
User SDK Android app instruction

1. Structure:

(1) Toolkit Utils :this SDK encapsulates most of the commands of the barcode printer and thermal pos printer, the commands are encapsulated in the Utils class: DataForSendToPrinterTSC, DataForSendToPrinterPos58, DataForSendToPrinterPos76 and DataForSendToPrinterPos80, all those are static methods, can be called directly by class name. The function of these tool class is to transfer all types of printer commands into (byte[]) so it can be sent directly.

(2) Service component kit: Service. This SDK encapsulates all the service logic of the connection of APP and printer, data sending/reading, disconnection, printer status monitor and network monitor; and the time-consuming operations has also been encapsulated in the asynchronous task, developers just need to call the returned object which is bound in the service in the code to implement service. The class under Service kit is PosprinterService, inherited from service based class.

(3) There are 4 interfaces in Posprinterinterface kit.

- ① BackgroundInit interface is used to implement the processing of the working thread in the asynchronous task, not available for developers.
- ②IMyBinder interface, the inside abstract method is a variety of methods for the printer operations, implemented in PosprinterService, The bounding of the returned interface object is the implementation of this interface.
- ③UiExecute interface, Implementation of the main thread in the end of the code execution will run in the work thread of the asynchronous task.
- ④ProcessData interface, the method for sending date, The implementation of the abstract method in this interface with return of the list<byte[]> is the date be sent to the printer. (kit) asynncTask kit: Class in this kit is the subclasses of asynchronous task, is also called by the SDK itself, developers just ignore it please.

2. Registration problem of Configuration list file

(1) User

Permission: <uses-feature android:required="true"

android:name="android.hardware.usb.host" />

<uses-permission android:name="android.hardware.usb.UsbAccessory" />

<uses-permission

android:name="android.permission.READ_EXTERNAL_STORAGE"/>

<uses-permission

android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

<uses-permission android:name="android.permission.BLUETOOTH"/>

<uses-permission

```

android:name="android.permission.BLUETOOTH_ADMIN"/>
    <uses-permission
android:name="android.permission.BLUETOOTH_PRIVILEGED"/>
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE"/>
    <uses-permission
android:name="android.permission.CHANGE_NETWORK_STATE"/>
    <uses-permission
android:name="android.permission.ACCESS_WIFI_STATE"/>
    <uses-permission
android:name="android.permission.CHANGE_WIFI_STATE"/>
    <uses-permission android:name="android.permission.CAMERA"/>
<uses-permission android:name="android.permission.VIBRATE" />
    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION"/>

    <uses-feature android:name="android.hardware.camera" />
    <uses-feature android:name="android.hardware.camera.autofocus" />
<uses-permission
android:name="android.permission.RESTART_PACKAGES" />
    <uses-permission
android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS" />
<uses-permission
android:name="android.permission.READ_PHONE_STATE"/>
<uses-permission android:name="android.permission.READ_LOGS"/>

```

(2) Service components must be registered

```

        <service
android:name="net.posprinter.service.PosprinterService"></service>
    </application>

```

3. Bind service at Activity or Fragment:

/ bind service, get IMyBinder object

```
Intent intent=new Intent(this,PosprinterService.class);
```

```
bindService(intent, conn, BIND_AUTO_CREATE);
```

in the rewrite room of conn to the return of Ibinde object

```
ServiceConnection conn=new ServiceConnection() {
```

```
    @Override
```

```
    public void onServiceDisconnected(ComponentName name) {
```

```
        // TODO Auto-generated method stub
```

```
    }
```

```

        @Override
        public void onServiceConnected(ComponentName name, IBinder
service) {
            // TODO Auto-generated method stub
            //Bind success
            binder=(IMyBinder) service;
        }

```

The binder obtained is the object of IMyBinder, can call the method in the interface

4.IMyBinder instruction:

```

/**
 *@author chensong
 *@param UiExecute execute :Interface implement class, implemented
method proceed in the UI threads.
 *@param ethernetIP : IP address of printer
 *@param ethernetPort : Port of printer
 *@category :connect network port : Port of the network connection
 * */
 void connectNetPort(String ethernetIP,int ethernetPort,UiExecute
execute);
/**
 *@author chensong
 *@param UiExecute execute: Interface implement class, implemented
method proceed in the UI threads.
 *@param bluetoothID : Bluetooth ID
 *@category : Bluetooth connection port
 * */
 void connectBtPort(String bluetoothID,UiExecute execute);
/**
 *@author chensong
 *@param UiExecute execute : interface implement class, implemented
method proceed in the UI threads.
 *@param context : Context
 *@param bluetoothID : Bluetooth ID
 *@category connect USB port : USB connection port
 * */
 void connectUsbPort(Context context,String bluetoothID,UiExecute
execute);
/**
 *@author chensong
 *@param UiExecute execute : interface implement class, implemented

```

method proceed in the UI threads.

***@category** disconnect current port, use this way when do not need to connect the printer, disconnect, release memory, reduce system load

* */

void disconnectCurrentPort(UiExecute execute);

/**

* **@author** chensong

* **@category** : A thread is opened to receive the data sent by the printer, and the data is stored in an annular queue buffer of size 500.

* **@param** data : The data received by the printer is byte[], byte array

* */

void acceptdatafromprinter(UiExecute execute);

/**

* **@category** : Read data from annular queue

* **@return** : Returns an annular queue object.

* */

RoundQueue<**byte**[]> readBuffer();

/**

* clear date in the buffer

* */

void clearBuffer();

/**

* **@author** chensong

* **@category** send the date to the printer

* **@param** data the date printer gets is byte[],

* */

/**

* Detect the connect status between the APP and printer

* */

void checkLinkState(UiExecute execute);

void write(**byte**[] data,UiExecute execute);

/**

* **@author** chensong

* **@param** execute execute : Interface implement class, implemented method proceed in the UI threads.

* **@param** processData : Interface implement class, the processing method for rewrite data which need to send .

* **@category** : Can implement the ProcessData method by itself , to use the method with the command of tool class to send to user defined command, suitable to use this interface to achieve the date process operation.

```
* */  
void writeDataByYouself(UiExecute execute,ProcessData processData);
```

For more details please refer to the sample codes.

5.Please refer to the index.html in the folder of index.files for the introduction of tool class.