Apple Home Kit: Overview

Table of Contents

1 Introduction

API (and library) which can be used by iPhone App developers to discover devices, request services to devices (synchronous call), and add hook to events (attach nonblocking, asynchronous handler).

1.1 Questions

- Then, what software API/middleware the device manufacturers (e.g. Honeywell thermostat developers) use?

2 Basic Classes

2.1 HMAccessory

Accessories (**HMAccessory**) are installed into homes and assigned to rooms. These are the actual physical home automation devices, such as a garage door opener or a thermostat. If the user doesn't configure any rooms, HomeKit assigns accessories to a special default room for the home.

Each physical accessory in the home is represented by one and only one accessory object. A single accessory provides one or more services, represented by instances of **HMService**. Corresponds to AllJoyn or Meadow device.

2.2 HMAction

HMAction is an abstract base class for actions in Home Kit.

2.2.1 HMActionSet

An **HMActionSet** object represents a set of actions (instances of **HMAction**) to be applied as a single set.

Action sets can be executed as a result of evaluating a trigger (instances of **HMTrigger**) or manually with **startExecutingActionSet:**. Actions in an action set are performed in an unspecified order. You create new action sets using the **addActionSetWithName:completionHandler:** method of **HMHome**. Very primitive but similar to the notion of "Workflow" in Meadow. Note that AllJoyn has no explicit, reified notion of "Action". In AllJoy, each "work" is directly implemented using C++/Java/etc – i.e. there is NO class named "Work".

2.3 HMCharacteristic

An **HMCharacteristic** object represents a specific characteristic of a service – for example, if a light is on or off, or what temperature a thermostat is set to. Loosely related to AllJoyn/Meadow property.

2.4 HMHome

Homes (**HMHome**) are the top level container, and represent a structure that a user would generally consider to be a single home. Users might have multiple homes that are far apart, such as a primary home and a vacation home. Or they might have two homes that are close together, but that they consider different homes – for example, a main home and a guest cottage on the same property.

An **HMHome** object allows you to communicate with and configure the different accessories in a home. Homes are the central organizing object for Home Kit.

Homes have three main purposes:

- Organize accessories into a number of rooms, which are themselves optionally grouped into zones.
- Serve as the main access point for communicating with and configuring accessories.
- Allow the user to define sets of actions that can be performed with a single operation, and triggers that can cause an action set to be performed at a specific time.

You dont create homes directly. Instead, you create them with the **addHomeWithName:completionHandler:** method of **HMHomeManager**.

AllJoyn/Meadow does not have explicit notion of "Container" of devices. Instead, it has a "tree-structured namespace and based on the name of device, we can impose some notion of containment..

2.4.1 HMHomeManager

A home manager object manages a collection of one or more homes. Use the home manager to add homes, get the list of homes, and track changes to homes with the home managers delegate.

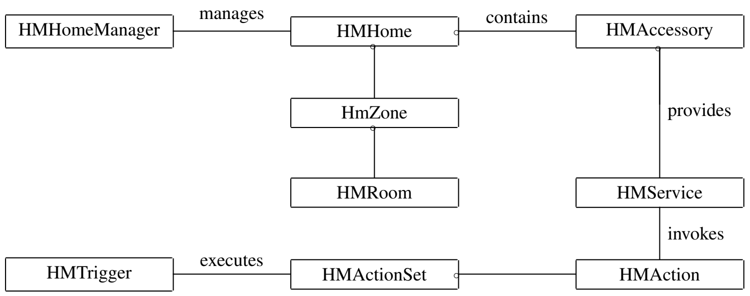

Figure 1: Class diagram

2.4.2 HMRoom

Rooms (**HMRoom**) are optional parts of homes, and represent individual rooms in the home. Rooms dont have any physical characteristicssize, location, etc. Theyre simply names that are meaningful to the user, such as living room or “kitchen”. Meaningful room names enable commands like, “Siri, turn on the kitchen lights.”

An **HMRoom** object is used to represent a room in a home. Accessories can be assigned to rooms.

You create new rooms using the **addRoomWithName:completionHandler:** method of **HMHome**. You can also group rooms into zones using instances of **HMZone**.

2.4.3 HMZone

Zones (**HMZone**) are optional groupings of rooms in a home. “Upstairs” and “downstairs” would be represented by zones. Zones are completely optional – rooms dont need to be in a zone. By adding rooms to a zone, the user is able to give commands to Siri such as, “Siri, turn on all of the lights downstairs.”

An **HMZone** object represents a collection of rooms that the user thinks of as a single area or zone – for example, “Living Room” and “Kitchen” might be grouped into a zone called “Downstairs”.

You create new zones using the **addZoneWithName:completionHandler:** method of **HMHome**. A zone can not span homes – that is, you cant create a zone that includes rooms from more than one home.

2.5 HMService

Services (**HMService**) are the actual services provided by an accessory. Accessories have both user-controllable services, like a light, and services that are for their own use, like a firmware update service. Home Kit is most concerned with user-controllable services.

A single accessory may have more than one user-controllable service. For example, most garage door openers have a service for opening and closing the door, and another service for the light on the garage door opener.

Services have characteristics that can be queried to discover their state or modified to cause the accessory to modify its behavior.

Roughly corresponds to Meadow/AllJoyn methods.

2.5.1 HMServiceGroup

An **HMServiceGroup** object represents a collection of accessory services, making it easier to address the services as a single entity. For example, a user might choose to group a set of lights together as “desk lamps”, and have another set of lights grouped as “ceiling lights”.

You create service groups using the **addServiceGroupWithName:completionHandler:** method of **HMHome**. Service groups are visible to Siri and allow users to control a group of services through Siri.

2.6 HMTrigger

An **HMTrigger** object represents a trigger event, used to trigger one or more action sets (instances of **HMActionSet**) when the conditions of the trigger are satisfied.

This class defines the basic behavior of triggers, but does not itself specify any criteria for firing a trigger. You should use instances of subclasses of **HMTrigger** to set up concrete triggers for actions.

Roughly corresponds to Meadow events or AllJoyn signals.

2.6.1 HMTimerTrigger

An **HMTimerTrigger** object represents a trigger based on periodic timers.

When a timer trigger is enabled using **enable:completionHandler:**, the system checks to verify that the timer triggers fire date, time zone, and recurrence rules yield a next fire date that is in the future.