

Programming Guide

RFID Android SDK

unitech electronics Documentation
Issue2, revision 8
March 2021

Revision History

Release	Revision	Date	Changes
2	0	2020-09-25	Update Preface, Pogo Control and MercuryAPI
2	1	2020-09-29	Update Preface, initial Setup and MercuryAPI
2	2	2020-09-30	Update Preface and initial Setup
2	3	2020-10-15	Add Import/Export Key Setting for rfidlib.jar V1.2.4
2	4	2020-10-19	Modify for rfidlib.jar V1.2.5 <ul style="list-style-type: none"> - Remove import/export Key Setting functions - Add getKeyRemapStatus, setKeyRemapStatus, getTriggerKeyCode and setTriggerKeyCode
2	5	2020-11-04	Add support list Modify Chapter 4
2	6	2020-12-19	Revised Preface to add more info about package directory.
2	7	2021-03-12	Modify for rfidlib.jar V1.3.6 to support EA630 A10 <ul style="list-style-type: none"> - Add changeTriggerKeyMode, getRFIDTriggerKeyCode, getScannerTriggerKeyCode
2	8	2021-03-15	Modify the description about supported DMSerersion version

Table of Contents

1.	PREFACE	1
1.1	PREREQUISITES.....	3
2.	INITIAL SETUP	4
2.1	INITIALIZE POGOCTRL.....	4
2.2	TURN ON POWER.....	4
2.3	TURN OFF POWER.....	4
2.4	GET UART PATH	5
3.	TRIGGER KEY SETUP	5
3.1	INITIALIZE KEYSSETTING.....	5
3.2	GET KEY REMAP STATUS.....	6
3.3	SET KEY REMAP STATUS.....	6
3.4	GET TRIGGER KEY CODE.....	6
3.5	SET TRIGGER KEY CODE.....	7
3.6	CHANGE TRIGGER KEY MODE.....	7
3.7	RFID DEFAULT KEY CODE.....	8
3.8	SCANNER DEFAULT KEY CODE.....	8
4.	MERCURY API.....	8

1. Preface

This document describes the functionalities of the RFID Android SDK. The SDK is delivered through the JAVA-based library “rfidlib.jar” for user with the supported portable Android RFID readers from unitech. Those supported RFID readers use the ThingMagic RFID modules and therefore the majority of the library is provided on the basis of the ThingMagic Mercury APIs.

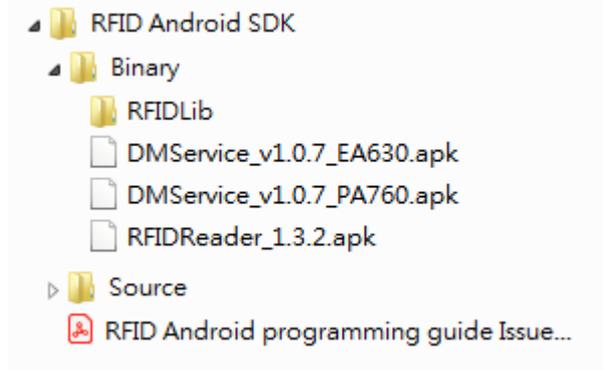
The “rfidlib.jar” library is consisted of three key functional areas:

- **The PogoCtrl API** – the unitech portable RFID readers use serial connections to communicate with the Android host device. The PogoCtrl API manages the pogo pin power on/off between the reader and the host devices. Turning the power to ON and getting connected are the first steps in getting the unitech RFID reader peripheral connected to your applications.
- **The KeySetting API** – allows you to configure the behavior of the button on the portable RFID reader peripherals. Configure the keys for trigger an RFID inventory session could be the second step in getting your applications to work with the RFID hardware.
- **The ThingMagic Mercury API-based function calls** – for simple and rapid setup and control of the supported unitech portal RFID readers. Developers can use these APIs for the read functions; tag operations; privacy and security; and the parameters for performance optimization (Please refer to the Mercury API Programming Guide for detailed information).

The key work flow in getting the unitech portal RFID reader connected with your application consists of the following key steps:

- **Initialization Phase**
Initialize PogoCtrl and then use the PogoCtrl API functions to:
 - turn the pogo pin power to on (Power has to be on in order to make serial connections)
 - get the UART path (UART path is the serial connection between reader and host, the UART path information will be used when you make a call to the Mercury API-based functions)
- **Key Setup Phase**
Use the KeySetting APIs to configure Trigger Key Remapping.
To use the trigger key to control the RFID reader, the trigger key needs to be reprogrammed to specific value as key code, and then the application (e.g. RFIDReader or customer’s application) would recognize the key code value and drive specific function whose value is match.
- **Operational Phase**
Use the Mercury-based APIs to setup and control the ThingMagic-based RFID modules on unitech portable RFID peripheral.

The figure below shows the structure of SDK. The *Binary* folder includes the library and one demo application, and the *Source* folder contains the source codes of demo application.



Folders & Files	Description
/RFID Android SDK	The SDK package
/RFID Android SDK/Binary/	The Binary folder contains the RFID Library (JAR) and the dependency apps & demo app.
/RFID Android SDK/Binary/RFIDLib/	The RFIDLib folder contains the rfidlib.jar itself, as well as the documentations.
/RFID Android SDK/Binary/RFIDLib/doc/	The doc folder inside RFIDLib contains the JAVADOC of the rfidlib.jar library file. Please load index.html into your browsers to view
/RFID Android SDK/Binary/RFIDLib/MercuryAPI_ProgrammerGuide_for_v1.27.3.pdf	The Mercury API documentation for use when you write an application that uses the Mercury API functions.
/RFID Android SDK/Binary/RFIDLib/rfidlib_V1.2.1.jar	The rfidlib.jar library file. Include this library in your Android Studio project to access the API functions.
/RFID Android SDK/Binary/DMService_v1.0.7_EA630.apk	The dependency app DMService required for EA630. Please make sure this APK is installed on the host device before using the rfidlib.jar
/RFID Android SDK/Binary/DMService_v1.0.7_PA760.apk	The dependency app DMService required for PA760. Please make sure this APK is installed on the host device before using the rfidlib.jar
/RFID Android SDK/Binary/RFIDReader_1.3.2.apk	The demo application that demonstrate the use of the rfidlib.jar, including the PogoCtrl,

	KeySetting, and Mercury APIs. The source code of this application is also included in this package for studying.
/RFID Android SDK/Source/	The source code of the RFIDReader application.
/RFID Android SDK/RFID Android programming guide issue2 rev6.pdf	This document.

Note: The library “rfidlib.jar” needs DMSERVICE V1.0.7 or later version to support. For EA630 A10, it needs V1.1.8 or later version to support. You could get it in the Binary folder.

Note: The PogoCtrl and KeySetting should be used in non-main thread.

1.1 Prerequisites

The SDK supports the following RFID Reader peripherals:

- RG760 attached to a PA760 with Android 9
- RG760 attached to a PA760 with Android 10
- RG630 attached to an EA630 with Android 9
- EA630 attached to an EA630 with Android 10

Note about the Dependency App “DMSERVICE”

The DMSERVICE application is one of the unitech helper applications that provide the functions of the UnitechSDK. The “rfidlib.jar” RFID Android SDK uses the UnitechSDK for some of its functionalities such as Key Settings. It is recommended that your application can take care of the installation of this DMSERVICE APK if the host device does not already have it installed, or if the minimum version required are not correct.

The RFID Android SDK requires DMSERVICE V1.0.7 or later (inclusive) to be installed on the supported host Android devices. On EA630 A10, the SDK requires DMSERVICE V1.1.8 or later version. The DMSERVICE APK files for the supported models can be found from the Binary folder inside the SDK package.

Note about Multithreaded Applications

The PogoCtrl and KeySetting should be used in non-main thread.

2. Initial Setup

Before communicating with the RFID reader, we need to turn on the RFID module. After using the RFID reader, we also suggest turning RFID module off to save power consumption. Those functions could be found in PogoCtrl. Please import `com.unitech.api.pogo.PogoCtrl` and we list the functions provided in PogoCtrl below.

2.1 Initialize PogoCtrl

Function Description:

Initialize PogoCtrl.

Function call:

`PogoCtrl getInstance (Context context)`

Parameter:

Context: The application's context

Return:

The initialized PogoCtrl.

2.2 Turn on power

Function Description:

Set the power pin to high.

Function call:

`Bundle powerOn ()`

Return:

Key: `errorCode`, Type: `Int`

Key: `errorMsg`, Type: `String`

2.3 Turn off power

Function Description:

Set the power pin to low.

Function call:

Bundle powerOff ()

Return:

Key: errorCode, Type: Int

Key: errorMsg, Type: String

2.4 Get UART path

Function Description:

Get the UART path of RFID module.

Function call:

Bundle getUartPath()

Return:

Key: UART, Type: String

3. Trigger Key Setup

To use the trigger key to control the RFID reader, we need to remap the trigger key to RFID key code, so it will trigger RFID function instead of barcode scanner.

The key remap functions could be found in KeySetting. Please import `com.unitech.api.key.KeySetting` and we list the functions provided in KeySetting below.

3.1 Initialize KeySetting

Function Description:

Initialize KeySetting and save current setting to file.

Function call:

KeySetting getInstance (Context context)

Parameter:

context: The application's context

Return:

The initialized KeySetting.

3.2 Get key remap status

Function Description:

Get the setting of key remap status from file which created by getInstance.

Function call:

Bundle getKeyRemapStatus ()

Return:

Key: errorCode, Type: Int

Key: errorMsg, Type: String

Key: keyRemapping, Type: boolean

3.3 Set key remap status

Function Description:

Set key remap status setting to the file which created by getInstance then import it to device. Please be noted that when the status is not enabled, the default trigger key function is barcdoe scanner.

Function call:

Bundle setKeyRemapStatus (Boolean enable)

Parameter:

enable: Enable/Disable the key remap setting

Return:

Key: errorCode, Type: Int

Key: errorMsg, Type: String

3.4 Get trigger key code

Function Description:

Get the trigger key code from file which created by getInstance.

Function call:

Bundle getTriggerKeyCode ()

Parameter:

enableSetting: Enable/Disable the key code setting

triggerKeyCode: The key code of trigger key

Return:

Key: errorCode, Type: Int

Key: errorMsg, Type: String

Key: keyCode, Type: String

3.5 Set trigger key code

Function Description:

Set trigger key code to the file which created by getInstance then import it to device.

Function call:

Bundle setTriggerKeyCode (String keyCode)

Parameter:

keyCode: The key code for trigger key.

Return:

Key: errorCode, Type: Int

Key: errorMsg, Type: String

3.6 Change trigger key mode

Function Description:

Change trigger key's key code to RFID or Scanner's default value to the file which created by getInstance then import it to device. The default key code, please refer to

Function call:

Bundle changeTriggerKeyMode (boolean toRFID)

Parameter:

toRFID: The trigger key mode, true for RFID and false for Scanner.

Return:

Key: errorCode, Type: Int

Key: errorMsg, Type: String

3.7 RFID default key code

Function Description:

Get the default trigger key code for RFID.

Function call:

Bundle getRFIDTriggerKeyCode ()

Return:

Key: errorCode, Type: Int

Key: errorMsg, Type: String

Key: keyCode, Type: String

3.8 Scanner default key code

Function Description:

Get the default trigger key code for Scanner.

Function call:

Bundle getScannerTriggerKeyCode ()

Return:

Key: errorCode, Type: Int

Key: errorMsg, Type: String

Key: keyCode, Type: String

4. Mercury API

Customized Mercury API is revised from Mercury API to support serial communication. Therefore, please specify the scheme as “ute” and Factory object as “new SerialTransportRS232.Factory()” while setting serial port of the reader. You could get the URI from [Get UART path](#) of PogoCtrl while making the connection. Below is the sample code, and you can also find the details in the ConnectionListener.java and ReaderConnect.java in the Source folder.

```
Pogoctrl pogoCtrl = PogoCtrl.getInstance(null);
```

unitech electronics co. ltd. Copyright @ 2019

```
Bundle path = pogoCtrl.getUartPath();  
String uniString = path.getString("UART");
```

```
private static Reader reader;  
if (reader == null) {  
    Reader.setSerialTransport("ute", new SerialTransportRS232.Factory());  
    reader = Reader.create(uniString);  
}  
reader.connect();
```

All other RFID related functions are kept the same. Please refer the official Mercury API programming guide and Java doc under the folder Binary/RFIDLib.

Note: The rfidlib.jar uses ltkjava-1.0.0.6.jar, please import it with rfidlib.jar