

Honeywell

PC43K Kiosk

User Development Guide

Disclaimer

Honeywell International Inc. (“HII”) reserves the right to make changes in specifications and other information contained in this document without prior notice, and the reader should in all cases consult HII to determine whether any such changes have been made. The information in this publication does not represent a commitment on the part of HII.

HII shall not be liable for technical or editorial errors or omissions contained herein; nor for incidental or consequential damages resulting from the furnishing, performance, or use of this material. HII disclaims all responsibility for the selection and use of software and/or hardware to achieve intended results.

This document contains proprietary information that is protected by copyright. All rights are reserved. No part of this document may be photocopied, reproduced, or translated into another language without the prior written consent of HII.

Copyright © 2018-2020 Honeywell International Inc. All rights reserved.

Web Address: www.honeywellaidc.com

Trademarks

Google, Android, Google Play, Google Pay and other marks are trademarks of Google LLC

Bluetooth trademarks are owned by Bluetooth SIG, Inc., U.S.A. and licensed to Honeywell.

microSD is a registered trademark of SD-3C, LLC.

Qualcomm and Snapdragon are registered trademarks or trademarks of Qualcomm Incorporated in the United States and/or other countries.

Other product names or marks mentioned in this document may be trademarks or registered trademarks of other companies and are the property of their respective owners.

Patents

For patent information, refer to www.hsmpats.com.

TABLE OF CONTENTS

| | |
|-----------------------------------------------------------------------------|----------|
| Customer Support | v |
| Technical Assistance | v |
| Product Service and Repair | v |
| Limited Warranty | v |
| Chapter 1 - Overview | 1 |
| Setup USB Devices | 1 |
| Enumerate Devices | 2 |
| Enumerate Printer | 2 |
| Initialize Printer | 3 |
| Print label | 3 |
| Configure Scanner | 6 |
| Configuring Scanner using Commands | 6 |
| Configuring Scanner using Barcodes | 8 |
| Chapter 2 - How to suppress pop-up to allow printer USB access | 9 |
| Introduction | 9 |
| Steps to suppress USB permission dialog with Honeywell Sign | 9 |

Customer Support

Technical Assistance

To search our knowledge base for a solution or to log in to the Technical Support portal and report a problem, go to www.hsmcontactsupport.com.

For our latest contact information, see www.honeywellaidc.com/locations.

Product Service and Repair

Honeywell International Inc. provides service for all of its products through service centers throughout the world. To find your service center, go to www.honeywellaidc.com and select Support. Contact your service center to obtain a Return Material Authorization number (RMA #) before you return the product.

To obtain warranty or non-warranty service, return your product to Honeywell (postage paid) with a copy of the dated purchase record. To learn more, go to www.honeywellaidc.com and select **Service & Repair** at the bottom of the page.

For ongoing and future product quality improvement initiatives, the mobile computer comes equipped with an embedded device lifetime counter function. Honeywell may use the lifetime counter data for future statistical reliability analysis as well as ongoing quality, repair and service purposes.

Limited Warranty

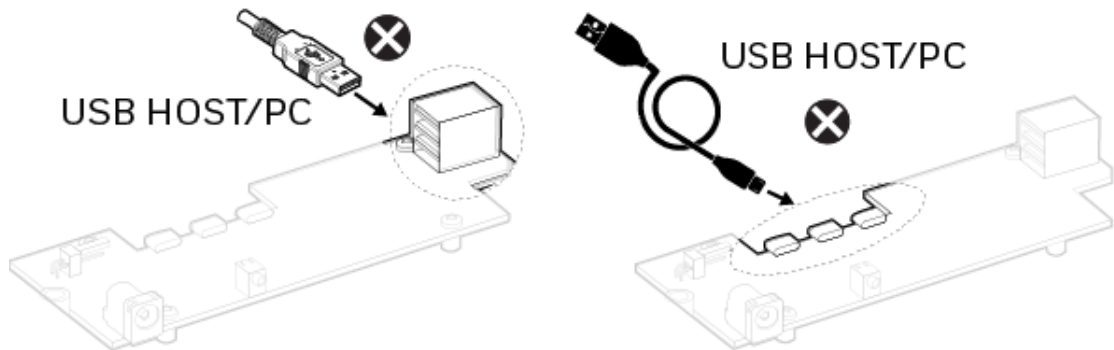
For warranty information, go to www.honeywellaidc.com and click **Resources > Product Warranty**.

This chapter details how to connect USB devices of the PC43K Kiosk to the android system and set-up communication with the USB devices. When the Android-powered device is in USB host mode, it acts as host, powers the bus, and enumerates connected USB devices.

The android powered EDA71 tablet acts as host to the USB devices (Printer and Scanner) in the PC43K Kiosk.



Caution: Do not connect a host, for example a PC or a laptop running on any OS, to any of the USB device ports. Voltage inrush from both hosts would damage the USB Hub of the Kiosk.



Setup USB Devices

When you connect USB devices to an Android-powered device, the Android system can determine whether your application is interested in the connected device. If so, you can set up communication with the device. To do this, your application must:

- Discover connected USB devices by using an intent filter to be notified when the user connects a USB device or by enumerating USB devices that are already connected.
- Ask the user for permission to connect to the USB device, if not already obtained.

- Communicate with the USB device by reading and writing data on the appropriate interface endpoints.

Please refer to the <https://developer.android.com/guide/topics/connectivity/usb/host#java> to implement your application for USB development.

Following sections, we will detail how to enumerate the devices with condition, how to Initialize printer before printing label and how to print label.

Enumerate Devices

If you want to enumerate all USB devices currently connected while your application is running. Please refer to the Enumerate devices section in following link: <https://developer.android.com/guide/topics/connectivity/usb/host#java>.

We need to enumerate either printer or scanner or both, to achieve this goal, please use product id as condition to filter out the USB devices that the app want to connect.

Enumerate Printer

The below table displays the product id of the printer.

| USB device type | Product Id |
|-----------------|---------------|
| Printer | 0x59 and 0x3F |

The sample code fragment as following:

```
if((device.getProductId() == 0x59) || (device.getProductId() == 0x3F))// printer
{

}

}
```

Note: The product id's represent different modes of the printer.

Note: All the programming follows Android USB host programming. For more information, refer to the <https://developer.android.com/guide/topics/connectivity/usb/host#java>.

Initialize Printer

Before printing the label, use the following commands to initialize the printer and check the status of the printer. User can write the following commands with android USB API to printer. Please also refer to <https://developer.android.com/guide/topics/connectivity/usb/host#java> to find how to write and read data on the appropriate interface endpoints.

Note: The commands may differ based on the selected printer language.

| Commands | Comments |
|------------|----------------------------------------------------------------------------------|
| VERBON | Specifies the verbosity level of the communication from the printer |
| INPUT OFF | Disables the Intermec Direct Protocol |
| SYSVAR(21) | Read the printhead density, expressed as number of dots per millimeter |
| ?PRSTAT | Return the current printer status or the current position of the insertion point |

For how to parse the **PRSTAT** command return value to check the printer status and how to parse the **SYSVAR** command return value to check the printhead density. Please refer the Connect id demo code.

Print label

The sample code commands in the below table uses Honeywell Fingerprint language. The commands include setting up the printer and printing the label.

Note: The printer supports multiple printer languages. For more information on the commands, refer to the respective printer command reference manual.

| Commands | Comments |
|-------------------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| CLIP | Enables or disables the printing of partial fields |
| CLIP BARCODE | CLIP BARCODE [HEIGHT INFORMATION X Y] is used for bar code fields only. Note that some bar codes, like Maxicode, consist of images and should in this context be regarded as image fields. |
| LBLCOND | Overrides the media feed setup |
| OPTIMIZE "BATCH" ON/OFF | Enables or disables optimization for batch printing. |
| CLL | Partial or complete clearing of the print image buffer |
| ALIGN | Specifies which part (anchor point) of a text field, bar code field, image field, line, or box will be positioned at the insertion point. |
| PP | Specifies the insertion point for a line of text, a bar code, an image, a box, or a line |
| FONTSIZE | The height of the font is given in points |
| NASC | Selects a single-byte character set, alternatively the multi-byte character set UTF-8 |
| PT | Prints data to the standard OUT channel. |

| Commands | Comments |
|-----------|--------------------------------------------------------------------------------------------------------------------------------------|
| FT | Selects a scalable TrueType or TrueDoc single-byte font, or a single-byte bitmap font, for printing the subsequent PRTXT statements. |
| FONTSLANT | Clockwise slant in degrees. Range is 0 (default) to 90. Does not work with bitmap fonts. |
| PRPOS | Specifies the insertion point for a line of text, a bar code, an image, a box, or a line. |
| BARSET | Specifies a bar code and sets additional parameters for complex bar codes. |
| PRBAR | Provides input data to a bar code. |
| BARFONT | Specifies fonts for the printing of bar code interpretation. |
| PRIMAGE | Selects an image stored in the printer's memory. |

Sample Code

The sample code for the print label used in the PC43K - Demo application is given below for your reference. You can find details on Connect id demo source code.

```
String strImage = "SYSVAR(43)=1\n" +
    "IMAGE LOAD \"BADGE300.PNG\",179504,\"\"\n";
byte[] testBytes = strImage.getBytes();

USBDeviceHandler.getInstance().sendRawData(testBytes);

WritePngFileDPI300();

String str1 = "CLIP ON\n" +
    "CLIP BARCODE ON\n" +
    "LBLCOND 3,2\n" +
    "CLL\n" +
    "OPTIMIZE \"BATCH\" OFF\n";

String str2 = "ALIGN 8\n" +
    "PP 600,1716\n" +
    "DIR 1\n" +
    "PRIMAGE \"BADGE300.PNG\" \n";
String str3 = "NASC 8\n" +
```

```
"FT \"Univers Concensed Bold\" \"\n" +  
"FONTSIZE 26\n" +  
"FONTSLANT 0\n" +  
"PP 600,1110\n" +  
"PT " + "\" + strFirstName + "\" + \"\n" +  
"FONTSIZE 18\n" +  
"PP 600,960\n" +  
"PT " + "\" + strLastName + "\" + \"\n";
```

```
String str4 = "FT \"Andale Mono\" \"\n" +  
"FONTSIZE 14\n" +  
"FONTSLANT 0\n" +  
"PP 600,750\n" +  
"PT " + "\" + strOrg + "\" + \"\n";
```

```
String str5 = "FT \"Andale Mono Bold\" \"\n" +  
"FONTSIZE 20\n" +  
"FONTSLANT 0\n" +  
"PP 600,690\n" +  
"PT " + "\" + strCountry + "\" + \"\n";
```

```
String str6 =  
"PRPOS 600,555\n" +  
"BARSET \"QRCODE\",1,1,12,2,4" + \"\n" +  
"BARFONT OFF\n" +  
"PRBAR" + "\" + strCode + "\" + \"\n";
```

```
String strText = "Monospace 821 Bold";
```

```
String str7 =  
"FT \"Monospace 821 Bold\" \"\n" +  
"FONTSIZE 20\n" +  
"FONTSLANT 0\n" +
```

```
"PP 600,150\n" +  
"PT \\"DELEGATE\\"\n";
```

```
String str8 = "PF\n";  
String str = str1 + str2 + str3 + str4 + str5 + str6 + str7 + str8;  
testBytes = str.getBytes();  
USBDeviceHandler.getInstance().sendRawData(testBytes);
```

Configure Scanner

The scan engine can be configured either by scanning configuration barcodes or by inputting commands in the application.

Configuring Scanner using Commands

The code fragments to make Scanner to scan barcode and stop scanning by broadcast.

Claim scanner

Following is the fragment of claim Scanner and is call on Activity.onResume(),for details you can refer connect id demo source code.

```
Bundle properties = new Bundle();  
    properties.putBoolean("DPR_DATA_INTENT", true);  
    properties.putString("DPR_DATA_INTENT_ACTION", ACTION_BARCODE_DATA);  
    properties.putString("DPR_DATA_INTENT_CATEGORY",  
Intent.CATEGORY_DEFAULT);  
    properties.putString("DPR_DATA_INTENT_PACKAGE_NAME",  
        ExplicitReceiver.class.getPackage().getName());  
    properties.putString("DPR_DATA_INTENT_CLASS_NAME",  
        ExplicitReceiver.class.getName());  
  
sendExplicitBroadcast(instance,  
new Intent("com.honeywell.aidc.action.ACTION_CLAIM_SCANNER")  
    .putExtra("com.honeywell.aidc.extra.EXTRA_SCANNER", "dcs.scanner.ring")  
    .putExtra("com.honeywell.aidc.extra.EXTRA_PROFILE", "MyProfile1")
```

```
.putExtra("com.honeywell.aidc.extra.EXTRA_PROPERTIES", properties));
```

Note: *ExplicitReceiver is a BroadcastReceiver to receive and handle decode intent from DCS; instance is the context for current Activity.*

Start to Scan

Following is the fragment of how to make Scanner to scan, for details you can refer connect id demo source code.

```
Intent controllIntent =  
new Intent("com.honeywell.aidc.action.ACTION_CONTROL_SCANNER");  
controllIntent.putExtra("com.honeywell.aidc.extra.EXTRA_SCAN",true);  
sendExplicitBroadcast(instance,controllIntent);
```

Stop scanning

Following is the fragment of how to stop scanning, for details you can refer connect id demo source code.

```
Intent controllIntent =  
new Intent("com.honeywell.aidc.action.ACTION_CONTROL_SCANNER");  
controllIntent.putExtra("com.honeywell.aidc.extra.EXTRA_SCAN",false);  
sendExplicitBroadcast(instance,controllIntent);
```

Handle barcode intent

Following is the fragment of handle barcode intent from ExplicitReceiver, for details you can refer connect id demo source code.

```
if ("com.honeywell.sample.action.ACTION_BARCODE_DATA"  
.equals(intent.getAction()))  
String data = intent.getStringExtra("data");
```

Note: *Intent is a broadcast from DCS layer after scanning a barcode;*

data is the decode result string.

When Scanner has successfully scan a barcode, DCS will prompt beep to indicate. And there is no need for App layer to do such thing again

Claim scanner

Following is the fragment of claim Scanner and is call on Activity.onPause (), for details you can refer connect id demo source code





```
sendExplicitBroadcast(instance,
```

```
new Intent("com.honeywell.aidc.action.ACTION_RELEASE_SCANNER"));
```

Configuring Scanner using Barcodes

You can create a set of menu commands as your own, custom defaults. To do so, scan the Set Custom Defaults bar code below before scanning the menu commands for your custom defaults. If a menu command requires scanning numeric codes from the back cover, then Save code, that entire sequence will be saved to your custom defaults. When you have entered all the commands you want to save for your custom defaults, scan the Save Custom Defaults bar code.

Note: For more information on the configuration code, refer the N3680 User Guide.

| Barcode | Comments |
|-------------------------------------------------------------------------------------------------|----------------------------------------------|
|  MNUCDP. | To set custom default |
|  MNUCDS. | To save custom default |
|  ALLENA1. | To enable all symbologies |
|  DEFAULT. | To reset the scan engine to factory default. |

HOW TO SUPPRESS POP-UP TO ALLOW PRINTER USB ACCESS

Introduction

When users connect a device that matches your device filter, the system presents them with a dialog that asks if they want to start your application. If users accept, your application automatically has permission to access the device until the device is disconnected. If you also check box to remember device, permission is granted until app uninstalls, or data is deleted through app manager. Normal application doesn't have permission to suppress it, but Honeywell is the vendor of this product, after sign with system key, application can suppress this dialog.

For details of how to suppress pop-up USB access, please refer to the Connect id demo source code.

Steps to suppress USB permission dialog with Honeywell Sign

To suppress USB permission dialog with Honeywell sign,

1. Call UsbManager add permission using reflection

```
public boolean grantAutomaticPermission(UsbDevice usbDevice)
{
    try
    {
        PackageManager pkgManager=mContext.getPackageManager();
        ApplicationInfo
        appInfo=pkgManager.getApplicationInfo(mContext.getPackageName(),
        PackageManager.GET_META_DATA);
```

```

        Class serviceManagerClass=Class.forName("android.os.ServiceManager");
        Method
getServiceMethod=serviceManagerClass.getDeclaredMethod("getService",String.
class);
        getServiceMethod.setAccessible(true);
        android.os.IBinder binder=(android.os.IBinder)getServiceMethod.invoke(null,
Context.USB_SERVICE);

        Class
iUsbManagerClass=Class.forName("android.hardware.usb.IUsbManager");
        Class stubClass=Class.forName("android.hardware.usb.IUsbManager$Stub");
        Method asInterfaceMethod=stubClass.getDeclaredMethod("asInterface",
android.os.IBinder.class);
        asInterfaceMethod.setAccessible(true);
        Object iUsbManager=asInterfaceMethod.invoke(null, binder);

        System.out.println("UID : " + appInfo.uid + " " + appInfo.processName + " " +
appInfo.permission);
        final Method grantDevicePermissionMethod =
iUsbManagerClass.getDeclaredMethod("grantDevicePermission",
UsbDevice.class,int.class);
        grantDevicePermissionMethod.setAccessible(true);
        grantDevicePermissionMethod.invoke(iUsbManager, usbDevice,appInfo.uid);

        System.out.println("Method OK : " + binder + " " + iUsbManager);
        return true;
    }
    catch(Exception e)
    {
        System.err.println("Error trying to assing automatic usb permission : ");
        e.printStackTrace();
        return false;
    }
}

```


2. Add share user id string in application manifest file

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    android:sharedUserId="android.uid.systemui"
    package="com.honeywell.connectiddemo">
```

3. Build APK, send apk file to Honeywell for sign.

Honeywell
9680 Old Bailes Road
Fort Mill, SC 29707

www.honeywellaidc.com