ALLSEEN ALLIANCE

Technical Steering Meeting

October 14, 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. Vote on Living Scenario Proposal
3. AllJoyn Build and SDK Improvements Proposal
4. Gateway Agent Status Update

Vote on Living Scenario Proposal

Living Scenario Proposal - Goal

The Living Scenario proposal has the objective to introduce a **functional layer** able to provide a basic interoperability among different devices/ecosystems **preserving the business model of each manufacturers.**

To reach this goal, every manufacturers could name with the same “word/sentence”, the very basic “needs” of the dwellers.

These sentences (“Living Scenario”) and can be seen as a list of “events” mapping some standard behavior of the inhabitants to actions from the Alljoyn ecosystem

- Enter Home
- Go out of home
- Go to sleep
- Wake up
- Cinema
- ….

Once the devices receives the “living scenario” events, they can react.

Living Scenario Proposal – Action

- To define new “living scenario” based on a common user activity.
  - Strong engagement of the marketing team to define “living scenario” (use cases analysis)

- To provide a common implementation that could be embedded in any AllJoyn compatible devices.
  - The common implementation should contain at least 2 methods:
    - The activation of a living scenario (probably the “events/action” mechanism is a good staring point).
    - The association of the status of a device/subsystem at a T time with a living scenario. (smart learning auto configuration)

AllJoyn Build and SDK Improvements Proposal

Summary

- AllJoyn Build and SDKs have become inconsistent and cumbersome over time.
- Proposal is to improve build and SDK.
- Hope to get some changes into 14.12 (for Core and Services), but need help.
- Hope is this becomes defacto for all AllSeen WGs

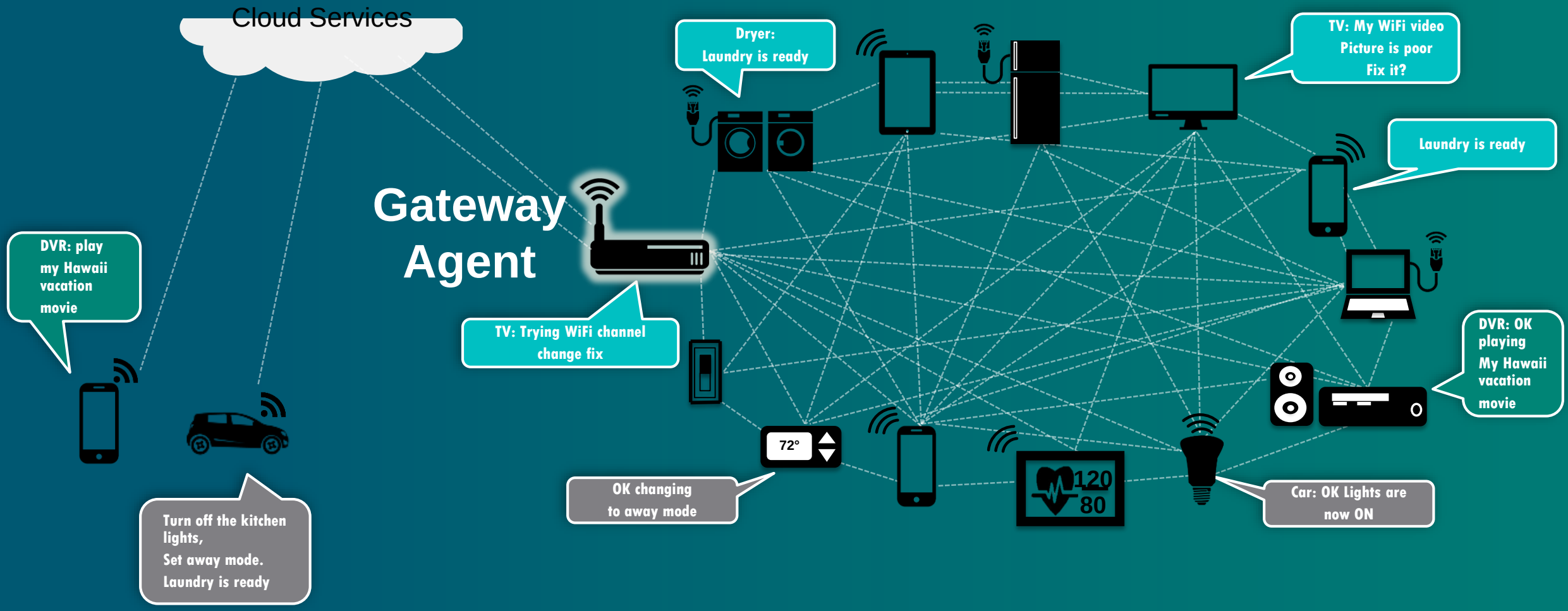
Proposal Summary

1. One source tarball per project
2. One SDK per platform per project
3. Consistent tarball and SDK names
4. Release “release” SDKs only
5. Single build products directory
6. Build from any directory
7. Make it easy to run build products
8. Restructure dist/sdk directory
9. Revisit iOS packages/libs

Gateway Agent Status Update

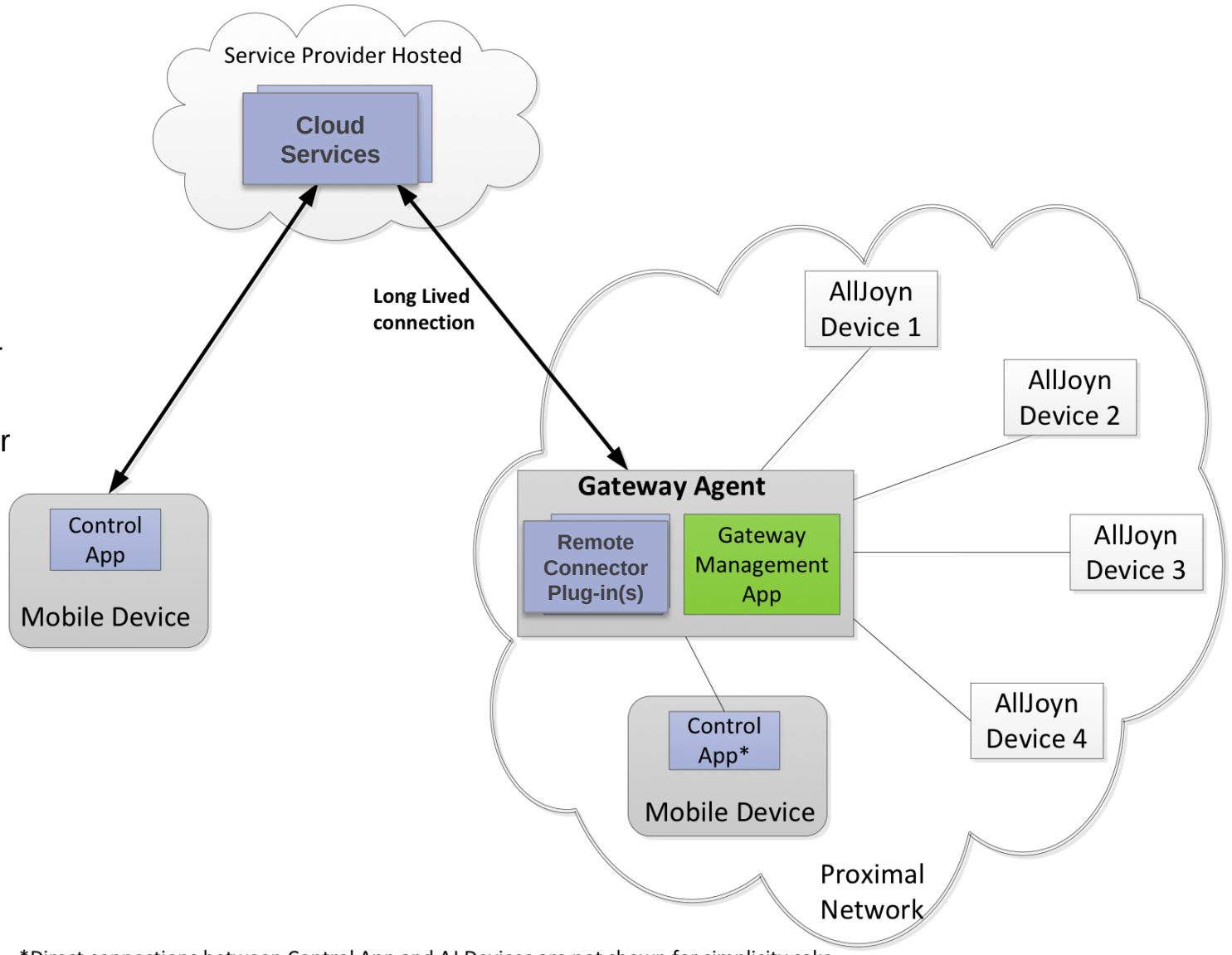
Art Lancaster
CTO Affinegy,
Chair Gateway W.G.

Gateway Agent – AllJoyn meets Cloud Services



Overall Architecture

- Gateway Agent
  - Gateway Management App
    - Embedded in gateway device
    - Management APIs for controlling the remote access and filtering for each cloud service to their included AllJoyn devices
    - Includes a TR-069 component for service provider remote provisioning management
  - Remote Connector Plug-in(s)
    - AllJoyn to Cloud (or other technology/network) Protocol connector
    - Installable and portable
    - One or more Connectors supported
- Control App
  - An add-on component for an AllJoyn mobile app to enable users to self-manage their cloud service connections
  - Required for Consumer Mode and not for Service Provider Mode (next slide)



*Direct connections between Control App and AJ Devices are not shown for simplicity sake.

New – Two Control and Management Modes

- **Consumer Mode** – self-service local management
  - Enables a consumer user to self-manage their AllJoyn connections to cloud services
  - Services can be initially provisioned via a mobile app that includes the Gateway Control App
  - The Control App follows the Gateway Agent APIs to install its own Connector App and to manage its Service Profiles
  - Consumer must be at home when configuring their cloud service using Consumer Mode
- **Service Provider Mode** – secure automated installation and remote management
  - Ideal for both connected device with services suppliers and for broadband operators
  - Remote management implemented with TR-069a5 enhanced with XMPP
    - Proven, secure, high-scale NAT traversal, compliant with Broadband Forum and XMPP standards
  - Remote software installation and update management for both the Connector App(s) and the complete firmware of the hub or gateway device.
  - Remote management of the AllJoyn Services Profile and of the gateway/hub configuration
  - Can coexist with Consumer Mode, or can remotely block Consumer Mode

Remote Connector Plug-In for AllJoyn ⟷ XMPP

- An XMPP Cloud Services Connector plug-in for the AllSeen Gateway Agent supporting its APIs and service profiles
- Relays local AllJoyn messages, to/from a remote AllJoyn application by wrapping these in XMPP sessions relayed via a standard XMPP server cloud service.
- Subscribers authenticate with their cloud service's XMPP ID – provides secure authentication and an encrypted channel for AllJoyn remote access.
- Affinegy is also providing an AllJoyn remote/mobile Android sample application that includes an XMPP client paired with the AllJoyn mobile app.
  - Normal case provides a remote UX only for the mobile device running this XMPP/AllJoyn mobile connector (non-routing AllJoyn app).
  - If used with a remote AllJoyn routing node can join two proximal networks over XMPP – needs care in setting up application AllJoyn security when used this way.

Gateway Agent Open Source Code and Status

Gateway Agent Component	Details	Status	Contributor
1. Gateway Management App	Main management embedded app	**Code available now in AllSeen GIT**	Qualcomm
2. TR-069 client	Service provider mode management client	Porting to AllSeen open source, available Oct. end	Affinegy
3. Connector Plug-in	XMPP cloud to AllJoyn connector.	Adding AllSeen gateway API support, available Oct. end	Affinegy
4. Package manager	Software installation manager for Connector Apps	**Code available now in AllSeen GIT**	Affinegy
5. Mobile Control App	Android based mobile Control App	**Code available now in AllSeen GIT**	Qualcomm
6. Full Gateway Agent Release 1 – Reference implementation	Integrated full Gateway Agent in OpenWRT & developer cloud reference	Q4 2014	Qualcomm & Affinegy

Cloud Services for Gateway Agent available from Affinegy

- Affinegy CHARIOT Server
  - Provides IOT cloud services fully compatible with AllSeen Gateway Agent
  - XMPP server for high scale, persistent IOT services and management
  - For Service Provider Mode – Advanced TR-069 ACS remoted management server deployed worldwide with millions of devices under management.
  - Provides full turn-key installation and support management for the hardware device running the Gateway Agent
- No cost developer hosted access available from Affinegy
- Commercial CHARIOT server licensing available from Affinegy, both hosted and installed options

AllSeen Gateway Working Group Contributors

- Affinegy
  - Art Lancaster, CTO – contributor and W.G. chair
  - Committers: Josh Spain, Kevin Sandifer, Jim Howard
- Qualcomm
  - Shane Dewing, Senior Director Product Management – contributor
  - Committers: Tsahi Asher, Tali Messing, Benita Gupta, Josh Hershberg

Backup

Details of SDK Update Proposal

1. One source tarball per project

- Problem: we currently don’t release source tarballs for all source.
- 1 per project keeps projects cleanly separated, allows projects to rev independently

2. One SDK per platform per project

- One SDK per “platform” per source tarball
- “Platforms” are:
  - Android SDK
  - Android NDK
  - Mac OSX
  - iOS
  - Windows (merge 32 bit and 64 bit into one SDK)

3. Consistent tarball and SDK names

- alljoyn-<component>-<ver>-<type>-<platform>.<ext>
- Examples:
  - alljoyn-core-14.12-src.tar.gz
  - alljoyn-core-14.12-sdk-ios.zip
  - alljoyn-base-services-14.12-src.tar.gz
  - alljoyn-lighting-14.12-src.tar.gz
  - alljoyn-lighting-14.12-ndk-android.zip

- Source Tarball format: .tar.gz
  - More flexible if we need symlinks
- SDKs format: .zip
  - Better supported natively on all platforms

4. Release “release” SDKs only

- Proposal:
  - Only release “release” variant of SDK
  - Update “release” variant with debug logging.
    - Disable logging by default
    - Can turn logging on at run time

- Why
  - debug logging with run-time control is most important to most developers.
  - most developers don’t need debug symbols
  - those who need debug symbols are contributors, who would likely be building from source anyhow
  - single SDK variant easier to manage
  - debug strings adds only small amount of code to library.
  - those needing the smallest lib can rebuilt without debug logging
- Some concerns over performance; analysis underway

5. Single build products directory

- single build products directory regardless of where build is executed from
- Will be located from top level, e.g.
  - alljoyn/
    - core/
    - services/
    - build/

6. Build from any directory

- same build/out dir regardless of where build started
- implementation may be related to larger scons cleanup

7. Make it easy to run build products

- Easy means:
  - no explicit typing LD_LIBRARY_PATH
  - no need to cd into bin dir or explicitly add path

- Potential implementations
  - wrapper scripts
  - scons install to install into system/specified dirs
  - perhaps automatically add path to dist libs and bin to shell env

8. Restructure dist/sdk directory

- Needs reorg now that more code has been added
- Optimize structure for common case

8. Restructure dist/sdk directory - Linux

Notes: applications will #include just core.h, config.h, etc; headers placed in core/ and config/ subdirs for c headers (as opposed to c++), #include c variant instead.

```
dist/
    bin/
        alljoyn-daemon
    include/
        alljoyn/
            core.h
            core_c.h
            config.h
            config_c.h
            core/
                <additional core headers>
            core_c/
                <c binding header files>
            config/
                <additional config headers>
            config_c/
                <config header files>
    lib/
        liballjoyn.a
        liballjoyn.so
        liballjoyn_c.so
        liballjoyn_c.a
        libajrouter.a
    alljoyn/
        BundledDaemon.o
    liballjoyn_about.a
    liballjoyn_about.so
    liballjoyn_config.a
    liballjoyn_config.so
    <etc...>
mozilla/
    libnpalljoyn.so
java/
    liballjoyn_java.so
    alljoyn.jar
    about.jar
    config.jar
docs/
    index.html
    cpp/
        index.html
        core/
            index.html (core cpp docs)
        config/
            index.html (config cpp docs)
    c/
        index.html
        core/
            index.html
        config/
            index.html
    java/
samples/
    cpp/
        core/
            basic/
                <buildable cpp source>
            chat/
        config/
            <buildable cpp source)
    c/
    java/
        core/
            basic/
                <buildable java source>
        config/
```

8. Restructure dist/sdk directory - Android

Two SDK zips: SDK and NDK

```
dist/
  sdk/
    apk/
        <pre-built sample apks>
    libs/
        armeabi/
            liballjoyn_java.so
        alljoyn.jar
        alljoyn_about.jar
        alljoyn_config.jar
        <etc>
    docs/
        index.html # links to all java docs
        core/
            index.html # java docs
        config/
            index.hml # java docs
    samples/
        core/
            basic/
                <buildable android source>
                <has libs/jars in here>
            chat/
        config/
```

```
ndk/
    apk
       <pre-built sample apks>
    lib
       liballjoyn.so
       alljoyn_about.so
       ...
    include/
        alljoyn/
            <mirror linux include dir>
    docs/
        index.html # links to all cpp docs
        core/
            index.html (core cpp docs)
        config/
            index.html (config cpp docs)
        <etc...>
    samples/
        core/
            basic/
                jni/ (cpp)
                src/ (java)
            chat/
```

8. Restructure dist/sdk directory – iOS and OSX

Separate iOS and OSX SDKs

```
dist/
  ios/
    docs/
        cpp/
            core/
            config/
        objc/
            core/
            config/
    samples/
        cpp/
        objc/
    cpp/
        inc/
            <match linux headers dir>
        lib/
            <match linux lib dir>
    objc/
        inc/
            <TBD>
        lib/
            <tbd>
```

```
osx/
    docs/
        cpp/
            core/
            config/
        objc/
            core/
            config/
    samples/
        cpp/
        objc/
    cpp/
        inc/
            <match linux headers dir>
        lib/
            <match linux lib dir>
    objc/
        inc/
            <TBD>
        lib/
            <tbd>
```

9. Revisit iOS packages/libs

- Currently, adding AllJoyn to iOS requires 20 some steps.
- Need to investigate simpler solutions to add AllJoyn in 1 minute.

Other suggestions?

Thank You

Follow Us On 

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