

OMNIKEY® Android Driver

User Guide

Software Version: 2.3
PLT-07678, A.3
November 2025

Copyright

© 2025 HID Global Corporation/ASSA ABLOY AB. All rights reserved.

This document may not be reproduced, disseminated, or republished in any form without the prior written permission of HID Global Corporation.

Trademarks

HID GLOBAL, HID, the HID Brick logo, and OMNIKEY are trademarks or registered trademarks of HID Global, ASSA ABLOY AB, or its affiliate(s) in the US and other countries and may not be used without permission. All other trademarks, service marks, and product or service names are trademarks or registered trademarks of their respective owners.

Contacts

For technical support, please visit: <https://support.hidglobal.com>.

What's new

Date	Description	Revision
November 2025	Updated for software version 2.3.	A.3

A complete list of revisions is available in [Revision history](#).

Introduction	4
1.1 Overview	5
1.1.1 Supported smart card readers	5
1.1.2 Minimum system requirements	5
1.1.3 Supported devices for USB smart card readers	5
1.1.4 Added features	5
1.1.5 Definitions, abbreviations and symbols	6
1.1.6 References	6
Application development	7
2.1 Introduction	8
2.2 Integrating the driver in a custom application	8
2.2.1 Linking the HID Android Driver AAR library file to the project	8
2.2.2 Permanent permission to access USB Smart Card Readers	8
2.3 Opening OMNIKEY Demo in Android Studio	9
2.3.1 How to build the OMNIKEY Demo sample app from sources	9
Using the built OMNIKEY Demo app (.apk)	10
3.1 Introduction	11
3.1.1 OMNIKEY Demo application installation and usage	12
Using escape commands	18
4.1 Overview	19
4.1.1 Using escape commands when a card is present	19
4.1.2 Using escape commands when a card is absent	20
Secure session	21
5.1 Overview	22

Section 01

Introduction

1.1 Overview

The OMNIKEY® Android driver v2 package brings support for OMNIKEY smart card readers to the Android Operating System (OS). The HID Android Driver AAR library provides access to smart cards via the well-known JSR 268 API.

This document describes the usage of smart cards and readers under Android OS, using the HID Android Driver library and OMNIKEY Demo sample app, and smart card access via the JSR 268 Smart Card IO API (JSR 268 API).

The HID Android Driver Library now supports Android API level 35 (Android 15) while maintaining backward compatibility with minimum SDK 19 (Android 4.4).

1.1.1 Supported smart card readers

The following smart card readers are supported:

- smart@link chipset.
- OMNIKEY USB Readers: 1021, 3021, 3121, 3621, 3821, 5427, 5427 Gen2, 5022, 5023, 5025, 5122, 5122 Dual, 5127 Mini, 5422, 6121.

1.1.2 Minimum system requirements

- Android device (tablet or smart phone) with On The Go (OTG) support.
- Android OS version 4.4 or higher.
- API support for android.hardware.usb.

The apps can be installed without special privileges, but you may have to grant access to USB devices during operation, if this is not already set.

1.1.3 Supported devices for USB smart card readers

Android tablets and smart phones which support the USB host (OTG) mode are supported. However, not all devices may offer a USB port to fit the USB Type A plug of the smart card readers. Please check what kind of adapter cables are needed to properly connect the reader to the device. Due to inconsistencies between device manufacturers, adapters from micro-USB to USB are not always compatible between different vendors.

1.1.4 Added features

Added public function for setting log level. A user can use it from application as described below:

setLogLevel(Log.INFO)

- Available log levels are: ASSERT, DEBUG, ERROR, INFO, VERBOSE and WARN.

1.1.5 Definitions, abbreviations and symbols

ARR	Android Archive
APDU	Application Protocol Data Unit
API	Application Programming Interface
APK	Android Application Package
OID	Object Identifier
PC/SC	Personal Computer/Smart Card
IFD	Interface Device (for accessing ICC card)
OTG	On The Go, USB interface feature
OS	Operating System
SDK	Software Development Kit
UID	Unique Identifier

1.1.6 References

ISO 7816-3	ISO 7816-3 Identification cards - Integrated circuit cards - Part 3: Cards with contacts - Electrical interface and transmission protocols Third edition, 2006-11-01
ISO 7816-4	ISO 7816-4 Identification cards - Integrated circuit cards - Part 4: Organization, security and commands for Interchange Second edition, 2005-01-15
PCSC-3	Interoperability Specification for ICCs and Personal Computer Systems Part 3. Requirements for PC-Connected Interface Devices Revision 2.01.09, June 2007
CCID	Specification for Integrated Circuit(s) Cards Interface Devices Revision 1.1, 22 April 2005
JSR 268 Doc	http://docs.oracle.com/javase/6/docs/jre/

Section 02

Application development

2.1 Introduction

Access to readers and inserted smart cards is provided from the management tool to user apps via the public JSR 268 API. This API provides several classes and methods for reader listing, card presence detection, and data transmission. For further information, refer to: <http://download.oracle.com/otndocs/jcp/jscio-4.0-fr-eval-oth-JSpec/>.

2.2 Integrating the driver in a custom application

2.2.1 Linking the HID Android Driver AAR library file to the project

To use the HID Android Driver library it must be included in a project as a dependency. Place the **HID_Android_Driver-release.aar** library file in the libs folder in the root directory of your project, then use the following code in the **build.gradle** file of your project:

```
dependencies {  
    implementation files('libs/HID_Android_Driver-release.aar')  
    /* (...) other dependencies */  
}
```

2.2.2 Permanent permission to access USB Smart Card Readers

It is necessary to add an intent-filter and meta-data to any activity in AndroidManifest.xml. This allows you to grant permanent permission to the application to access a USB Smart Card Reader. The intent-filter and meta-data placed in AndroidManifest.xml in a project compiled to the AAR library file are later ignored, so they must be explicitly added to the application.

Example of intent-filter and meta-data added to OmniKeyDemoActivity

```
<activity  
    android:name="com.hidglobal.omnikeydemo.OmniKeyDemoActivity"  
    android:label="@string/title_activity_OMNIKEYDemo"  
    android:screenOrientation="portrait"  
    android:theme="@style/OmniKeyDemoApp"  
    android:exported="true" >  
    <!-- Intent-filter is required to allow user to grant permanent permission to the USB device  
         to the application. Without it any granted permission is temporary only. It can't be  
         applied to AAR library file. To enable permanent permission to the device,  
         intent-filter must be added to activity in the final application. Intent-filter  
         triggers application opening after device attachment too. -->  
    <intent-filter>  
        <action android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED" />  
    </intent-filter>  
    <!-- Meta-data is the second required element to allow user to grant permanent permission  
         to the USB device to the application. List of all device filters is stored  
         in separate XML file. The file location is res/xml/device_filter.xml. -->  
    <meta-data  
        android:name="android.hardware.usb.action.USB_DEVICE_ATTACHED"  
        android:resource="@xml/device_filter" />  
</activity>
```

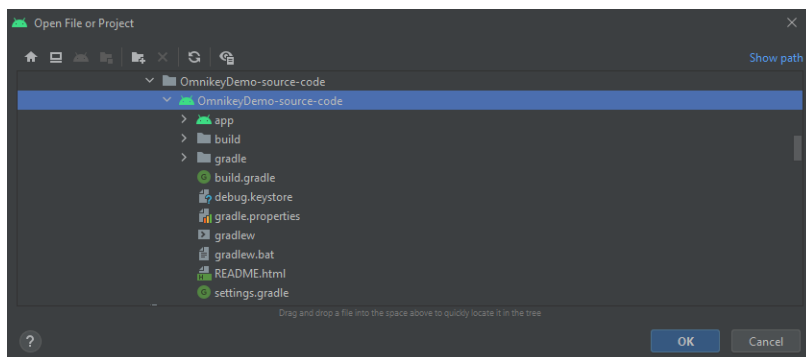
Example of device-filter.xml

```
<?xml version="1.0" encoding="utf-8"?>  
<resources>  
    <!-- HID devices have a VID of 0x076b = 1899 -->  
    <!-- XChip -->  
    <!-- OmniKey AG, Smart@Link -->  
    <usb-device vendor-id="1899" product-id="40993" /> <!-- PID: 0xa021 -->  
</resources>
```

2.3 Opening OMNIKEY Demo in Android Studio

2.3.1 How to build the OMNIKEY Demo sample app from sources

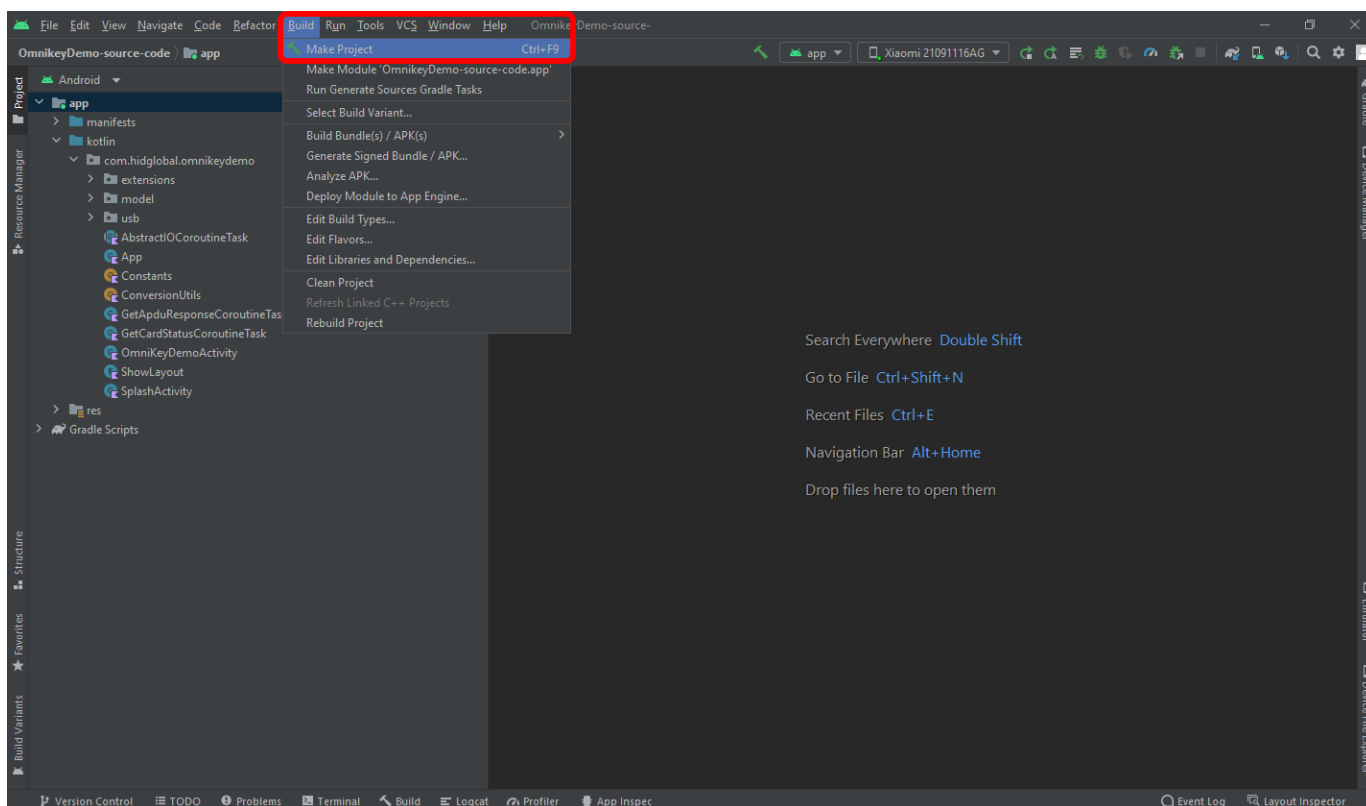
1. Unzip **hid-omnikey-demo.zip**.
2. Copy the **HID_Android_Driver-release.aar** library file from the AAR folder to the **hid-omnikey-demo/app/libs** folder. It is a required dependency.
3. Open the **hid-omnikey-demo** project in Android Studio. Wait while all the required dependencies are collected.



4. After launch, click **File > Sync Project with Gradle Files** to fetch all dependencies and configure the project.

Note: Use the OMNIKEY Demo application as an integration example.

5. Select **Build > Make Project** to build the project.



6. After a successful build, you can find the generated APK file in the folder:
hid-omnikey-demo\app\build\outputs\apk\debug

Section **03**

Using the built OMNIKEY Demo app (.apk)

3.1 Introduction

This section describes the installation of a compiled sample application that uses the HID Android Driver library.

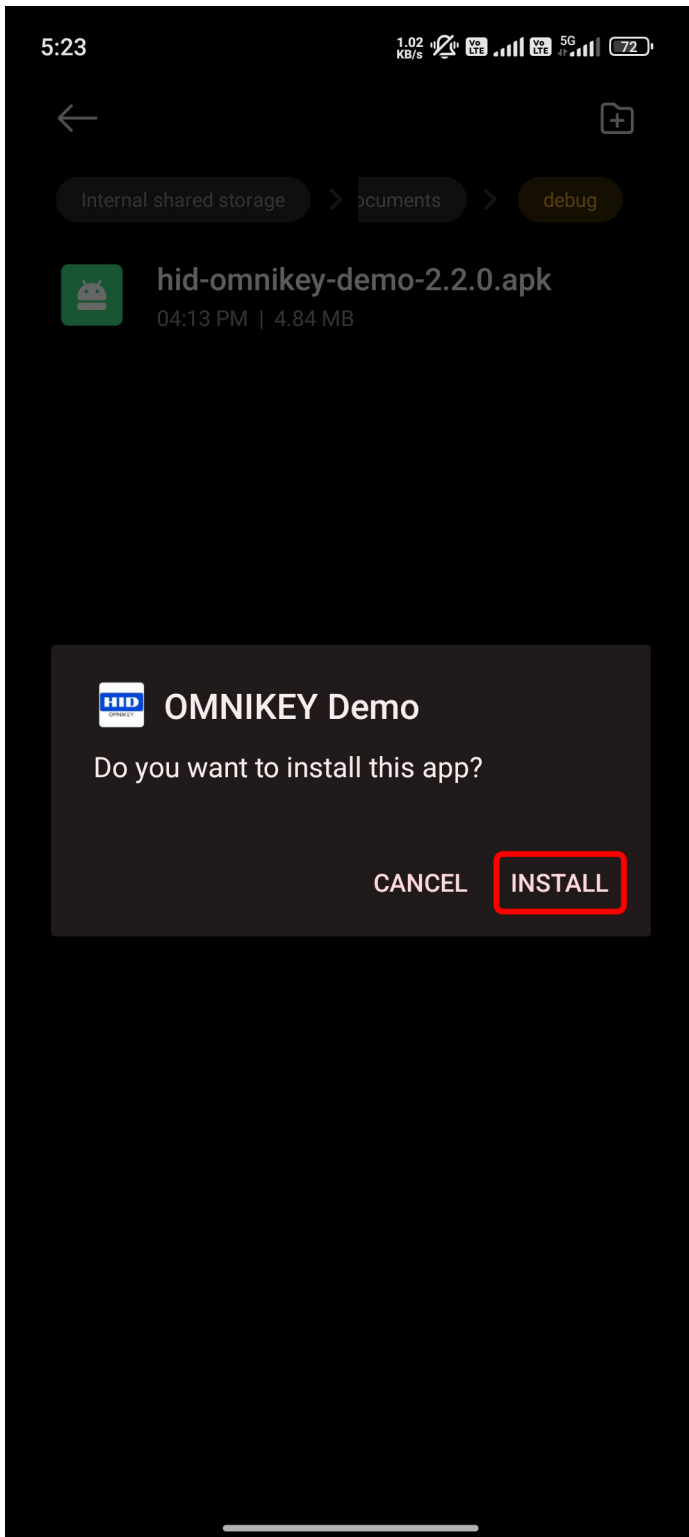
The APK file **hid-omnikey-demo-2.2.0.apk** is a built example application written in Kotlin and using the **HID_Android_Driver-release.aar** library from the AAR folder. The application is named **OmnikeyDemo** and allows the ATR and UID to be read from many types of Smart Card.

3.1.1 OMNIKEY Demo application installation and usage

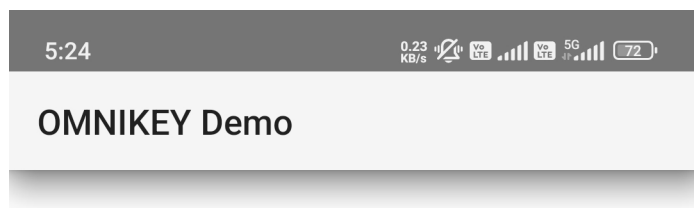
1. Copy **hid-omnikey-demo-2.2.0.apk** to your Android device and launch it.



2. Tap **INSTALL** to start the installation.



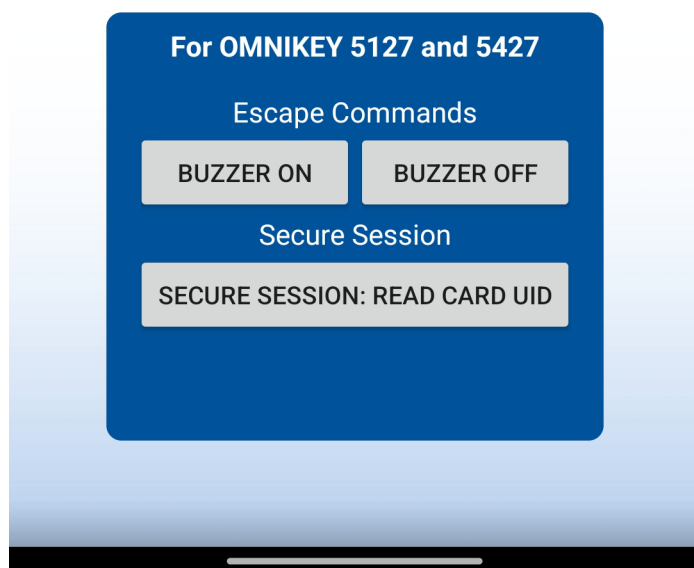
3. The application screen should look like this:



☐ Send APDU manually

ATR:

UID:

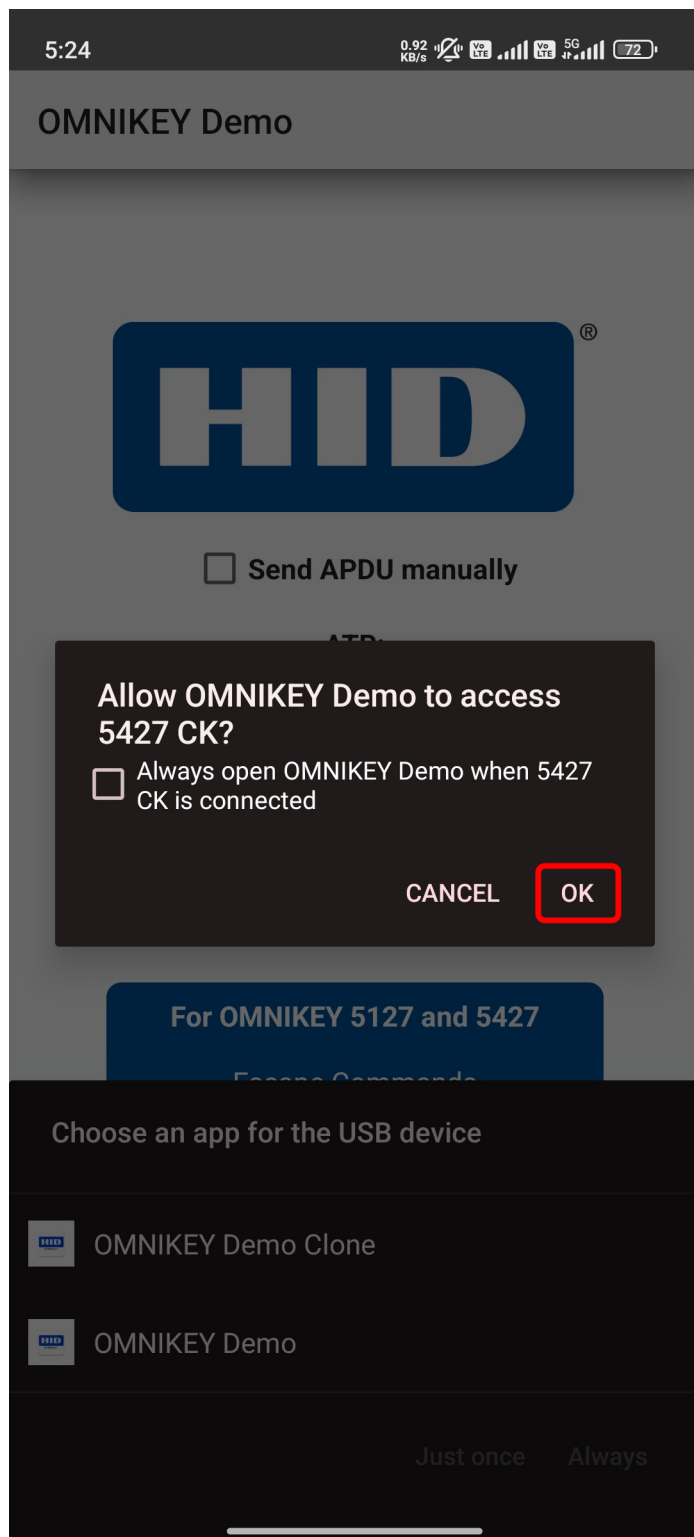


Note:

OMNIKEY Demo v2.2 adds two buttons which you can use to enable or disable the buzzer using escape commands when a card isn't presented to the reader. This feature works only with OMNIKEY 5127 and OMNIKEY 5427 readers.

OMNIKEY Demo v2.2 adds a button to read the UID of a card through a secured session established between the Android device and the OMNIKEY reader. Not all cards support this. This feature works only with OMNIKEY 5127 and OMNIKEY 5427 readers.

4. Connect an OMNIKEY RFID card reader to your Android device using USB.
Tap **OK** to allow the OMNIKEY Demo application to access your reader.

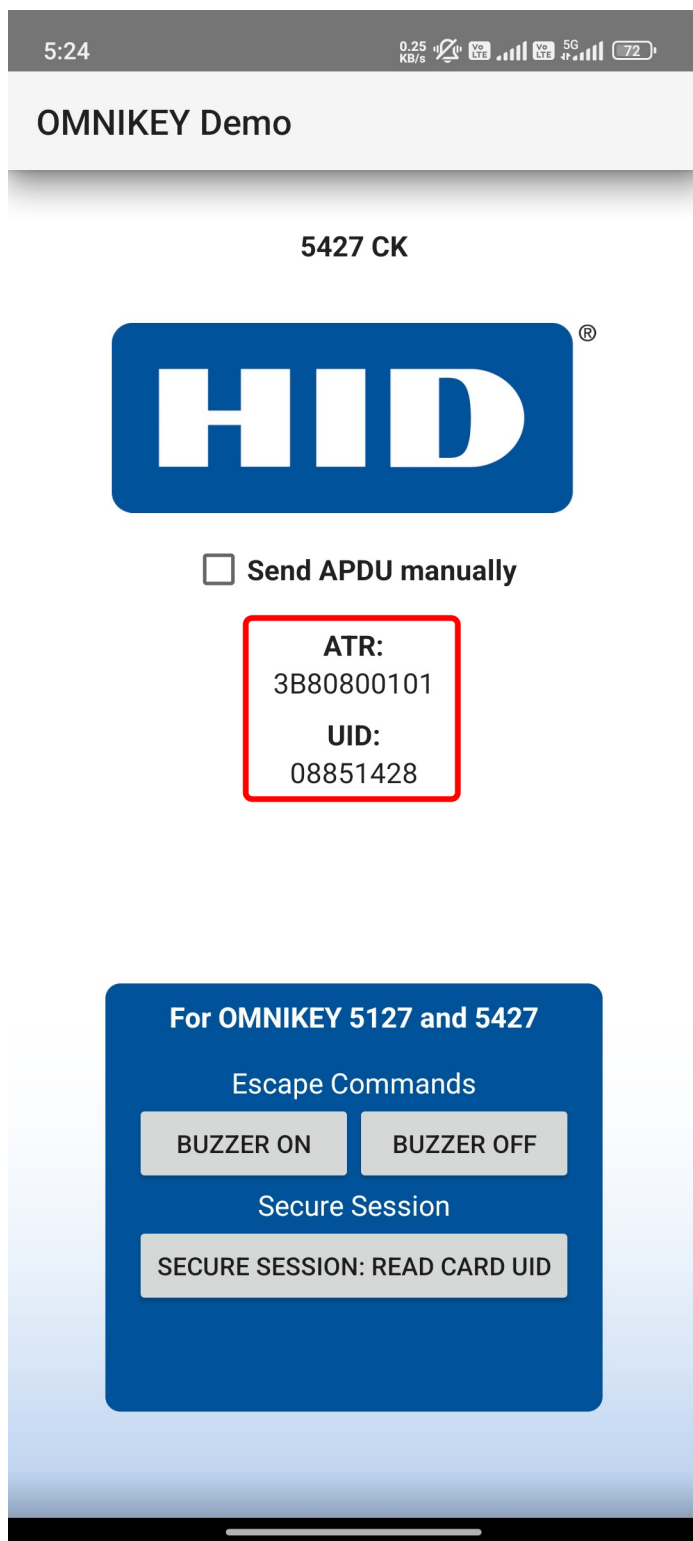


Note: When there is more than one app that uses the OMNIKEY Android Driver Library, and device identifiers match device filters in AndroidManifest.xml of each app, you must select which app is allowed to access the connected USB device.

5. The idle state with **Waiting for card...** should appear.



6. Present a smart card to the reader. The data read from the card should be displayed.



Section 04

Using escape commands

4.1 Overview

OMNIKEY Android Driver v2.1 adds support for sending escape commands to the OMNIKEY reader. You can see samples with escape commands in the "EscapeCommandsSample" class in the HID OMNIKEY Demo v2.1 application source code.

4.1.1 Using escape commands when a card is present

cardTerminal is the CardTerminal object corresponding to the selected smart card reader.

```
/**
 * Predefined ByteArray with APDU needed to enable buzzer of OMNIKEY 5247 G2 Reader.
 * It enables both LEDs (white and blue) too.
 */
val ENABLE_BUZZER_APDU = ConversionUtils.hexStringToByteArray("FF6803000113")

/**
 * Predefined ByteArray with APDU Status Word corresponding to operation success.
 */
val STATUS_WORD_SUCCESS = ConversionUtils.hexStringToByteArray("9000")

/**
 * Function to enable buzzer of the selected reader when card is presented to the reader.
 *
 * @param cardTerminal CardTerminal object corresponding to the selected smart card reader.
 *
 * @return True when operation successful, false otherwise.
 */
fun enableBuzzerCardPresent(cardTerminal: CardTerminal): Boolean {
    try {
        /* When card is present you can point protocol or use "*" to select any available
         * protocol. */
        val card = cardTerminal.connect("")
        /* Send Escape Command APDU through transmitControlCommand method. Other methods
         * are available.
         * controlCode argument is ignored, you can use any integer. It is kept to
         * be compliant with smartcardio API only. */
        val response = card!!.transmitControlCommand(0, ENABLE_BUZZER_APDU)
        /* Disconnect from the card if connected */
        card.disconnect(true)
        /* Receiving 0x9000 means success. */
        return response.contentEquals(STATUS_WORD_SUCCESS)
    } catch (e: Exception) {
        /* Log caught exception as error. */
        Timber.e(e)
        return false
    }
}
```

4.1.2 Using escape commands when a card is absent

cardTerminal is the CardTerminal object corresponding to the selected smart card reader.

```
/**
 * Predefined ByteArray with APDU needed to enable buzzer of OMNIKEY 5247 G2 Reader.
 * It enables both LEDs (white and blue) too.
 */
val ENABLE_BUZZER_APDU = ConversionUtils.hexStringToByteArray("FF6803000113")

/**
 * Predefined ByteArray with APDU Status Word corresponding to operation success.
 */
val STATUS_WORD_SUCCESS = ConversionUtils.hexStringToByteArray("9000")

/**
 * Function to enable buzzer of the selected reader when card isn't presented to the reader.
 * It enables both LEDs (white and blue) too.
 *
 * @param cardTerminal CardTerminal object corresponding to the selected smart card reader.
 *
 * @return True when operation successful, false otherwise.
 */
fun enableBuzzerCardAbsent(cardTerminal: CardTerminal): Boolean {
    try {
        /* When card is absent use "UNDEFINED" as protocol. Card object will be returned.
         * However, only transmitControlCommand method will be available. */
        val card = cardTerminal.connect("UNDEFINED")
        /* Send Escape Command APDU through transmitControlCommand method.
         * controlCode argument is ignored, you can use any integer. It is kept to
         * be compliant with smartcardio API only. */
        val response = card!!.transmitControlCommand(0, ENABLE_BUZZER_APDU)
        /* Receiving 0x9000 means success. */
        return response.contentEquals(STATUS_WORD_SUCCESS)
    } catch (e: Exception) {
        /* Log caught exception as error. */
        Timber.e(e)
        return false
    }
}
```

Section 05

Secure session

5.1 Overview

OMNIKEY Android Driver v2.2 adds a sample implementation of a secure session. This allows you to establish secure, encrypted communication between an Android application and the OMNIKEY reader. This feature is supported only by OMNIKEY 5127CK Mini and OMNIKEY 5427 CK readers.

To learn how a secure session is implemented, see the following files:

- **HidSecureSession.kt**: An example of the implementation of secure session logic.
- **HidSecureSessionSample.kt**: Examples of how the HidSecureSession class can be used.

Revision history

Date	Description	Revision
November 2025	Updated for software version 2.3.	A.3
July 2024	Updated for software version 2.2.	A.2
June 2024	Updated for software version 2.1.	A.1
April 2024	Initial release.	A.0



hidglobal.com

For technical support, please visit: <https://support.hidglobal.com>

© 2025 HID Global Corporation/ASSA ABLOY AB.

All rights reserved.

PLT-07678, Rev. A.3

Part of ASSA ABLOY