ALLSEEN ALLIANCE

Core Working Group Meeting

The contents of this document, and the interfaces described, and all the information herein, are the result of collaborative discussions by the Core Working Group. This summary documents the final consensus of the team.

Reminder:

This call is being recorded

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.

Agenda

- AJATS discussion
  - Discuss Josh's proposal: both conclusions and prioritization
  - Way: feedback is to at least to the minimum to get it up and running again – all platforms with current set up.
    - This includes "cutting all the fat" i.e. removing duplication from core test plan
  - Marcello to present to TSC.
- Post mortem?
  - Scheduled for 7/15 @ 10am PDT
- C&C discussion
  - https://wiki.allseenalliance.org/core/security_enhancements?image=core%3Ac_c_recomended_tests.xlsx&ns=core&tab_details=view&do=media
    - **ACTION:** Kris to provide final recommendation/thumbs up
    - **ACTION:** Marcello to send Ram and email to identify next steps on getting tests developed.

AJATS Recommendations

Josh Spain, Affinegy

What is AJATS?

- Tests AllJoyn Core
- … simultaneously on multiple platforms (currently Android, iOS, Linux, Windows)
- … to find interoperability issues (e.g., caused by endianness, architecture, OS-specific APIs, etc.)
- … by using existing test apps in the core/test.git and core/alljoyn.git repositories.

What is it NOT?

- Does NOT test anything other than AllJoyn Core
- Does NOT test interoperability with *real* apps and devices
- Is NOT comprehensive (i.e., does not test *all* bindings, features, or OS interactions)

How does it work?

- Isolated network
- Driven by Jenkins
  - Host systems on OSX, Ubuntu, and Windows (all Jenkins clients)
  - Mobile systems are connected via USB to host systems
  - Jenkins starts jobs on the hosts and runs scripts from the core/test/testbot folder
- BASH scripts are run on each host system
  - The scripts build everything for the platforms, including test mobile apps
  - Then the scripts deploy the mobile apps to the devices
  - The scripts load the test apps on the devices
    - … and watch for output on the command line
    - … while logging the output to files.
  - The test apps interact with other test apps on the network *(currently only ones connected to the same host)*

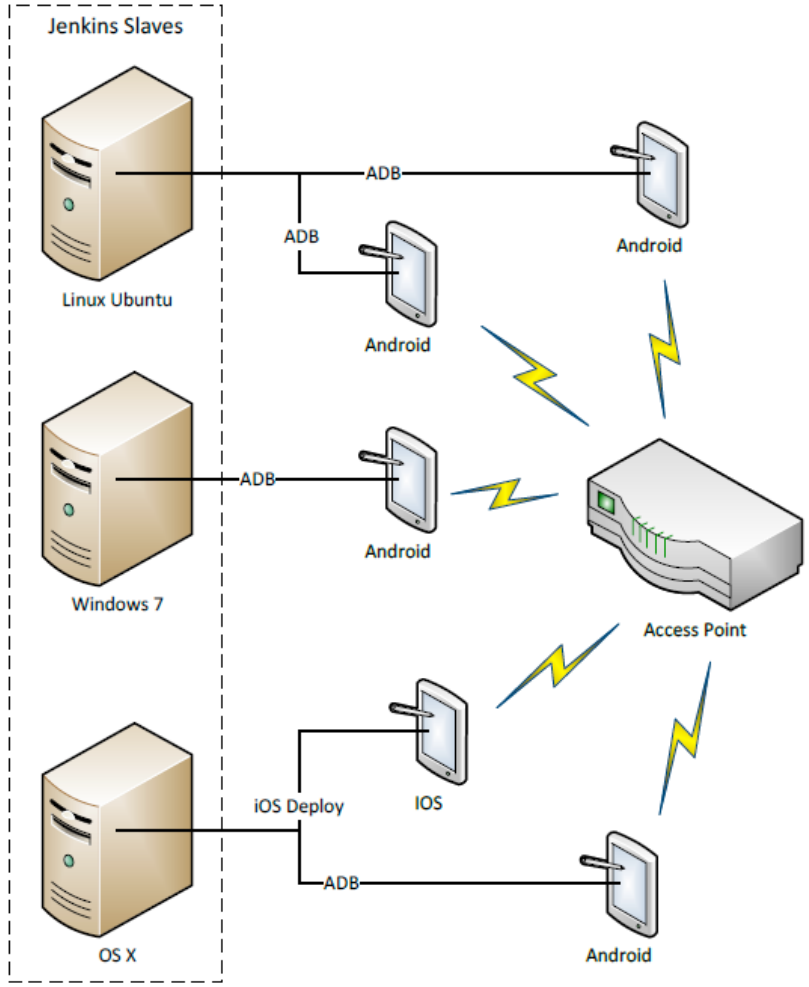

Figure 1 Block diagram of hardware for AJATS as operated by QCE.

Jenkins System Status

- Ubuntu (12.04?)
  - Building successfully on master branches for both Linux and Android
- Windows 7
  - Offline for 4 months
  - 5 months since last successful build
- Mac Mini (OSX 10.?)
  - Jenkins jobs not yet working

Current Tests

- **Windows local**
  - Security 2.0 - bbservice ←→bbclient
    - TCP and UDP – SRP, LOGON, ECDHE_NULL, ECDHE_PSK, ECDHE_ECDSA
  - rawservice ←→rawclient
  - basic_service ←→basic_client
  - bbservice ←→bbclient
- **Android ←→ Android**
  - Lots of tests
  - 2 or more Android devices in each test
  - Java code was written specifically for these tests
- **Linux local**
  - Events & Actions
  - Sessions
  - PolicyDB
  - Raw Sockets
  - Platform-independent tests
  - Multipoint session tests
  - About tests
  - Two daemons
  - Stress tests
  - Thin Core tests

Disabled Tests

- Windows
  - Windows ←→ Android
    - bbservice ←→bbclient
    - bbclient ←→bbservice
  - Java bindings tests
  - TC/SC interaction tests
- Android ←→ Android
  - About certification
  - JNI simple service/client
  - JNI chat between two devices
  - Property service and client
  - Java chat between two devices
  - Several variations on bbservice/bbclient tests
  - Bundled daemon application UI
- iOS ←→ Android (Jenkins system not fully set up)
  - BasicBusService.app ←→ bbclient
  - BasicBusClient.app ←→ bbservice

What is Missing?

- OpenWrt
- Windows ←→ iOS
- Windows ←→ Android (exists but disabled)
- Windows ←→ Linux
- Linux ←→ iOS

What needs Work?

- iOS Jenkins build needs to be finished
- Redundancy with single-host automated/unit tests could be removed (i.e., focus 100% on interaction and backwards-compatibility tests)
- Hard-coded paths (like usernames) means it is tied to the existing AJATS hardware
- Many tests are commented out due to bugs
  - These should be reviewed and uncommented if the bugs are already fixed
  - We should consider prioritizing fixes for these bugs to increase the value of AJATS
- Would be more maintainable and more integrated with Jenkins if written in Groovy instead of BASH
  - could be done over time
- Using a specific unit test framework would be better
  - Currently it parses the arbitrary output of existing test apps
    - When test apps change, the unit tests are often broken
  - We could rewrite test apps to be specific to AJATS
  - We could use something like Google Test, JUnit, or many others to provide clear, concise, consistent results

Proposed Plan

- Phase 1
  - Simplify and Solidify
- Phase 2
  - Rewrite
- Phase 3
  - Expand Coverage

Each Phase Estimated at:

1 FTE
3 months

Phase 1 – Simplify and Solidify

- Focus on only **interoperability** and **backwards-compatibility** tests and remove everything else.
- Do not add any new tests
- Specifically this means:
  - iOS
    - Upgrade to latest OSX
    - Get the Jenkins host (Mac Mini) up and running
    - Get the iOS ←→ Android tests working properly
  - Windows
    - Upgrade to Windows 10
    - Bring the system back online
    - Remove irrelevant tests
    - Make sure Windows ←→ Android tests are running properly
  - Linux
    - Upgrade to Ubuntu 14.04 or higher
    - Remove irrelevant tests

Phase 2 - Rewrite

- Redesign and rewrite the system based on several goals:
  - Easy to maintain
  - Functions across supported environments
  - Easy to integrate with Jenkins
  - Easy to expand functionality

- The blueprints of the necessary logistics are already written in the BASH scripts.
  - This significantly reduces the difficulty of such a rework.

- Potential candidate would be Groovy scripts (essentially Java as a scripting language)

Phase 3 – Expand Coverage

- Determine what manual tests can be automated in the context of AJATS
- Add backwards-compatibility tests
- Add more OS interaction
  - iOS ←→ Windows
  - Windows ←→ Linux
  - iOS ←→ Linux
  - OpenWrt
- Expand to an IPv6 network

Summary

Recommendation

AJATS is useful. We should resurrect and maintain it.

Resource Requirements

Minimum ¼ time engineer for maintenance

Preferred 1 FTE for 3 months or 1 half-time for 6 months

Ideally 1 FTE for 9 months

Enhancements (priority order)

Remove redundant tests (focus 100% on interop and back-compat)

Fix issues preventing some tests from running

Update system to a more versatile language like Groovy

Add more interoperability tests

Action Items

Action Items

- Proposal to outline the process for changing APIs
  - Has been handed over to Brian Rockwell
    - will craft proposal for review by Core WG and then presented to TSC
  - Expectation that this will be reviewed with WG in January.
- Arvind (MSFT) to create Wiki process page linked off of Core WG wiki
  - Remove references to java from mandatory binding list for 16.04 timeframe
  - Hope to have something by the end of Jan.

Action Items

- Add a way in JIRA to track compatibility issues and proposals
  - Proposal is the use a label: “compatibility impact” in the Jira ticket
  - Committers should require for these sorts of change that the release notes be updated
  - **Marcello** to document this process in the Core Process wiki
- Define process to require regression/unit tests for bugs
  - **Marcello** to add description of Committers requiring unit test for bug fixes to Core Wiki
- Define process for handling “Technical Debt”
  - Example: Took a shortcut to make a release and not loosing track of this to fix the shortcut
  - Jira items to track when things are missing – e.g. feature added, but missing from Java binding
  - If shortcut is taken: could create a Jira task to track the problem

Thank You

Follow Us On 

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