

AllSeen Alliance Self-Certification Developer Guide

Version 14.06 Update 1

October 1, 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.....	4
2 Terminology	5
3 Environment Overview.....	6
4 Hardware Requirements	9
4.1 Hardware requirements for the Self-Certification Tool	9
4.2 Hardware requirements for the optional emulated DUT	9
4.3 Hardware requirements for Wi-Fi Access Point	9
5 Meeting Software Prerequisites	10
5.1 PC or laptop.....	10
6 Selecting a TCCL release.....	Error! Bookmark not defined.
7 How to Troubleshoot your Test Environment by Building and Testing an Emulated DUT	11
7.1 Building an emulated DUT.....	11
7.1.1 For 14.02 release	12
7.1.2 For 14.06 release	12
7.1.3 For both 14.02 and 14.06 releases	12
7.2 Connecting the emulated DUT and the Self-Certification Tool	12
7.2.1 Connecting through a wireless access point	12
7.2.2 Connecting through tethering (optional)	13
7.3 Obtaining emulated DUT app and device IDs	14
7.4 Testing the emulated DUT with the Self-Certification Tool	16
8 Building the Self-Certification Tool.....	17
8.1 Getting the libraries	17
8.1.1 For 14.02 release	17
8.1.2 For 14.06 release	17
8.2 Downloading AllJoyn Self-Certification Tool source code	17
8.3 Building the dependencies	18
8.4 Installing the Self-Certification Tool APK in the Android device that will become the Self-Certification Tool	19
Appendix A Hints to troubleshoot adb commands	21
A.1 What to do when adb command is not recognized.....	21
A.2 What to do when “adb devices” does not show my connected device	21

Figures

Figure 1. Overview of the Self-Certification Tool and the DUT	6
Figure 2. Overview of the Self-Certification Tool and its Wi-Fi connections	6
Figure 3. PC or Laptop connection to download the Self-Certification Tool APK.....	7
Figure 4. PC or Laptop connection to download the sample APK into the Emulated DUT	7
Figure 5. PC or Laptop connection to download the sample APK with the Onboarding client	7
Figure 6. General overview of the troubleshooting environment*	11
Figure 7. Enabling Wi-Fi hotspot in the emulated DUT	13
Figure 8. Self-Certification Tool APK information view	20

Tables

Table 1. PC or laptop software requirements	10
Table 2. PC or laptop software requirements	10

1 Introduction

Instructions on how to download and run the Self-Certification Tool tests needed to certify your product are given in the *AllSeen Alliance Self-Certification Program User Guide*. This document provides instructions on how to verify the downloaded Self-Certification Tool is working and on how to build yourself the Self-Certification Tool.

1.1 Document organization

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

Section 3 provides an overview of the test environment, troubleshooting environment, and the developer's development.

Section 4 and 5 provide details on the hardware you need and the software that needs to be installed in the corresponding hardware elements.

Section 6 provides instructions as to how to troubleshoot your test environment before you start testing. This is a recommended step if you are not familiar with the technology or it is the first time you test a product.

Section 7 provides instructions on how to compile and build the Self-Certification Tool. Testing with your self-built Self-Certification Tool should not be used for Certification. For Certification testing you must use the Self-Certification Tool APK provide by the AllSeen Alliance.

This document is intended for software engineers and assumes familiarity with the AllJoyn SDK and Java programming language.

1.2 Release history

Release version	Date	What changed
14.06	9/2/2014	Initial release
14.06 Update 1	10/1/2014	Corrected hyperlinks

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.
Emulated DUT	An Android device, different from the one hosting the Self-Certification Tool, where you can download sample applications to emulate a real product.
Maven	Maven is a software project management and comprehension tool. Based on the concept of a project object model (POM), Maven can manage a project's build, reporting and documentation from a central piece of information.
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 Environment Overview

This section describes the connections or environments you need to set up to: test with the Self-Certification Tool, to install the Self-Certification Tool, to download and install the emulated DUT provided by the AllSeen Alliance, to troubleshoot the Self-Certification Tool installation, or to compile and build the Self-Certification Tool for purposes other than certifying your product.

Figure 1 provides a general overview of how the Self-Certification Tool and your product under test (i.e., the DUT) are connected via the same multicast group.

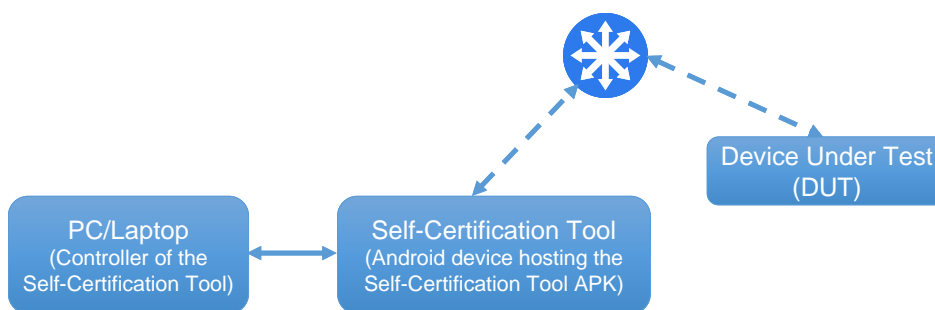


Figure 1. Overview of the Self-Certification Tool and the DUT

Figure 2 shows how the Self-Certification Tool and the DUT use the Wi-Fi interface provided by an external wireless access point to connect to each other. Also the PC/laptop and the Android device hosting the Self-Certification Tool APK need to be connected via USB. This is the configuration to use for official testing.

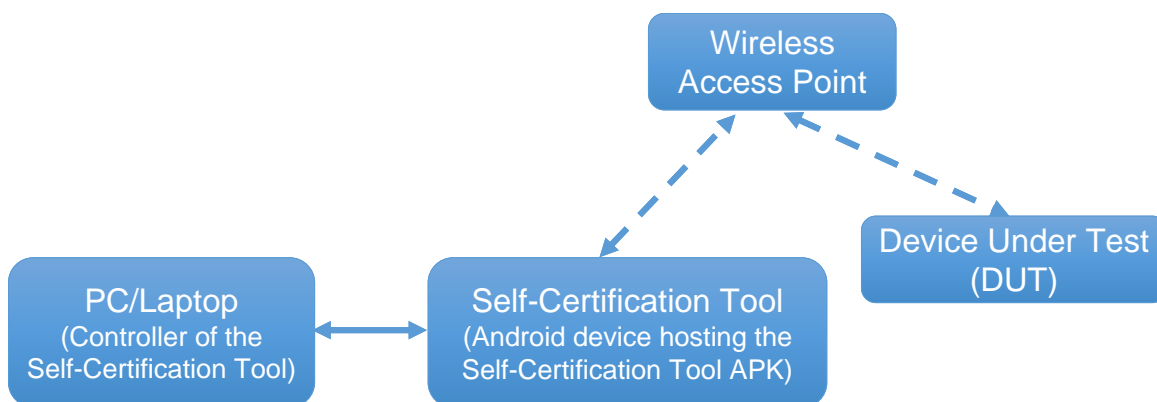


Figure 2. Overview of the Self-Certification Tool and its Wi-Fi connections

An alternative wireless Wi-Fi interface that does not need an external wireless access point can be achieved using the tethering mode if the DUT is a mobile phone that has internet access through a cellular network.

Figure 3 shows the USB connection between the PC/laptop and the Android device you must have in order to download the Self-Certification Tool APK into the Android device to be used as Self-Certification Tool.

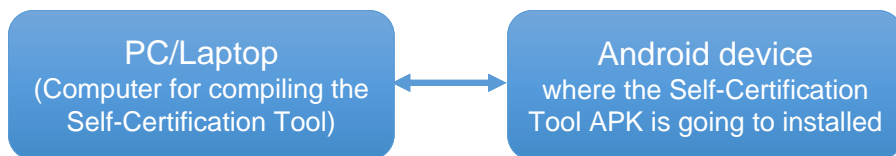


Figure 3. PC or Laptop connection to download the Self-Certification Tool APK

To ensure the Self-Certification Tool is working before you actually start testing your product, you need another Android device that will host the sample application including the Onboarding server (AboutOnbConfServ.apk). This Android device with the sample application becomes an emulator of a DUT (Device Under Test). Building an emulated DUT is optional. You do not need to that if you already have a device ready for testing.

To download the AboutOnbConfServ.apk to the emulated DUT, you need a PC or laptop as shown in Figure 4. This PC or laptop can be the same that is used to control the Self-Certification Tool. The emulated DUT Android device cannot be the same as the Android device hosting the Self-Certification Tool APK.

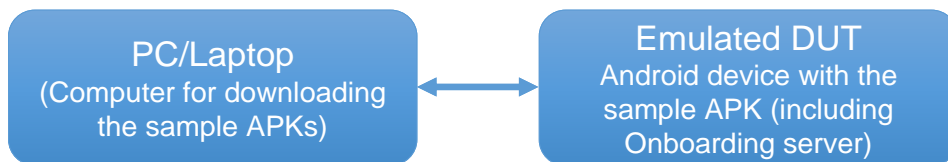


Figure 4. PC or Laptop connection to download the sample APK into the Emulated DUT

To onboard the emulated DUT, you need an Onboarding client. The sample APK with the Onboarding client (OnboardingSampleClient.apk) has to be installed in a separate Android device. The Android device hosting the OnboardingSampleClient.apk can be the same Android device as the one used to host the Self-Certification Tool APK. Figure 5 depicts the USB connection between the PC/laptop and the Android device you need to download the Onboarding client.

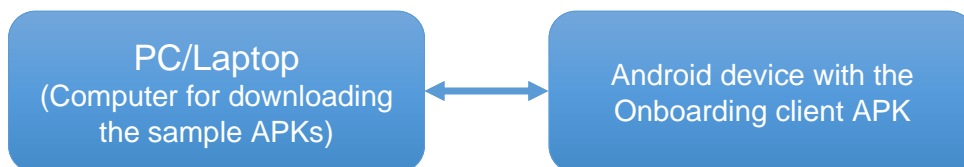


Figure 5. PC or Laptop connection to download the sample APK with the Onboarding client

The PC or laptop used to download the OnboardingSampleClient.apk can be the same used to control the Self-Certification Tool.

If you want to compile and build the Self-Certification Tool yourself, you need a PC or laptop with the required software installed (see section on software prerequisites). The PC or laptop used to compile the Self-Certification Tool can be the same as the one used to compile and generate the Self-Certification Tool APK or to control the Self-Certification Tool. You may want to compile the Self-Certification Tool to pretest future releases (or to create sample APKs that are not yet available).

4 Hardware Requirements

As a summary to cover all environments, you need the following hardware:

- PC or laptop (the controller of the Self-Certification Tool)
- Android device (the Self-Certification Tool itself), which can also host the Onboarding client
- Wireless Access Point, which provides the transport between the Self-Certification Tool and your DUT
- Another Android device to host the samples applications (the emulated DUT) with the Onboarding server.

4.1 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) or to host the OnboardingSampleClient.apk
 - 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

4.2 Hardware requirements for the optional emulated DUT

Android device to build an emulated DUT (optional).

- Processor: ARM based with at least 1 GHz.
- RAM memory: 512 MB or more.
- Mobile network connection (optional).
- Minimum system version is Jelly Bean 4.1.2 (API 16).

4.3 Hardware requirements for Wi-Fi Access Point

Wireless Access Point which supports 2.4 GHz 802.11n

5 Meeting Software Prerequisites

5.1 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, go to Appendix A.

If you also want to compile and build the Self-Certification Tool yourself instead of using the APK provided for the Android device by the AllSeen Alliance, you need to install the software listed in Table 2 on your PC or laptop.

Table 2. PC or laptop software requirements to build the self-certification tool

Software	Minimum version	URL
Apache Maven	3.1.1	http://maven.apache.org/download.cgi
Git	N/A	http://git-scm.com/downloads

6 How to Troubleshoot your Test Environment by Building and Testing an Emulated DUT

If your product is not yet ready you can ensure the Self-Certification Tool works by following the instructions given in this section. This process is optional.

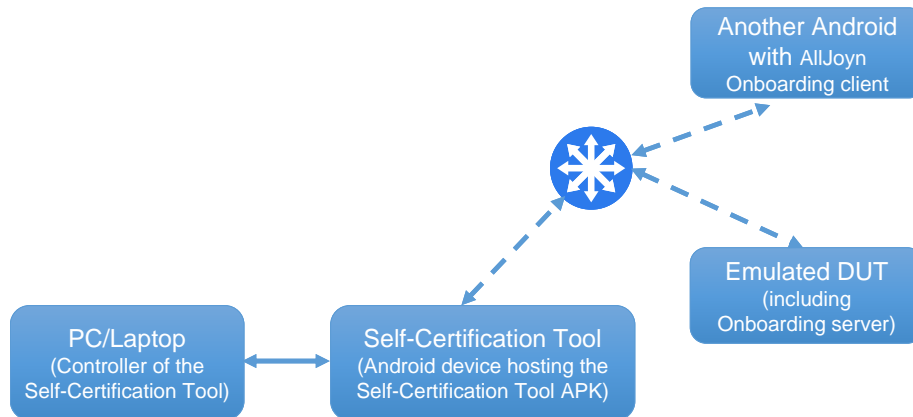


Figure 6. General overview of the troubleshooting environment (see note below)

NOTE

Although the picture depicts two Android devices (one hosting the Self-Certification Tool APK and another hosting the Onboarding client APK) both APKs can be installed in the same Android physical device, but both APKs need to run in parallel.

6.1 Building an emulated DUT

This section assumes you already have installed the Self-Certification Tool for your selected TCCL release in an Android device (the Self-Certification Tool itself and Android device hosting the Self-Certification Tool APK) according to Section 6 in the separate Self-Certification Program User Guide.

To build an emulated DUT, the AllSeen Alliance has provided a sample application which includes the Onboarding server. In order to be able to test the emulated DUT you need to know its deviceId and appId. Another sample APK with an Onboarding client has been provided by the AllSeen Alliance in order to help extract the deviceId and appId of the emulated DUT.

Both sample APKs are available for releases 14.02 and 14.06 here:
<http://mirrors.us.kernel.org/allseenalliance/alljoyn/apk/>.

6.1.1 For 14.02 release

1. Download the file "AllJoyn Onboarding Service Framework SDK 14.02.00 - Android" from: <https://www.alljoyn.org/docs-and-downloads>.
2. Extract the file you have just downloaded.
3. Sample applications to be installed can be found in "alljoyn-android\services\alljoyn-onboarding-14.02.00-rel\tools".

6.1.2 For 14.06 release

1. Download the file "Onboarding SDK" in Android section from: <https://allseenalliance.org/source-code>.
2. Extract the file you have just downloaded.
3. Sample applications to be installed can be found in "alljoyn-android\services\alljoyn-onboarding-14.06.00-rel\tools".

6.1.3 For both 14.02 and 14.06 releases

1. Move to the previous folder using the shell.
2. Connect the Android device that will work as an emulated DUT to the PC or laptop via USB cable (see Figure 4).
3. Use the following adb command to install "AboutConfOnbServer.apk" in the emulated DUT.

```
adb install AboutConfOnbServer.apk
```
4. Connect the PC or laptop to the Android device where you will install the Onboarding client sample APK. This second Android device can be the same that you have used to host the Self-Certification Tool (see Figure 5).
5. Use the following adb command to install "OnboardingSampleClient.apk" in the Android device.

```
adb install OnboardingSampleClient.apk
```

6.2 Connecting the emulated DUT and the Self-Certification Tool

This section describes two methods to connect the emulated DUT to the Self-Certification Tool.

6.2.1 Connecting through a wireless access point

This section describes how to connect the Android device acting as an emulated DUT with the Onboarding client using the Wi-Fi network provided by a wireless access point. Note that the Android device hosting the Onboarding client can be the same as the Android device hosting the Self-Certification Tool APK.

1. Connect the Android device that works as an emulated DUT to a wireless access point.
2. Connect the Android device that hosts the sample Onboarding client to the same wireless access point.
3. Open "AboutConfOnbServer" in the emulated DUT. Nothing should apparently happen when you touch on the icon. This is because the service starts in the background.
4. Open the Onboarding Client in the other Android device.
5. Find the wireless access point inside the "OnboardingSampleClient", and join it. Then press "Connect to AllJoyn" button.
6. If process was successful you should be able to see the "AJ connect done" message in the Android device.

6.2.2 Connecting through tethering (optional)

This section describes how to configure a Wi-Fi hotspot in the emulated DUT by tethering and how to use "OnboardingSampleClient" APK to connect this device to the recently created network. You must have a mobile network connection to connect both devices by tethering.

1. Configure a Wi-Fi hotspot in the emulated DUT.

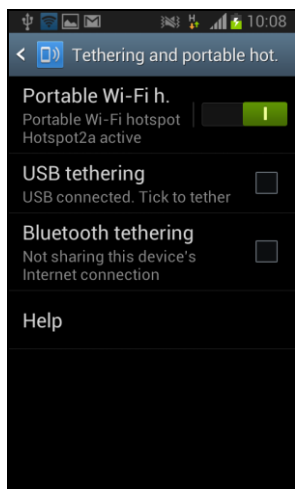


Figure 7. Enabling Wi-Fi hotspot in the emulated DUT

2. Open "AboutConfOnbServer" in the emulated DUT. Nothing should apparently happen when you touch on the icon. This is because the service starts in the background.
3. Open "OnboardingSampleClient" in the other device.
4. Find the recently created network (the one that is being shared by tethering) inside "OnboardingSampleClient", and join it. Then press "Connect to AllJoyn" button.

5. If process was successful you should be able to see the "AJ connect done" message in the Android device.

6.3 Obtaining emulated DUT app and device IDs

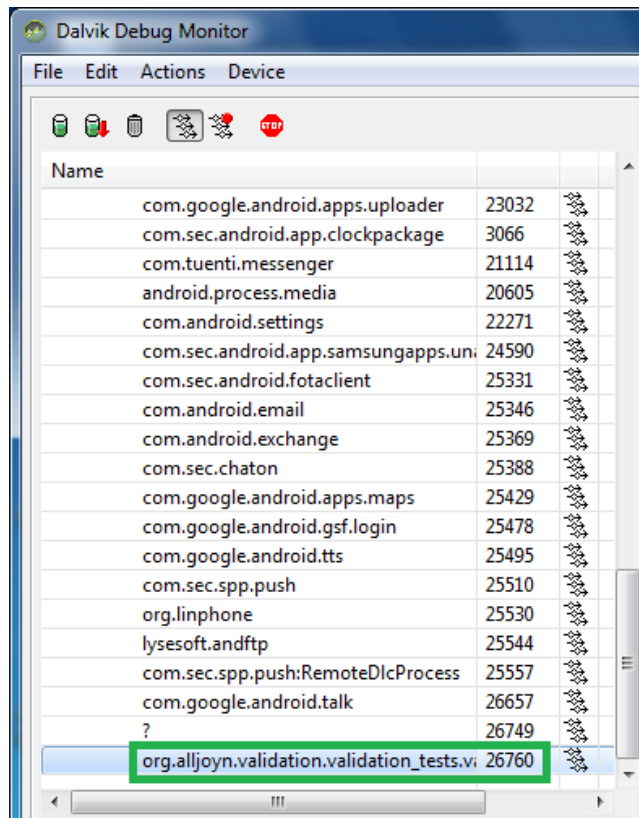
The following will provide you both the "deviceId" and "appId" from the emulated DUT.

Once the emulated DUT has the "AboutConfOnbServer" APK running, the other Android has the Onboarding client running and both are connected to the same Wi-Fi network and have been onboarded, follow the next steps to execute the Self-Certification Tool.

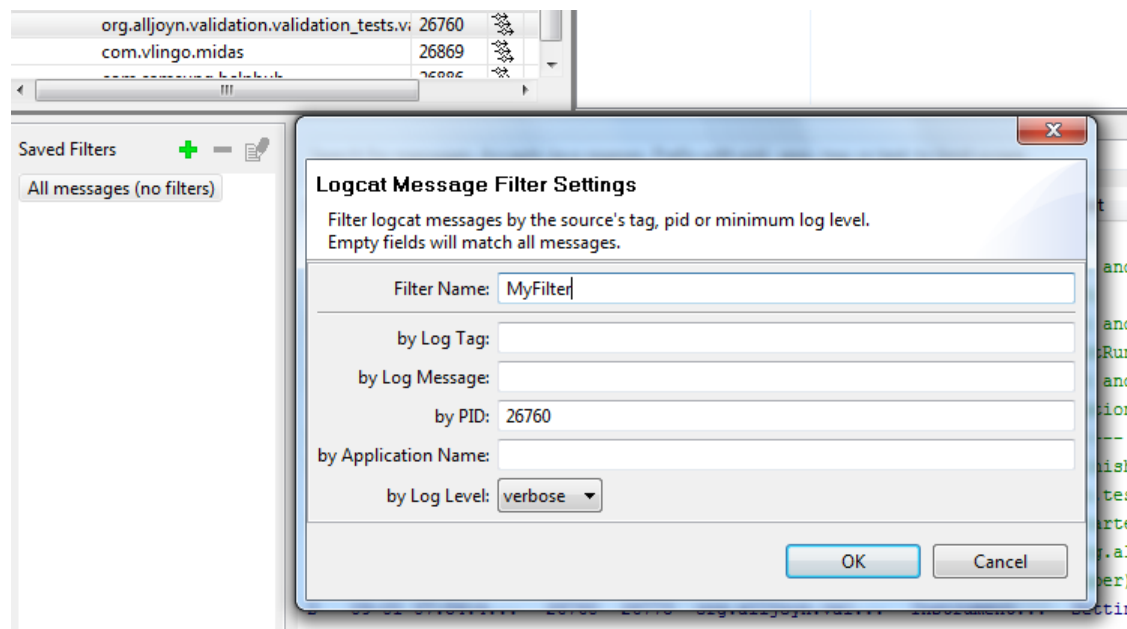
1. Connect the Android device hosting the Self-Certification Tool to your PC or laptop via USB cable
2. Open Dalvik Debug Monitor by clicking on "ddms.bat" executable that is located in: "<androidSDKPath>\adt-bundle-windows-x86_64-20140702\sdk\tools". This is needed in order to be able to examine the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

```
adb shell am instrument -w -e appId ba477de5-f430-4b4c-b3f3-88e8dc98128d
-e deviceId 4564545455454 -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
```

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



- 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):



- 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
```

7. You should expect the test to FAIL this time, but you will be able to determine the real deviceId and appId of your emulated DUT.

NOTE

“actualDeviceId” in the logcat file equals the DeviceId of your device. “actualAppId” in the logcat file equals the AppId of your emulated DUT. Please make note of these two id’s as they must be used for each of the adb commands to run the tests for your emulated DUT.

6.4 Testing the emulated DUT with the Self-Certification Tool

Once you have obtained the correct “deviceId” and “appId”, you can test the communication between the sample application and the Self-Certification Tool.

1. Execute the adb command again but replace “appId” and “deviceId” with your current ones just obtained in the instructions above.

```
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.v
alidation.testing.instrument.ValidationInstrumentationTestRunner
```

2. Look over the PC console and see if you passed the test. You should get the following output:

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

OK (1 test)
```

In case you do not obtain the correct output, follow the instructions given below:

1. Go to application manager in both devices.
2. Force stop "ValidationTestIT" application.
3. Uninstall "AboutConfOnbServer" and "OnboardingSampleClient".
4. Repeat the whole section "6.2 Connecting the emulated DUT and the Self-Certification Tool".
5. You do not need to repeat the steps describe in section "6.3 Obtaining emulated DUT app and device Ids" if you already have the correct "appId" and "deviceId".
6. Then you can follow the steps described in the AllSeen Alliance Self-Certification User Guide to test the emulated DUT with the Self-Certification Tool.

7 Building the Self-Certification Tool (APK)

In order to compile the Self-Certification Tool from the source code repositories follow the instructions provided in this section. This self compiled version of the Self-Certification Tool should only be used for pre-testing or debugging future services and not for official Certification.

To compile the Self-Certification Tool, and deploy it into an Android device of your own, you will need to download AllJoyn libraries and the Self-Certification Tool code.

7.1 Getting the libraries

This section provides detailed instructions on how to download the AllJoyn libraries that are required as a dependency for compiling the Self-Certification Tool. You have to download some precompiled libraries according to the TCCL release you selected (see section 5 in the AllSeen Self-Certification User Guide).

7.1.1 For 14.02 release

1. Download the following file:
<http://mirrors.us.kernel.org/allseenalliance/alljoyn/libs-14.02.zip>
2. Extract it and you should see the "libs" directory.
3. Extract downloaded file content inside recently created "libs" directory.

7.1.2 For 14.06 release

1. Download the following file:
<http://mirrors.us.kernel.org/allseenalliance/alljoyn/libs-14.06.zip>
2. Create a directory called "libs".
3. Extract the file you have just downloaded into "libs" directory.

Once you have completed these steps you have the necessary libraries to build the Self-Certification Tool. You will need to use them in section "7.3 Building the dependencies".

7.2 Downloading AllJoyn Self-Certification Tool source code

At this time, you need to download some source code from the AllSeen Alliance validation repositories with the objective of compiling the Self-Certification Tool. Instructions are provided below:

1. Go to a directory where you want to download the Self-Certification Tool source code.

2. Relying on Git, execute the following command:

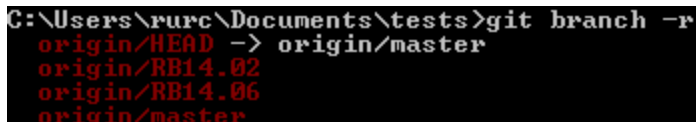
```
git clone https://git.allseenalliance.org/gerrit/compliance/tests
```

3. Now you have the Self-Certification Tool source code. Hereinafter, the "tests" directory is referred to as VAL_DIR.

4. Execute the following git command to list all remote branches:

```
git branch -r
```

You should get an output similar to the one in the picture below.



```
C:\Users\rurc\Documents\tests>git branch -r
origin/HEAD -> origin/master
origin/RB14.02
origin/RB14.06
origin/master
```

5. According to the TCCL release you want to use, you have to switch to the correct branch with the next command:

```
git checkout BRANCH_NAME
```

In the previous command, BRANCH_NAME has to be replaced by the branch of the TCCL release you want to use.

Branch Name	TCCL release
RB14.02	14.02
RB14.06	14.06

7.3 Building the dependencies

Once you have the Self-Certification Tool source code that corresponds to the TCCL release you have decided to use, you will have to move the downloaded libraries (section 7.1) to a specified directory and execute some commands in the PC console with the purpose of building necessary dependencies.

1. Assuming you are in the main directory of your Self-Certification Tool source code, move to the following path: "VAL_DIR\java\components\validation-dependencies\HEAD\libs".
2. Move every file from the "libs" directory you previously created in section "7.1 Getting the libraries" to this "libs" directory where we have just moved in.

Now you have every library in the correct place to build necessary dependencies. Continue with the next instructions that are part of the building process.

3. Move to "VAL_DIR/java/components/validation-base/HEAD"
4. Execute the following maven command:

```
mvn clean install
```

You should get an output similar to this:

```
[INFO] -----  
[INFO] BUILD SUCCESS  
[INFO] -----  
[INFO] Total time: 13.651 s  
[INFO] Finished at: 2014-08-22T14:08:06+02:00  
[INFO] Final Memory: 11M/95M  
[INFO] -----
```

5. Move to "VAL_DIR/java/components/validation-dependencies/HEAD"

6. Execute the following maven command:

```
mvn clean install
```

Again, the output should give you BUILD SUCCESS.

7. Move to "VAL_DIR/java/components/validation-framework/HEAD"

8. Execute the following maven command:

```
mvn clean install
```

Expect BUILD SUCCESS this time too.

9. Move to "VAL_DIR/java/components/validation-tests/HEAD"

10. Execute the following maven command. Please pay attention to this command because this time it has two extra parameters. Note that SDK_PATH must be replaced by your Android SDK main directory. e.g.: "C:\Users\rurc\Desarrollo\adt-bundle-windows-x86_64-20140702\sdk":

```
mvn clean install -Dmaven.test.skip=true -Dandroid.sdk.path=SDK_PATH
```

Again you should read BUILD SUCCESS in the console.

7.4 Installing the Self-Certification Tool APK in the Android device that will become the Self-Certification Tool

Assuming you are in "VAL_DIR/java/components/validation-tests/HEAD/validation-test-it" directory, use this maven command to copy the Self-Certification Tool to the device.

```
mvn android:deploy
```

If you obtained BUILD SUCCESS you should now have the Self-Certification Tool APK successfully installed in your Android device.

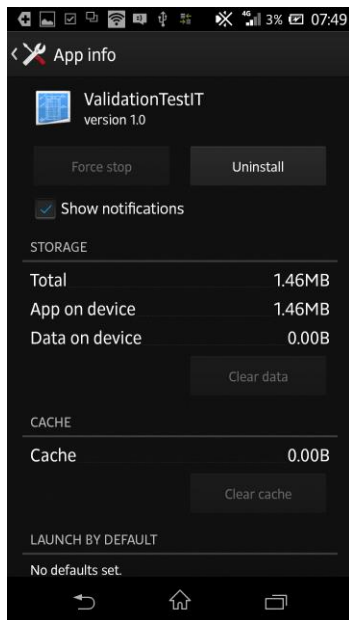


Figure 8. Self-Certification Tool APK information view

Appendix A Hints to troubleshoot adb commands

A.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.

A.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>