ALLSEEN ALLIANCE

Technical Steering Meeting

October 28, 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

1. Approve minutes from previous meeting
2. Topics for TSC face-2-face at Alliance Summit
3. Vote on Lighting App Proposal
4. C&C Requirements for WGs
5. AllJoyn.js Status Update

TSC F2F Topics

Some ideas for the F2F

- Opportunity to have to in depth discussion around
  - Workgroup and Project Workflow
  - Workgroup structure
  - Hackfests and developer outreach
  - Roadmap and priorities
  - What are we doing right and what are we doing wrong?
  - Other thoughts and ideas

Vote on Lighting App Proposal

Lighting Apps Proposal Summary

- Container project for Android and iOS sample applications that interact with the Lighting Service Framework,
- One place to go to find LSF applications
- Proposed Project Name and Working Group
  - lighting/apps.git
  - Will reside in the existing Connected Lighting WG
- Release date
  - Code is ready to be submitted on project approval

C&C Requirements for WGs
Telis Kaleas

C&C Requirements for WGs

The Committer or Contributor of Core (About…)and each Service Framework (Config, Control Panel, Notifications, Lighting, etc…) MUST Provide to C&CWG:

- *The source code of the new (or updates) of Core and Service Frameworks (Email of the location))*
- *Interface Definitions associated with the Core Interfaces or Service Framework Interface Definitions*
- *Test Case Specs which are written against the Interface Definition of the Service Framework*
- *Provide Test Code implementing the Test Case Specs*

- Send documents to the Certification and Compliance (Technical) mail list.
- Allow 2 weeks lead time for C&C WG to approve documents
- Documents MUST be approved by C&CWG BEFORE a product can be certified with that service framework
- Interface definition and Test Case Spec templates can be found here: https://wiki.allseenalliance.org/compliance/overview

AllJoyn.js Status Update

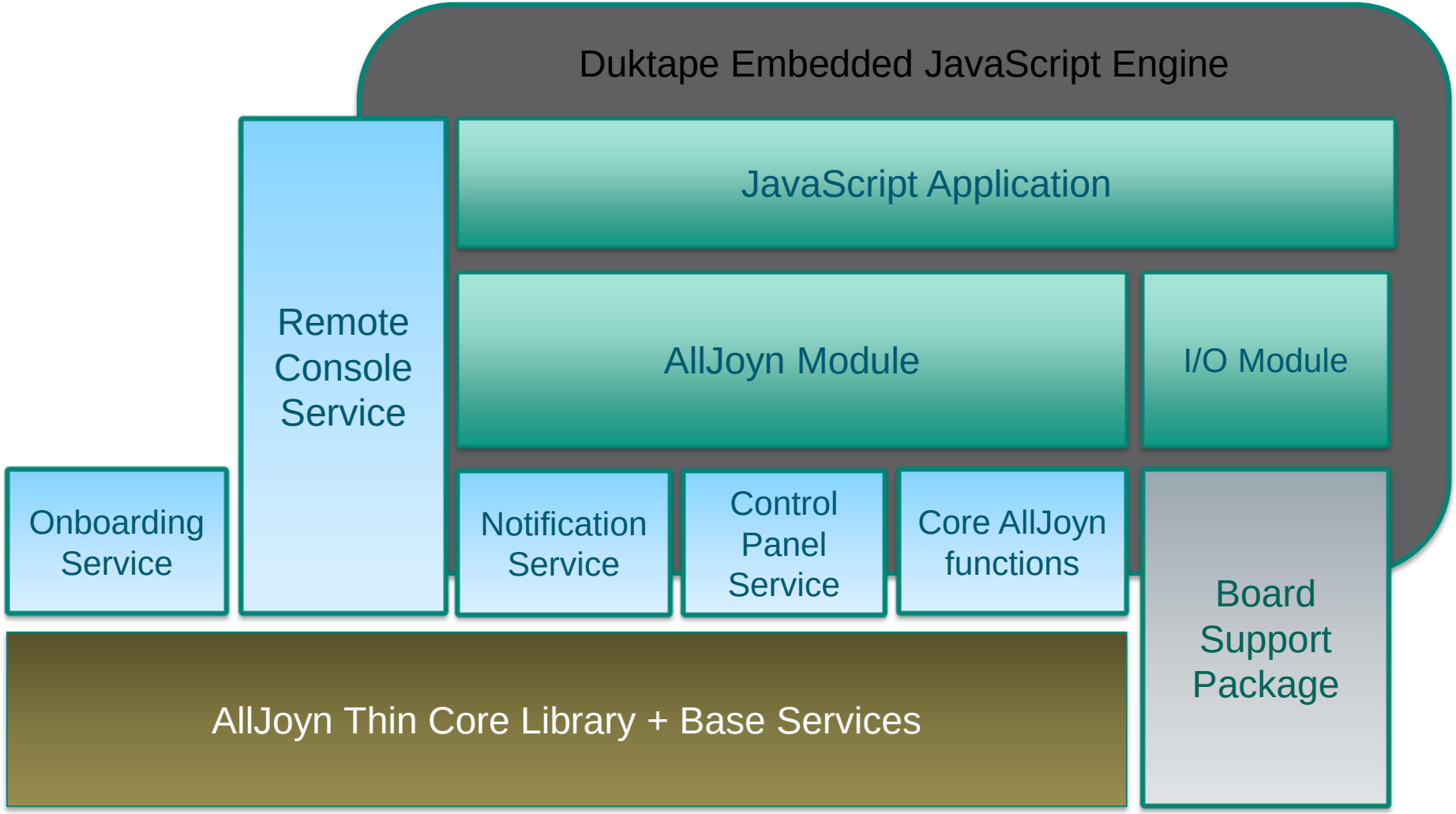
Alljoyn.js Update

- Code available for download from Alliance git repo
  - https://git.allseenalliance.org/cgit/core/alljoyn-js.git/
- Builds and runs on Windows, Linux, and Mac
  - Some code pushed for ST Micro “Discovery” board but not fully functional.
  - Getting started and other documentation can be found here:
    https://git.allseenalliance.org/cgit/core/alljoyn-js.git/tree/doc/html
  - Deep-dive presentation is available here:
    https://wiki.allseenalliance.org/_media/training/programming_alljoyn.js.pdf
- Requires an AllJoyn routing node on same CPU or elsewhere on the network

What is AllJoyn.js?

- AllJoyn.js combines the AllJoyn Thin Core Library (AJTCL) and base services with “duktape” an open source small-footprint ECMAScript 5.0-compliant runtime engine.
  - For more information on “duktape” see www.duktape.org
- A set of JavaScript APIs provide an easy to use abstraction layer over the AllJoyn core, base services, and the device I/O peripherals.
- The combined implementation is targeted at microcontrollers having a minimum of 128K RAM (preferably 256K for “real applications”) and 500K Flash.
- Also designed to run on Linux, Windows, and OS X
- Includes a “console” service for installing scripts and debugging application code.

Alljoyn.js Architecture



Console Service

- An AllJoyn service that runs alongside the JavaScript app
  - Functionality is exposed as an AllJoyn interface
  - AllJoyn.js source tree includes a command line console client
- Provides remote access to running JavaScript application
  - OTA flashing of new JavaScript applications
  - Execute JavaScript code on target in real-time
  - Logging of output from JavaScript `print()` and `alert()` functions
  - Displays notifications from running JavaScript program

Programming model

- 100% event driven.
  - No threading, no blocking calls
- Functions registered with the AllJoyn object (AJ) are called when various AllJoyn events happen:
  - AllJoyn bus attachment events:

    `onAttach onDetach`
  - AllJoyn messages:

    `onSignal onMethodCall onPropSet onPropGet onPropGetAll`
- Functions can be registered with one-shot and interval timers

  `setTimeout clearTimeout setInterval resetInterval clearInterval`
- Functions can be registered to be called on input and output triggers
  - `setTrigger`

Interface and Object definitions

- A definition is required for the interfaces and objects used by an AllJoyn.js application.
- These definitions supply essential information required to send and receive signals, make and handle method calls, and access properties.

```
AJ.interfaceDefinition[‘test.InterfaceA'] = {
    mySignal:{ type:AJ.SIGNAL, args:[’s’] },
    myProperty:{ type:AJ.PROPERTY, signature:’u’ },
    myMethod:{ type:AJ.METHOD, args:[‘i’, ‘i’], returns:[’i’] }
};

AJ.interfaceDefinition[‘org.example.Interface2’] = {
     /* signals, methods, and properties */
};

AJ.objectDefinition['/myApp'] = {
    interfaces:[‘test.InterfaceA’,‘org.examples.Interface2’]
};
```

Service Discovery

- AJ.findService()
  - Takes two arguments: an interface name and the callback function to be called when the service has been found.
  - If the second argument is omitted, discovery of specified interface is canceled.
  - If the required service is discovered, the callback function is called with a service object that provides information about the discovered service.

Accepting remote connections

- To explicitly accept or reject a connection from a remote peer, register the “onPeerConnected” callback function.
  - Return *true* to accept the connection or *false* to reject it

```
AJ.onPeerConnected = function(peer) {
    connectedPeer = peer; // Save the service object
    return true;          // Accept the connection
}
```

  - The argument to the callback function is a service object.
    - Use the service object to send signals and make method calls to the connected peer.
- When no “onPeerConnected” callback registered:
  - AllJoyn.js will automatically accept all connections
    - Note: The Application will not have access to a service object that is needed to send signals or make method calls to the remote peer.

Base service integration

- AllJoyn.js currently integrates with four base services:
  - Onboarding – gets a device onto a Wi-Fi network
  - Configuration – sets up authentication credentials, friendly name, etc.
  - Notification – send text messages for human consumption
  - Control Panel – a generic UI toolkit
- Onboarding and Configuration are mostly transparent to JavaScript applications.
  - Some parameters from the Configuration service can be read and set
- AllJoyn.js implements APIs to support Notification and Control Panel services.

Code Samples

Send a notification on GPIO interrupt

```
var pbA = IO.digitalIn(IO.pin[8], IO.pullDown);
var pbB = IO.digitalIn(IO.pin[9], IO.pullDown);

AJ.onAttach = function()
{
    pbA.setTrigger(IO.fallingEdge, function() {
        AJ.notification(1, "Button A pressed").send(200); });

    pbB.setTrigger(IO.risingEdge, function() {
        AJ.notification(0, "Button B released").send(200); });
}

AJ.onDetach = function()
{
    pbA.setTrigger(IO.disable);
    pbB.setTrigger(IO.disable);
}
```

Controlling LED flash rate

```
var cp = AJ.controlPanel();

var c1 = cp.containerWidget(cp.VERTICAL, cp.HORIZONTAL);
var rate = c1.propertyWidget(cp.SLIDER, 500, "Flash rate:");
rate.range = { min:20, max:1000, increment:50, units:"msec" };

var led = IO.digitalOut(IO.pin[0]);

var blinky = setInterval(function(){led.toggle();}, rate.value);

rate.onValueChanged = function(val) { resetInterval(blinky, val); }

AJ.onAttach = function() { cp.load(); }
```

AllJoyn.js Workshop at Alliance Summit

- Afternoon of the second day
- Tutorial and opportunity to get hands-on and write some AllJoyn code
- Good way to become quickly familiar with the AllJoyn concepts
- Hope you will be able to join in!

Thank You

Follow Us On 

- For more information on AllSeen Alliance, visit us at: allseenalliance.org & allseenalliance.org/news/blogs