

iScan Pro Development Notes

Documentation

Current Version: v1.3
Version Status: Released

Documentation Update Notes

Document Version	Update time	Adapted version	Updates	Participants
v1.0	2020/07/23	V1.0	Initial content	Wang Hang
v1.1	2020/11/12	V1.1	1、 Update the interface	Lin Zhenyu / Dong Wenbin / Lu Quanfeng /
V1.2	2020/12/29	V1.2	1、 Update the interface definition	Lu Quanfeng/Wang Hang
V1.3	2022/05/16	V1.3	1、 Correction of some interface parameters description	Lin Zhen Yu

Catalog

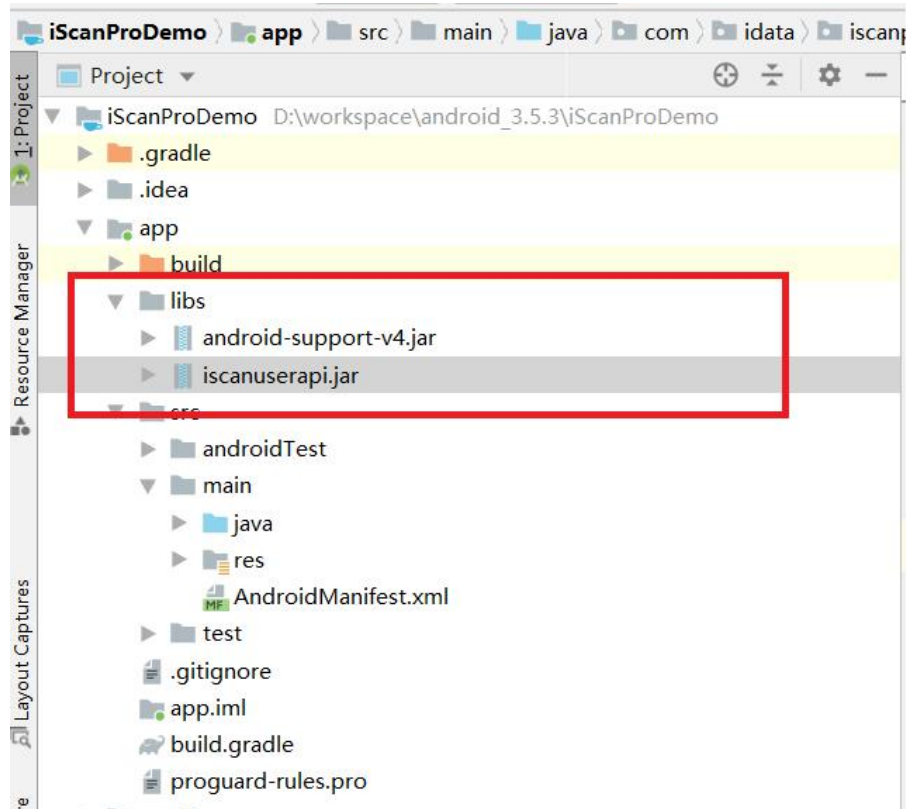
1、 Documentation description	3
2、 Development instructions	3
3、 Interface list	4
3.1 Turn on scanning	4
3.2 Closing the scan	5
3.3 Start scanning the code	5
3.4 Stop Sweep	5
3.5 Disabling the scan button	5
3.6 Configuring the scan result output method	5
3.7 Configuring the scan prompt method	6
3.8 Configuring additional keys	7
3.9 Configuring Barcode Data Processing Rules	7
3.10 Configuring the trigger method	7
3.11 Configuring the Scan Timeout Time	8
3.12 Configuring the Continuous Scan Interval	8
3.13 Setting the character encoding format	8
3.14 Whether to delete the existing content of the edit box	9
3.15 Restoring the default configuration parameters	9
3.16 Configuring Decode Region Mode	9
3.17 Setting the laser fill light mode	9
3.18 Configuring Barcode Control Switches	10
3.19 Image output mode	10
3.20 Registering a scan result listener object	11
3.21 Logging off the scan result listener object	12

1、 Document description

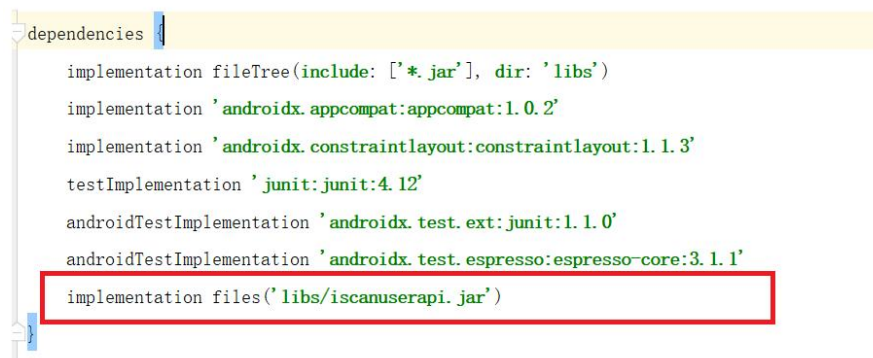
This document describes the development instructions for scanning barcodes on iData terminals for reference by developers in related industries and integrators.

2、 Development Notes

- 1) Please copy "iscanuserapi.jar" from the libs directory to the libs directory of the project, as follows.



- 2) Add "iscanuserapi.jar" to the compile path of the project, and then add the "build.gradle" file as follows.



- 3) Initialize the miScanInterface in the onCreate method in the Context.
- 4) The miScanInterface method is called after initialization to perform the scanning operation.

```

//定义接口操作对象
private iScanInterface miScanInterface ;

//定义数据回调接口
private IScanListener scanResltListener = new IScanListener() {
    /**
     * @param data 条码数据
     * @param type 条码类型
     * @param decodeTime 解码耗时
     * @param keyDownTime 按键时间
     * @param imagePath 图片路径, 注意: 默认图片输出为关闭状态;
     * 如果需要条码图片需要打开图片输出选项。
     */
    @Override
    public void onScanResults(String data, int type, long decodeTime, long keyDownTime, String imagePath) {
        Log.d(TAG, msg: "onScanResults: data="+data);
        Log.d(TAG, msg: "onScanResults: type="+type);
        Log.d(TAG, msg: "onScanResults: decodeTime="+decodeTime);
        Log.d(TAG, msg: "onScanResults: keyDownTime="+keyDownTime);
        Log.d(TAG, msg: "onScanResults: imagePath="+imagePath);
    }
};

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    //初始化接口对象
    miScanInterface = new iScanInterface( context: this);
    //注册数据回调接口
    miScanInterface.registerScan(scanResltListener);
}

```

5) Application exit destruction

```

@Override
protected void onDestroy() {
    super.onDestroy();
    //应用关闭后注销回调
    miScanInterface.unregisterScan(scanResltListener);
}

```

3、Interface list

3.1 Turn on scanning

void open()

Description: Power up the scanning head, power up before scanning, perform this operation before starting scanning.

Input parameters: empty

Return value: null

Note: This interface can not be called frequently, for power saving needs to be called once when the program is opened, or not called to iScanPro automatic control, frequent switching of the serial port is likely to cause the scanning engine to seize.

3.2 Closing the scan

`void close()`

Description: Close the scan head, after closing, the scan head will stop working and no longer respond to the decoding request.

Input parameters: empty

Return value: null

Note: By default, the system will automatically manage the scan head, if there is no special case, it is recommended that application developers do not call this interface.

3.3 Start scanning code

`void scan_start()`

Description: Start to scan the code, the scanning head will come out of the light and recognize the barcode after the call.

Input parameters: empty

Return value: null

3.4 Stop Sweep

`void scan_stop()`

Description: Stop scanning, after this operation is executed, the scanning head light will be extinguished

Input parameters: empty

Return value: null

3.5 Disabling the scan button

`void lockScanKey(boolean enable)`

Description: Disable the scan button. Pressing the scan button after disabling it will no longer respond to scan requests.

Input parameter: `enable`, scan key disable status.

false: disable the scan button, after disabling, users can develop their own definition of the scan button function.

true: disable, press the scan button to automatically trigger the scan function after disabling.

Return value: null

Note: After the application is disabled, **please un-disable it when the application is exited**, otherwise it will cause the problem of not being able to scan.

3.6 Configuring the scan result output method

void setOutputMode(int mode)

Description: Configure the scan result output mode, see the "Output Mode" section for the description of the output mode.

Input parameter: mode, output mode, default: 0, supports the following values.

0: Scan results are sent directly to the focus edit box

1: Scan results are sent in broadcast mode, when broadcast mode is selected, you can listen to the scan results through Android broadcast mechanism, the default broadcast definition of scan results is as follows.

action: android.intent.action.SCANRESULT

extras: **value**, the result of the recognition, the value is null if the recognition fails.

length, the length of the barcode.

2: Scan results are sent in analog keystroke mode

Return value: null

3.7 Configuring the scan prompt method

void enablePlayBeep(boolean enable)

Description: Configure whether to play sound after successful scanning

Input parameter: enable, sound playback status after successful scanning

true: play sound after successful scan

false: After successful scanning, no sound is played

Return value: null

void enableFailurePlayBeep(boolean enable)

Description: Whether to play the sound after the scan fails

Input parameter: enable, beep status after scan failure

true: Allow sound to be played after a failed scan

false: no sound is played after a failed scan

Return value: null

void enablePlayVibrate(boolean enable)

Description: Whether or not to vibrate after successful scanning

Input parameter: enable, vibration prompt status after successful scanning

true: After successful scanning, the vibration prompt

false: after successful scanning, no vibration

Return value: null

void lightSet(boolean enable)

Description: Configure the scan indicator status

Input parameter: enable, scan indicator status

true: scanning process on indicator

false: scanning process off indicator

Return value: null

3.8 Configuring additional keys

void enableAddKeyValue(int value)

Description: Append the key value of the specified key to the scan result.

Input parameters: value, parameter value, additional value type. The supported values are as follows.

0: No additional content

1: Additional carriage return key

2: Additional TAB key

3: Additional line break(\n)

Return value: null

3.9 Configuring Barcode Data Processing Rules

void addPrefix(String text)

Description: Add a prefix to the scan result string after a successful scan

Input parameter: text, the prefix character added before the scan result

Return value: null

void addSuffix(String text)

Description: Add a suffix to the scan result string after a successful scan

Input parameter: text, the suffix added to the scan result

Return value: null

void filterCharacter(String text)

Description: Filter specific characters

Input parameter: text, the character to be filtered

Return value: null

3.10 Configuring the trigger method

void continuousScan(boolean enable)

Description: Configure the continuous scan state

Input parameter: enable, continuous sweep status

true: turn on continuous scanning.

Note: After setting open continuous scanning, you need to click the scan button to scan normally.

false: turn off continuous scanning.

Return value: null

void effortScan(boolean enable)

Description: Power saving mode, press the button until the barcode is solved or the scan timeout will stop scanning

Input parameter: enable, power saving mode status

 true: Turn on labor-saving mode

 false: turn off power-saving mode

Return value: null

3.11 Configuring the Scan Timeout Time

void setTimeOut(int value)

Description: Set the timeout time to stop scanning automatically after the specified time.

Input parameter: value, continuous scan interval time, time unit: ms (milliseconds)

System default: 30000

Return value: null

3.12 Configuring the Continuous Scan Interval

void setIntervalTime(int value)

Description: Configure the interval time of continuous scanning, the interval time after each successful scan in continuous scanning mode

Input parameter: value, continuous sweep interval time, time unit: ms (milliseconds)

System default: 5000

Return value: null

3.13 Setting the character encoding format

void setEncodeFormart(int mode)

Description: Configure the decoding character encoding format

Input parameter: mode, encoding type value, optional encoding as follows.

 0: Auto

 1: GB2312

 2: GBK

 3: GB18030

 4: UTF-8

 5: ISO-8859-1

 6: BIG5

 7: SJIS

8: EUC-JP

Return value: null

3.14 Whether to delete the existing content of the edit box

`void setDelete(boolean enable)`

Description: Whether to delete the content of the edit box after the barcode is scanned

Input parameter: enable, automatically empty the existing content state

true: Delete the content of the edit box after scanning the barcode

false: do not delete the content of edit box after scanning the barcode, default value.

Return value: null

3.15 Restore default configuration parameters

`void resetScan()`

Description: Restores the default settings of the scan parameters, and restores the scan configuration parameters to the factory state after restoration.

Input parameters: empty

Return value: null

3.16 Configuring Decoding Region Mode

`void setCenterMode(int mode)`

Description: Configure the decoding region mode.

Input parameter: mode, decoding region mode, supports the following values.

0: Region decoding, default value.

1: Center decoding, after opening, the system only recognizes the barcode in the center area of the aiming light.

Return value: null

3.17 Setting the laser fill light mode

`void setAimLightMode(int mode)`

Description: Set the laser fill light mode

Input parameter: mode, light mode.

0: fill light and laser are bright

1: Only the laser is bright

2: Only fill light bright

3: None of them are bright
Return value: null

3.18 Configuring Barcode Control Switches

`void setBarcodeEnable(int barcodeId,boolean enable)`

Description: Configure barcode control switch

Input parameter: barcodeId, barcode type, supports the following values.

- 0: Aztec
- 1: Codabar
- 2: Code11
- 3: Code128
- 4: Code39
- 6: Code93
- 8: DataMatrix
- 9: EAN8
- 10: EAN13
- 11: Interleaved 2 of 5
- 12: Maxicode
- 13: Micropdf
- 15: Pdf417
- 17: QR
- 19: UPCA
- 20: UPCE0
- 48: HANXIN

enable, barcode status, supports the following values.

true: Support specified barcode decoding

false: turn off the specified barcode decoding

Return value: null

3.19 Image output mode

`void saveImageMode(int mode)`

Description: Configure the image output mode

Input parameter: mode, the image output mode, supports the following values.

- 0: off, default value
- 1: Always output
- 2: Decode the output successfully
- 3: Decoding failure output

Return value: null

Note: 1) This interface is only valid for DS7000/7000Pro scanning head
2) Callback images need the device to get read access, Android 10.0 version need to be in
AndroidManifest.xml file Application to add
android:requestLegacyExternalStorage="true"
3) APP must run in the main thread

3.20 Registering a scan result listener object

void registerScan(IScanListener mIScanListener)

Description: Register the listener and implement the listener callback method to get the path to the saved image

Input parameters: mIScanListener: listener object, example of use is as follows.

Implement the callback method as follows.

```
public void onScanResults(String data, int type, long decodeTime,  
long keyDownTime, String imagePath) {
```

```
/**
```

Can not directly perform UI operations, we recommend calling
runOnUiThread to switch the main thread for operations

Parameter description: data:barcode result, type:barcode type
decodeTime: decoding time, keyDownTime: scanning key press
time

imagePath: the path where the scan head gets the image

```
*/
```

```
}
```

Return value: None

Note: 1) This interface is only valid for DS7000/7000Pro scanning head, if you need to get the image path.

You need to call the saveImageMode method first to enable the image output function.

(2) The interface callback will return the barcode and picture, in addition, the barcode return of this interface is not affected by the setting parameters of the output mode, and **it is recommended to cooperate with the setting of the output mode as broadcast to avoid accidentally touching the UI control after the barcode is scanned by the default input box output mode.**

3) After turning on the picture output, please delete the picture after use to save the storage space of the device.

3.21 Logging off the scan result listener object

`void unregisterScan(IScanListener mIScanListener)`

Description: Logout of the scan result listener, called when the application is destroyed.

Input parameters: mIScanListener: listener object, listener object at initialization

Return value: None