AllJoyn™ Android Environment Setup Guide

HT80-BA001-2 Rev. D

October 2012

Submit technical questions at:

http://www.alljoyn.org/forums

Qualcomm Innovation Center, Inc.
5775 Morehouse Drive
San Diego, CA 92121
U.S.A.

Contents

Tables

1 Introduction

1.1 Scope

This document is a guide to using AllJoyn-enabled applications on Android devices. It covers the following topics:

- Setting up the programming environment
- Obtaining and building AllJoyn for Android from source
- Modifying an Android device to support AllJoyn-enabled applications

1.2 Audience

This document provides important information for application developers who want to use AllJoyn on an Android-enabled device.

1.3 Prerequisites

- This document uses many terms and concepts that are described in the document *Introduction to AllJoyn*. Therefore, it is strongly recommended that you read that document before beginning this document.
- Before proceeding with development as described in this document, make sure that you have set up the development environment as described in the documents *Setting Up the Build Environment (Microsoft Windows Platform)* or *Setting Up the Build Environment (Linux Platform)*.

1.4 Revision history

The table below provides the revision history for this document.

Table 1: Revision history

Version	Date	Description
A	June 2011	Initial release
B	November 2011	Formatting changes and insertion of chapter named "Building AllJoyn from Source for Android"
C	September 2012	Updated sample commands in section "Building AllJoyn for Android" for release 2.6.
D	October 2012	Updated sample commands in section "Building AllJoyn for Android" for release 3.0.

1.5 Related documents

The table below is a list of supplemental documents and downloads that relate to the information in this guide.

Table 2: Related documents

Location	Title
http://www.alljoyn.org	Introduction to AllJoyn
http://www.alljoyn.org	Guide to AllJoyn Development Using the Java SDK
http://www.alljoyn.org	C++ API Reference Manual
http://www.alljoyn.org	Java API Reference Manual
http://www.alljoyn.org	AllJoyn SDK
http://www.alljoyn.org	Source Code
http://www.alljoyn.org	Configuring the Build Environment (Microsoft Windows Platform)
http://www.alljoyn.org	Configuring the Build Environment (Linux Platform)

1.6 Acronyms and abbreviations

ADT	Android Development Tools
API	Application Programming Interface
APK	Android Package
BT	Bluetooth
IP	Internet Protocol
NDK	Native Developers Kit
OEM	Original Equipment Manufacturer
SDK	Software Developers Kit

2 Setting Up the Programming Environment

This chapter explains how to set up the programming environment for developing AllJoyn-enabled Android applications. It covers the following topics:

- Install the Android SDK and NDK
- Install Eclipse and the ADT plug-in

Note The procedures described in this chapter require the specified tool versions.

2.1 Installing the Android SDK

The Android software development kit (SDK) provides the tools needed for building Android applications and transferring applications to or from an Android device.The ‘adb’ tool is used to:

- Transfer/pull files to/from the phone
- Run the AllJoyn daemon
- Install/uninstall applications

If you want to run AllJoyn on Android 2.2 (Froyo), you can install Android SDK version r9 or above.

If you want to run AllJoyn on Android 2.3 (Gingerbread), you can install at least Android SDK version r11 or above.

Download Android SDK version r11 or above from the following location:

http://developer.android.com/sdk/index.html

Install the SDK by following the directions given on the download page.

In order to run, the SDK requires certain software packages to be pre-installed on your system. For more information see the following location:

http://developer.android.com/sdk/requirements.html

After installing the SDK, you must install the Android platform support packages you wish to use. See:

http://developer.android.com/sdk/installing.html#AddingComponents

AllJoyn runs on any SDK platform with Android API levels 8 through 11. Note that installing these packages may take some time.

2.2 Installing the Android NDK

The Android native development kit (NDK) enables developers to build Java native libraries (JNI libraries) which can be called from Android (Java) applications. Android NDK is required

only to write Java native libraries. The Android NDK is not required to use the Android Java bindings.

The main tool used from the Android NDK is ‘ndk-build’, which is used to build the native library of the JNI application.

To run Android JNI applications on Android 2.3 (Gingerbread), you need to install Android NDK version r5b or above.

Download Android NDK version r5b or above from the following location:

http://developer.android.com/sdk/ndk/index.html

Install the NDK by following the directions given on the download page.

To run, the NDK requires that the following software packages are pre-installed on your system:

- Latest Android SDK (including all dependencies)
- GNU Make 3.81 or later
- Recent version of awk (GNU awk or nawk)

For more information, see the NDK download page.

2.3 Installing Eclipse and the ADT plug-in

The Android SDK operates in the Eclipse integrated development environment, with the addition of the Eclipse plug-in for the Android development tools (ADT).

Since Android applications are Java-based, installing Eclipse for Java development may be helpful.

Download Eclipse from the following location:

http://eclipse.org

Install Eclipse by following the directions given on the download page.

Instructions for installing the Eclipse ADT plug-in can be found at the following location:

http://developer.android.com/guide/developing/projects/projects-eclipse.html

Note If the plug-in can’t find the SDK executable, it displays an error and then prompts for its location. (Point it to where you installed the SDK.) Also, if you haven't already selected the packages that need installing (refer to *Installing the Android SDK*), you are prompted to do so upon launching ADT.

2.4 Downloading the OpenSSL header files and library

AllJoyn uses the OpenSSL crypto library for end-to-end encryption and authentication.

The prebuilt library is required to link AllJoyn applications. It can be downloaded directly from the Android device or emulator into the `lib` folder of the AllJoyn distribution. Attach the device (or launch the Android emulator), then run the following commands:

```
cd <alljoyn_dir>/lib
adb pull /system/lib/libcrypto.so libcrypto.so
```

The above command means:

adb pull <location of the file on the phone that you want to pull> <destination on your machine where you want to store the pulled file with the name that you want>

The library can also be built from the Android source repository. For details on building the Android source tree, see the Android source repository web site:

http://source.android.com/source/download.html

Important If you are building for Froyo, the `libcrypto.so` library must be pulled from Froyo, not Gingerbread. Conversely, if you are building for Gingerbread, the `libcrypto.so` library must be pulled from Gingerbread.

3 Building AllJoyn from Source for Android

For most developers, the SDK package available to download from www.alljoyn.org is sufficient for developing Android applications using AllJoyn. However, if you wish to obtain and compile AllJoyn from source, follow the directions in this chapter.

To compile AllJoyn from source, the following items are required:

- Android SDK
- Android NDK
- Eclipse and the ADT plug-in
- Android source

Instructions for obtaining the Android SDK, Android NDK, Eclipse and the ADT plug-in are in *Setting Up the Programming Environment*.

3.1 The Android source

The Android source (*http://source.android.com*) is required for building Android targets. For most developers, downloading and building Android source is the most complicated step in building AllJoyn for Android. Google has detailed instructions for downloading and building Android source.

For a list of system requirements and instructions for obtaining the required tools, see *http://source.android.com/source/initializing.html*

For instructions on obtaining the Android Source Tree, see *http://source.android.com/source/downloading.html*. When running the `repo init` command specify:

- `-b froyo-release` for Froyo source
- `-b gingerbread-release` for Gingerbread source

For instructions on building and running the build source, see *http://source.android.com/source/building.html*

- Build the "generic" version of Android.
- There is no need to run the code. Only the build libraries that are not available in the NDK are used.

3.2 Obtaining the AllJoyn source

If you followed the instructions in *The Android source*, you should have the repo tool and git installed on your system. Enter the following commands to get the AllJoyn source:

```
$ mkdir $HOME/alljoyn # for example
$ cd $HOME/alljoyn
$ repo init -u git://github.com/alljoyn/manifest.git
$ repo sync
$ repo start master --all
```

3.3 Building AllJoyn for Android

At this point you have all of the files and programs required to build AllJoyn for Android.

The following commands assume you have installed the Android NDK at `/usr/local/android-ndk-r5b`, you have downloaded and built the Android source, and it is located in `$HOME/android-platform`.

Important If you are building for Froyo, the Android source must be built for Froyo, not Gingerbread. Conversely, if you are building for Gingerbread, the Android source must be built for Gingerbread.

Use the following commands to build AllJoyn for Android:

```
$ export JAVA_HOME="/usr/lib/jvm/java-6-sun" # or java-5-sun
$ export CLASSPATH="/usr/share/java/junit.jar"
$ scons OS=android CPU=arm ANDROID_NDK=/usr/local/android-ndk-r5b
                  ANDROID_SRC=$HOME/android-platform WS=off
```

It is possible to specify that AllJoyn use additional tools during the build process. For example, AllJoyn can use Uncrustify to check white space compliance and Doxygen for producing API documentation for the C++ APIs. See the document, *Configuring the Build Environment (LINUX Platform)*, for detailed instructions for installing these two tools.

4 Device Configuration

This chapter explains how to modify an Android mobile device to support AllJoyn-enabled applications.

The AllJoyn software architecture consists of two parts:

- Client library which provides APIs to applications
- AllJoyn daemon that implements routing between applications and devices, registration of services, discovery of devices, and services

Applications communicate with the AllJoyn daemon, and it is the daemon that handles all external communications over the available interfaces (currently Bluetooth and Wi-Fi). The AllJoyn daemon is designed to work with the ‘standard’ Wi-Fi and Bluetooth (BT) interfaces that are supported on Linux.

On the Android platform, there are three options for implementing the AllJoyn daemon: as an Android service or as one of two forms of daemons mentioned below. All of these approaches are supported, but each has associated benefits and limitations. Note that applications are unaware of the implementation of the AllJoyn daemon, and work the same way, regardless of implementation since these applications are written using the client library.

The three options are listed below, starting with the easiest option:

- Installing AllJoyn as an Android service
- Installing the AllJoyn Linux daemon with Wi-Fi-only support
- Installing the AllJoyn Linux daemon with Wi-Fi and BT support

Additional installation, file building, and testing information and instructions for each option is provided in the following sections.

4.1 Installing AllJoyn as an Android service

This approach provides the most flexibility in choosing which Android device to use. The background process can be installed on any Android device running Froyo (version 2.2) or a newer version of Android, and does not requiring any knowledge of Linux or the Android shell.

In this approach, the AllJoyn background process is implemented as an Android service, and can be installed and executed, just like any other Android application.

There are limitations to this approach, however. Access to BT in the Android environment is restricted to a defined set of BT profiles (e.g., OBEX) and requires pairing of devices before interaction is possible. Since AllJoyn requires access to lower-level BT APIs, BT is not supported if the AllJoyn background process is implemented as an Android service. Such an application can use only the Wi-Fi radio.

When run as an Android service, the Alljoyn background process is subject to the service shutdown policies of the Android system.

4.2 Installing the AllJoyn daemon with Wi-Fi-only support

As the name implies, and similar to running as an AllJoyn Android service, the application can only use the Wi-Fi radio with this option.

The benefits of this approach are that the daemon can be automatically started at device boot time, and does not require the user to install or activate the daemon.

The disadvantage of this approach is that the daemon either needs to be installed by the device original equipment manufacturer (OEM), which is not available today, or installed manually by the user and started via a command shell.

A developer phone (i.e., a privileged-access device) is required for this option. This is a fairly straightforward process, and most developers should be comfortable with this.

4.3 Installing the AllJoyn daemon with Wi-Fi and Bluetooth support

This is the most complicated of the three options. This process requires a developer phone with a custom built image for the particular platform.

Attention Rooting or hacking a device to achieve privileged access may void any manufacturer warranty, damage a device, or both. If you elect to root or hack your device, then you do so at your own risk and you hereby waive any liability and claims against Qualcomm Innovation Center, Inc. and its affiliates.

This approach is not recommended for most developers. As noted earlier, applications are unaware of the implementation of the background process, and work the same way regardless of implementation. The AllJoyn APIs will not change, and we highly recommend pursuing either of the options described in *Installing AllJoyn as an Android service* or *Installing the AllJoyn daemon with Wi-Fi-only support* until AllJoyn becomes more widely adopted by OEMs.

BT stack modifications have been built and verified on the HTC Google Nexus One. Instructions are provided below to use those modifications. Stack modifications for any other devices are not provided.

In some cases, the underlying Linux OS is modified by OEMs and can contain differences relative to the Android baseline as delivered by Google. These differences can include custom BT stacks and Wi-Fi Internet Protocol (IP) filtering algorithms that can affect compatibility with AllJoyn.

While support for multiple radio technologies is a key feature of AllJoyn, it is not necessary to use BT during application development.

As BT is integrated as part of the device platform by OEMs, an application works exactly the same as it did when developed using the Wi-Fi-only options.

To validate BT functionality is not a simple process, it requires pulling and installing patches from the BlueZ open source project and dealing with specific issues that may arise for the platform being used. See *Installing on an unmodified device (Android service)*, *Installing on a privileged access device (Linux daemon)*, and *Verifying that the AllJoyn daemon is running* with specific instructions on how to install each option.

Note As Android releases incorporate more recent BlueZ patches, these steps may become unnecessary.

4.4 Installing on an unmodified device (Android service)

The AllJoyn SDK contains an APK file that is used to implement this feature:

- `AllJoyn.apk`: This is the service that implements the AllJoyn background process.

To install this package, connect the device to a computer and use the following commands:

```
$ cd <alljoyn_dir>/alljoyn_android/alljoyn/bin
$ adb install -r AllJoyn.apk
```

Note Drop the “-r” option if this is the first installation.

Note When determining which <alljoyn_dir> to use, note that the AllJoyn SDK contains both the debug version and the release version.

- For the debug build, <alljoyn_dir> = alljoyn_2_X_X_dbg
- For the release build, <alljoyn_dir> = alljoyn_2_X_X_rel

To start the service, go to the list of Android applications and run the “AllJoyn” application, which launches the AllJoyn service.

AllJoyn-enabled applications execute using the AllJoyn service, but can only use Wi-Fi as a communication mechanism between devices.

The AllJoyn icon will show in the list of services running at the top of the screen, as shown in *Figure 1*.

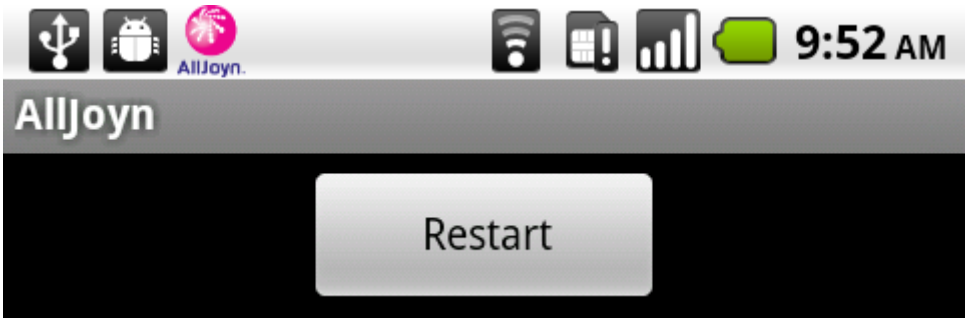

Figure 1: Indication that Alljoyn service is running

The AllJoyn service provides a single interface button named **Restart**. Pressing **Restart** causes the service to reset itself.

Important Restarting the daemon affects all applications using AllJoyn. Restarting the AllJoyn service means that all applications that use AllJoyn would also require a restart.

4.5 Installing on a privileged access device (Linux daemon)

1. Push alljoyn-daemon to the phone using these commands:

```
$ cd <alljoyn_dir>/bin
$ adb shell mkdir /data/alljoyn
$ adb push alljoyn-daemon /data/alljoyn/alljoyn-daemon
```

2. Start the daemon by typing the following command while the device is connected to a computer:

```
$ adb shell /data/alljoyn/alljoyn-daemon --internal --no-bt --no-switch-user
```

Note The daemon must be manually restarted every time the device is rebooted.

To get the daemon to continue running when the adb session has exited, start it in the background. To do this, run the following commands (instead of the command shown above):

```
$ adb shell
$ su
# cd /data/alljoyn
# ./alljoyn-daemon --internal --no-bt --no-switch-user &
```

Note To get the daemon to start automatically every time the device is booted, an entry to the device's `/rc.init` file must be added. This can only be done by modifying and re-flashing the device's boot image. This procedure is beyond the scope of this document.

4.6 Verifying that the AllJoyn daemon is running

The quickest way to find out if the AllJoyn daemon is running is to launch one of the samples from the AllJoyn SDK, similar to the following example:

```
$ cd <alljoyn_dir>/java/samples/simple/client/bin
$ adb install -r SimpleClient.apk
$ cd <alljoyn_dir>/java/samples/simple/service/bin
$ adb install -r SimpleService.apk
```

1. On the phone, launch the `SimpleService` application and then the `SimpleClient` application. The client does not run unless the service is running first.
2. In the `SimpleClient` application, type any string into the text box and press **Enter**. If the string entered is echoed back to the application, the AllJoyn daemon is running as expected.
3. Test out any of the other sample applications found in the java/samples directory. Try running the service on one device and the client on another.