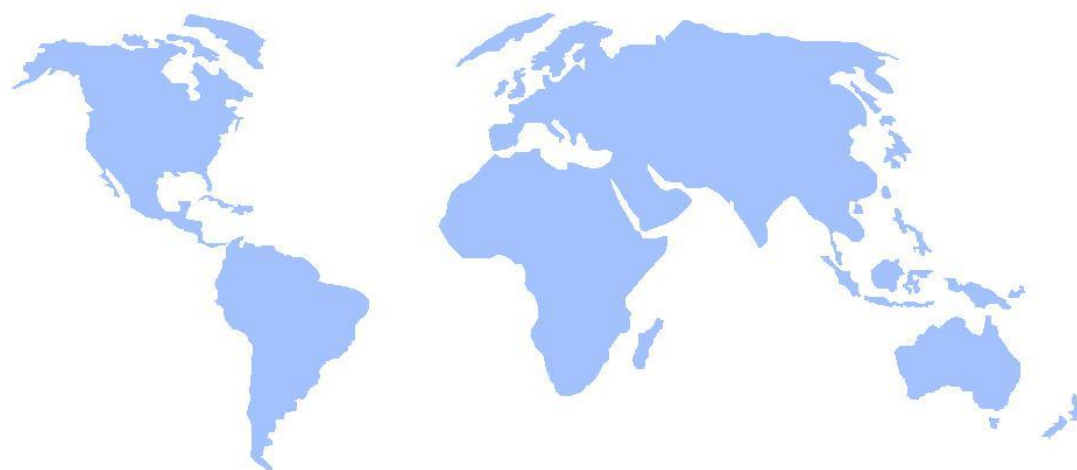




ScanKey Interface User Guide



<https://en.seuic.com/>

Revision Record

| Version | Revision Date | Revised content | Revisionist |
|---------|---------------|----------------------|-------------|
| 1.0 | 2017/03/01 | Create this document | Liangwei Xu |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

Catalogs

| | |
|--------------------------------------|---|
| Overview..... | 3 |
| Hardware Platform..... | 3 |
| Software Platform | 3 |
| Interface Definition..... | 4 |
| ScanKeyService Class | 4 |
| Method of use | 4 |
| Function Interface | 5 |
| 1. Register callback function | 5 |
| 2. Uninstall callback function | 5 |
| IKeyEventCallback Interface | 6 |
| Function Interface | 6 |
| 1. Press response button..... | 6 |
| 2. Lift response button | 6 |
| Scan Code - Key List | 6 |
| D700..... | 6 |
| D500P | 7 |

Overview

Scan key is a special key for scanning application, originally it was only used in scanning SDK, other programs (such as UHF application) need to write their own JNI library to monitor it if they want to use it, but they cannot do mutual exclusion when scanning application is open.

This interface not only opens the scan key interface, but also provides monitoring of other physical keys.

Hardware Platform

The SDK is available for the following end devices.

- D500P
- D700

The specific scope of use is specified in the function interface description, if not specified means that all support.

Software Platform

The SDK is based on Android version 5.1 and supports Eclipse and Android Studio development tools.

Interface Definition

| | |
|------------------|---|
| Package Name | com.seuic.scankey |
| Package file | scankey.jar |
| System package | Yes |
| Included Classes | ScanKeyService |
| Function | Provides interface to scan keys and other physical keys |

ScanKeyService Class

Method of use

The ScanKeyService class is simple to use, it is implemented in Singleton mode and calls getInstance() to return the object. It provides two functions to implement the registration callback function and uninstallation callback function. Suitable for these two functions called onResume and onPause, respectively, but of course you can also call in other places.

To facilitate programming, ScanKeyService through the callback function to achieve the key response. We need to implement an instance of IKeyEventCallback to pass as a parameter to the registration function, and write code in onKeyDown and onKeyUp to do the processing after receiving a keystroke.

The following are code examples.

```
import com.seuic.scankey.IKeyEventCallback;
import com.seuic.scankey.ScanKeyService;

private ScanKeyService mScanKeyService = ScanKeyService.getInstance();

private IKeyEventCallback mCallback = new IKeyEventCallback.Stub() {
    @Override
    public void onKeyDown(int keyCode) throws RemoteException {
        Log.d(TAG, "onKeyDown: keyCode=" + keyCode);
    }

    @Override
    public void onKeyUp(int keyCode) throws RemoteException {
        Log.d(TAG, "onKeyUp: keyCode=" + keyCode);
    }
};

@Override
protected void onResume() {
    super.onResume();
    mScanKeyService.registerCallback(mCallback, null);
}

@Override
protected void onPause() {
```

```

super.onPause();
mScanKeyService.unregisterCallback(mCallback);
}

```

Function Interface

| Function | Description |
|--------------------|-----------------------------|
| registerCallback | Register callback function |
| unregisterCallback | Uninstall callback function |

1. Register callback function

void registerCallback(IKeyEventCallback callback, String cookie)

Parameters

callback

The object that implements the IKeyEventCallback interface needs to implement both onKeyDown and onKeyUp interfaces.

cookie

The key value of the key to be monitored, separated by a comma if multiple keys are to be monitored. Set to null to monitor scanned keys. Note that the key value is the key value of the underlying hardware (scan code), not the KeyEvent key value of Android, the two have correspondence, the values are not equal. Example.

“212”: Indicates monitoring of a key with scan code 212, corresponding to KeyEvent.KEYCODE_CAMERA.

“100,101,102”: indicates that the three keys with scan codes 100, 101 and 102 are monitored, and any of them triggered will call the registered callback function.

Return Value

None

Remarks

Calling registerCallback multiple times does not register multiple times if the callbacks are the same. The underlying layer will check if the incoming callback is already registered, and if it is, it will not be re-registered, but the cookie value will be updated to the last value. Accordingly, the unregisterCallback is called only once.

If multiple programs monitor the same keystroke by registering different callbacks, which program will receive the response to that keystroke? The answer is: the one that was registered last. If there are three programs A, B and C registered in turn to monitor the scan key, C will receive a notification when the scan key is pressed. If C unloads the callback function, then B will be notified, and if B exits, A will be notified. This mechanism is used to ensure that a key press only works for one program.

2. Uninstall callback function

void unregisterCallback(IKeyEventCallback callback)

Parameters

callback

The object when registerCallback is called.

Return Value

None

IKeyEventCallback Interface

As a callback class that responds to keystrokes, IKeyEventCallback requires developers to implement two overloaded functions onKeyDown and onKeyUp.

Function Interface

| Function | Description |
|-----------|-----------------------|
| onKeyDown | Press response button |
| onKeyUp | Lift response button |

1. Press response button

void onKeyDown(int keyCode) throws RemoteException;

Parameters

keycode

key value to inform the program which key was pressed.

Return Value

None

2. Lift response button

void onKeyUp(int keyCode) throws RemoteException;

Parameters

keycode

key value to inform the program which key was pressed.

Return Value

None

Scan Code - Key List

Use scan codes to monitor keystrokes.

D700

| Button | Key Scan Code | KeyEvent Key Value | Remarks |
|--------|---------------|--------------------|---------|
|--------|---------------|--------------------|---------|

| | | | |
|--------------------------------------|-----|-------------|--|
| Function keys on the upper left side | 102 | F1 | |
| Left and right side scan keys | 250 | F12 | |
| Volume + | 115 | VOLUME_UP | |
| Volume - | 114 | VOLUME_DOWN | |

D500P

| Button | Key Scan Code | KeyEvent Key Value | Remarks |
|----------------------|---------------|--------------------|---------|
| Scan key | 250 | F12 | |
| PTT | 252 | PROG_GREEN | |
| 1 | 2 | 1 | |
| 2 | 3 | 2 | |
| 3 | 4 | 3 | |
| 4 | 5 | 4 | |
| 5 | 6 | 5 | |
| 6 | 7 | 6 | |
| 7 | 8 | 7 | |
| 8 | 9 | 8 | |
| 9 | 10 | 9 | |
| 0 | 11 | 0 | |
| Blue function key +1 | 59 | F1 | |
| Blue function key +2 | 60 | F2 | |
| Blue function key +3 | 61 | F3 | |
| Blue function key +4 | 62 | F4 | |
| Blue function key +5 | 63 | F5 | |
| Blue function key +6 | 64 | F6 | |
| Blue function key +7 | 65 | F7 | |
| Blue function key +8 | 66 | F8 | |
| Blue function key +9 | 67 | F9 | |
| Blue function key +0 | 68 | F10 | |
| Delete key | 14 | DEL | |
| Enter key | 28 | ENTER | |