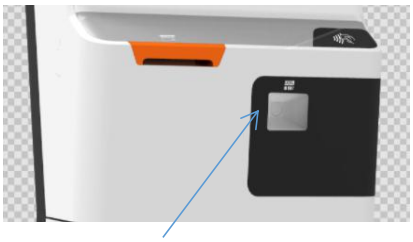
Barcode Scanner User Guide

SUNMI barcode scanner + keyboard component can avoid conflicts between a handheld scanner and input method.
The introductions to USB barcode scanner and serial barcode scanner are listed below:

USB Barcode Scanner



Serial Barcode Scanner (Much Smaller)

1. USB Barcode Scanner

A USB barcode scanner functions like a USB keyboard, **which can only collect data.** **Two collection methods** are available (alternative, please set according to your needs. **KeyEvent is used by default**):
Method 1: KeyEvent. Use dispacthKeyEvent.
Method 2: Broadcast. While using this mode, data cannot be filled onto the input box shown on the App interface like using a keyboard. Please **switch receive mode** and **collect data scanned through broadcast** following the instructions below.

a. Switch Receive Mode:

Method 1: “Settings”->change “Barcode Scanning and Keyboard” into “Do Not Output” + “Broadcast Output”

Method 2 (recommended):
action:com.sunmi.scanner.ACTION_BAR_DEVICES_SETTING
<p>field descriptions:

Key (*Required)	Description	Field Type
*name	Device Name	String (can be obtained by enumerating the UsbDevice)
*pid	Barcode Scanner pid	Integer (same as above)
*vid	Barcode Scanner vid	Integer (same as above)
*type	Data receiving mode	Integer (see type descriptions below)
toast	Whether to show Toast debugging info	Boolen (false by default)

<p> name/pid/vid list:

Name	pid	vid
Synmbol Bar Code Scanner	0x1200	0x05E0
Point of Sale Fixed Barcode Scanner	0x2514	0x05F9
SM-S100W USB HID Keyboard	0x0022	0x324F

SM-S100W USB HID Keyboard	0x00C1	0x324F

<p> type descriptions:
0-->Keyboard.
1-->Barcode scanner. The data received will be directly shown on UI (KeyEvent) .
2-->Barcode scanner. The data received won't be directly shown on UI (Broadcast Mode).
3-->Barcode scanner, Acceleration Mode (data will be filled onto the input box at one go. 1.0.18 is required).

<p>sample (set broadcast output for a device):

```
Intent intent = new Intent();
intent.setAction("com.sunmi.scanner.ACTION_BAR_DEVICES_SETTING");
intent.putExtra("name","Point of Sale Fixed Barcode Scanner.");
intent.putExtra("pid",9492);
intent.putExtra("vid",1529);
intent.putExtra("type",2);        //1 KeyEvent output    2Broadcast output
intent.putExtra("toast",true);
context.sendBroadcast(intent);
```

b. Collect Data Scanned Through Broadcast

Listen to a broadcast: "com.sunmi.scanner.ACTION_DATA_CODE_RECEIVED"
Sample: private static fifinal String ACTION_DATA_CODE_RECEIVED = "com.sunmi.scanner.ACTION_DATA_CODE_RECEIVED";

```
private static fifinal String DATA = "data";
private BroadcastReceiver receiver = new BroadcastReceiver()
{
@Override
public void onReceive(Context context, Intent intent)
{
String code = intent.getStringExtra(DATA);
if (code != null && !code.isEmpty())
{
mCode.setText(code);
}
}
};
private void registerReceiver()
{
IntentFilter fifilter = new IntentFilter();
fifilter.addAction(ACTION_DATA_CODE_RECEIVED);
registerReceiver(receiver, fifilter);
}
```

2. Serial Barcode Scanner

A serial barcode scanner is suitable for scanning screen barcodes, like payment barcodes on phone screens or digital membership barcodes, etc. It supports **KeyEvent output and Broadcast output, so no mode switching or setting is needed.**
Method 1: KeyEvent. Same as USB barcode scanner. Use dispacthKeyEvent.
Method 2: Broadcast. Same as USB barcode scanner. Use BroadcastReceiver.

In addition, commands can be sent through broadcast to control a serial barcode scanner (for example, turn it on or off):
Serial Barcode Scanner Commands Manual
Serial Barcode Scanner SourcecodeDemo

The Way to Send Commands With Broadcast

action：com.sunmi.scanner.Setting_cmd
cmd byte[]：cmd_data: command + 2-digit check bit (check and computation)
Demo:

```
/*
**send serial command
*/
public void onSendSerialCmd(View view) {
try {
String s = "NLS0302010;";//serial command, for example: NLS0302010;
byte[] bytes = s.getBytes();
byte[] cmd = new byte[bytes.length + 2];
System.arraycopy(bytes, 0, cmd, 0, bytes.length);
lrcCheckSum(cmd);
// send cmd
Intent intent = new Intent("com.sunmi.scanner.Setting_cmd");
intent.putExtra("cmd_data", cmd);
sendBroadcast(intent);
} catch (Exception e) {
e.printStackTrace();
}
}
private void lrcCheckSum(byte[] content) {
int len = content.length;
int crc = 0;
for (int l = 0; l < len - 2; l++) {
crc += content[l] & 0xFF;
}
crc = ~crc + 1;
```

```
content[len - 2] = (byte) ((crc >> 8) & 0xFF);
content[len - 1] = (byte) (crc & 0xFF);
}
```

Common Commands:

a. "Sense mode" for common automatic scanning (default mode):

·"@SCNMOD2" sets the mode to "sense mode". The barcode scanner scans automatically in this mode.

·"@ORTSET$" sets waiting time. $ indicates time (ms). 1000 is recommended.

·"@RRDDUR$" sets the time interval between 2 scans of one barcode. $ indicates time (ms). 800~1000 is recommended.

·"@SENIST$" sets the time interval between scans of different barcodes. $ indicates time (ms). 200~400 is recommended. This command actually controls the time interval between senses. It also applies to the scans of one barcode when the value is larger than the value in "@RRDDUR$".

b. "Trigger mode" for payment acceptance use:

·"@SCNMOD0" sets the "trigger mode". In this mode, the barcode scanner is off by default, and the command "trigger a scan" below needs to be sent to trigger one scan.

·"#SCNTRG1" triggers one scan. Once a barcode has been scanned or in case a timeout, it will be turned off.

·"#SCNTRG0" turns off scanning.

·"@ORTSET$" sets waiting time. $ indicates time (ms). 60000 is recommended.

c. Other Common Commands:

·"@TSUENA1" activates suffix.

·"@TSUSET0D0A" sets the suffix as carriage return/line feed.

·"@GRBENA1" enables the buzzer.