccessories { error in
        guard error == nil else { /* Handle error */ return }
    }
}
```

```swift
// Add an accessory
func addAccessory(to home: HMHome) {
    home.addAndSetupAccessories { error in
        guard error == nil else { /* Handle error */ return }
    }
}
```

Scenes

Scenes

Integrate accessories together

Scenes

Integrate accessories together

Predefined scenes with suggested actions

Scenes

Integrate accessories together

Predefined scenes with suggested actions

User customizable

Scenes

Integrate accessories together

Predefined scenes with suggested actions

User customizable

Built-in Siri support

Scenes

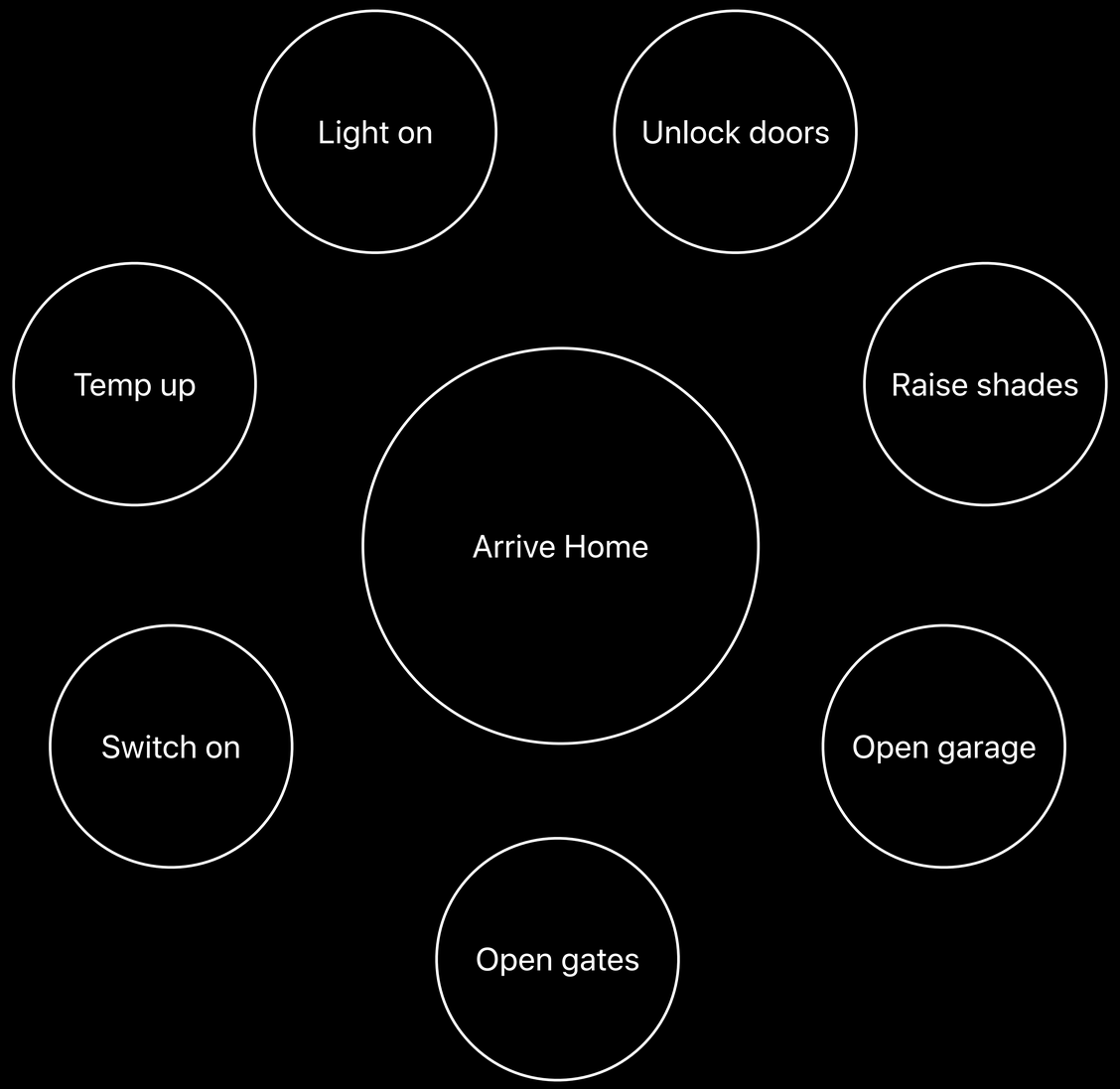
HMActionSet

Scenes

HMActionSet

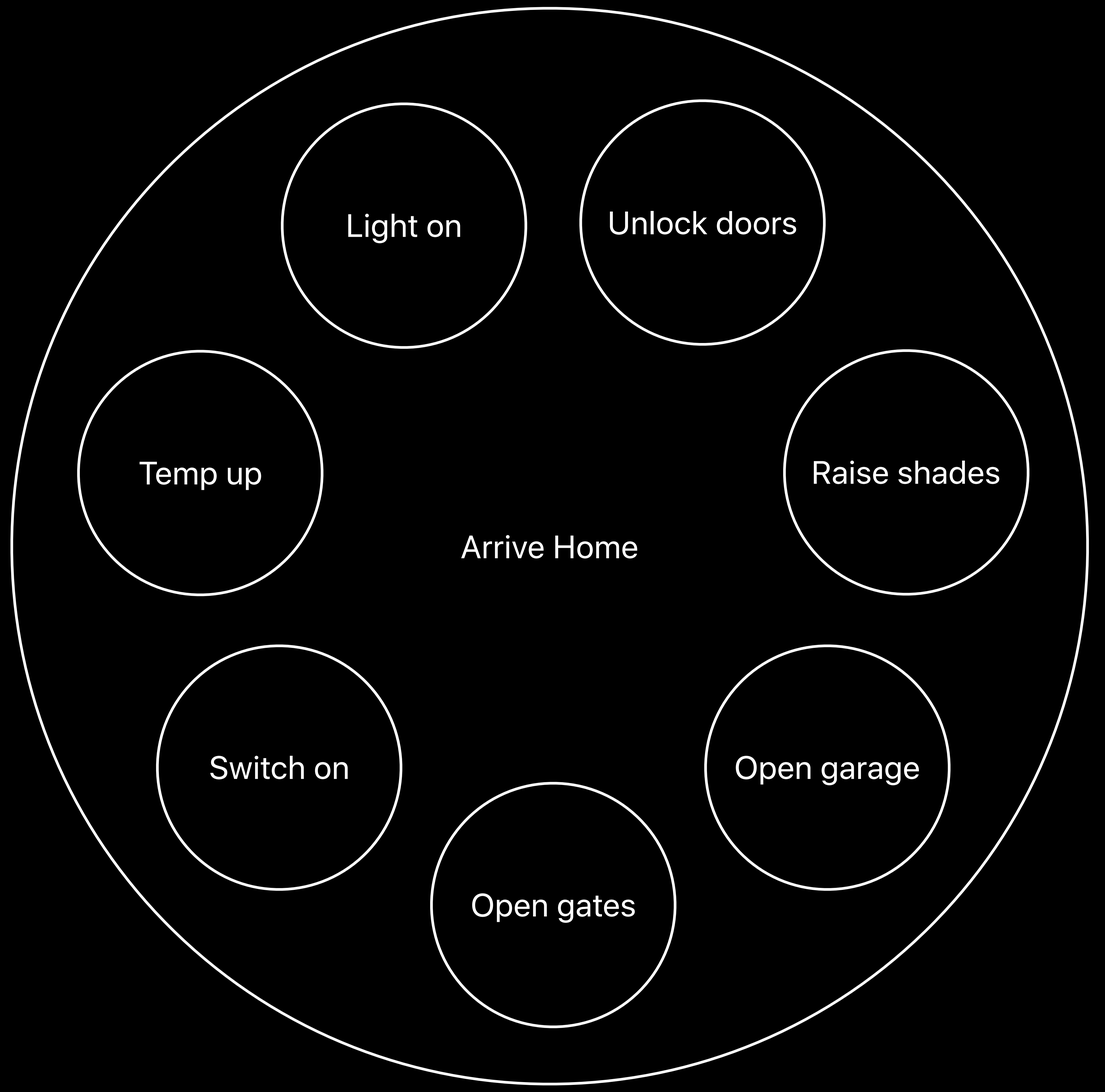
Scenes

HMActionSet



Scenes

HMActionSet



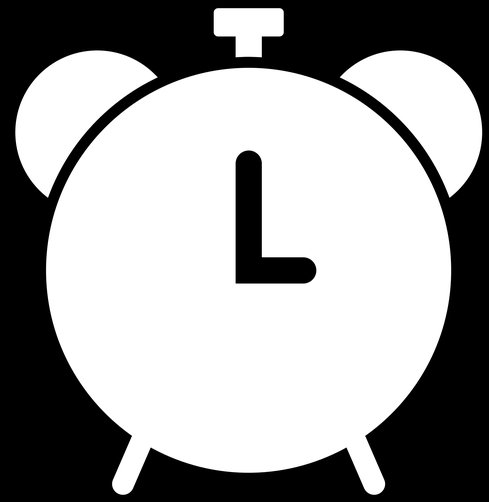
Scenes

Predefined action set types

Scenes

Predefined action set types

HMActionSetTypeWakeUp

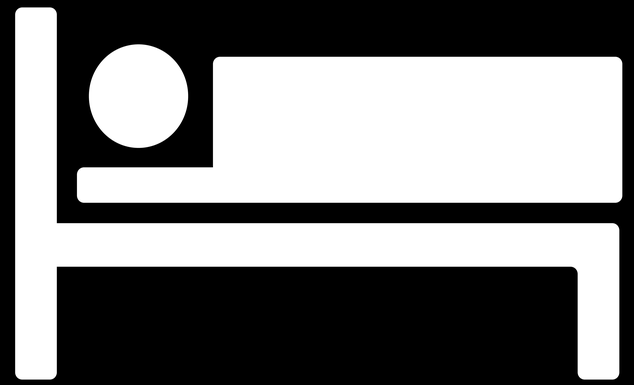


Scenes

Predefined action set types

HMActionSetTypeWakeUp

HMActionSetTypeSleep



Scenes

Predefined action set types

HMActionSetTypeWakeUp

HMActionSetTypeSleep

HMActionSetTypeHomeDeparture

Scenes

Predefined action set types

HMActionSetTypeWakeUp

HMActionSetTypeSleep

HMActionSetTypeHomeDeparture

HMActionSetTypeHomeArrival

Scenes

Predefined action set types

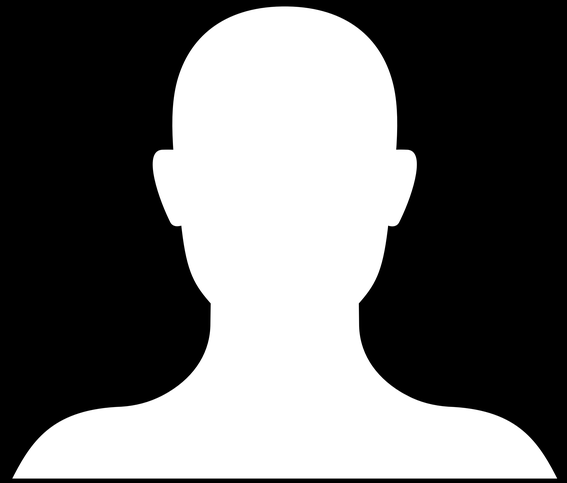
HMActionSetTypeWakeUp

HMActionSetTypeSleep

HMActionSetTypeHomeDeparture

HMActionSetTypeHomeArrival

HMActionSetTypeUserDefined



Scenes

HMHome integration

```
extension HMHome {
  var actionSets: [HMActionSet] { get }
  func addActionSet(withName: String, completionHandler: @escaping (HMActionSet?, Error?)
                                                                  -> Void)
  func removeActionSet(_: HMActionSet, completionHandler: @escaping (Error?) -> Void)
  func executeActionSet(_: HMActionSet, completionHandler: @escaping (Error?) -> Void)
  func builtinActionSet(ofType actionSetType: String) -> HMActionSet?
}
```

Scenes

HMHome integration

```
extension HMHome {
  var actionSets: [HMActionSet] { get }
  func addActionSet(withName: String, completionHandler: @escaping (HMActionSet?, Error?)
                                                                  -> Void)
  func removeActionSet(_: HMActionSet, completionHandler: @escaping (Error?) -> Void)
  func executeActionSet(_: HMActionSet, completionHandler: @escaping (Error?) -> Void)
  func builtinActionSet(ofType actionSetType: String) -> HMActionSet?
}
```

Scenes

HMHome integration

```
extension HMHome {
  var actionSets: [HMActionSet] { get }
  func addActionSet(withName: String, completionHandler: @escaping (HMActionSet?, Error?)
                                                                  -> Void)
  func removeActionSet(_: HMActionSet, completionHandler: @escaping (Error?) -> Void)
  func executeActionSet(_: HMActionSet, completionHandler: @escaping (Error?) -> Void)
  func builtinActionSet(ofType actionSetType: String) -> HMActionSet?
}
```

Scenes

HMHome integration

```
extension HMHome {
  var actionSets: [HMActionSet] { get }
  func addActionSet(withName: String, completionHandler: @escaping (HMActionSet?, Error?)
                                                                  -> Void)
  func removeActionSet(_: HMActionSet, completionHandler: @escaping (Error?) -> Void)
  func executeActionSet(_: HMActionSet, completionHandler: @escaping (Error?) -> Void)
  func builtinActionSet(ofType actionSetType: String) -> HMActionSet?
}
```

```
func createActionSet(withName name: String, actions: [HMAction], home: HMHome) {
    home.addActionSet(withName: name) { actionSet, error in
        guard let actionSet = actionSet else { /* Handle error */ return }

        for action in actions {
           actionSet.addAction(action) { error in
                guard error == nil else { /* Handle error */ return }
           }
        }
    }
}

func executeActionSet(_ actionSet: HMActionSet, home: HMHome) {
    home.executeActionSet(actionSet) { error in
        guard error == nil else { /* Handle error */ return }
    }
}
```

```swift
func createActionSet(withName name: String, actions: [HMAction], home: HMHome) {
    home.addActionSet(withName: name) { actionSet, error in
        guard let actionSet = actionSet else { /* Handle error */ return }

        for action in actions {
            actionSet.addAction(action) { error in
                guard error == nil else { /* Handle error */ return }
            }
        }
    }
}

func executeActionSet(_ actionSet: HMActionSet, home: HMHome) {
    home.executeActionSet(actionSet) { error in
        guard error == nil else { /* Handle error */ return }
    }
}
```

```
func createActionSet(withName name: String, actions: [HMAction], home: HMHome) {
    home.addActionSet(withName: name) { actionSet, error in
        guard let actionSet = actionSet else { /* Handle error */ return }

        for action in actions {
            actionSet.addAction(action) { error in
                guard error == nil else { /* Handle error */ return }
            }
        }
    }
}

func executeActionSet(_ actionSet: HMActionSet, home: HMHome) {
    home.executeActionSet(actionSet) { error in
        guard error == nil else { /* Handle error */ return }
    }
}
```

```swift
func createActionSet(withName name: String, actions: [HMAction], home: HMHome) {
    home.addActionSet(withName: name) { actionSet, error in
        guard let actionSet = actionSet else { /* Handle error */ return }

        for action in actions {
            actionSet.addAction(action) { error in
                guard error == nil else { /* Handle error */ return }
            }
        }
    }
}

func executeActionSet(_ actionSet: HMActionSet, home: HMHome) {
    home.executeActionSet(actionSet) { error in
        guard error == nil else { /* Handle error */ return }
    }
}
```

```
func createActionSet(withName name: String, actions: [HMAction], home: HMHome) {
    home.addActionSet(withName: name) { actionSet, error in
        guard let actionSet = actionSet else { /* Handle error */ return }

        for action in actions {
           actionSet.addAction(action) { error in
                guard error == nil else { /* Handle error */ return }
           }
        }
    }
}

func executeActionSet(_ actionSet: HMActionSet, home: HMHome) {
    home.executeActionSet(actionSet) { error in
        guard error == nil else { /* Handle error */ return }
    }
}
```

Event Triggers

Event Triggers

Location

Set your lights to turn on as soon as you pull up to the house.

Event Triggers

Location

Set your lights to turn on as soon as you pull up to the house.

Time

Have your home turn up the heat at 6:00 a.m., before you get out of bed.

Event Triggers

Location

Set your lights to turn on as soon as you pull up to the house.

Time

Have your home turn up the heat at 6:00 a.m., before you get out of bed.

Action

Use a motion sensor in the doorway to turn your kitchen lights on When you walk in.

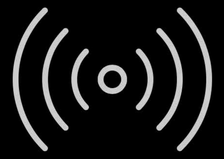
Event Triggers

Location

Set your lights to turn on as soon as you pull up to the house.

Time

Have your home turn up the heat at 6:00 a.m., before you get out of bed.



Action

Use a motion sensor in the doorway to turn your kitchen lights on When you walk in.

Occupancy

Trigger a scene based on who’s home, like automatically turning the lights off as you leave if nobody else is there.

Event Triggers

HMTrigger

```
class HMTrigger : NSObject {
  var name: String { get }
  var isEnabled: Bool { get }
  var actionSets: [HMActionSet] { get }
  func updateName(_ name: String, completionHandler: @escaping (Error?) -> Void)
  func addActionSet(_ actionSet: HMActionSet, completionHandler: @escaping (Error?) -> Void)
  func enable(_ enable: Bool, completionHandler: @escaping (Error?) -> Void)
}
```

Event Triggers

HMTrigger

```
class HMTrigger : NSObject {
  var name: String { get }
  var isEnabled: Bool { get }
  var actionSets: [HMActionSet] { get }
  func updateName(_ name: String, completionHandler: @escaping (Error?) -> Void)
  func addActionSet(_ actionSet: HMActionSet, completionHandler: @escaping (Error?) -> Void)
  func enable(_ enable: Bool, completionHandler: @escaping (Error?) -> Void)
}
```

Event Triggers

HMTrigger

```
class HMTrigger : NSObject {
  var name: String { get }
  var isEnabled: Bool { get }
  var actionSets: [HMActionSet] { get }
  func updateName(_ name: String, completionHandler: @escaping (Error?) -> Void)
  func addActionSet(_ actionSet: HMActionSet, completionHandler: @escaping (Error?) -> Void)
  func enable(_ enable: Bool, completionHandler: @escaping (Error?) -> Void)
}
```

Event Triggers

HMTrigger

```
class HMTrigger : NSObject {
  var name: String { get }
  var isEnabled: Bool { get }
  var actionSets: [HMActionSet] { get }
  func updateName(_ name: String, completionHandler: @escaping (Error?) -> Void)
  func addActionSet(_ actionSet: HMActionSet, completionHandler: @escaping (Error?) -> Void)
  func enable(_ enable: Bool, completionHandler: @escaping (Error?) -> Void)
}
```

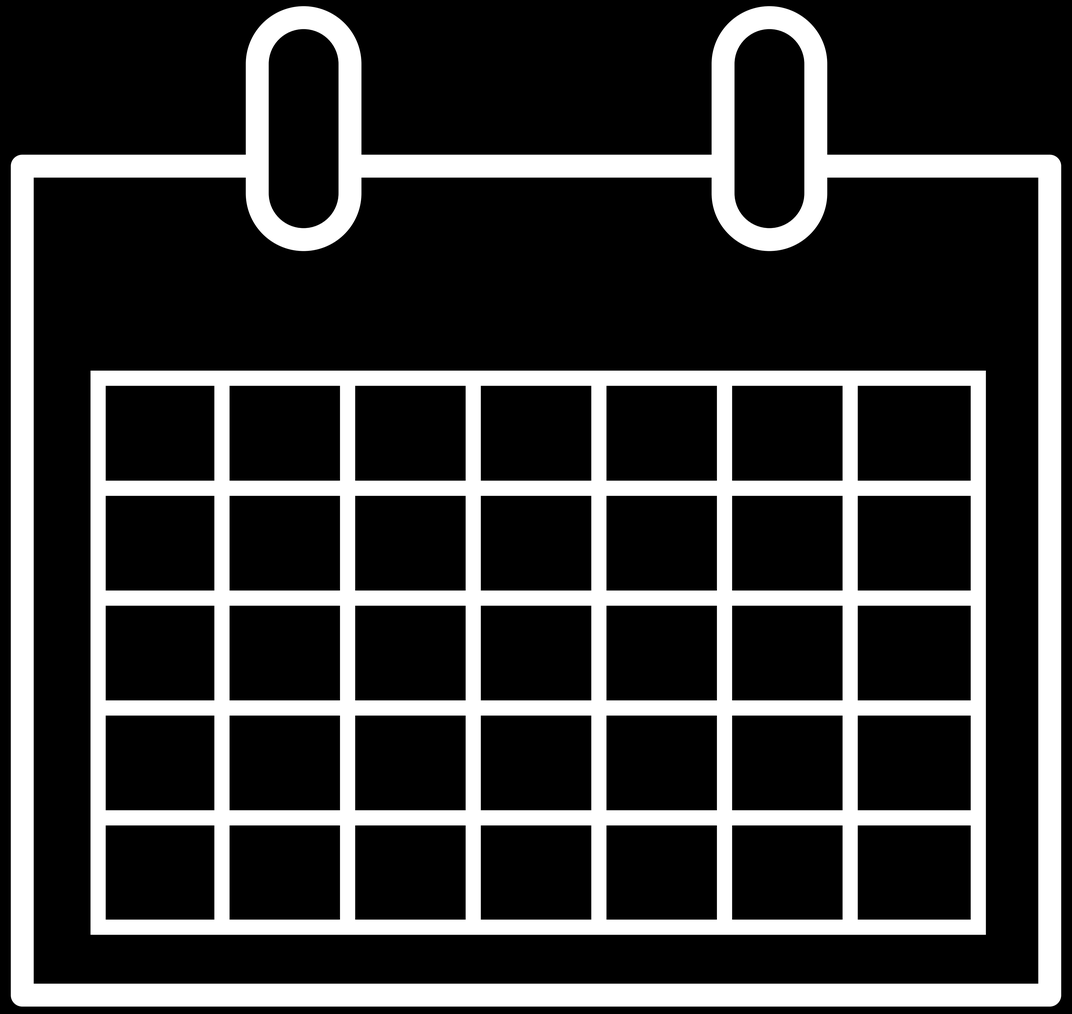
Event Triggers

HMTrigger

```
class HMTrigger : NSObject {
  var name: String { get }
  var isEnabled: Bool { get }
  var actionSets: [HMActionSet] { get }
  func updateName(_ name: String, completionHandler: @escaping (Error?) -> Void)
  func addActionSet(_ actionSet: HMActionSet, completionHandler: @escaping (Error?) -> Void)
  func enable(_ enable: Bool, completionHandler: @escaping (Error?) -> Void)
}
```

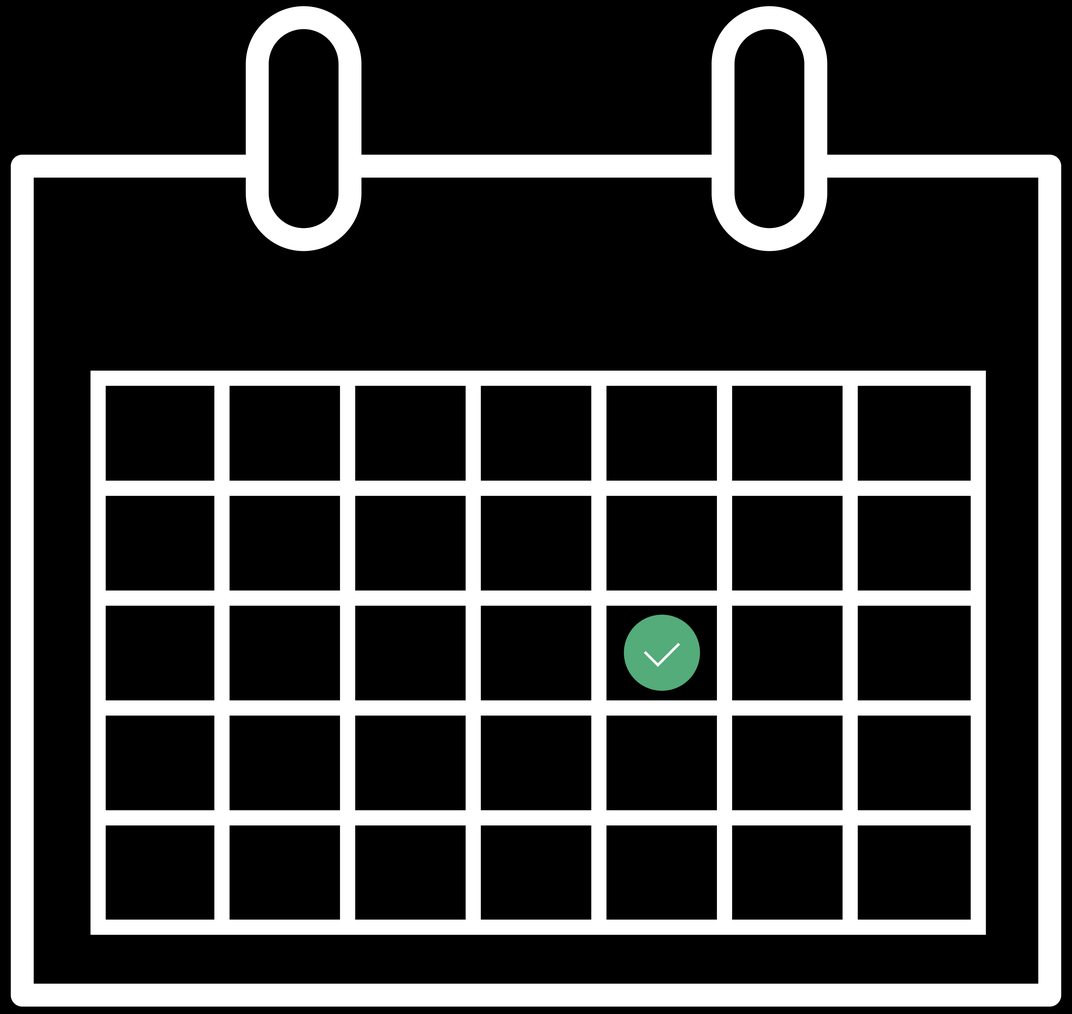
Event Triggers

HMHome integration



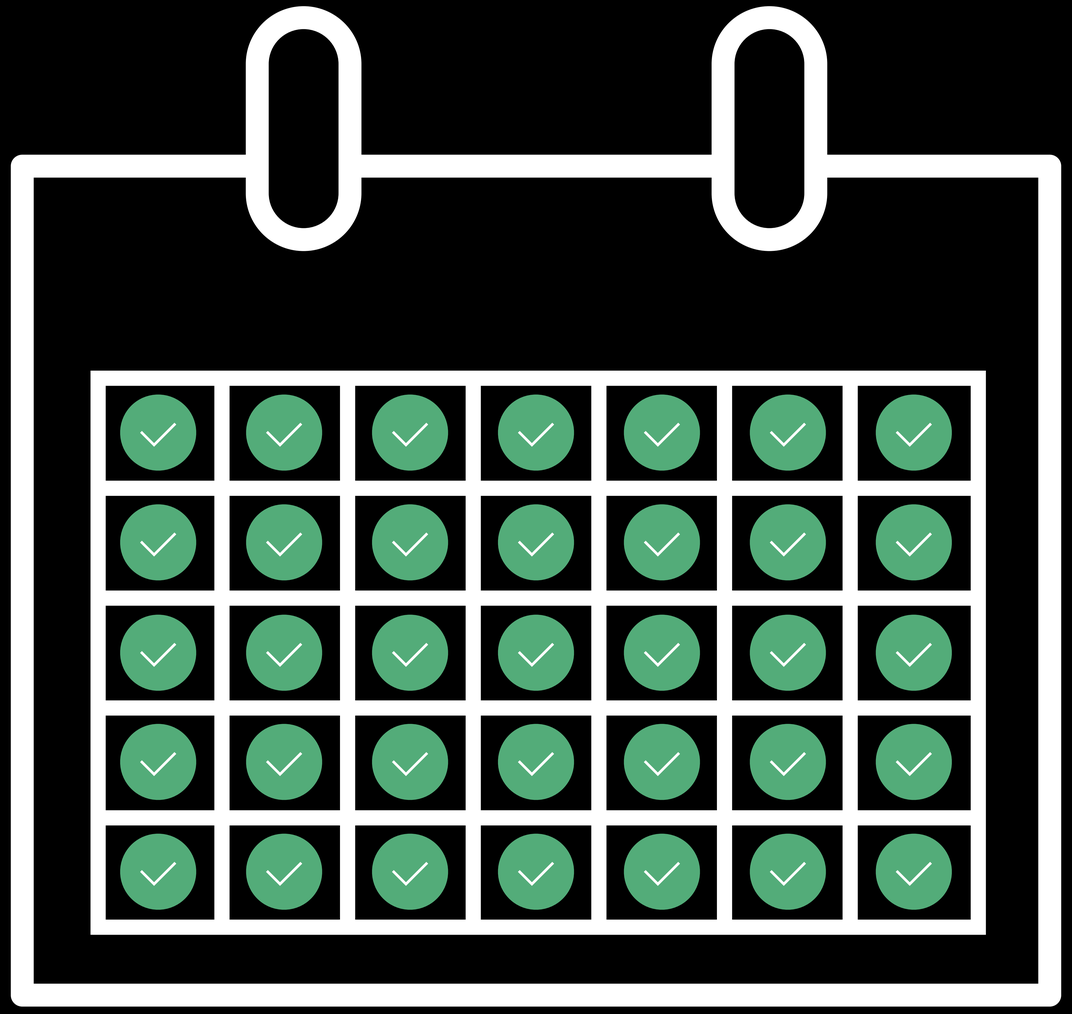
Event Triggers

HMHome integration



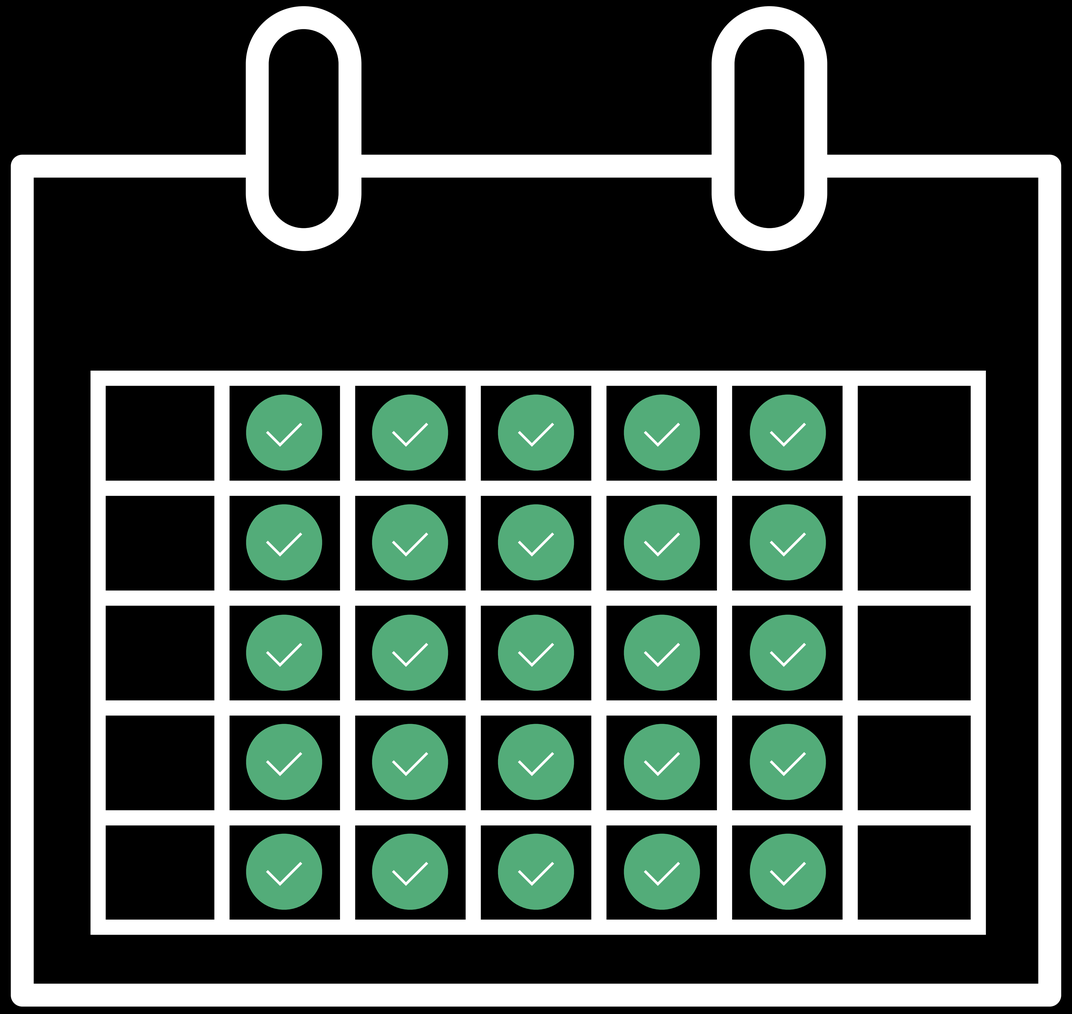
Event Triggers

HMHome integration



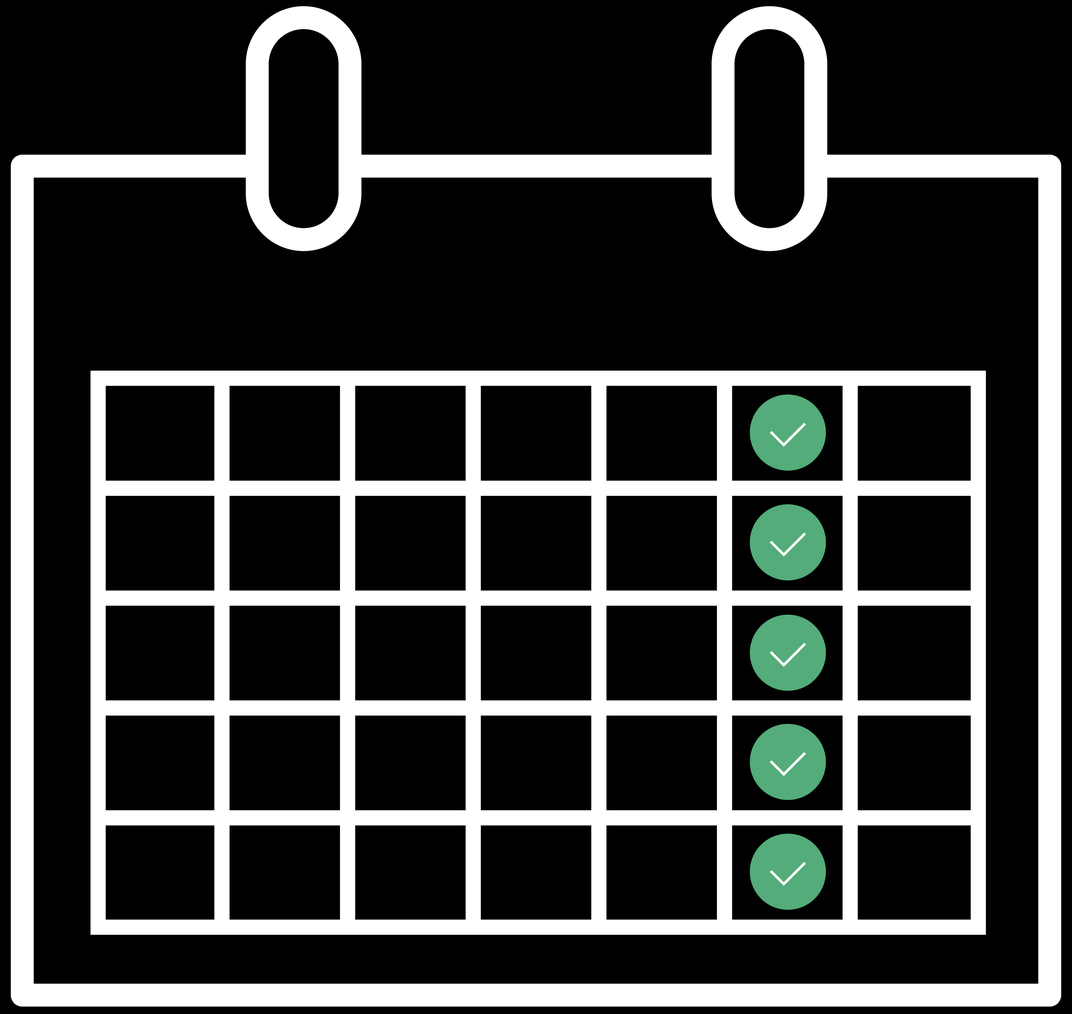
Event Triggers

HMHome integration



Event Triggers

HMHome integration



```swift
// Create 7PM Trigger
func create7PMTrigger(withName name: String, actionSet: HMActionSet, home: HMHome) {
    var components = DateComponents()
    components.hour = 19

    let timeEvent = HMCalendarEvent(fire: components)
    let trigger = HMEventTrigger(name: name, events: [timeEvent], predicate: nil)

    home.addTrigger(trigger) { error in
        guard error == nil else { /* Handle error */ return }

        trigger.addActionSet(actionSet) { error in
           guard error == nil else { /* Handle error */ return }
       }
    }
}
```

```swift
// Create 7PM Trigger
func create7PMTrigger(withName name: String, actionSet: HMActionSet, home: HMHome) {
    var components = DateComponents()
    components.hour = 19

    let timeEvent = HMCalendarEvent(fire: components)
    let trigger = HMEventTrigger(name: name, events: [timeEvent], predicate: nil)

    home.addTrigger(trigger) { error in
        guard error == nil else { /* Handle error */ return }

        trigger.addActionSet(actionSet) { error in
            guard error == nil else { /* Handle error */ return }
        }
    }
}
```

```swift
// Create 7PM Trigger
func create7PMTrigger(withName name: String, actionSet: HMActionSet, home: HMHome) {
    var components = DateComponents()
    components.hour = 19

    let timeEvent = HMCalendarEvent(fire: components)
    let trigger = HMEventTrigger(name: name, events: [timeEvent], predicate: nil)

    home.addTrigger(trigger) { error in
        guard error == nil else { /* Handle error */ return }

        trigger.addActionSet(actionSet) { error in
           guard error == nil else { /* Handle error */ return }
       }
    }
}
```

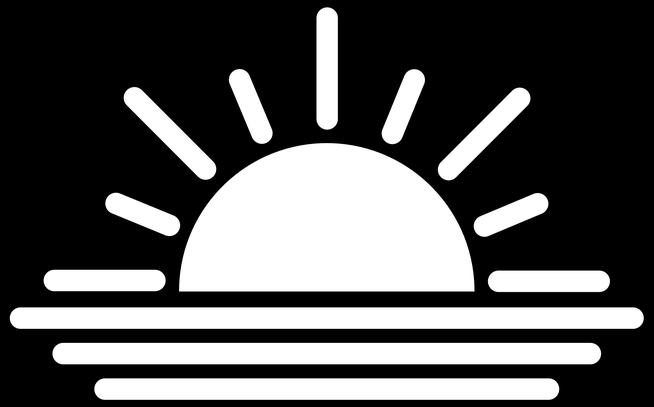
```
// Create 7PM Trigger
func create7PMTrigger(withName name: String, actionSet: HMActionSet, home: HMHome) {
    var components = DateComponents()
    components.hour = 19

    let timeEvent = HMCalendarEvent(fire: components)
    let trigger = HMEventTrigger(name: name, events: [timeEvent], predicate: nil)

    home.addTrigger(trigger) { error in
        guard error == nil else { /* Handle error */ return }

        trigger.addActionSet(actionSet) { error in
           guard error == nil else { /* Handle error */ return }
       }
    }
}
```

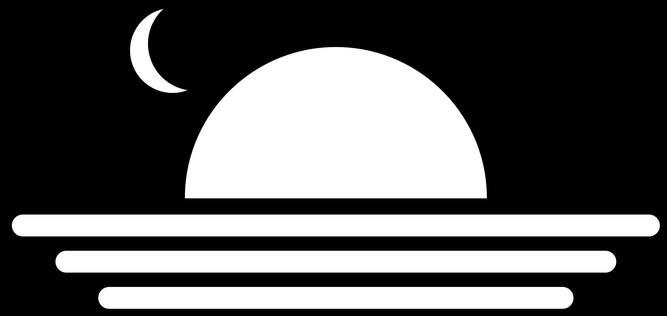
```swift
// Create 7PM Trigger
func create7PMTrigger(withName name: String, actionSet: HMActionSet, home: HMHome) {
    var components = DateComponents()
    components.hour = 19

    let timeEvent = HMCalendarEvent(fire: components)
    let trigger = HMEventTrigger(name: name, events: [timeEvent], predicate: nil)

    home.addTrigger(trigger) { error in
        guard error == nil else { /* Handle error */ return }

        trigger.addActionSet(actionSet) { error in
            guard error == nil else { /* Handle error */ return }
        }
    }
}
```

```swift
// Create 7PM Trigger
func create7PMTrigger(withName name: String, actionSet: HMActionSet, home: HMHome) {
    var components = DateComponents()
    components.hour = 19

    let timeEvent = HMCalendarEvent(fire: components)
    let trigger = HMEventTrigger(name: name, events: [timeEvent], predicate: nil)

    home.addTrigger(trigger) { error in
        guard error == nil else { /* Handle error */ return }

        trigger.addActionSet(actionSet) { error in
            guard error == nil else { /* Handle error */ return }
        }
    }
}
```

Significant Time Events

HMSignificantTimeEvent





```swift
// Create Sunset Trigger
func createSunsetTrigger(withName name: String, actionSet: HMActionSet, home: HMHome) {
    var offset = DateComponents()
    offset.minute = -30

    let sunsetEvent = HMSignificantTimeEvent(significantEvent: .sunset, offset: offset)
    let trigger = HMEventTrigger(name: name, events: [sunsetEvent], predicate: nil)

    home.addTrigger(trigger) { error in
        guard error == nil else { /* Handle error */ return }

        trigger.addActionSet(actionSet) { error in
            guard error == nil else { /* Handle error */ return }
        }
    }
}
```

```swift
// Create Sunset Trigger
func createSunsetTrigger(withName name: String, actionSet: HMActionSet, home: HMHome) {
    var offset = DateComponents()
    offset.minute = -30

    let sunsetEvent = HMSignificantTimeEvent(significantEvent: .sunset, offset: offset)
    let trigger = HMEventTrigger(name: name, events: [sunsetEvent], predicate: nil)

    home.addTrigger(trigger) { error in
        guard error == nil else { /* Handle error */ return }

        trigger.addActionSet(actionSet) { error in
            guard error == nil else { /* Handle error */ return }
        }
    }
}
```

```swift
// Create Sunset Trigger
func createSunsetTrigger(withName name: String, actionSet: HMActionSet, home: HMHome) {
    var offset = DateComponents()
    offset.minute = -30

    let sunsetEvent = HMSignificantTimeEvent(significantEvent: .sunset, offset: offset)
    let trigger = HMEventTrigger(name: name, events: [sunsetEvent], predicate: nil)

    home.addTrigger(trigger) { error in
        guard error == nil else { /* Handle error */ return }

        trigger.addActionSet(actionSet) { error in
            guard error == nil else { /* Handle error */ return }
        }
    }
}
```

```swift
// Create Sunset Trigger
func createSunsetTrigger(withName name: String, actionSet: HMActionSet, home: HMHome) {
    var offset = DateComponents()
    offset.minute = -30

    let sunsetEvent = HMSignificantTimeEvent(significantEvent: .sunset, offset: offset)
    let trigger = HMEventTrigger(name: name, events: [sunsetEvent], predicate: nil)

    home.addTrigger(trigger) { error in
        guard error == nil else { /* Handle error */ return }

        trigger.addActionSet(actionSet) { error in
            guard error == nil else { /* Handle error */ return }
        }
    }
}
```

```swift
// Create Sunset Trigger
func createSunsetTrigger(withName name: String, actionSet: HMActionSet, home: HMHome) {
    var offset = DateComponents()
    offset.minute = -30

    let sunsetEvent = HMSignificantTimeEvent(significantEvent: .sunset, offset: offset)
    let trigger = HMEventTrigger(name: name, events: [sunsetEvent], predicate: nil)

    home.addTrigger(trigger) { error in
        guard error == nil else { /* Handle error */ return }

        trigger.addActionSet(actionSet) { error in
            guard error == nil else { /* Handle error */ return }
        }
    }
}
```

Event Trigger Conditions

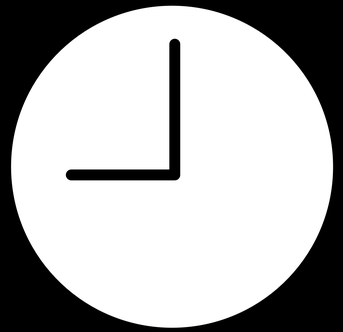
Event Trigger Conditions

Conditionalize automated controls

Event Trigger Conditions

Conditionalize automated controls

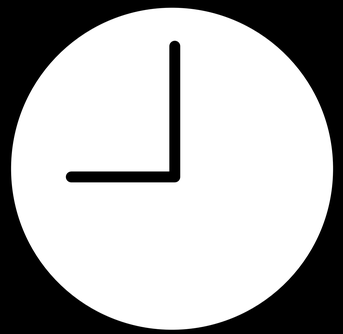
- Time

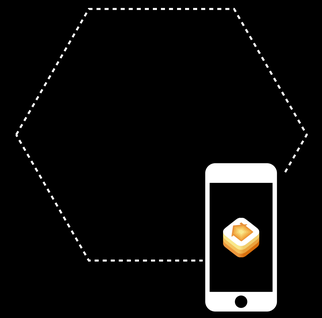


Event Trigger Conditions

Conditionalize automated controls

- Time
- Presence

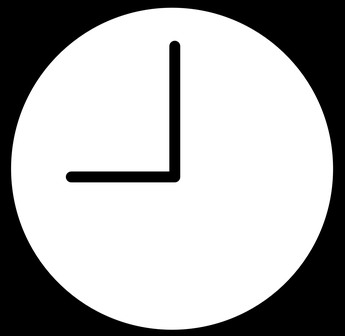


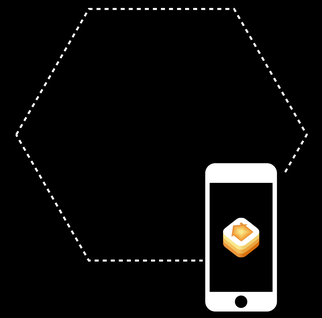


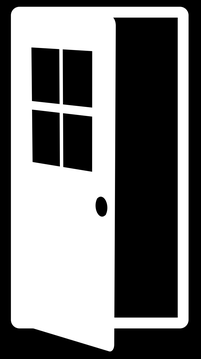
Event Trigger Conditions

Conditionalize automated controls

- Time
- Presence
- Accessory conditions



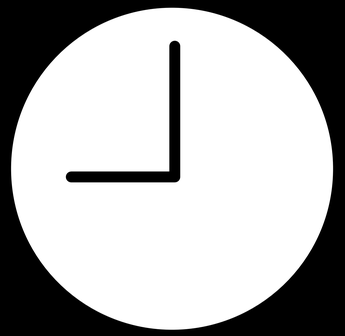


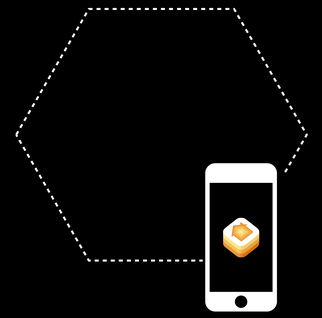


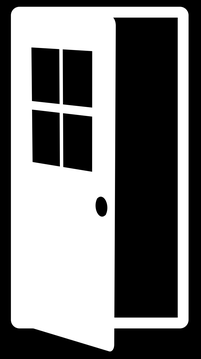
Event Trigger Conditions

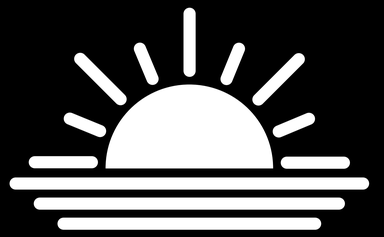
Conditionalize automated controls

- Time
- Presence
- Accessory conditions
- Significant events









```swift
// Create Sunset Presence Trigger
func createSunsetPresenceTrigger(withName name: String, actionSet: HMActionSet, home: HMHome) {
    var offset = DateComponents()
    offset.minute = -30
    let sunsetEvent = HMSignificantTimeEvent(significantEvent: .sunset, offset: offset)

    let presence = HMPresenceEvent(presenceEventType: .atHome, presenceUserType: .homeUsers)
    let predicate = HMEventTrigger.predicateForEvaluatingTrigger(withPresence: presence)
    let trigger = HMEventTrigger(name: name, events: [sunsetEvent], predicate: predicate)

    home.addTrigger(trigger) { error in
        guard error == nil else { /* Handle error */ return }

        trigger.addActionSet(actionSet) { error in
            guard error == nil else { /* Handle error */ return }
        }
    }
}
```

```swift
// Create Sunset Presence Trigger
func createSunsetPresenceTrigger(withName name: String, actionSet: HMActionSet, home: HMHome) {
    var offset = DateComponents()
    offset.minute = -30
    let sunsetEvent = HMSignificantTimeEvent(significantEvent: .sunset, offset: offset)

    let presence = HMPresenceEvent(presenceEventType: .atHome, presenceUserType: .homeUsers)
    let predicate = HMEventTrigger.predicateForEvaluatingTrigger(withPresence: presence)
    let trigger = HMEventTrigger(name: name, events: [sunsetEvent], predicate: predicate)

    home.addTrigger(trigger) { error in
        guard error == nil else { /* Handle error */ return }

        trigger.addActionSet(actionSet) { error in
            guard error == nil else { /* Handle error */ return }
        }
    }
}
```

```swift
// Create Sunset Presence Trigger
func createSunsetPresenceTrigger(withName name: String, actionSet: HMActionSet, home: HMHome) {
    var offset = DateComponents()
    offset.minute = -30
    let sunsetEvent = HMSignificantTimeEvent(significantEvent: .sunset, offset: offset)

    let presence = HMPresenceEvent(presenceEventType: .atHome, presenceUserType: .homeUsers)
    let predicate = HMEventTrigger.predicateForEvaluatingTrigger(withPresence: presence)
    let trigger = HMEventTrigger(name: name, events: [sunsetEvent], predicate: predicate)

    home.addTrigger(trigger) { error in
        guard error == nil else { /* Handle error */ return }

        trigger.addActionSet(actionSet) { error in
            guard error == nil else { /* Handle error */ return }
        }
    }
}
```

```swift
// Create Sunset Presence Trigger
func createSunsetPresenceTrigger(withName name: String, actionSet: HMActionSet, home: HMHome) {
    var offset = DateComponents()
    offset.minute = -30
    let sunsetEvent = HMSignificantTimeEvent(significantEvent: .sunset, offset: offset)

    let presence = HMPresenceEvent(presenceEventType: .atHome, presenceUserType: .homeUsers)
    let predicate = HMEventTrigger.predicateForEvaluatingTrigger(withPresence: presence)
    let trigger = HMEventTrigger(name: name, events: [sunsetEvent], predicate: predicate)

    home.addTrigger(trigger) { error in
        guard error == nil else { /* Handle error */ return }

        trigger.addActionSet(actionSet) { error in
            guard error == nil else { /* Handle error */ return }
        }
    }
}
```

```swift
// Create Sunset Presence Trigger
func createSunsetPresenceTrigger(withName name: String, actionSet: HMActionSet, home: HMHome) {
    var offset = DateComponents()
    offset.minute = -30
    let sunsetEvent = HMSignificantTimeEvent(significantEvent: .sunset, offset: offset)

    let presence = HMPresenceEvent(presenceEventType: .atHome, presenceUserType: .homeUsers)
    let predicate = HMEventTrigger.predicateForEvaluatingTrigger(withPresence: presence)
    let trigger = HMEventTrigger(name: name, events: [sunsetEvent], predicate: predicate)

    home.addTrigger(trigger) { error in
        guard error == nil else { /* Handle error */ return }

        trigger.addActionSet(actionSet) { error in
            guard error == nil else { /* Handle error */ return }
        }
    }
}
```

```swift
// Create Sunset Presence Trigger
func createSunsetPresenceTrigger(withName name: String, actionSet: HMActionSet, home: HMHome) {
    var offset = DateComponents()
    offset.minute = -30
    let sunsetEvent = HMSignificantTimeEvent(significantEvent: .sunset, offset: offset)

    let presence = HMPresenceEvent(presenceEventType: .atHome, presenceUserType: .homeUsers)
    let predicate = HMEventTrigger.predicateForEvaluatingTrigger(withPresence: presence)
    let trigger = HMEventTrigger(name: name, events: [sunsetEvent], predicate: predicate)

    home.addTrigger(trigger) { error in
        guard error == nil else { /* Handle error */ return }

        trigger.addActionSet(actionSet) { error in
            guard error == nil else { /* Handle error */ return }
        }
    }
}
```

Camera Accessories

Camera Accessories

Show live streams

Camera Accessories

Show live streams

Display still images

Camera Accessories

Show live streams

Display still images

Control the camera settings

Camera Accessories

Show live streams

Display still images

Control the camera settings

Control the speaker and microphone

```swift
import HomeKit

class MyCameraClass: UIViewController {
    var liveStreamView: HMCameraView?

    func startCameraStream(for accessory: HMAccessory) {
        // Ensure this is a camera accessory
        guard let cameraStreamControl = accessory.cameraProfiles?.first?.streamControl else
                                                                                { return }

        cameraStreamControl.delegate = self
        cameraStreamControl.startStream()

        let liveStreamView = HMCameraView()
        self.view.addSubview(liveStreamView)
        self.liveStreamView = liveStreamView
    }
}
```

```
import HomeKit

class MyCameraClass: UIViewController {
    var liveStreamView: HMCameraView?

    func startCameraStream(for accessory: HMAccessory) {
        // Ensure this is a camera accessory
        guard let cameraStreamControl = accessory.cameraProfiles?.first?.streamControl else
                                                                                { return }

        cameraStreamControl.delegate = self
        cameraStreamControl.startStream()

        let liveStreamView = HMCameraView()
        self.view.addSubview(liveStreamView)
        self.liveStreamView = liveStreamView
    }
}
```

```swift
import HomeKit

class MyCameraClass: UIViewController {
    var liveStreamView: HMCameraView?

    func startCameraStream(for accessory: HMAccessory) {
        // Ensure this is a camera accessory
        guard let cameraStreamControl = accessory.cameraProfiles?.first?.streamControl else
                                                                                { return }

        cameraStreamControl.delegate = self
        cameraStreamControl.startStream()

        let liveStreamView = HMCameraView()
        self.view.addSubview(liveStreamView)
        self.liveStreamView = liveStreamView
    }
}
```

```swift
import HomeKit

class MyCameraClass: UIViewController {
    var liveStreamView: HMCameraView?

    func startCameraStream(for accessory: HMAccessory) {
        // Ensure this is a camera accessory
        guard let cameraStreamControl = accessory.cameraProfiles?.first?.streamControl else
                                                                                { return }

        cameraStreamControl.delegate = self
        cameraStreamControl.startStream()

        let liveStreamView = HMCameraView()
        self.view.addSubview(liveStreamView)
        self.liveStreamView = liveStreamView
    }
}
```

```
import HomeKit

class MyCameraClass: UIViewController {
    var liveStreamView: HMCameraView?

    func startCameraStream(for accessory: HMAccessory) {
        // Ensure this is a camera accessory
        guard let cameraStreamControl = accessory.cameraProfiles?.first?.streamControl else
                                                                        { return }

        cameraStreamControl.delegate = self
        cameraStreamControl.startStream()

        let liveStreamView = HMCameraView()
        self.view.addSubview(liveStreamView)
        self.liveStreamView = liveStreamView
    }
}
```

```swift
import HomeKit

class MyCameraClass: UIViewController {
    var liveStreamView: HMCameraView?

    func startCameraStream(for accessory: HMAccessory) {
        // Ensure this is a camera accessory
        guard let cameraStreamControl = accessory.cameraProfiles?.first?.streamControl else
                                                                                { return }

        cameraStreamControl.delegate = self
        cameraStreamControl.startStream()

        let liveStreamView = HMCameraView()
        self.view.addSubview(liveStreamView)
        self.liveStreamView = liveStreamView
    }
}
```

```
extension MyCameraClass: HMCameraStreamControlDelegate {
    func cameraStreamControlDidStartStream(_ cameraStreamControl: HMCameraStreamControl) {
        liveStreamView?.cameraSource = cameraStreamControl.cameraStream
    }
}
```

```
extension MyCameraClass: HMCameraStreamControlDelegate {
    func cameraStreamControlDidStartStream(_ cameraStreamControl: HMCameraStreamControl) {
        liveStreamView?.cameraSource = cameraStreamControl.cameraStream
    }
}
```

```
extension MyCameraClass: HMCameraStreamControlDelegate {
    func cameraStreamControlDidStartStream(_ cameraStreamControl: HMCameraStreamControl) {
        liveStreamView?.cameraSource = cameraStreamControl.cameraStream
    }
}
```

```
extension MyCameraClass: HMCameraStreamControlDelegate {
    func cameraStreamControlDidStartStream(_ cameraStreamControl: HMCameraStreamControl) {
        liveStreamView?.cameraSource = cameraStreamControl.cameraStream
    }
}
```

Summary

Summary

Join the MFi program

Summary

Join the MFi program

Take advantage of the ADK

Summary

Join the MFi program

Take advantage of the ADK

Create a great HomeKit accessory

Summary

Join the MFi program

Take advantage of the ADK

Create a great HomeKit accessory

Build a HomeKit app

More Information

https://developer.apple.com/wwdc18/231

HomeKit Lab	Technology Lab 12	Friday 12:00PM


WWDC18