Q7SDK Document

Q7SDK Document 1
Printer Device 1
Interface 2
Example 9
Scanner Device 10
Interface 10
Example 12
Scanner Camera Device 10
Interface 10
Example 12
System Device 15
Interface 15
Example 20
Psam Device 20
Interface 151
Example 23
CustomKeys Device 24
Interface 154
Example 24

Printer Device

The built-in printer has two modes.And the print mode needs to enter Settings-Advanced Settings-Print advances settings to switch. And after selecting the Label mode, click "Label Study" to make print adjustments.

1. Label printer mode

2. Receipt printer modeThe reciept printer supports API calls and virtual Bluetooth printer calls.The following is the description of the API call.

Interface

initPrinterService

```
/**
  * initPrinterService
  * @description Initialize the printer device
  * @param callback Printer callback parameters
  * @return
  */
public void initPrinterService(InnerResultCallback callback)
```

sendRawData

```
/**
  * sendRawData
  * @description send ESC command to printer
  * @param data ESC data
  * @param data callback printer call back
  * @return
  */
public void sendRawData(byte[] data, InnerResultCallback callback)
```

printLines

```
/**
  * printLines
  * @description print empty line
  * @param n count of empty line
  * @param callback printer callback
  * @return
  */
public void printLines(int n, InnerResultCallback callback)
```

setAlign

```
/**
  * setAlign
  * @description set print align
  * @param align align type
  * @return
  */
public void setAlign(AlignType align)
```

setStyleBold

```
/**
  * setStyleBold
  * @description set print data bold
  * @param isBold true->set bold, false->set normal
  * @return
  */
public void setStyleBold(boolean isBold)
```

setStyleUnderline

```
/**
  * setStyleUnderline
  * @description set print data with underline
  * @param isUnderLine true->set underline, false->set normal
  * @return
  */
public void setStyleUnderline(boolean isUnderLine)
```

setFontSize

```
/**
  * setFontSize
  * @description set print text font size
  * @param size text size
  * @return
  */
public void setFontSize(int size)
```

setFontName

```
/**
  * setFontName
  * @description set print text font type, it should be a .ttf file
  * @param typeface the path of .ttf file
  * @return
  */
public void setFontName(String typeface)
```

printTextWithFont

```
/**
  * printTextWithFont
  * @description print text with customer fort
  * @param content print text
  * @param size print text size
  * @param isBold print text bold or not
  * @param isUnderLine print text with underline or not
  * @param typeface typeface the path of .ttf file
  * @param callback printer callback
  * @return
  */
public void printTextWithFont(String content, int size, boolean isBold, boolean
isUnderLine, String typeface, InnerResultCallback callback)
```

printText

```
/**
  * printText
  * @description print text
  * @param content text to print
  * @param callback printer callback
  * @return
  */
public void printText(String content, InnerResultCallback callback)
```

printBarCode

```
/**
```

```
  * printBarCode
  * @description print barcode
  * @param data barcode information
  * @param symbology barcode type
  * @param height barcode print height
  * @param width barcode print width
  * @param textposition barcode text position
  * @param callback printer callback
  * @return
  */
public void printBarCode(String data, int symbology, int height, int width, int
textposition, InnerResultCallback callback)
```

printQr

```
/**
  * printQr
  * @description print QrCode
  * @param data Qrcode information
  * @param modulesize Qrcode size
  * @param errorlevel Qrcode error level QrCodeErrorLevel
  * @param callback printer callback
  * @return
  */
public void printQr(String data, int modulesize, int errorlevel, InnerResultCallback
callback)
```

printTable

```
/**
```

```
  * printTable
  * @description print table
  * @param txts text information
  * @param weight weight of each text
  * @param align text align type: 0->left, 1->center, 2->right
  * @param callback printer callback
  * @return
  */
public void printTable(String[] txts, int[] weight, int[] align, InnerResultCallback
callback)
```

printBitmap

```
/**
  * printBitmap
  * @description print bitmap
  * @param bitmap bitmap for print
  * @param callback printer callback
  * @return
  */
public void printBitmap(Bitmap bitmap, InnerResultCallback callback)
```

isReceiptMode

```
/**
  * isReceiptMode
  * @description check if printer is receipt mode
  * @param
  * @return true -> receipt mode, false -> label mode
  */
```

```
public boolean isReceiptMode()
```

isLabelMode

```
/**
 * isLabelMode
 * @description check if printer is label mode
 * @param
 * @return true -> label mode, false -> receipt mode
 */
public boolean isLabelMode()
```

labelLocate

```
/**
  * labelLocate
  * @description Position the label to the initial printing position, this interface needs to be called before printing each label
  * @param
  * @return
  */
public void labelLocate()
```

labelOutput

```
/**
  * labelOutput
  * @description End the printing of the current label and position the label to the cutting position.
  * This interface needs to be called when each label finishes printing mouth.
```

```
  * @param
  * @return
  */
public void labelOutput()
```

Example

```
private void printTextWithFont() {
    try {
        mPrinter.printTextWithFont("Hello Q7!", 30, true, true, "/font/test.ttf", new
InnerResultCallback.Stub() {
            @Override
            public void onRunResult(boolean b) throws RemoteException {
                showNormalMessage("task result : " + b);
            }

            @Override
            public void onReturnString(String s) throws RemoteException {
                showNormalMessage("task feedback : " + s);
            }

            @Override
            public void onRaiseException(int i, String s) throws RemoteException {
                showErrorMessage("task exception");
                showErrorMessage("error code :" + i);
                showErrorMessage("message :" + s);
            }
        });
    } catch (Exception e) {
        e.printStackTrace();
```

```
    }
}
```

Scanner Device

Interface

initScan

```
/**
  * initScan
  * @description Initialize the scanning device
  * @param callback Scanning device callback parameters
  * @return
  */
public void initScan(IScanInitCallback callback)
```

startScan

```
 /**
  * startScan
  * @description Start the scan code function
  * @param scanParam Scanning parameters
  * @param callback Scan code result callback parameters
  * @return
  */
public void startScan(ScanQ7Bean scanParam, IScanQCallback callback)
```

stopScan

```
/**
  * stopScan
  * @description stop scan
  * @param
  * @return
  */
public void stopScan()
```

destroy

```
/**
  * destroy
  * @description Release scanning device resources
  * @param
  * @return
  */
public void destroy()
```

getVersion

```
/**
  * getVersion
  * @description
  * @param type 0->Decode Scheme Version, 1->Scanner Version, 2->Decode Version
  * @return
  */
public String getVersion(int type)
```

getCodeType

```
/**
  * getCodeType
  * @description get support decode type
  * @param type 0->1D Code, 1->2D Code, 2->Ohter
  * @return
  */
public List<String> getCodeType(int type)
```

configurationSymbology

```
/**
  * configurationSymbology
  * @description Configure Scanning Device Parameters
  * @param
  * @return
  */
public void configurationSymbology()
```

Example

```
private void startScan() {
    ScanQ7Bean scanQ7Bean = new ScanQ7Bean();
    try {
        scanQ7Device.startScan(scanQ7Bean, new IScanQCallback.Stub() {
            @Override
            public void onSuccess(String s) throws RemoteException {
                showNormalMessage(s);
```

```
            @Override
            public void onFailed(int i, String s) throws RemoteException {
                showErrorMessage(s);
            }
        });
    } catch (RemoteException e) {
        e.printStackTrace();

}
```

Scan Camera Device

Interface

scan

```
/**
  *  scan

  * @description start scanning

  * @param callback Scanning device callback parameters

  * @return
  */
public void scan(ScanCameraBean scanParam, IScanCallback callback);
```

stopScan

```
/**
  * stopScan
```

```
  * @description stop scanning
  * @return
  */
public void stopScan()
```

Example

```
private void startScan() {

    Bundle scanParams = new Bundle();

        ScanCameraBean.Builder scanCameraBean = new ScanCameraBean.Builder()

                .setCameraId(1)

                .setContinuous(false)

                .setTimeOut(60_000)

                .setBeep(true);

        scanDevice.scan(scanCameraBean.build(), new IScanCallback.Stub() {

            @Override

            public void onSuccess(byte[] bytes) throws RemoteException {

                Log.d(TAG, "onSuccess: " + HexUtils.bcd2str(bytes));

                showNormalMessage("Scan result is " + new String(bytes));

            }

            @Override
```

```
            public void onFailed(int i, String s) throws RemoteException {

                showErrorMessage("Scan failed with " + i + "," + s);

            }

        });

}
```

System Device

Interface

reboot

```
/**
  * reboot
  * @description reboot terminal
  */
public void reboot()
```

getSystemInfo

```
/**
  * getSystemInfo
  * @description Get the sn/imei/imsi... info string
  * @param type information type
  * @return
  */
public String getSystemInfo(SystemInfoType type)
```

getInet4Address

```
/**
  * getInet4Address
  * @description  Get the IP address of WIFI
  * @return address or null
  */
@RequiresPermission(Manifest.permission.INTERNET)
public String getInet4Address()
```

installApkBack

```
/**
  * Install an app
  * @param apkPath  the apk file path, Android S has restrictions on memory access.
Please refer to SDK Sample to use this method.
  * @param listener result listener
  */
public void installApkBack(String apkPath, FinishListener listener)
```

uninstallApkBack

```
/**
  * Uninstall an app
  * @param pkgName  the package name of app
  * @param listener result listener
  */
public void uninstallApkBack(String pkgName, FinishListener listener)
```

setSystemDateTime

```
/**
  * setSystemDateTime
  * @description set system date and time
  * @param format type: yyyy-MM-dd hh-mm-ss
  * @return setting result
  */
public boolean setSystemDateTime(int year, int month, int date, int hour, int minite,
int second)
```

getHardwareAddress

```
/**
  * Get the MAC address of WIFI
  * @return address or null
  */
@RequiresPermission(Manifest.permission.INTERNET)
public String getHardwareAddress()
```

The following are three types of whitelists, each with independent whitelist functions

three types of whitelists : （ InstallWhiteList 、PermissionWhitelist、KeepAliveList）

setEnableInstallWhiteList

```
/**
  * setEnableInstallWhiteList
  * @param isEnableWhiteList
  * @return setting result
  */
public boolean setEnableInstallWhiteList(boolean isEnableWhiteList)
```

setInstallWhiteList

```
/**
     * setInstallWhiteList
     * @param list
     * @return setting result
     */
public boolean setInstallWhiteList(List<String> list)
```

getInstallWhiteList

```
/**
```

```
     * getInstallWhiteList

     * @return get install white list

     */

public boolean getInstallWhiteList(List<String> list)
```

deleteInstallWhiteList

```
/**
  * deleteInstallWhiteList
  * @param pkgName@param pkgName
  * @return delete result
  */
public boolean deleteInstallWhiteList(String pkgName)
```

clearWhiteList

```
/**
  * clearWhiteList
  * @description clear all install whitelist
  * @return clear result
  */
public boolean clearWhiteList()
```

getPermissionWhitelist

```
/**
  * getPermissionWhitelist
```

```
  * @description get default authorization whitelist,permission granted by default
  * @return get default authorization whitelist
  */
public List<String> getPermissionWhitelist()
```

addPkgNamesToPermissionWhitelist

```
/**
  * addPkgNamesToPermissionWhitelist
  * @description add default authorization whitelist,permission granted by default
  */
public void addPkgNamesToPermissionWhitelist(List<String> list)
```

getKeepAliveList

```
/**
  * getKeepAliveList
  * @description get keep alive whitelist,When low memory the process will not be killed
  * @return get keep alive whitelist
  */
public List<String> getKeepAliveList()
```

addPkgNamesToKeepAliveList

```
/**
  * addPkgNamesToKeepAliveList
  * @description add keep alive whitelist,the process will not be killed
  */
public void addPkgNamesToKeepAliveList(List<String> list)
```

deletePkgNameFromPermissionWhitelist

```
/**
  * deletePkgNameFromPermissionWhitelist
  * @description delete by package name default authorization whitelist
  */
public void deletePkgNameFromPermissionWhitelist(String pkgName)
```

deletePkgNameFromKeepAliveList

```
/**
  * deletePkgNameFromKeepAliveList
  * @description delete by package name keep alive whitelist
  */
public void deletePkgNameFromKeepAliveList(String pkgName)
```

Example

```
private void getSystemInfo() {
    showNormalMessage("-----------------------------------\nGet              System
Info:\n------------------");
    for (int i = 0; i < SystemDevice.SystemInfoType.values().length; i++) {
        SystemDevice.SystemInfoType type = SystemDevice.SystemInfoType.values()[i];
        showNormalMessage("%s:\n%s\n", type.name(), device.getSystemInfo(type));
    }
    showNormalMessage("-----------------------------------");
}
```

Psam Device

Interface

reset

```
/**
  * reset
  *  To reset the Psam card to its original state
  *  @return byte[] reset result
  */
public byte[] reset()
```

open

```
/**
  * open psam deivce
  *  @return
  *
  */
public byte[] sendApdu(byte[] apdu)
```

sendApdu

```
/**
  * To send the APDU command to psam card
  *  @param apdu apdu command
  *  @return byte[] return from psam card. 9000 means success
  */
```

```
public byte[] sendApdu(byte[] apdu)
```

close

```
/**
  * close psam deivce
  *  @return close rusult
  *
  */
public boolean close()
```

Example

```
/private void reset() {

        byte[] result = mPsamDevice.reset();

        showNormalMessage("Reset result = " + HexUtils.bcd2str(result));

    }

    private void sendApdu() {

        showNormalMessage("send apdu: FE020005030084000008");

        byte[]                                result                                =
mPsamDevice.sendApdu(HexUtils.hexStringToByte("FE020005030084000008"));

        showNormalMessage("get result:" + HexUtils.bcd2str(result));

    }
```

CustomKeys Device

Interface

show

```
/**
  * show
  *  Show scan result
  *  resultBean: resultData type
  */
public void show(CustomKeysResultBean resultBean)
```

Example

```
private void show(){

        customKeysDevice.show(new CustomKeysResultBean());

    }
```