Harman Developer SmartThings Integration Documentation

Release 0.0.1

Tyler Freckmann

January 11, 2016

Contents

Note: Github: github.com/tylerfreckmann/smartthings

This documentation is about integrating Harman Kardon WirelessHD SDK into the SmartThings platform. You can download the SDK at the Harman Developer web site. Please visit SmartThings to read about their platform.

This documentation is a work in progress.

Welcome to the HKWirelessHD SDK SmartThings Integration documentation. Here you will find information about how to hook up your HKWirelessHD SDK into SmartThings, and the basics of creating SmartApps for HKWirelessHD devices on the SmartThings platform.

This documentation is organized into a few different chapters:

HKWirelessHD SmartThings Integration Overview This chapter is the overview of how the HKWirelessHD SDK and SmartThings integrate, as well as presenting the sample SmartApps.

Getting Started This chapter describes how you can quickly get started building SmartApps that incorporate HK-WirelessHD devices.

Programming Guide This chapter explains the overall API you will use to build HKWirelessHD compatible SmartApps.

CHAPTER 1

Contents

1.1 Overview

1.1.1 HKWirelessHD SmartThings Integration Overview

HKWirelessHD is an SDK that enables control of certain Harman Kardan devices (speakers) over Wi-Fi. It supports iOS/Android development and utilizes iOS to enable a RESTful Web API. Please visit the Harman Developer web site to download and familiarize yourself with this SDK.

Note: As this SDK is still in its early stages, the HKWHub iOS app is necessary in order to support REST Web API functionality, which SmartThings integration relies upon. We hope to eliminate this limitation in the future by exploring cross-platform and true "cloud" solutions as well as exploring other ways to integrate with SmartThings, but please bear with us for the time being.

SmartThings is a platform designed for the Internet of Things and "Smart Home" automation. They offer a platform to develop "SmartApps," which connect different smart devices in the home. This platform allows integration of SmartThings devices as well as smart devices developed by other companies by abstracting away much of the architecture behind the Internet of Things. Please familiarize yourself with the SmartThings platform as well.

What's Included

- **SmartThings Integration files (for reference):**
  - HKSpeaker.groovy
  - HKServiceManager.groovy

How It Works

The basic idea of the SmartThings platform is to connect all of your devices and define "SmartApps" to automate them. (These SmartApps run on the SmartThings platform).

From the SmartThings documentation:

> Each device in SmartThings has "capabilities", which define and standardize available attributes and commands for a device. This allows you to develop an application for a device type, regardless of the connection protocol or the manufacturer.

This abstraction is defined by the "Device Type Handler", which in our case is HKSpeaker.groovy.

Devices connect to the SmartThings platform in one of three ways:

- connecting directly to the SmartThings Hub
- connecting through the "cloud"
- connecting through the Local Area Network (LAN)

HKWirelessHD compatible speakers connect through the cloud by providing SmartThings with a REST API, handled by the HKWHub iOS app. Devices that don't connect directly to the SmartThings Hub (as in our case) require a "Service Manager" SmartApp. The HKServiceManager.groovy is what allows SmartThings to make REST requests and control HKWirelessHD speakers through the HKWHub app.

1.2 Getting Started

1.2.1 Getting Started

Setup

Download HKWHub iOS app

- Go to the Harman Developer web site and click on Developer Tools > Download SDK > iOS > Sample Apps.
- Find the HKWHub App and download it.

Prepare your router so that the HKWHub app can handle REST requests

- Configure your router's DMZ host to be the IP address that is displayed on the HKWHub app's home page.

Register for a free SmartThings developer account

- Go to the SmartThings web site, click on "Getting Started", and follow the brief SmartThings tutorial.

Incorporate the HKWirelessHD speaker Device Handler and Service Manager into your account

- **Create a Device Handler called `HKSpeaker` and paste the contents of HKSpeaker.groovy into it.**
  - Publish this Device Handler for yourself in "My Apps"
- **Create a SmartApp called `HKServiceManager` and paste the contents of HKServiceManager.groovy into it.**
  - Configure the Service Manager's ipAddress setting to be your router's public IP address (which you can find by googling "my ip address" from the device running the HKWHub app).
  - Publish this Service Manager for yourself in "My Apps"

Now you should be able to create your own SmartApps that incorporate HKWirelessHD compatible speakers! In the SmartThings app on your phone, you can press the "+" symbol and scroll all the way to the right to My Apps to download the HK Service Manager. When you open the Service Manager, you should be walked through a process to select which HKWirelessHD compatible speakers you want to connect to SmartThings. Once you've installed the Service Manager (by clicking "Done"), you will see HKWirelessHD speakers in your list of devices, and you should be able to control them from that interface.

The Service Manager has now incorporated your HKWirelessHD speakers into the SmartThings platform as `musicPlayers`, which have the capabilities defined in the SmartThings documentation under Music Player.

Creating a Sample SmartApp

In this section we will explain how to create a SmartApp that incorporates an HKWirelessHD device. We will create a simple SmartApp called “Welcome Home,” that will play a song when a door is opened.

Step 1: Create a new SmartApp

Go to your developer environment page, and create a new SmartApp.

Step 2: Select the Contact Sensor and Speaker

In the `preferences` definition, create sections for the user to select a `contactSensor` and a `musicPlayer`:

```
preferences {
        // Select the contact sensor
        section ("Which contact sensor?") {
                input "contact", "capability.contactSensor", title: "Which?"
        }
        // Select the speaker
        section ("Which HK Speaker?") {
                input "speaker", "capability.musicPlayer", title: "Which?"
        }
}
```

Step 3: Subscribe to Event

In the `initialize` method, subscribe to the `contactSensor`:

```
def initialize() {
        subscribe(contact, "contact", contactHandler)
}
```

Step 4: Define the event handler

Create a new method to handle the `contact` event:

```
def contactHandler(evt) {
        log.debug "door ${evt.value}"

        // If the door is open, play a song
        if (evt.value == "open") {
                speaker.play();
        }
}
```

Step 5: Publish it and test it

Publish your SmartApp for yourself and see if it works!

1.3 Programming Guide

1.3.1 Programming Guide

In this chapter, we will explain what the Device Type Handler for the HKWirelessHD speaker does.

Music Player Device Type Handler for HKWirelessHD Speakers

Note: The Music Player was originaly defined for Sonos speakers, so some of the attributes and commands are just stubs, or do not make the most logical sense. Hopefully, in the future, the SmartThings Music Player definition will also be well suited for HKWirelessHD speakers.

Attributes:

Attribute	Type	Possible Values
status	String	`"playing"` `"paused"`
level	Number	`0-50`
trackDescription	String	`description of the current playing track`
trackData	JSON	`not implemented`
mute	String	`"muted"` `"unmuted"`

Commands:

play() If the speaker is paused, resume playback, otherwise, play first song in playlist

pause() Pause music playback

stop() Stop music playback

nextTrack() Advance to next track

playTrack(string) Play the track matching the given string (the string is the title for the track to be played)

setLevel(number) Set the volume to the specified level (0-50)

playText(string) Not implemented

mute() Mute playback

previousTrack() Go back to the previous track

unmute() Unmute playback

setTrack(string) Not implemented

resumeTrack() Resume music playback

restoreTrack(map) Not implemented

SmartApp Example:

```
preferences {
  section("Title") {
    input "player", "capability.musicPlayer", title: "music player", required: true, multiple: false
    input "frontDoor", "capability.contactSensor", title: "front door", required: true, multiple: fal
  }
}

def installed() {
  subscribe(frontDoor, "contact", myHandler)
}
```

```
def myHandler(evt) {
  if("open" == evt.value) {
    player.play()
  }
}
```

CHAPTER 2

Search

- search