



AllSeen Alliance Self-Certification User Guide

Version 14.06 Update 3

November 25, 2014

This work is licensed under a Creative Commons Attribution 4.0 International License.

<http://creativecommons.org/licenses/by/4.0/>

Any and all source code included in this work is licensed under the ISC License per the AllSeen Alliance IP Policy.

<https://allseenalliance.org/allseen/ip-policy>

Contents

1 Introduction.....	5
1.1 Document organization	5
1.2 Release history	6
2 Terminology	7
3 Meeting Hardware Requirements.....	8
3.1 Hardware requirements for the Self-Certification Tool.....	8
3.2 Hardware requirements for Wi-Fi Access Point.....	8
4 Meeting Software Prerequisites	9
4.1 PC or laptop.....	9
5 Selecting a TCCL release to certify your product.....	10
6 Installing the Self-Certification Tool	11
7 Determining service or services you would like to certify.....	14
8 Testing a sample of your product	15
8.1 Getting the appld and deviceId.....	15
8.2 Testing your DUT	17
8.2.1 Run the tests	17
9 Reporting your results and obtaining Certification.....	22
10 Reporting Technical Issues	23
Appendix A List of test cases for each service	24
Appendix B List of adb command parameters needed for running the test cases.....	29
Appendix C List of valid values for the “testCaseName” parameter	30
Appendix D Writing adb commands to execute tests	34
D.1 Case 1 (About Service).....	35
D.2 Case 2 (About and Config)	36
D.3 Case 3 (Notification Consumer)	36
D.4 Case 4 (Notification Producer).....	36
D.5 Case 5 (Onboarding)	37

D.6 Case 6 (About-v1-01)	37
D.7 Case 7 (About-v1-01 and About-v1-02 Service)	37
Appendix E PC console results examples	38
E.1 One test case executed with result FAIL	38
E.2 Two test cases executed both resulting in FAIL.....	38
E.3 One test case executed with result PASS	39
E.4 Two test cases executed with result PASS.....	39
E.5 One test suite executed with two test cases FAIL.....	39
Appendix F Logcat output example	41
Appendix G Hints to Troubleshoot adb Commands	42
G.1 What to do when adb command is not recognized	42
G.2 What to do when “adb devices” does not show my connected device	42
G.3 Common adb command errors	43

Figures

Figure 1. Overview of the Self-Certification Tool and its connections	8
Figure 2. Successful Self-Certification Tool APK installation	11
Figure 3. Self-Certification Tool APK listed in the Android device.....	12
Figure 4. Self-Certification Tool APK information in the Android device.....	12

Tables

Table 1. PC or laptop software requirements to control the Self-Certification Tool.....	9
Table 2. Services in scope of the AllSeen Alliance Certification by release	14
Table 3. Test suite names	18
Table 4. adb command parameters for running test cases.....	29
Table 5. testCaseName parameter values.....	30
Table 6. Test suite names	34

1 Introduction

This document provides a general overview of the steps you must follow to certify a product according to the AllSeen Certification Program. It also provides instructions as to how to download and run the tests needed to certify your product.

1.1 Document organization

Section 2 provides details about some of the terms used in the document.

Section 3 provides details on the hardware you need to build and operate the Self-Certification Tool needed to run the tests required to certify a product according to the AllSeen Certification Program.

Section 4 provides details about the software that needs to be installed in the hardware elements described in section 3, before you can attempt to download the AllSeen Self-Certification Tool.

Section 5 describes what you need to take into account to decide the TCCL release to use before you actually start testing your product.

Section 6 provides instructions to download the Self-Certification Tool APK according to the TCCL release you have chosen for certification.

Section 7 provides indications so that you can properly select the test suites and test cases you must run to be able to be granted certification by the AllSeen Alliance. The test cases are grouped in test suites for each service defined by the AllJoynTM framework. In each test suite there are a number of test cases. Please note that you must run all the test suites and test cases in each test suite for all the services/roles supported by your product.

Section 8 provides detailed instructions on how to operate the Self-Certification Tool to run the tests.

Section 9 provides instructions to report the results to the AllSeen Alliance Certification Authority who grants certification and the rights to use the AllSeen logo once the outcome of the testing has been reviewed.

Section 10 and the Appendixes provide additional information that you may find useful.

1.2 Release history

Release version	Date	What changed
14.06	9/2/2014	Initial release
14.06 Update 1	10/1/2014	<ul style="list-style-type: none">■ Added a statement at the start of section 8.2 about the requirement to use the same deviceId and appId during testing.■ Updated the adb commands in section 8.2.1 for starting the execution of tests.■ Modified Appendix A to reflect the current valid set of test cases.■ Updated the Onboarding adb command parameter descriptions in Appendix B.■ Updated the testCaseName parameter values in Appendix C to reflect the current test cases.■ Updated the adb command information in Appendix D.5.
14.06 Update 2	10/29/2014	<ul style="list-style-type: none">■ Updated table: services in scope of the AllSeen Alliance certification by release.■ Added Audio and Lighting test suite names in table 3.■ Added Audio and Lighting test cases in Appendix A.■ Added an adb command parameter for Audio test suite■ Updated Appendix C with new testCaseName values.■ Updated table 6 with mandatory parameters for Audio and Lighting test suites.
14.06 Update 3	11/24/2014	<ul style="list-style-type: none">■ Gateway Service added.■ Events & Actions Feature from Core added.■ Typo errors fixed.■ Missing parameter in some adb command examples added.

2 Terminology

Some terms used in this document are listed below.

Term	Description
adb commands	A command line tool that, among others, allows you to communicate with a connected Android device. http://developer.android.com/tools/help/adb.html
AllJoyn™ framework	An open source IoT software framework and set of services that enables horizontal interoperability across product platforms and allows for proximal peer-to-peer discovery & communications over various transports and OSes.
appld	Globally unique identifier for the application given by its About interface's Appld metadata field.
DDMS	A debugging tool called the Dalvik Debug Monitor Server, which provides port-forwarding services, screen capture on the device, thread and heap information on the device, logcat, process, and radio state information, incoming call and SMS spoofing, location data spoofing, and more. http://developer.android.com/tools/debugging/ddms.html
deviceId	Device identifier set by platform-specific means found in the About interface's DeviceId metadata field.
DUT	Device Under Test. A real sample of the product device which is going to be subject to the testing.
Self-Certification Tool	An Android device hosting the Self-Certification Tool APK, used as the actual Self-Certification Tool. The Self-Certification Tool only supports Wi-Fi transport.
Self-Certification Tool APK	An APK with the Self-Certification Tool software provided by the AllSeen Alliance that has to be installed on an Android device. The Android device together with the Self Certification Tool APK become the Self-Certification Tool.
PC or Laptop	Any computer used as controller of the Self-Certification Tool. The control of the Self-Certification Tool takes place via adb commands.
PID	Process Identifier. Is a number used to temporarily uniquely identify a process
TCCL	Test Case Control List. A list of test cases that are required by the AllSeen Alliance to certify a product.
Wireless Access Point	A device that allows wireless devices to connect to a wired network using Wi-Fi.

3 Meeting Hardware Requirements

Figure 1 provides an overview of the elements you need to build the Self-Certification Tool and how to connect it to your product under test (i.e., the DUT).

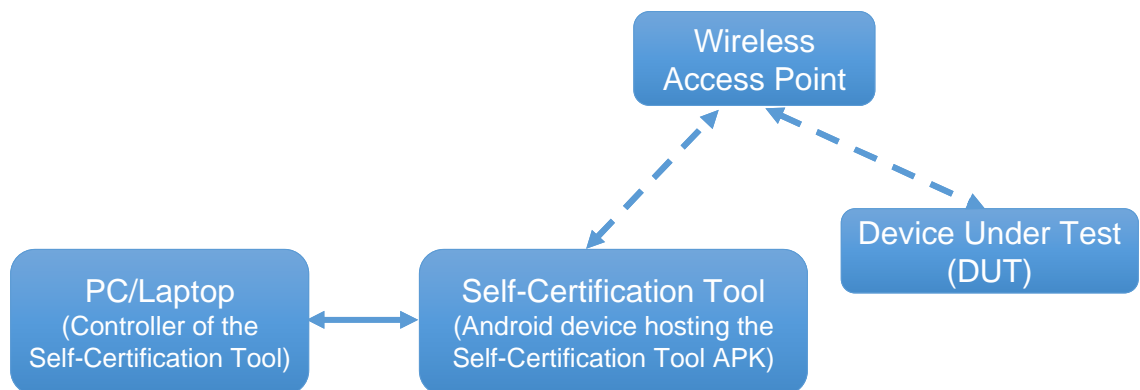


Figure 1. Overview of the Self-Certification Tool and its connections

To run the Self-Certification Tool, you need a PC or laptop (the controller of the Self-Certification Tool), an Android device (the Self-Certification Tool itself), a Wireless Access Point, which provides the transport between the Self-Certification Tool and your DUT.

3.1 Hardware requirements for the Self-Certification Tool

- PC or laptop (the controller of the Self-Certification Tool)
 - Processor: Intel Pentium 4 / AMD Athlon XP with 2.4 GHz or superior.
 - RAM memory: 2 GB or more.
 - Integrated / external graphic card.
- Android device to host the Self-Certification Tool APK (the Self-Certification Tool itself)
 - Processor: ARM based processor with at least 1 GHz.
 - RAM memory: 512 MB or more.
 - Minimum system version is Jelly Bean 4.1.2 (API 16).
- USB cable

3.2 Hardware requirements for Wi-Fi Access Point

Wireless Access Point that supports 2.4 GHz 802.11n.

4 Meeting Software Prerequisites

4.1 PC or laptop

The software to control the Self-Certification Tool can be installed on a PC or laptop running on any of the following operating systems: Windows, Linux or OS X.

In order to be able to control the Self-Certification Tool running on the Android device, you need to install on your PC or laptop the software listed in Table 1 and accept the default configuration.

NOTE

You must have Administrator privileges in your PC or laptop to be able to install the software.

Table 1. PC or laptop software requirements to control the Self-Certification Tool

Software	Minimum version	URL
Java SE Development Kit (JDK)	1.6	http://www.oracle.com/technetwork/es/java/javase/downloads/index.html
Android SDK	API / Platform 16	http://developer.android.com/sdk/index.html

Do not forget to install API / Platform 16 in SDK manager. You can open it from your SDK folder.

To ensure your PC or laptop console can find the adb executable, add the “platform-tools” folder (placed in Android SDK) to your system environment variables. To ensure adb is able to detect devices please execute “adb devices” command. For further hints on basic problems solving, see Appendix D.

5 Selecting a TCCL release to certify your product

AllSeen Certification requirements change over time and are published as TCCL releases. You can choose any of the active TCCL releases (no more than 24 months old) to certify your product.

At this point in time the only active TCCL releases are 14.02 and 14.06.

NOTE

The Core release version plus each service framework release version must match the same active TCCL release to pass certification for your product.

6 Installing the Self-Certification Tool

To use the Self-Certification Tool, you have to download the Self-Certification Tool APK into an Android Device. The Android device hosting the Self-Certification Tool APK becomes the Self-Certification Tool.

You can download the Self-Certification Tool APK into the Android device from the Android device's browser, according to the TCCL release version you plan to use (see section 5 on how to choose the TCCL release).

Navigate to the URL indicated below and choose the software version of the wanted TCCL release.

<http://mirrors.us.kernel.org/allseenalliance/alljoyn/apk/>

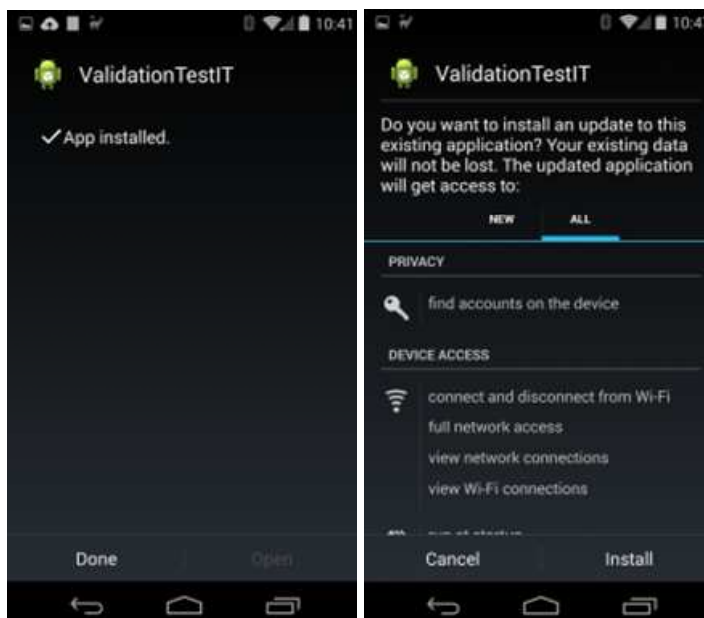


Figure 2. Successful Self-Certification Tool APK installation

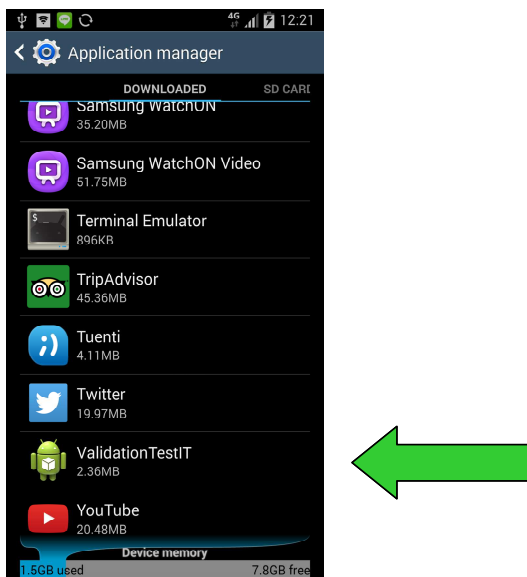


Figure 3. Self-Certification Tool APK listed in the Android device

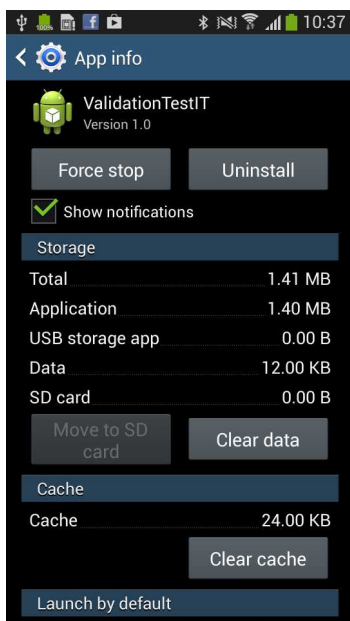


Figure 4. Self-Certification Tool APK information in the Android device

To validate that the Self-Certification Tool APK has been properly installed on the Android device, navigate to “Settings → Apps” and you should be able to see an App named “ValidationTestIT”.

You now have the Self-Certification Tool ready to be used to certify your product according to TCCL release you have chosen.

NOTE

If you have already installed the Self-Certification Tool for a TCCL release on an Android device and you want to install a new release in the same Android device, make sure you first delete the previously downloaded APK.

7 Determining service or services you would like to certify

You shall test the core (About Feature and Events & Actions feature) and all services implemented by your product for which tests suites have been made official. In order to be able to have your product certified it must implement AllJoyn core (About Feature and Events & Actions feature) plus at least one service framework.

Table 2 lists the services under the scope of the AllSeen Certification and the availability of test suites in the Self-Certification Tool for each TCCL release.

Table 2. Services in scope of the AllSeen Alliance Certification by release

Service release	TCCL release	
	14.02	14.06
Core (About Feature)	Y	Y
Core (Events & Actions Feature)	N	Y
Control Panel Service	Y	Y
Notification Service	Y	Y
Onboarding Service	Y	Y
Configuration Service	Y	Y
Audio Service	N	Y
Lighting Service	N/A	Y
Gateway Service	N/A	Y

NOTE

“Y” means test suite officially available; “N” means test suite not officially available; “N/A” means test suite not available.

The list of test cases officially released in the Self-Certification Tool can be found in Appendix A. Test case descriptions for the core and each service can be found in the corresponding specifications documents

(<https://wiki.allseenalliance.org/compliance/overview?&>).

8 Testing a sample of your product

A sample of your product that undergoes testing is referred to as the DUT (Device Under Test).

The Self-Certification Tool APK cannot be launched directly from the Android device. To run the tests on the Self-Certification Tool, (i.e., the Android device hosting the Self-Certification Tool APK described in section 6) you must use adb commands from the PC or laptop (i.e., from the Self Certification Tool controller) where you have already installed the software listed in section 4, Meeting Software Prerequisites.

NOTE

To be able to command the Self-Certification Tool from the PC or laptop using adb commands, both PC or laptop and Android device must be connected via a USB cable.

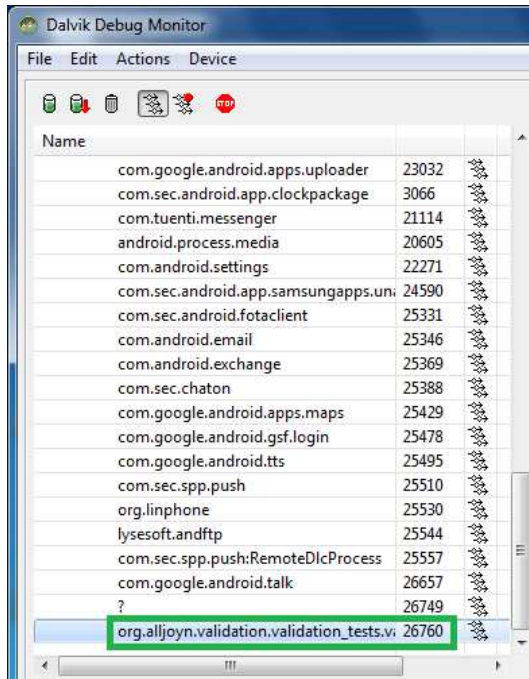
8.1 Getting the appId and deviceId

Before starting testing, ensure you know the “deviceId” and “appId” of the DUT. If you know both “deviceId” and “appId” go to section 8.2, Testing your DUT. If you do not know the “deviceId” or “appId” of your DUT, follow the instructions provided below.

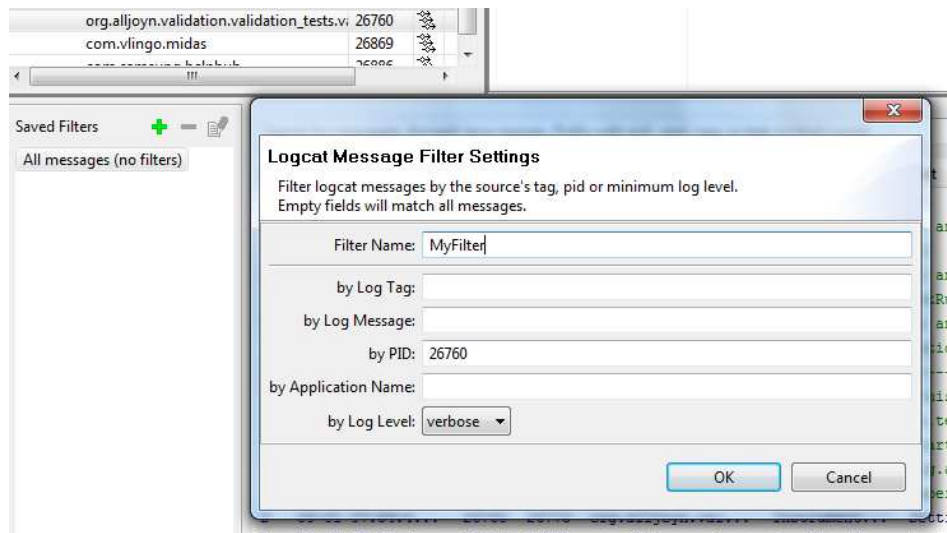
1. Connect the Android device hosting the Self-Certification Tool to your PC or laptop via USB cable.
2. Open Dalvik Debug Monitor by clicking on “ddms.bat” executable that is located in: “<androidSDKPath>\adt-bundle-windows-x86_64-20140702\sdk\tools”. This is needed in order to be able to examine the log and get “deviceId” and “appId” from your DUT.
3. Ensure your DUT is onboarded to the same Wi-Fi network as the Self-Certification Tool.
4. Execute the following adb command from your PC or laptop. Please make sure everything is written in just one long line.

```
adb shell am instrument -w -e appId 5b7a70f0-fd5b-49be-8e0a-9640f49cd587
-e deviceId 5b7a70f0-fd5b-49be-8e0a-9640f49cd597 -e testSuiteList
org.alljoyn.validation.testing.suites.about.AboutTestSuite -e
testCaseName About-v1-01
org.alljoyn.validation.validation_tests.validation_tests_it/org.alljoyn.v
alidation.testing.instrument.ValidationInstrumentationTestRunner
```

5. While the adb command is executing look into the DDMS (Dalvik Debug Monitor) to identify the PID of the Self-Certification Tool process (shown as ValidationToolIT in the list). In the example below, the PID is 26760.



6. Once you know the PID (which changes with every execution) using the filtering options, filter the logcat to only display the Self-Certification Tool messages. (See figure below.)



7. Look over the logcat file output on your PC or laptop to find this message:
 Waiting for About announcement signal from device: 5b7a70f0-fd5b-49be-8e0a-9640f49cd597; appId: 5b7a70f0-fd5b-49be-8e0a-9640f49cd587
 Ignoring About announcement signal from DUT with deviceId:
 actualDeviceId, appId: actualAppId
8. You should expect the test to FAIL this time, but you will be able to determine the real deviceId and appId of your DUT.

NOTE

“actualDeviceId” in the logcat file equals the DeviceId of your device. “actualAppId” in the logcat file equals the AppId of your device. Please make note of these two IDs as they must be used for each of the adb commands to run the tests for your DUT (sample of your product).

8.2 Testing your DUT

Once you know the deviceId and appId, you can start testing.

You must execute all test suites of the core and each individual official service release supported by your product. The test cases and tests suites release must match the selected TCCL release.

You must execute all test suites, and all test cases in each test suite, of each individual official service release supported by your product using the same deviceId and appId

8.2.1 Run the tests

Ensure you have opened DDMS before running the tests as you will need this to save your logcat output. To open DDMS go to “<androidSDKPath>\adt-bundle-windows-x86_64-20140702\sdk\tools” and execute “ddms.bat” file.

To run the tests, follow the next steps:

1. Connect the Android device hosting the Self-Certification Tool APK to your PC or laptop via USB cable.
2. Ensure your DUT is onboarded to the same Wi-Fi network as the Self-Certification Tool.
3. Use adb commands on the PC or laptop to start the execution of tests from the Self-Certification Tool. The adb command to run all test suites (i.e., all test cases for all services) is given below.

Note that the whole command has to be written in just one long line.

```
adb shell am instrument -w -e appId DUT_APPID -e deviceId DUT_DEVICE_ID -e
userInputTimeoutValueInMS 5000 -e enableInteractive true -e
org.alljoyn.Onboarding.SoftApSsid softapssid -e
org.alljoyn.Onboarding.SoftApSecurity softapsecurity -e
org.alljoyn.Onboarding.SoftApPassphrase softappassphrase -e
org.alljoyn.Onboarding.PersonalApSsid personalapssid -e
org.alljoyn.Onboarding.PersonalApSecurity personalapsecurity -e
org.alljoyn.Onboarding.PersonalApPassphrase personalappassphrase -e
testSuiteList
org.alljoyn.validation.testing.suites.about.AboutTestSuite,org.alljoyn.va
lidation.testing.suites.controlpanel.ControlPanelTestSuite,org.alljoyn.va
lidation.testing.suites.notification.NotificationProducerTestSuite,org.al
ljoyn.validation.testing.suites.notification.NotificationConsumerTestSui
te,org.alljoyn.validation.testing.suites.onboarding.OnboardingTestSuite,or
g.alljoyn.validation.testing.suites.config.ConfigTestSuite
```

```
org.alljoyn.validation.validation_tests.validation_tests_it/org.alljoyn.v
alidation.testing.instrument.ValidationInstrumentationTestRunner
```

NOTE

If you copy the example above using a rich text editor like Microsoft Word instead of a plain text editor like Notepad, it may change “-“ to “–“ and the command will not be valid.

If the device under test does not support all services, remove the corresponding Test_Suite_Name parameter from the adb command for services not supported and other parameters which may be optional for the supported services. The list of Test_suite_names for each service is provided in Table 3.

NOTE

The asterisks for the Notification and Onboarding test suites below correspond to notes provided in Appendix D.3 and D.5.

Table 3. Test suite names

Service	TEST_SUITE_NAME
Core (About Feature)	org.alljoyn.validation.testing.suites.about.AboutTestSuite
Core (Events & Actions Feature)	org.alljoyn.validation.testing.suites.eventsactions.EventsActionsTestSuite
Control Panel Service	org.alljoyn.validation.testing.suites.controlpanel.ControlPanelTestSuite
Notification Service (producer)*	org.alljoyn.validation.testing.suites.notification.NotificationProducerTestSuite
Notification Service (consumer)*	org.alljoyn.validation.testing.suites.notification.NotificationConsumerTestSuite
Onboarding Service**	org.alljoyn.validation.testing.suites.onboarding.OnboardingTestSuite
Configuration Service	org.alljoyn.validation.testing.suites.config.ConfigTestSuite
Audio Service	org.alljoyn.validation.testing.suites.audio.AudioTestSuite
Lighting Service	org.alljoyn.validation.testing.suites.LSF_LampTestSuite
Gateway Service	org.alljoyn.validation.testing.suites.gwagent.GWAgentTestSuite

For a device that only supports the About, Control Panel and Configuration services, the adb command is shown below.

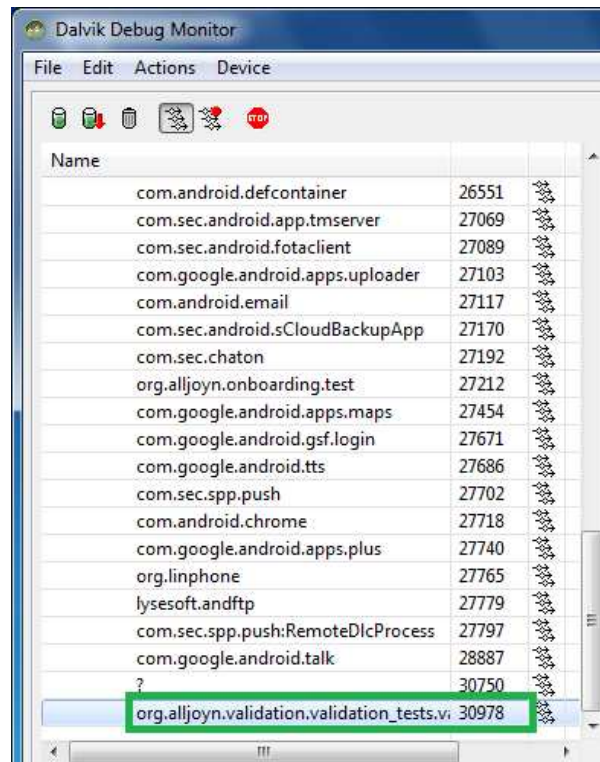
```
adb shell am instrument -w -e appId DUT_APPID -e deviceId DUT_DEVICE_ID -
e testSuiteList
org.alljoyn.validation.testing.suites.about.AboutTestSuite,org.alljoyn.va
lidation.testing.suites.controlpanel.ControlPanelTestSuite,org.alljoyn.va
lidation.testing.suites.config.ConfigTestSuite
org.alljoyn.validation.validation_tests.validation_tests_it/org.alljoyn.v
alidation.testing.instrument.ValidationInstrumentationTestRunner
```

Alternatively, you can use adb commands to run each test suite individually or each test case individually. To apply for certification you have to run all test cases in all test suites corresponding to the services supported by the product.

NOTE

The general structure of the adb command and some examples on how to use adb commands to run tests are provided in Appendix D.

4. While the adb command is executing look into the DDMS (Dalvik Debug Monitor) to identify the PID of the Self-Certification Tool process (shown as ValidationToolIT in the list). In the example below, the PID is 30978.



5. Once you know the PID, filter the logcat to only display the Self-Certification Tool messages using the filtering options. (See figure below.)



6. The aggregated results PASS/FAIL will be displayed in the PC or laptop console after the test has finished. Below is an example of a PC console output. More examples are provided in Appendix E.

```
Test results for ValidationInstrumentationTestRunner=.
Time: 3.099
OK (2 tests)
```

7. Save your console output because you will need to send every log at the end of the testing process.





For Windows right-click on the console, point to mark, and then click. After that, you will be able to select text. Once text is selected, right-click the title bar, point to edit, and then click to copy.

For Linux select the text in the console and press [ctrl] + [shift] + [c] to copy it.

8. In case of FAIL you can review the logcat file to identify the reason for failure.
9. If you have PASS for all tests, save your logcat file and move to the section "Reporting results". An example of a logcat file is provided in Appendix F.

You can save the logcat file by selecting in your logcat window the lines you want to export and pressing the save button (the one with the diskette icon). You can select all lines with [ctrl] + [a].

ts Java regexes. Prefix with pid; app; tag; or text: to limit scope.

verbose    

Export Selected Items To Text File..

PID	TID	Application	Tag	Text
26760	26773	org.alljoyn.val...	ServiceHelper	Adding AnnouncementHandler for About
26760	26773	org.alljoyn.val...	ServiceHelper	Stopping AboutClient
26760	26773	org.alljoyn.val...	ServiceHelper	Starting AboutClient
26760	27235	org.alljoyn.val...	ioeAboutSe...	MyBusListener.foundAdvertisedName: ':_YgDqkMv.2'
26760	26773	org.alljoyn.val...	ioeAboutSe...	BusAttachment.registerBusObject(AnnouncementReceiver) status = 'OK', []
				ObjPath: '/About'
26760	26773	org.alljoyn.val...	ioeAboutSe...	BusAttachment.registerSignalHandlers(AnnouncementReceiver) status = []
				'OK'
26760	26773	org.alljoyn.val...	ioeAboutSe...	BusAttachment.addMatch() status = OK
26760	26773	org.alljoyn.val...	ServiceHelper	Adding match rule: sessionless='t',type='error'
26760	26773	org.alljoyn.val...	DeviceAnno...	Waiting for About announcement signal from device: 4564545455454; a []
				ppId: 815f638c-6976-4278-8f0f-244fd18d61e2
26760	27235	org.alljoyn.val...	ioeAboutSe...	MyBusListener.nameOwnerChanged(:_YgDqkMv.1, null, :_YgDqkMv.1)
26760	27261	org.alljoyn.val...	ioeAboutSe...	MyBusListener.nameOwnerChanged(org.alljoyn.sl.x_YgDqkMv.x0, null, : []
				_YgDqkMv.1)
26760	27263	org.alljoyn.val...	ioeAboutSe...	MyBusListener.nameOwnerChanged(org.alljoyn.sl.x_YgDqkMv.x0, :_YgDqk []
				Mv.1, null)
26760	27263	org.alljoyn.val...	ioeAboutSe...	MyBusListener.nameOwnerChanged(:_YgDqkMv.1, :_YgDqkMv.1, null)

NOTE

Please use the following coding scheme to name your logcat files:

company_product-name_date_#no.txt

(example: ACME_smartwatch-xyz_2014-08-27_#01.txt)

9 Reporting your results and obtaining Certification

Complete the Submission Form (<https://wiki.allseenalliance.org/certification?&>) and email it along with every log from the PC console output, and every logcat file collected during testing (see section 8.2.1) to certification@allseenalliance.org.

NOTE

Please use this coding scheme to name your files:

company_product-name_date_#no.txt

Here is an example: ACME_smartwatch-xyz_2014-08-27_#01.txt

Once reviewed by the Certification Authority and found that your device complies with the AllSeen Alliance Certification Program requirements, you will receive an email with confirmation granting product certification along with the “Designed for AllSeen” mark and marketing guidelines.

10 Reporting Technical Issues

You can ask questions about the Self-Certification Tool using the following link:

<https://ask.allseenalliance.org/questions/>

NOTE

Make sure you tag it with “validation”.

TIP

The HTML to wiki tool <http://w-i-k-i.appspot.com/> can save Word as HTML and convert it.

You can submit technical issues to JIRA using the following link:

<https://jira.allseenalliance.org/browse/ASACOMP>

Appendix A List of test cases for each service

This appendix provides the list of test cases in each test suite for each service under the scope of the AllSeen Alliance certification.

NOTES

- The list will be replaced by the TCCL once it has been approved.
- Test cases ~~strikeout~~ mean they have been removed from the list of official test cases.

About-v1-01;About announcement received

About-v1-02;Version property consistent with the About announcement

About-v1-03;GetObjectDescription() consistent with the About announcement

About-v1-04;Bus objects consistent with the About announcement

About-v1-05;Standardized interfaces match definitions

About-v1-06;GetAboutData() with default language

About-v1-07;GetAboutData() with each supported language

About-v1-08;GetAboutData() without a specified language

About-v1-09;GetAboutData() for an unsupported language

About-v1-10;GetContent() on the About DeviceIcon

About-v1-11;GetUrl() on the About DeviceIcon

EventsActions-v1-01;Description tag existence in introspection XML and identical XML across different description languages

Config-v1-01;System App AppId equals DeviceId

Config-v1-02;Call a Config interface method without proper authentication

~~Config-v1-03: Version property on Config bus object~~

Config-v1-04;GetConfigurations() method with default language

Config-v1-05;GetConfigurations() method with unspecified language

Config-v1-06;GetConfigurations() method for each supported language

Config-v1-07;GetConfigurations() method with unsupported language

Config-v1-08;UpdateConfigurations() method with a new DeviceName

~~Config-v1-09: UpdateConfigurations for DeviceName == MaxLength~~

~~Config-v1-10: UpdateConfigurations for DeviceName > MaxLength~~

~~Config-v1-11; UpdateConfigurations() method with an empty DeviceName~~

Config-v1-12;UpdateConfigurations() method with a DeviceName containing special characters

Config-v1-13;UpdateConfigurations() method with an unsupported language

Config-v1-14;UpdateConfigurations() method with a DefaultLanguage of another language

Config-v1-15;UpdateConfigurations() method for DefaultLanguage with an unsupported language

Config-v1-16;UpdateConfigurations() method for DefaultLanguage with an unspecified language

~~Config-v1-18; UpdateConfigurations() method for a read-only field~~

Config-v1-19;UpdateConfigurations() method for an invalid field

Config-v1-20;ResetConfigurations() method for DeviceName

Config-v1-21;ResetConfigurations() method for DefaultLanguage (at least one supported language)

Config-v1-22;ResetConfigurations() method for DefaultLanguage (more than one supported language)

~~Config-v1-23; ResetConfigurations() method for a read-only field~~

Config-v1-24;ResetConfigurations() method with an unsupported language

Config-v1-25;ResetConfigurations() method for an invalid field

Config-v1-26;Restart() method

Config-v1-27;Restart() method persists configuration changes

~~Config-v1-28; SetPasscode() method with an empty password~~

Config-v1-29;SetPasscode() method with a new value

Config-v1-30;SetPasscode() method with a one-character value

Config-v1-31;SetPasscode() method with special characters

Config-v1-32;Restart() method persists changed passcode

Config-v1-33;FactoryReset() method

Config-v1-34;FactoryReset() method clears configured data

Config-v1-35;FactoryReset() method resets the passcode

ControlPanel-v1-01;Verify all ControlPanel bus objects

ControlPanel-v1-02;Verify all Container bus objects

ControlPanel-v1-03;Verify all Property bus objects

ControlPanel-v1-04;Verify all LabelProperty bus objects

ControlPanel-v1-05;Verify all Action bus objects
ControlPanel-v1-06;Verify all Dialog bus objects
ControlPanel-v1-07;Verify all ListProperty bus objects
ControlPanel-v1-08;Verify all NotificationAction bus objects
ControlPanel-v1-09;Verify all HTTPControl bus objects
ControlPanel-v1-10;Verify all secured ControlPanel bus objects

Notification-Consumer-v1-01;Basic text message
Notification-Consumer-v1-02;Multiple languages
~~Notification-Consumer-v1-03; Empty language field~~
Notification-Consumer-v1-04;Invalid language field
Notification-Consumer-v1-05;Message priority
Notification-Consumer-v1-06;Custom attributes
Notification-v1-01;Sending of notifications (Notification Producer)

Onboarding-v1-01;Offboard the DUT
Onboarding-v1-02;Onboard the DUT
Onboarding-v1-03;Session joined on Soft AP
~~Onboarding-v1-04;Invalid authType provided to ConfigureWiFi() returns OutOfRange error~~
Onboarding-v1-05;Nonexistent personal AP SSID provided to ConfigureWiFi()
Onboarding-v1-06;Invalid passphrase for personal AP provided to ConfigureWiFi()
Onboarding-v1-07;AuthType value of "any" provided to ConfigureWiFi()
Onboarding-v1-08;GetScanInfo() returns results or FeatureNotAvailable error
Onboarding-v1-09;Call Onboarding method without proper authentication
Onboarding-v1-10;Call Onboarding method after changing the passcode
Onboarding-v1-11;Factory reset
Onboarding-v1-12;Factory reset resets passcode

Audio-v1-01;Validate Stream bus objects
Audio-v1-02;Opening a Stream bus object
Audio-v1-03;Opening and closing a Stream bus object
Audio-v1-04;Closing an unopened Stream bus object
Audio-v1-05; Verify any AudioSink capabilities

Audio-v1-06; Verify any ImageSink capabilities
Audio-v1-07;Verify any Application.MetadataSink capabilities
Audio-v1-08;Configure any AudioSink port
Audio-v1-09;Configure any AudioSink bus object with an invalid configuration
Audio-v1-10;Configure any AudioSink bus object twice.
Audio-v1-11;Check for OwnershipLost signal
Audio-v1-12;Playback on an AudioSink
Audio-v1-13;Pausing a playback on an AudioSink
Audio-v1-14;Flushing a paused AudioSink
Audio-v1-15;Flushing a playing AudioSink
Audio-v1-16;Paused AudioSink remains paused after sending data
Audio-v1-17;Playing an empty AudioSink remains IDLE
Audio-v1-18;Flushing an idle AudioSink
Audio-v1-19;Sending data to an ImageSink
Audio-v1-20;Sending data to an Application.MetadataSink
Audio-v1-21;Setting the mute state on an AudioSink
Audio-v1-22;Setting the volume on an AudioSink
Audio-v1-23;Setting an invalid volume on an AudioSink
Audio-v1-24;Independence of mute and volume on an AudioSink
Audio-v1-25;Synchronize clocks on an AudioSink

LSF_Lamp-v1-01;Service Interface Version equals 1
LSF_Lamp-v1-02;Lamp Service Version equals 1
LSF_Lamp-v1-03;ClearLampFault() method
LSF_Lamp-v1-04;SetOnOff() property
LSF_Lamp-v1-05;SetHue() property
LSF_Lamp-v1-06;SetSaturation()
LSF_Lamp-v1-07;SetColorTemp()
LSF_Lamp-v1-08;SetBrightness()
LSF_Lamp-v1-09;TransitionLampState and verify state and signal
LSF_Lamp-v1-10;ApplyPulseEffect
LSF_Lamp-v1-11;Service interface XML matches
LSF_Lamp-v1-12;Parameters interface version equals 1

LSF_Lamp-v1-13;GetEnergyUsageMilliwatts
LSF_Lamp-v1-14;GetBrightnessLumens
LSF_Lamp-v1-15;Details interface version equals 1
LSF_Lamp-v1-16;GetMake
LSF_Lamp-v1-17;GetModel
LSF_Lamp-v1-18;GetType
LSF_Lamp-v1-19;GetLampType
LSF_Lamp-v1-20;GetLampBaseType
LSF_Lamp-v1-21;GetLampBeamAngle
LSF_Lamp-v1-22;GetDimmable
LSF_Lamp-v1-23;GetColor
LSF_Lamp-v1-24;GetVariableColorTemp
LSF_Lamp-v1-25;GetLampID
LSF_Lamp-v1-26;GetHasEffects
LSF_Lamp-v1-27;GetMinVoltage
LSF_Lamp-v1-28;GetMaxVoltage
LSF_Lamp-v1-29;GetMaxWattage
LSF_Lamp-v1-30;GetIncandescentEquivalent
LSF_Lamp-v1-31;GetMaxLumens
LSF_Lamp-v1-32;GetMinTemperature
LSF_Lamp-v1-33;GetMaxTemperature
LSF_Lamp-v1-34;GetColorRenderingIndex

GWAgent-v1-01;Interfaces Match Definition

Appendix B List of adb command parameters needed for running the test cases

Table 4 provides the list of parameters in an adb command to start testing.

Table 4. adb command parameters for running test cases

Parameter	Description
appld	Globally unique identifier for the application given by its About interface's Appld metadata field
deviceId	Device identifier set by platform-specific means found in the About interface's DeviceId metadata field
testSuiteList	Comma-separated list of the test suites to run (for example, org.alljoyn.validation.testing.suites.about.AboutTestSuite).
testCaseName	Comma-separated list of the test cases to run (for example, About-v1-01, About-v1-02, About-v1-03).
enableInteractive	Determines if user interaction is required during test case execution. Default value is "false". This parameter must be set to "true" for NotificationProducerTestSuite and NotificationConsumerTestSuite test cases.
userInputTimeoutValueInMS	Time in milliseconds to wait for a test to complete before considering it failed. This will override the default timeout value.
org.alljoyn.Onboarding.SoftApSsid	SSID name for the DUT's software access point (required for OnboardingTestSuite test cases) *Note 1.
org.alljoyn:Onboarding.SoftApSecurity	Security type for the DUT's software access point. It can be none, WPA or WPA2_TKIP (required for OnboardingTestSuite test cases) *Note 1.
org.alljoyn:Onboarding.SoftApPassphrase	Passphrase for the DUT's software access point (required for OnboardingTestSuite test cases if the software access point has security) *Note 1.
org.alljoyn:Onboarding.PersonalApSsid	SSID (name) of the access point (required for OnboardingTestSuite test cases) *Note 2.
org.alljoyn:Onboarding.PersonalApSecurity	Security type of the access point. It can be WPA or WPA2 (required for OnboardingTestSuite test cases) *Note 2.
org.alljoyn:Onboarding.PersonalApPassphrase	Passphrase of the access point. (required for OnboardingTestSuite test cases) *Note 2.
org.alljoyn:Audio.TestObjectPath	The path of the Stream bus object being tested

*Note 1: The SSID name, the security type and the passphrase values used by the onboarder (the Self-Certification Tool) to configure the onboarder (DUT software access point, e.g., the tethering mode AP).

*Note 2: SSID name, security type and passphrase of the Wireless Access Point used in the test (i.e., the personal AP, as referred in the AllJoyn Test Case Specifications).

Appendix C List of valid values for the “testCaseName” parameter

Table 5 lists the valid values for the parameter testCaseName in the adb command.

Table 5. testCaseName parameter values

testCaseName
About-v1-01
About-v1-02
About-v1-03
About-v1-04
About-v1-05
About-v1-06
About-v1-07
About-v1-08
About-v1-09
About-v1-10
About-v1-11
EventsActions-v1-01
Config-v1-01
Config-v1-02
Config-v1-04
Config-v1-05
Config-v1-06
Config-v1-07
Config-v1-08
Config-v1-12
Config-v1-13
Config-v1-14
Config-v1-15
Config-v1-16
Config-v1-19
Config-v1-20
Config-v1-21
Config-v1-22
Config-v1-24
Config-v1-25

testCaseName
Config-v1-26
Config-v1-27
Config-v1-29
Config-v1-30
Config-v1-31
Config-v1-32
Config-v1-33
Config-v1-34
Config-v1-35
ControlPanel-v1-01
ControlPanel-v1-02
ControlPanel-v1-03
ControlPanel-v1-04
ControlPanel-v1-05
ControlPanel-v1-06
ControlPanel-v1-07
ControlPanel-v1-08
ControlPanel-v1-09
ControlPanel-v1-10
Notification-Consumer-v1-01
Notification-Consumer-v1-02
Notification-Consumer-v1-04
Notification-Consumer-v1-05
Notification-Consumer-v1-06
Notification-v1-01
Onboarding-v1-01
Onboarding-v1-02
Onboarding-v1-03
Onboarding-v1-05
Onboarding-v1-06
Onboarding-v1-07
Onboarding-v1-08
Onboarding-v1-09
Onboarding-v1-10
Onboarding-v1-11
Onboarding-v1-12
Audio-v1-01

testCaseName
Audio-v1-02
Audio-v1-03
Audio-v1-04
Audio-v1-05
Audio-v1-06
Audio-v1-07
Audio-v1-08
Audio-v1-09
Audio-v1-10
Audio-v1-11
Audio-v1-12
Audio-v1-13
Audio-v1-14
Audio-v1-15
Audio-v1-16
Audio-v1-17
Audio-v1-18
Audio-v1-19
Audio-v1-20
Audio-v1-21
Audio-v1-22
Audio-v1-23
Audio-v1-24
Audio-v1-25
Audio-v1-26
Audio-v1-27
Audio-v1-28
LSF_Lamp-v1-01
LSF_Lamp-v1-02
LSF_Lamp-v1-03
LSF_Lamp-v1-04
LSF_Lamp-v1-05
LSF_Lamp-v1-06
LSF_Lamp-v1-07
LSF_Lamp-v1-08
LSF_Lamp-v1-09
LSF_Lamp-v1-10
LSF_Lamp-v1-11
LSF_Lamp-v1-12
LSF_Lamp-v1-13

testCaseName
LSF_Lamp-v1-14
LSF_Lamp-v1-15
LSF_Lamp-v1-16
LSF_Lamp-v1-17
LSF_Lamp-v1-18
LSF_Lamp-v1-19
LSF_Lamp-v1-20
LSF_Lamp-v1-21
LSF_Lamp-v1-22
LSF_Lamp-v1-23
LSF_Lamp-v1-24
LSF_Lamp-v1-25
LSF_Lamp-v1-26
LSF_Lamp-v1-27
LSF_Lamp-v1-28
LSF_Lamp-v1-29
LSF_Lamp-v1-30
LSF_Lamp-v1-31
LSF_Lamp-v1-32
LSF_Lamp-v1-33
LSF_Lamp-v1-34
GWAgent-v1-01

Appendix D Writing adb commands to execute tests

The general format of an adb command is:

```
adb shell am instrument -w -e appId DUT_APPID -e deviceId DUT_DEVICE_ID -e  
PARAMETER VALUE -e PARAMETER VALUE -e testSuiteList -e testSuiteList  
TEST_SUITE_NAME -e testCaseName TEST_CASE_NAME  
org.alljoyn.validation.validation_tests.validation_tests_it/org.alljoyn.validat  
ion.testing.instrument.ValidationInstrumentationTestRunner
```

NOTE

If you copy any of the examples using a rich text editor like Microsoft Word instead of a plain text editor like Notepad, it may change “-” to “—” and the command will not be valid.

To run individual test cases, you need to include “-e testCaseName TEST_CASE_NAME” after the reference to the test suite. Various test cases of a test suite can be aggregated by just separating each TEST_CASE_NAME by commas without spaces between them. The list of valid values for testCaseName needed to build adb commands for individual test cases can be found in Appendix C. If you want to run the complete test suite, do not include “-e testCaseName TEST_CASE_NAME” in the adb command.

The appId and deviceId parameters are always required in the adb commands. Additional parameters can be added in the adb commands. To do that, include “-e PARAMETER VALUE” in the adb command line, where PARAMETER is the name of the parameter and VALUE is its value. Parameters can be found in Appendix C.

Table 6 lists the test suite names to be used in the adb commands to run a specific test suite as well as the list of other mandatory parameters to be added to the adb command.

NOTE

The asterisks for the Notification and Onboarding test suites below correspond to notes provided in Appendix D.3 and D.5.

Table 6. Test suite names

Service	TEST_SUITE_NAME	MANDATORY PARAMETERS FOR THE TEST SUITE
Core (About Feature)	org.alljoyn.validation.testing.suites.about.AboutTestSuite	deviceId appId
Core (Events & Actions Feature)	org.alljoyn.validation.testing.suites.eventsactions.EventsActionsTestSuite	deviceId appId
Control Panel Service	org.alljoyn.validation.testing.suites.controlpanel.ControlPanelTestSuite	deviceId appId

Service	TEST_SUITE_NAME	MANDATORY PARAMETERS FOR THE TEST SUITE
Notification Service (producer)*	org.alljoyn.validation.testing.suites.notification.NotificationProducerTestSuite	deviceId appId enableInteractive userInputTimeoutValueInMS
Notification Service (consumer)*	org.alljoyn.validation.testing.suites.notification.NotificationConsumerTestSuite	deviceId appId enableInteractive userInputTimeoutValueInMS
Onboarding Service**	org.alljoyn.validation.testing.suites.onboarding.OnboardingTestSuite	deviceId appId org.alljoyn.Onboarding.SoftApSsid org.alljoyn:Onboarding.SoftApSecurity org.alljoyn:Onboarding.SoftApPassphrase org.alljoyn:Onboarding.PersonalApSsid org.alljoyn:Onboarding.PersonalApSecurity org.alljoyn:Onboarding.PersonalApPassphrase
Configuration Service	org.alljoyn.validation.testing.suites.config.ConfigurationTestSuite	deviceId appId
Audio Service	org.alljoyn.validation.testing.suites.audio.AudioTestSuite	deviceId appId org.alljoyn:Audio.TestObjectPath
Lighting Service	org.alljoyn.validation.testing.suites.LSF_LampTestSuite	deviceId appId
Gateway Service	org.alljoyn.validation.testing.suites.gwagent.GWAgentTestSuite	deviceId appId

NOTE

You shall run all the tests cases in the corresponding test suite of the core or service your product supports. If a product only has one role in a service (e.g., Notification Service) it only has to pass the tests for that role in that test suite service (e.g., Notification Consumer).

Below are several examples on how to build an adb command for one test suite, a combination of test suites, a single test case or a combination of test cases.

D.1 Case 1 (About Service)

Running the complete test suite for the About Service for a DUT with Device Id = DUT_DEVICE_ID and App Id = DUT_APP_ID

```
adb shell am instrument -w -e appId DUT_APPID -e deviceId DUT_DEVICE_ID -e testSuiteList org.alljoyn.validation.testing.suites.about.AboutTestSuite
```

```
org.alljoyn.validation.validation_tests.validation_tests_it/org.alljoyn.validation.testing.instrument.ValidationInstrumentationTestRunner
```

Multiple test suites can be put into one command line by just separating each TEST_SUITE_NAME by commas without spaces between them.

D.2 Case 2 (About and Config)

Running two test suites (About and Config) for a DUT with Device Id = DUT_DEVICE_ID and App Id = DUT_APP_ID

```
adb shell am instrument -w -e appId DUT_APPID -e deviceId DUT_DEVICE_ID -e testSuiteList org.alljoyn.validation.testing.suites.about.AboutTestSuite,org.alljoyn.validation.testing.suites.config.ConfigTestSuite org.alljoyn.validation.validation_tests.validation_tests_it/org.alljoyn.validation.testing.instrument.ValidationInstrumentationTestRunner
```

D.3 Case 3 (Notification Consumer)

NOTE

The Notification Consumer and Notification Producer test suite requires user interaction at the Self-Certification Tool to complete the test cases. Thus, the parameter for user interaction enabling and timeout must be in the adb command.

Running a test suite (Notification Consumer) for a DUT with Device Id = DUT_DEVICE_ID and App Id = DUT_APP_ID

```
adb shell am instrument -w -e appId DUT_APPID -e deviceId DUT_DEVICE_ID -e userInputTimeoutValueInMS 5000 -e enableInteractive true -e testSuiteList org.alljoyn.validation.testing.suites.notification.NotificationConsumerTestSuite org.alljoyn.validation.validation_tests.validation_tests_it/org.alljoyn.validation.testing.instrument.ValidationInstrumentationTestRunner
```

D.4 Case 4 (Notification Producer)

Running a test suite (Notification Producer) for a DUT with Device Id = DUT_DEVICE_ID and App Id = DUT_APP_ID

```
adb shell am instrument -w -e appId DUT_APPID -e deviceId DUT_DEVICE_ID -e userInputTimeoutValueInMS 5000 -e enableInteractive true -e testSuiteList org.alljoyn.validation.testing.suites.notification.NotificationProducerTestSuite org.alljoyn.validation.validation_tests.validation_tests_it/org.alljoyn.validation.testing.instrument.ValidationInstrumentationTestRunner
```

D.5 Case 5 (Onboarding)

NOTE

To ensure Onboarding can take place, you need to add information, i.e. some parameters in the command line, to select the correct Wi-Fi network. Thus you need to add the corresponding parameters in the adb command line.

Running a test suite (Onboarding) for a DUT with Device Id = DUT_DEVICE_ID and App Id = DUT_APP_ID

```
adb shell am instrument -w -e appId DUT_APPID -e deviceId DUT_DEVICE_ID -e
org.alljoyn.Onboarding.SoftApSsid softapssid -e
org.alljoyn.Onboarding.SoftApSecurity softapsecurity -e
org.alljoyn.Onboarding.SoftApPassphrase at4wireless -e
org.alljoyn.Onboarding.PersonalApSsid personalapssid -e
org.alljoyn.Onboarding.PersonalApSecurity personalapsecurity -e
org.alljoyn.Onboarding.PersonalApPassphrase personalappassphrase -e
testSuiteList
org.alljoyn.validation.testing.suites.onboarding.OnboardingTestSuite
org.alljoyn.validation.validation_tests.validation_tests_it/org.alljoyn.validat
ion.testing.instrument.ValidationInstrumentationTestRunner
```

D.6 Case 6 (About-v1-01)

Running one test case (About-v1-01) from a suite for a DUT with Device Id = DUT_DEVICE_ID and App Id = DUT_APP_ID

```
adb shell am instrument -w -e appId DUT_APPID -e deviceId DUT_DEVICE_ID -e
testSuiteList org.alljoyn.validation.testing.suites.about.AboutTestSuite -e
testCaseName About-v1-01
org.alljoyn.validation.validation_tests.validation_tests_it/org.alljoyn.validat
ion.testing.instrument.ValidationInstrumentationTestRunner
```

D.7 Case 7 (About-v1-01 and About-v1-02 Service)

Running two test cases (About-v1-01 and test case About-v1-02) from one suite for a DUT with DeviceId = DUT_DEVICE_ID and App Id = DUT_APP_ID

```
adb shell am instrument -w -e appId DUT_APPID -e deviceId DUT_DEVICE_ID -e
testSuiteList org.alljoyn.validation.testing.suites.about.AboutTestSuite -e
testCaseName About-v1-01,About-v1-02
org.alljoyn.validation.validation_tests.validation_tests_it/org.alljoyn.validat
ion.testing.instrument.ValidationInstrumentationTestRunner
```

Appendix E PC console results examples

Below are some examples of results in the PC console.

E.1 One test case executed with result FAIL

One test case executed (About-v1-01) with result FAIL

```
Failure in testAbout_v1_01_AboutAnnouncement:
junit.framework.AssertionFailedError: Timed out waiting for About announcement
at
org.alljoyn.validation.testing.suites.about.AboutTestSuite.setUp(AboutTestSuite
.java:115) at
org.alljoyn.validation.testing.instrument.InstrumentationTestCaseWrapper.runTes
t(InstrumentationTestCaseWrapper.java:105) at
android.test.AndroidTestRunner.runTest(AndroidTestRunner.java:190) at
android.test.AndroidTestRunner.runTest(AndroidTestRunner.java:175) at
android.test.InstrumentationTestRunner.onStart(InstrumentationTestRunner.java:5
55) at
android.app.Instrumentation$InstrumentationThread.run(Instrumentation.java:1698
)
```

```
Test results for ValidationInstrumentationTestRunner=.F
Time: 30.203
```

```
FAILURES!!!
Tests run: 1, Failures: 1, Errors: 0
```

E.2 Two test cases executed both resulting in FAIL

Two test cases (About-v1-01 and About-v1-02) executed both resulting in FAIL.

```
Failure in testAbout_v1_01_AboutAnnouncement:
junit.framework.AssertionFailedError: Timed out waiting for About announcement
at
org.alljoyn.validation.testing.suites.about.AboutTestSuite.setUp(AboutTestSuite
.java:115) at
org.alljoyn.validation.testing.instrument.InstrumentationTestCaseWrapper.runTes
t(InstrumentationTestCaseWrapper.java:105) at
android.test.AndroidTestRunner.runTest(AndroidTestRunner.java:190) at
android.test.AndroidTestRunner.runTest(AndroidTestRunner.java:175) at
android.test.InstrumentationTestRunner.onStart(InstrumentationTestRunner.java:5
55) at
android.app.Instrumentation$InstrumentationThread.run(Instrumentation.java:1698
)
```

```
Failure in testAbout_v1_02_AboutVersion:
```

```
junit.framework.AssertionFailedError: Timed out waiting for About announcement
at
org.alljoyn.validation.testing.suites.about.AboutTestSuite.setUp(AboutTestSuite
.java:115) at
org.alljoyn.validation.testing.instrument.InstrumentationTestCaseWrapper.runTes
t(InstrumentationTestCaseWrapper.java:105) at
android.test.AndroidTestRunner.runTest(AndroidTestRunner.java:190) at
android.test.AndroidTestRunner.runTest(AndroidTestRunner.java:175) at
android.test.InstrumentationTestRunner.onStart(InstrumentationTestRunner.java:5
55) at
android.app.Instrumentation$InstrumentationThread.run(Instrumentation.java:1698
)
```

```
Test results for ValidationInstrumentationTestRunner=.F.F
Time: 60.436
```

```
FAILURES!!!
Tests run: 2, Failures: 2, Errors: 0
```

E.3 One test case executed with result PASS

One test case executed (About-v1-01) with result PASS

```
Test results for ValidationInstrumentationTestRunner=.
Time: 2.222
```

```
OK (1 test)
```

E.4 Two test cases executed with result PASS

Two test cases executed (About-v1-01 and About-v1-02) with result PASS

```
Test results for ValidationInstrumentationTestRunner=.
Time: 3.099
```

```
OK (2 tests)
```

E.5 One test suite executed with two test cases FAIL

One test suite executed (About test suite) with two test cases FAIL

```
Failure in testAbout_v1_06_GetAboutForDefaultLanguage:
junit.framework.AssertionFailedError: DateOfManufacture field value 10/10/2014
does not match expected date pattern YYYY-MM-DD at
org.alljoyn.validation.testing.suites.about.AboutTestSuite.validateDateOfManufa
cture(AboutTestSuite.java:572) at
org.alljoyn.validation.testing.suites.about.AboutTestSuite.verifyAboutMap(About
TestSuite.java:503) at
org.alljoyn.validation.testing.suites.about.AboutTestSuite.testAbout_v1_06_GetA
boutForDefaultLanguage(AboutTestSuite.java:226) at
```

```
java.lang.reflect.Method.invokeNative(Native Method) at
org.alljoyn.validation.testing.instrument.InstrumentationTestCaseWrapper.runTest(InstrumentationTestCaseWrapper.java:105) at
android.test.AndroidTestRunner.runTest(AndroidTestRunner.java:190) at
android.test.AndroidTestRunner.runTest(AndroidTestRunner.java:175) at
android.test.InstrumentationTestRunner.onStart(InstrumentationTestRunner.java:55) at
android.app.Instrumentation$InstrumentationThread.run(Instrumentation.java:1698)
```

```
Failure in testAbout_v1_07_GetAboutForSupportedLanguages:
junit.framework.AssertionFailedError: DateOfManufacture field value 10/10/2014
does not match expected date pattern YYYY-MM-DD at
org.alljoyn.validation.testing.suites.about.AboutTestSuite.validateDateOfManufacture(AboutTestSuite.java:572) at
org.alljoyn.validation.testing.suites.about.AboutTestSuite.verifyAboutMap(AboutTestSuite.java:503) at
org.alljoyn.validation.testing.suites.about.AboutTestSuite.validateSupportedLanguagesAboutMap(AboutTestSuite.java:469) at
org.alljoyn.validation.testing.suites.about.AboutTestSuite.testAbout_v1_07_GetAboutForSupportedLanguages(AboutTestSuite.java:243) at
java.lang.reflect.Method.invokeNative(Native Method) at
org.alljoyn.validation.testing.instrument.InstrumentationTestCaseWrapper.runTest(InstrumentationTestCaseWrapper.java:105) at
android.test.AndroidTestRunner.runTest(AndroidTestRunner.java:190) at
android.test.AndroidTestRunner.runTest(AndroidTestRunner.java:175) at
android.test.InstrumentationTestRunner.onStart(InstrumentationTestRunner.java:55) at
android.app.Instrumentation$InstrumentationThread.run(Instrumentation.java:1698)
```

```
Test results for ValidationInstrumentationTestRunner= F.F
Time: 14.473
```

```
FAILURES!!!
Tests run: 11, Failures: 2, Errors: 0
```

In Appendix F, you can find some extracts of the logcat output for the execution in this example.

Appendix F Logcat output example

Below you can find an extract of the logcat output for one test suite executed (About test suite) with two test cases FAIL.

```
08-20 09:57:51.510: E/SELinux(2021): selinux_android_seapp_context_reload:
seapp_contexts file is loaded from /data/security/spota/seapp_contexts
08-20 09:57:51.631: E/ValidationInstrumentationTestRunnerHelper(2021):
testCaseKeywords: null
08-20 09:57:51.661: W/dalvikvm(2021): method
Landroid/test/InstrumentationTestRunner$StringResultPrinter;.print incorrectly
overrides package-private method with same name in Ljunit/textui/ResultPrinter;
08-20 09:57:51.661: I/TestRunner(2021): started:
testAbout_v1_01_AboutAnnouncement(org.alljoyn.validation.testing.instrument.Ins
trumentationTestCaseWrapper)
08-20 09:57:51.711: E/PERMISSION_MGR(2021): 1.081 ***** ERROR
PERMISSION_MGR lepDisp ...ndroid/PermissionDB.cc:202 | 0x0001
08-20 09:57:51.721: E/NETWORK(2021): 1.090 ***** ERROR NETWORK lepDisp
common/os/posix/Socket.cc:283 | 0x0004
08-20 09:57:51.741: I/ioeAboutServiceImpl(2021):
BusAttachment.registerBusObject(AnnouncmentReceiver) status = 'OK', ObjPath:
'/About'
08-20 09:57:51.751: I/ioeAboutServiceImpl(2021):
BusAttachment.registerSignalHandlers(AnnouncmentReceiver) status = 'OK'
08-20 09:57:51.751: I/ioeAboutServiceImpl(2021): BusAttachment.addMatch()
status = OK
08-20 09:57:51.751: I/DeviceAnnounceHandler(2021): Waiting for About
announcement signal from device: 4564545455454; appId: 6475a67c-4d00-481e-9e56-
7fba5c514778
08-20 09:58:15.896: I/ServiceHelper(2021): Session established to peer:
:j8WNer7W.2
08-20 09:58:16.187: E/ALLJOYN_JAVA(2021): 25.565 ***** ERROR ALLJOYN_JAVA
external ...a/jni/alljoyn_java.cc:6042 | 0x909b
08-20 09:58:16.187: I/ServiceHelper(2021): Ignoring
ALLJOYN_JOINSESSION_REPLY_ALREADY_JOINED error code
08-20 09:58:16.207: E/ALLJOYN_JAVA(2021): 25.587 ***** ERROR ALLJOYN_JAVA
external ...a/jni/alljoyn_java.cc:6170 | 0x908a
08-20 09:58:16.337: I/TestRunner(2021): finished:
testAbout_v1_01_AboutAnnouncement(org.alljoyn.validation.testing.instrument.Ins
trumentationTestCaseWrapper)
08-20 09:58:16.337: I/TestRunner(2021): passed:
testAbout_v1_01_AboutAnnouncement(org.alljoyn.validation.testing.instrument.Ins
trumentationTestCaseWrapper)
```

Appendix G Hints to Troubleshoot adb Commands

G.1 What to do when adb command is not recognized

Instructions for Windows

If your console does not detect the command, make sure you have added “adb.exe” path to Windows’ environment variables. For instance: “C:\Users\rurc\development\adt-bundle-windows-x86_64-20140702\sdk\platform-tools”.

Complete the following instructions to add an environment variable in Windows:

1. From the Desktop, right-click **My Computer** and click **Properties**.
2. Click **Advanced System Settings** link in the left column.
3. In the System Properties window, click the **Environment Variables** button.
4. Add a new environment variable whose name is going to be “Path” and its value the path to the “platform-tools” folder where adb executable is located inside the Android SDK.

Otherwise, you may execute “adb” using the full path, for example:

“C:\Users\rurc\Development\adt-bundle-windows-x86_64-20140702\sdk\platform-tools\adb.exe” followed by its parameters.

Instructions for Linux

Make sure you have added the path where adb executable is located to Linux path.

1. Execute the following command:

```
$ export PATH=$PATH:ADB_FOLDER
```

Replace ADB_FOLDER with the path where your adb executable is located, you may need to add 'sudo' to the beginning of this command.

2. After that, execute below command:

```
$ echo $PATH
```

You should now see your adb executable directory added to the end of the \$PATH variable.

G.2 What to do when “adb devices” does not show my connected device

The expected output is this:

```
List of devices attached
3096d3ee          device
```

Instructions for Windows

If you get something that different from the above, please make sure your device is correctly detected in Windows' device manager and that you have adb drivers installed in your Android SDK device manager.

Instructions for Linux

If your device does not appear, make sure you have added the USB Vendor ID that corresponds to your device manufacturer to Linux "udev" rules. Official instructions are provided by Google on the link below:

<http://developer.android.com/tools/device.html>

G.3 Common adb command errors

Below some typical adb command errors are listed.

- adb command with “–” instead of “-” (while copying a command with a rich text editor be careful as it may change “-” to “–”)
- Parameter missing in the adb command. (Check Appendix B to find adb command parameters.)
- Wrong value in a parameter (e.g., wrong ssid)
- Value in the parameter out of range
- One or more blank spaces in a command line (testSuiteNames must not have blank spaces between them).
- Commas missing (testCaseNames or testSuiteNames must be separated by commas)