ALLSEEN ALLIANCE

Technical Steering Meeting

April 1, 2014

Antitrust Compliance Notice

- AllSeen Alliance meetings involve participation by industry competitors, and it is the intention of AllSeen Alliance to conduct all of its activities in accordance with applicable antitrust and competition laws. It is therefore extremely important that attendees adhere to meeting agendas, and be aware of and not participate in any activities that are prohibited under applicable US state, federal or foreign antitrust and competition laws.

- Examples of types of actions that are prohibited at AllSeen Alliance meetings and in connection with AllSeen Alliance activities are described in the AllSeen Alliance Antitrust Policy. If you have questions about these matters, please contact your company counsel, or if you are a member of AllSeen Alliance, feel free to contact Lee Gesmer or Andrew Updegrove, of the firm of Gesmer Updegrove LLP, which provides legal counsel to AllSeen Alliance.

Reminder:

This call is being recorded

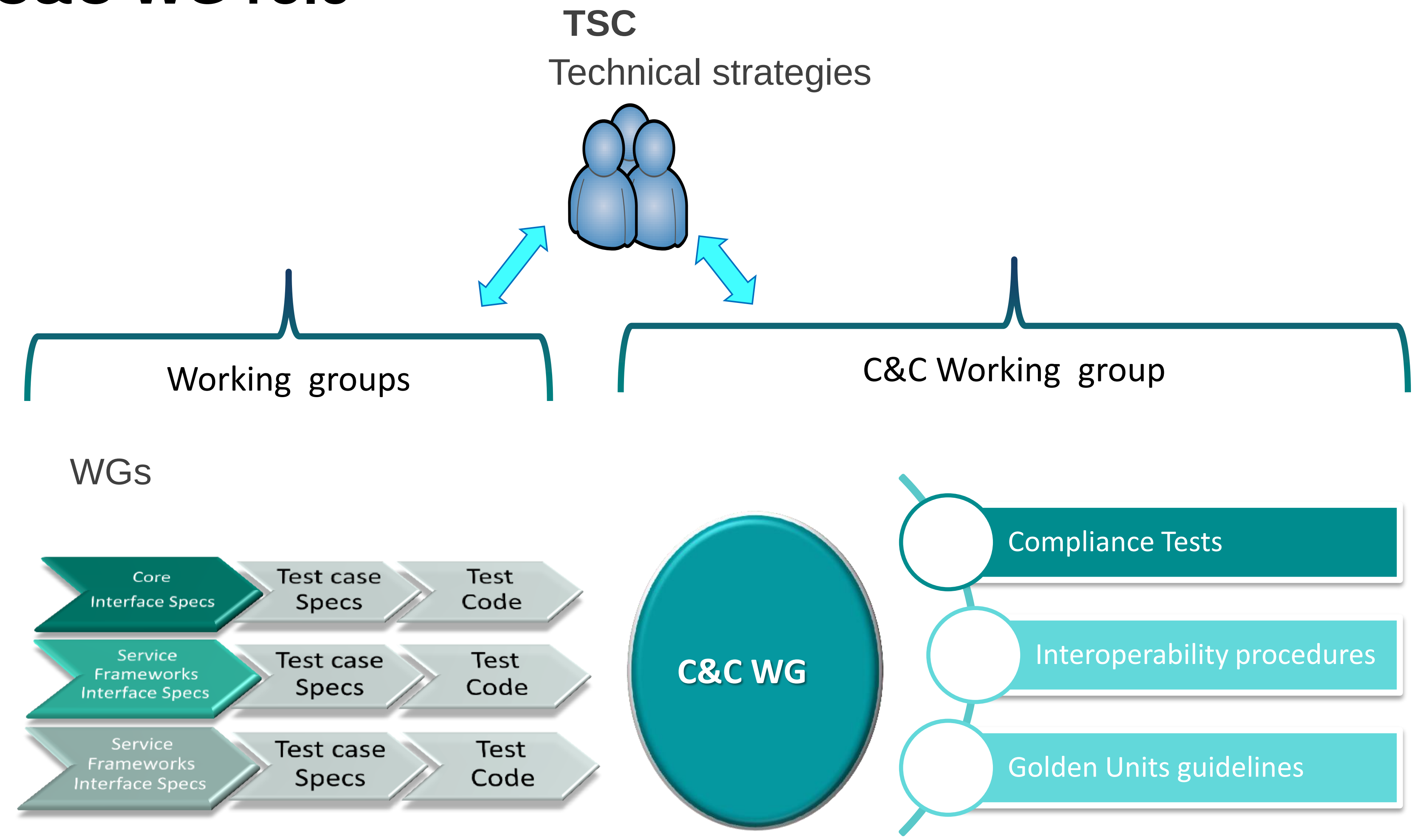
Agenda

- Approve minutes from last call
- C&C WG requirements for core and service compliance testing
  - Discussion and vote
- Hackfest logistics
  - Vote to authorize H/W purchase
- Lighting service proposal
  - Presentation by LIFX and vote on proposal

C&C Requirements for Core and Service WGs

Telis Kaleas

C&C WG role



TSC

Technical strategies

Working groups

C&C Working group

WGs

Core Interface Specs | Test case Specs | Test Code

Service Frameworks Interface Specs | Test case Specs | Test Code

Service Frameworks Interface Specs | Test case Specs | Test Code

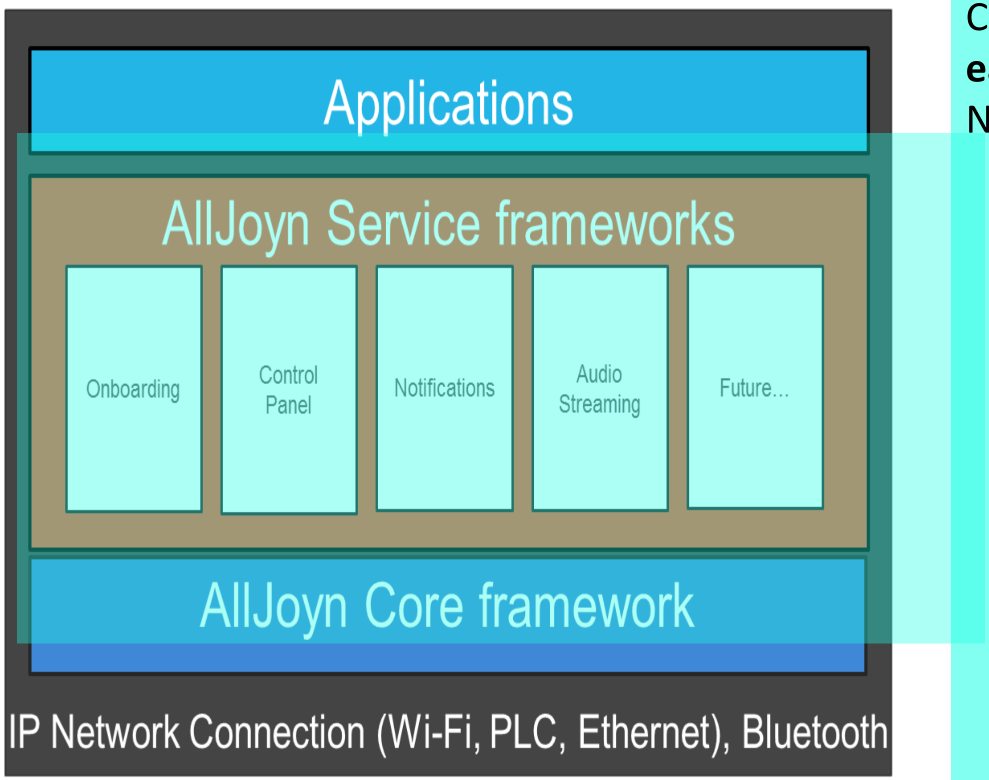
C&C WG

Compliance Tests

Interoperability procedures

Golden Units guidelines

C&C Requirements



Committer or Contributor of **Core** (About…) **and each Service Frameworks** (Config, Control Panel, Notifications,Lighting…) MUST Provide:

- The source code of the new (or updates) of core and Service Frameworks
- Interface Specs associated with that Core Interfaces or Service Framework
- Test case Specs which are written against the Interface Specs
- Provide Test code implementing the Test case Specs

Service Frameworks Interface Specs

- https://allseenalliance.org/sites/default/files/resources/aj_notif_sf_interface_spec.pdf
- Common Template sections:
  - Introduction
  - Service Messages
  - Service Interface
  - Call Flows
  - Introspect XML

Test Case Specification Template

- Proposed template for writing specs
- Includes instructions to assist with writing
- Template sections:
  - Introduction
  - Environment setup (Requirements, Preconditions, Test Execution Notes, and Test Parameters)
  - List of Test Cases (Test Case Id, Title, Objective, Procedure, and Expected results)

Lighting Service Framework Proposal

Marc Alexander
VP Engineering, LIFX Labs

Introduction

- We believe there is a strong need in the market for a connected lighting framework based on open standards to accelerate the pace of innovation of connected lighting over IP-based networks such as Wi-Fi.
- We believe that AllJoyn provides the foundation for delivering a compelling solution for the market with the addition of a lighting service.

LSF Objectives

- **Standard** - Provide a standard AllJoyn Framework for Lighting Control over IP Networks.
- **Interoperable** - Make lighting interoperable with other connected things. Across brands, platforms, and verticals.
- **Open** - Open Source, Open Governance. Code is the specification.

LSF Core Features

- Control
- Groups
- Scenes
- Effects

Control

- Lamps can be controlled via the LSF
- On / Standby
- Hue
- Saturation
- Brightness
- Color Temperature
- Lamp Output (read-only in Lumens)
- Power Draw ((read-only in mA)

Groups

- Lamps can be grouped and controlled together
- Create / Update / Delete Group
- Groups can be nested
- Groups can be named by the user
- Example:
  - Whole Home Group
  - Kitchen, Family Room, Dining Room Groups
  - Kitchen Nook, Kitchen Under Counter, Kitchen Floods

Scenes

- Scenes are preferences saved by the user to set a particular lighting mood or simply store a setting for convenience and future recall.
- Create / Update / Delete Scene
- Scenes can be grouped
- Scenes can be named by the user
- Scenes can contain groups and individual Lamps
- Scenes can contain an effect
- Example “Dining” Scene comprised of:
  - Wall sconce group transitions to blue hue, 30% brightness
  - Art light group transitions to warm color temp, 40% brightness
  - Chandelier group transitions to cool color temp, 15% brightness
  - group transitions to cool color temp, 15% brightness

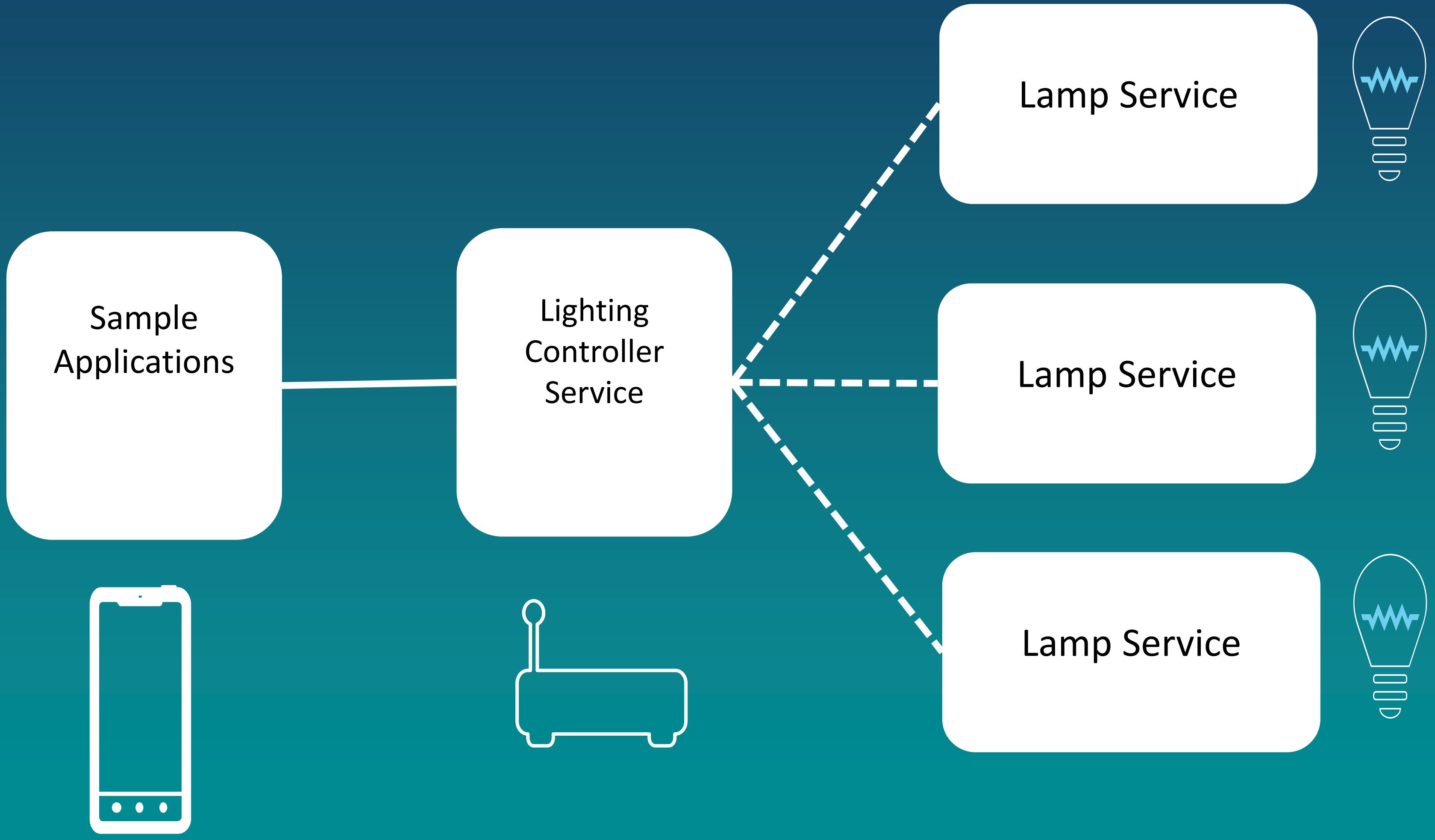
Effects

- The LSF defines a transition effect that the Light OEM may elect to implement.
- Custom effects may be developed by the Light OEM and made discoverable by developers.
- Effects are applied to Lamps or Lamp Groups as a part of a scene.

Developing with LSF

- Sample applications including client API’s available for both Android and iOS.

LSF Software Architecture



AllSeen Alliance
May 1st Hackfest Details

Event Summary

- May 1. 8:30a-5:30p. San Jose. Co-located with Embedded Linux Conference
- Agenda:
  - 4 hours of tutorial/samples + hands-on exercises
  - 4 hours of tinkering in groups of 6
- Maximize time by loaning:
  - LinuxVM on USB stick with build env and software preloaded
  - Arduino Yuns with LED and joystick shields

Goal

- Get developers excited about AllJoyn
  - Hands-on with Linux on Arduino Yuns with LED matrix and joystick shields
    - Everyone likes to make hardware do something
  - Teams of 6 to build something creative

- Teach developers about AllJoyn on Linux:
  - Core AllJoyn: Interfaces, methods, properties, signals, sessions
  - Notifications: sending, receiving.
  - Control Panel: XML, adding buttons, labels,

Detailed Agenda

Start	End	Item
8:30	9:30	Registration, breakfast, equipment loan pickup
9:30	10:00	Intro: Overview, LinuxVM, software, hardware
10:00	10:30	AllJoyn Overview
10:30	11:15	AllJoyn Interface tutorial walkthrough + hands-on exercise
11:15	11:30	Break
11:30	12:15	AllJoyn Notification tutorial walkthrough + hands-on exercise
12:15	1:00	AllJoyn Control Panel tutorial walkthrough + hands-on exercise
1:00	2:00	Lunch
2:00	5:30	Team tinkering and hacking

Provide H/W

- Important to get hands-on with real hardware
  - Closer to real IoT use cases
- Simplifies setup by providing attendees with everything needed to get started
  - Hardware
  - USB Stick with LinuxVM with all software, build pre-loaded
  - Attendees only needs to pre-install VM player.
- Hardware can be reused for future events
  - One time expense for this Hackfest

Equipment Costs

Item	What/Why	Quantity	Unit Cost
Arduino Yun	Linux + Arduino on one board; one per attendee	100	$72
Arduino LED Matrix Shield	14x9 LED matrix for output	75	$40
Arduino Joystick Shield	Joystick for input	25	$10
Wi-Fi Router	One per team of 6	20	$110
USB Stick	32GB for LinuxVM with software preloaded	100	$20
	Total with estimated 20% tax/shipping		**$17,580**

Budget

- Event Costs
  - $30K for venue rental, AV equipment rental, Wi-Fi, breakfast/lunch for attendees, logistics
  - $20K for one time equipment spend
- Budget for 2014: $120K
  - Originally allocated $30K for 4 Hackfests.
  - Will only have 3 Hackfests this year, so should have $10K left:
    - $30K x 3 + $20K (one time)

Tidbits

- Course content will be repurposed for online hands-on tutorial
- Next hack fest can potentially be focused on building service frameworks

Course Content

AllJoyn Interface Tutorial

- Provided software
  - Yun:
    - Sends about announcement with helloworld() method on helloworld interface.
    - Helloworld() prints to console
    - Sends helloworld signal after session joined
  - Linux:
    - Listen for about announcements an calls helloworld() method
    - Signal handler to receive helloWorldSignal() and prints to console
- Exercises (depending on shield)
  - Yun
    - Add putChar() that prints letter to LED matrix. Maybe also add putBitmap() that puts a bitmap on the LED matrix
    - Sends buttonChanged signal when button state changes. Add buttonState property to read buttonState
  - Linux
    - Accepts text from console and calls putChar() and/or putBitmap()
    - Register to receive buttonChanged signal. Calls getProperty on buttonState when signal received and prints to console

AllJoyn Notification Tutorial

- Provided Software
  - Yun
    - Notification consumer prints to console with notification received
    - Add producer code lib (that does nothing)
  - Linux
    - Accepts text from console and sends out as notification
    - When receives notification, prints to console
- Exercises (depending on shield)
  - Yun
    - When receive notification, prints firsts char to LED matrix.
    - When receives button or joystick changes, send notification

AllJoyn Control Panel Tutorial

- Provided Software
  - Yun
    - Implement Control Panel Controllee with basic button and spinner. When activated, prints to console
    - Implements Control Panel Controlee with a counter as a label. Update label and publish every 5 seconds
  - Linux
    - Control Panel CLI viewer
- Exercises (depending on shield)
  - Yun
    - Add button to update the LED maybe blink, modify picker to print different characters,
    - Add label for joystick/button states with publish; add button to change Arduino LED state

Team Project Ideas

- Create a digital scrolling marquee with Yuns with LED matrixes when an AllJoyn notification is received
- Create an AllJoyn Control panel to display fun things on a set of Yuns with LEDs
- Create a simple game (pong, breakout, Flappy Bird!) with the Yuns and the LED matrix and Joystick shields
- Display text on the Yun's LED matrix and scroll with the joystick

Thank You.