Ambimat Smart SDK - AmbiPay

Document Details

Document Name	AmbiSmartSDKIntegrationDoc
Document Manager	Gaurav Chopra
Date Effective from	11-05-2020
Date last Modified	11-05-2020
Document Type	FSD
Version	0.2
Document ID	AMBI_SDKI_001
Applicability	3rd Party

Version Control Log

Date	Version	Authored By	Remarks
11-10-2019	0.1	Mobile Development Team	Prepared the first draft of the AmbiSmart SDK Functional Specification Document.
13-05-2020	0.2	Mobile Development Team	Added IResponseHandler

Document Approvers

S. No.	Approver	Author
1	Neel Shah	Gaurav Chopra (Head, Android Development, Ambimat Electronics)

1. Objective

The objective of this document is to provide the functional specifications of the AmbiSmart SDK to enable the 3rd parties to integrate with the SDK.

2. Scope

This document will provide the requirements to integrate with the AmbiPay device.

3. Definitions & Acronyms

- Client – **N/A**
- TSP – Technology Solutions Provider (Ambimat Electronics)

4. Introduction

Each AmbiPay device has proprietary software on it and the communication with these devices can only be established with a set protocol. These AmbiPay devices can communicate with an Android Application as well, provided the Android app uses the AmbiPaySmart SDK. The steps involved in integrating the SDK with the application will be covered in this document.

5. Steps To Integrate

The steps to integrate are provided in 2 parts:

5.1 Dependency Resolution

The 3rd party app developer needs to follow the following steps:

- Copy AmbimatSmartSDKx.x.x.aar to the libs folder of your application.
- Edit app's buid.gradle to include the newly added aar file as a dependency. For eg: add the following line in the build.gradle "***implementation fileTree(dir: 'libs', include: ['*.aar'])***". This will ensure that the aar file is can be referred to at compile time.

5.2 How To Use The SDK

After the dependencies have been added, as given in the section 5.1 of this document, in the launcher activity of the 3rd party application following steps needs to be followed

- Take the handle of the SDK in the 'onCreate()' of the launcher activity by:

  IAmbiPayDeviceCommunicator ambiPayDevice = AmbimatSDKManager.getSmartDeviceCommunicator(this);

  Where AmbimatSDKManager is the class that provides the reference of the object to access various features of the AmbiPay device.
- Next make ambiPayDevice object aware of the Activity life cycle in the "onCreate()" by:

  getLifecycle().addObserver(ambiPayDevice);

  Correspondingly we need to remove it on the **onDestroy()** of the launcher activity by:

  getLifecycle().removeObserver(ambiPayDevice);

- In the event of a UI input or a life cycle event call the '**openConnection**()' method on the **ambiPayDevice** to connect.

- Next to send data to the device use the '**writeData(byte[], IResponseHandler handler)**' method on the **ambiPayDevice** object. This method will give you the response of the write in the form of a byte[]. The activity or the component calling the write method has to implement an interface IResponseHandler in order to get the response of the write command. The details of this interface are given at the end.

- Once the trans receive is complete you can call the '**disconnect()**' method on the **ambiPayDevice.**

6.0 IAmbiPayDeviceCommunicator javadocs

```
/**
 * Interface to connect, communicate and disconnect from the AmbiPay device.
 */
public interface IAmbiPayDeviceCommunicator extends LifecycleObserver {

   /**
    * This method opens the connection between the 3rd party app and the AmbiPay    device.
    * Request USB permission before calling this method.
    *
    * @throws AmbimatSDKException will be thrown in case an error is encountered.
    */
   void openConnection() throws AmbimatSDKException;

   /**
    * This method transmits the given apdu to the AmbiPay device.
    *
    * @param apdu data in the form of byte[] that is to be sent to the AmbiPay device.
    * @return
    * @throws AmbimatSDKException
    */
   byte[] writeData(byte[] apdu) throws AmbimatSDKException;


   /**
    * This method will help in disconnecting the device gracefully.
    */
   void disconnect();

}
```

7.0 IResponseHandler

```
/**
 * Ambipay device would return the response of the write over this interface.
 * The calling party has to implement this interface in order to receive the response.
 */
public interface IResponseHandler {
   /**
    * This method gets invoked when the device responds to the command sent in {@link
IAmbiPayDeviceCommunicator#writeData(byte[], IResponseHandler)}
    * @param data response received from the Ambipay device.
    */
   void onResponseReceived(byte[] data);
}
```