Getting Started with the AllJoyn™ Lighting Service Framework 15.04

Lighting Controller Service

October 05, 2015

Contents

1 Introduction

The AllJoyn™ Lighting service framework provides Lighting OEMs and application developers a means to build a complete lighting solution supported by the AllJoyn framework.

The Lighting service framework comprises two service components:

- Lamp service: Enables an embedded lighting device (such as a connected light bulb) to be controlled by the Controller service.
- Controller service: Enables AllJoyn applications (e.g., an application running on a smartphone) to control the Lamp service.

Lighting OEMs can embed the Lamp service in the firmware of their lighting products to enable Smart Lighting features.

The Controller service is designed to find all devices running the Lamp service on the AllJoyn network and provide additional functionality to control the lighting devices in a variety of ways (e.g., create and control a group of lights simultaneously and applying custom lighting effects). The Controller service can reside in lighting device, a hub, router, gateway, mobile device (smartphone or tablet), desktop, or home automation controller.

Developers can build AllJoyn applications that will communicate with the Controller service.

Note This documentation is for the Controller Service.

Figure 1 illustrates the Lighting service framework components and their interaction with an OEM device.

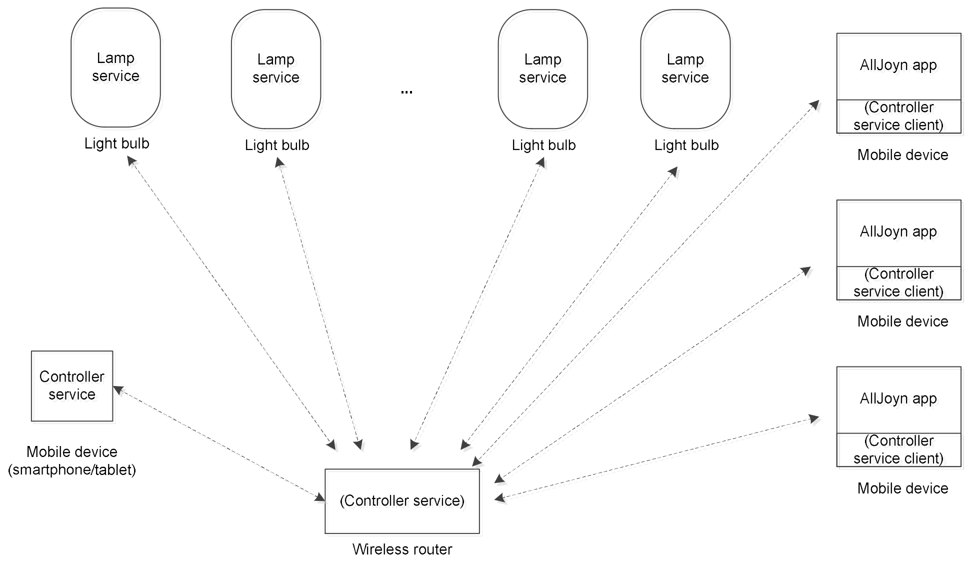

Figure 1: Lighting service framework components

1.1 Purpose

This document provides an overview of the Lighting service framework and detailed instructions on how to start using the Controller service. The following topics are covered:

- Overview of the Lighting service framework
- Configuring the development environment for the Controller Service
- Implementing device-specific methods of the Controller Service
- Building and running the Controller service on Linux

1.2 Scope

This document is written for OEMs and developers and assumes competent knowledge of AllJoyn development. It will focus on the main components of the Lighting service framework.

1.3 References

The following documents are references found on the AllSeen Alliance web site's *Docs/Downloads* section.

- *AllJoyn™ Framework Tutorial*
- *Introduction to the AllJoyn™ Framework*
- *Introduction to AllJoyn™ Thin Library*
- *AllJoyn™ Troubleshooting Guide*
- *Configuring the Build Environment (Linux Platform)*
- *Getting Started AllJoyn Lighting Service Framework – Lamp Service*

1.4 Acronyms and terms

Term	**Definition**
AllJoyn Standard Core Library	An application or AllJoyn daemon process that contains the full implementation of the AllJoyn message bus.
Announcement	A sessionless signal whose payload includes published services' interfaces and metadata that are used for discovery.
Configuration service framework	Software layer that enables devices to provide remote configuration of AllJoyn service frameworks' metadata (ConfigData) in a session.
Controller service	Maintains the Lamp Group, Scene, and Presets on behalf of the Lamps. The Controller service is a logical entity that can reside in a hub, router, gateway, smartphone/tablet, desktop, or home automation controller.

Term	Definition
Lamp group	A logical grouping of Lamps allowing them to be controlled simultaneously as they are a single Lamp
Lamp service	Client service implemented in a Lamp, enabling it to be controlled by the Controller service. A Lamp service can reside in a lamp, luminaire, light switch, plug, outlet, or socket adapter and can leverage the AllJoyn Standard Core or Thin Client Library.
Lighting service framework	AllJoyn framework that provides a means to build a complete lighting solution. It consists of the Controller service and Lamp service.
Notification service framework	Software layer that enables devices to send or receive human-consumable notifications.
Onboarding service framework	Software layer that enables devices to provide remote configuration (OnboardingData) and control (driver mode) over a device's onboarding process to a Wi-Fi AP over an AllJoyn session.
Preset	Saving the Lamp or Lamp group state (Hue, Saturation, Color Temperature, Brightness) and giving it a friendly name.
Scene	Lighting setup for a particular event, for example, dimming the lights when a movie is on. The user can create a scene that they can apply to a set of lights.

2 Overview

The Lighting service framework is part of a comprehensive lighting system that allows OEMs and developers to build lighting hardware that performs the following functions:

- Runs the Lamp service
- Create applications that interface with the Controller service to control lighting hardware.

2.1 Lamp service

The Lamp service uses the AllJoyn Thin Core Library and is meant to be run on embedded hardware like a light bulb. The Lamp service uses several AllJoyn service frameworks to allow the embedded hardware to communicate with other devices on the AllJoyn network.

1. The Onboarding service framework connects the hardware to the Wi-Fi network.
2. Once onboarded, the About feature allows the Lamp service to broadcast its presence on the network.
3. Any device that receives the About announcement can connect directly to the Lamp service and use the Configuration service framework to modify settings as well as perform other operations like factory reset.
4. The Lamp service uses the Notification service framework to notify the user of any problems with the light.

2.2 Controller service

The Controller service uses the AllJoyn Standard Library and is configured to listen for any lighting devices running the Lamp service that may be present on the network. The Controller service will discover any lighting device connected to the network by receiving the About announcement and then connecting directly to the device using an AllJoyn session. The Controller service can then perform operations on the Lamp service such as:

- Toggle the light on and off
- Retrieve power consumption and light details
- Set the brightness or hue of the light

The Controller service can control every lighting device connected to your network. Developers can then build AllJoyn applications that communicate directly with the Controller service and control their lighting devices. The Controller service also provides developers the ability to grouped lights together and control them simultaneously, and create a custom lighting experience using scenes. Developers have the freedom to create a variety of applications to control their lights.

In order to provide a easy way for developers to use Controller service, Controller service client was created. Controller service client provides a set of C/C++ APIs to application developers to access Controller service.

Figure 2 illustrates the Controller service and Controller service client framework

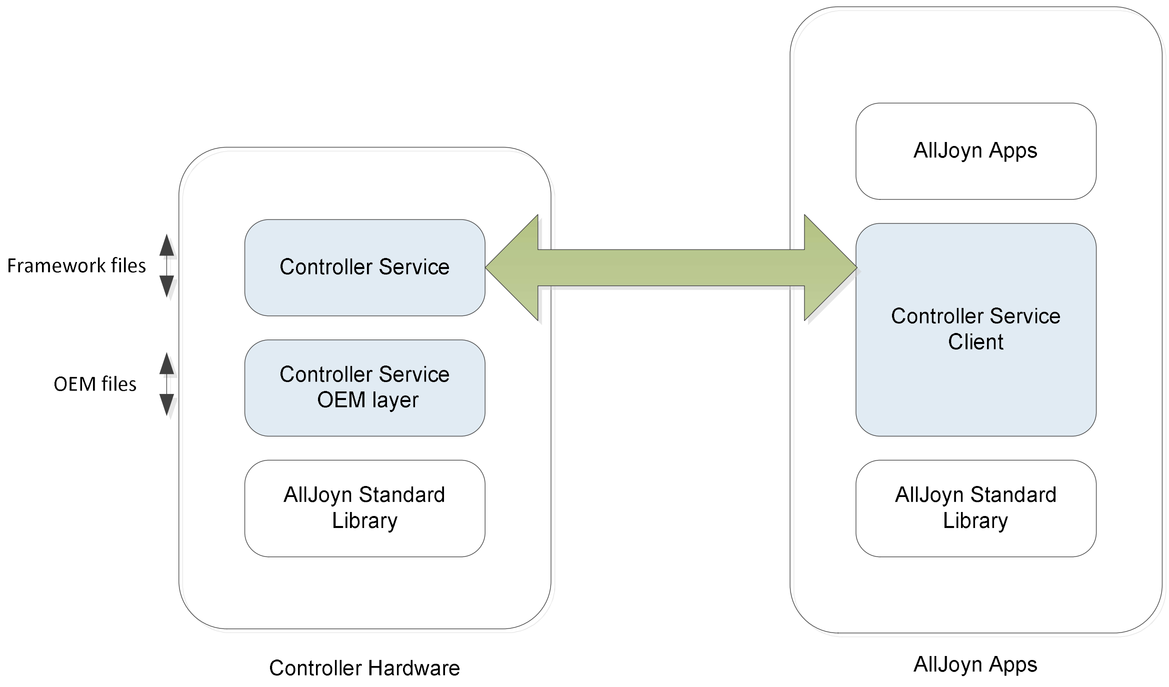

Figure 2: Controller Service and Controller Service Client

3 Configuring the Development Environment

This chapter provides instructions to configure your Linux environment to build the Controller service as well as the AllJoyn core.

Verify your allseenalliance.org account has been created and you have received email confirmation before completing the procedures in this chapter.

3.1 Configure the Linux environment

1. Access the Configuring the Build Environment (Linux Platform) document on the AllSeen Alliance website.
2. Follow the instructions in Section 2 only up to the section titled Obtain the AllJoyn source.

4 Obtaining and building the Controller Service

This chapter provides instructions to download the Controller service and AllJoyn core modules from the AllSeen Alliance web site.

4.1 Obtain the source code

1. Open a terminal window.
2. Change to your working directory.

   ```
   cd <path_to_working_directory>
   ```

3. In your working directory, create a folder called allseen.

   ```
   mkdir allseen/
   ```

4. Use the following commands to clone the base_tcl/, and base/ source code repositories.

   ```
   git clone https://git.allseenalliance.org/gerrit/services/base_tcl.git
   git clone https://git.allseenalliance.org/gerrit/services/base.git
   ```

5. Create a folder called core/ inside the allseen directory and change into the core folder.

   ```
   mkdir core/
   cd core/
   ```

6. Use the following commands to clone the alljoyn/, ajtcl/, and service_framework/ repositories.

   ```
   git clone https://git.allseenalliance.org/gerrit/core/alljoyn.git
   git clone https://git.allseenalliance.org/gerrit/core/ajtcl.git
   git clone
   https://git.allseenalliance.org/gerrit/lighting/service_framework.git
   ```

Note Please refer to https://wiki.allseenalliance.org/tsc/connected_lighting for the particular Lighting release and the corresponding release tag/commitIDs.

Note Technically, the lamp service (required /base_tcl.git and /ajtcl.git) is not required to build Controller Service. However, this is included for enabling the End-to-End Lighting testing.

4.2 Build Controller Service

1. Open a terminal window
2. Change to your working directory.

   ```
   cd <path_to_working_directory>/core/service_framework
   ```
3. Call scons to build the Controller Service for 64-bit Linux

   ```
   scons OS=linux CPU=x86_64 WS=off V=1
   ```

   Note For 32-bit Linux, use the, `scons OS=linux CPU=x86 WS=off V=1`

5 Integrating the Controller Service

This chapter provides instructions on how to integrate the Controller service into a OEM's Lighting Controller hardware platform.

Note This chapter does not target any specific hardware platform. The Lamp service partitions two set of files: framework and OEM. Framework files are intended to be OEM- and hardware-independent. It is strongly recommended not to change framework files in order to maintain compatibility across all AllJoyn devices. OEM files are designed to consist of hardware dependent implementation such as drivers.

Figure 3 illustrates the Controller service components.

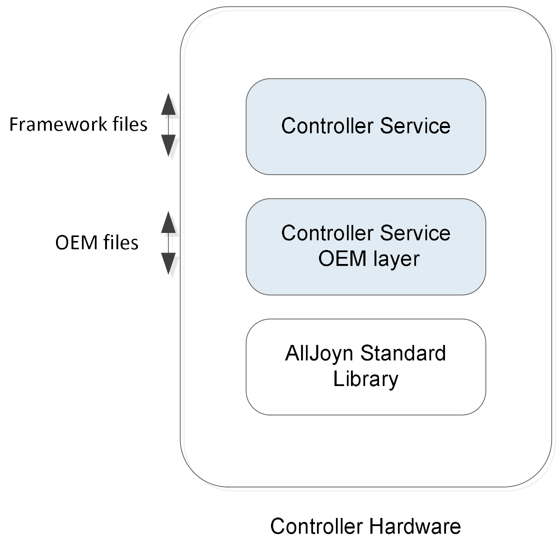

Figure 3: Controller service components

5.1 Modify the Controller service's properties

The Controller service contains a hard-coded list of read-only Lighting-related OEM properties. These include properties such as the maximum number of lamps and maximum number of Lighting entities such as Groups and Scenes that depends on the supported behavior of the hardware running the Controller service.

1. In a terminal, change into the following directory:

   ```
   cd <path_to_working_directory>/allseen/core/service_framework/
      standard_core_library/lighting_controller_service/inc/
   ```

2. In your favorite text editor, open OEM_CS_Config.h.
3. Scroll to the bottom to view a definition of the lsf{} data structure.

   Note You can see the definition of the #define in OEMCode.h in the …./standard_core_library/lighting_controller_service/inc/ directory.

4. Changes the values in each #define to match your hardware specification. For example, how many lamps your hardware can be supported?
5. Save and close all modified files.

5.2 Modify the Controller service's About details

The Controller service uses the AllJoyn About feature to broadcast its existence, along with its About properties such as model number, over the AllJoyn network. Any device on the network that is configured to listen for the Controller service About announcements can receive the announcement, discover what the device is, and see which AllJoyn interfaces are supported by the device.

Complete the following procedure to modify the About details.

1. In a terminal, change into the following directory.

   ```
   cd <path_to_working_directory>/allseen/core/service_framework/
      standard_core_library/lighting_controller_service/src.
   ```

2. In your favorite text editor, open OEM_CS_Config.cc.
3. Modify default values defined in PopulateDefaultProperties() as needed

   Throughout the file, you can see the default values of the PropertyStore object that is used by the AllJoyn About feature. For example, you can see the default language is defined as "en" for English.

   This is where you can add support for other languages, change the manufacturer of the device, and modify fields like mode, software version, hardware version, etc. These are the details that are broadcast by the AllJoyn About announcement.

Note If you have never used the About feature, refer to the AllJoyn documentation listed in *References* for an explanation of the fields that are present in the AllJoyn About announcement.

4. Save and close the modified files.

Note If you have a OEMConfig.ini in the current working directory, AllJoyn About service will use the About properties defined in OEMConfig.ini instead of the ones defined in PopulateDefaultProperties in OEM_CS_Config.cc.

5.3 Configure OEM-specific functions

The Controller service defines a specific set of Lighting-related OEM functions such as Synchronization timestamp, Lighting Controller Service ranking, and defining the default Lamp state whose implementation may vary based on hardware specification.

Complete the following procedure to implement the functions that start with a "OEM_" prefix.

1. In a terminal, change into the following directory:

```
cd <path_to_working_directory>/allseen/core/service_framework/
   standard_core_library/lighting_controller_service/src/.
```

2. In your favorite text editor, open OEM_CS_Config.cc. You should see a set of functions whose names start with "OEM_".
3. Implement the following OEM functions based on your hardware implementation. Details on each OEM function are documented in the code. Some examples about these OEM functions follow:
   - OEM_CS_GetFactorySetDefaultLampState() – provides the default Lamp State to Lamp Service
   - OEM_ CS_GetSynctimeStamp() – provides the 64-bit timestamp to all Lamp services in case Lamp OEMs require to have time for time synchronization feature.
   - OEM_ CS_GetRankParam_Power() – provides a enum value (in OEM_CS_Config.h::_OEM_CS_RankParam_Power) indicating how the device running the Lighting Controller is powered.
4. Save and close the modified files

6 Test Harness for Controller Service

A test harness is bundled with the Controller service source code in AllSeen Alliance git repository, and is based on Linux Ubuntu platform.

The test harness is located at:

- `…/allseen/core/service_framework/stanard_core_library/lighting_controller_client/samples/`

  Note For Lighting Controller service

The test harness executable is located in

- `…/allseen/core/service_framework/build/linux/standard_core_library/lighting_controller_service/bin/`**`lighting_controller_service`**

The build and execution instructions are in the README.txt in the folder specified.