01.代码结构

02.代码逻辑

03.APK中无法连接到这个扫码器。

- 问题描述：APK中无法连接到这个扫码器。
- 日志如下：
  - 

1.1 查看系统日志并进行问题解决

- 扫码转成USB HID模式，确保是HID，而非键盘输入模式
- 查看系统日志，确保成功连接到Android pos机中。
- adb命令：
  - ```
    adb devices
    adb logcat | grep -i usb
    adb logcat | grep UsbHidManager
    ```
- 日志如下
  - ```
    05-29 14:41:00.012    61    61 I usb 1-1.1: new full-speed USB device
    number 3 using ehci-platform

    05-29 14:41:00.113    61    61 I usb 1-1.1: New USB device found,
    idVendor=0218, idProduct=0210, bcdDevice= 0.00

    05-29 14:41:00.113    61    61 I usb 1-1.1: New USB device strings:
    Mfr=1, Product=2, SerialNumber=3

    05-29 14:41:00.113    61    61 I usb 1-1.1: Product: USBScn Module

    05-29 14:41:00.113    61    61 I usb 1-1.1: Manufacturer: USBScn Chip

    05-29 14:41:00.113    61    61 I usb 1-1.1: SerialNumber:
    2027300413413333
    ```

```
05-29 14:41:00.132    61    61 I input   : USBScn Chip USBScn Module as
/devices/platform/fed00000.usb/usb1/1-1/1-1.1/1-
1.1:1.0/0003:0218:0210.0001/input/input7

05-29 14:41:00.193    61    61 I hid-generic 0003: 0218:0210.0001:
input,hidraw0: USB HID v1.10 Keyboard [USBScn Chip USBScn Module] on
usb-fed00000.usb-1.1/input0

05-29 14:40:58.976   502   605 D EventHub: No input device
configuration file found for device 'USBScn Chip USBScn Module'.

05-29 14:40:58.985   502   605 I EventHub: New device: id=8, fd=517,
path='/dev/input/event7', name='USBScn Chip USBScn Module',
classes=KEYBOARD | ALPHAKEY | EXTERNAL, configuration='',
keyLayout='/system/usr/keylayout/Generic.kl',
keyCharacterMap='/system/usr/keychars/Generic.kcm',
builtinKeyboard=false,

05-29 14:40:58.987   502   605 I InputReader: Device added: id=6,
eventHubId=8, name='USBScn Chip USBScn Module',
descriptor='4cc968938b847d3dcaa32261c1d126231806d979',sources=KEYBOARD
```

    - 扫码枪设备信息
    - 硬件识别
        - 厂商ID : 0x0218 (536)
        - 产品ID : 0x0210 (528)
        - 制造商 : USBScn Chip
        - 产品名 : USBScn Module
        - 序列号 : 2027300413413333
    - 设备类型确认
        - 从日志可以看出，扫码枪被识别为：
        - USB HID设备 : USB HID v1.10 Keyboard
        - 键盘设备 : classes=KEYBOARD | ALPHAKEY | EXTERNAL
        - 输入设备 : 映射到 /dev/input/event7
- 为什么我的UsbHidManager检测不到设备
    - 我的 `UsbHidManager.kt` 中的 SUPPORTED_SCANNERS 列表没有包含这个厂商ID：
    - ```
      private val SUPPORTED_SCANNERS = mapOf(
          0x05e0 to "Symbol/Zebra",
          0x0536 to "Hand Held Products",
          0x0c2e to "Metrologic",
          0x1a86 to "QinHeng Electronics",
          0x0483 to "STMicroelectronics",
          0x04b4 to "Cypress"
          // 缺少 0x0218 -> "USBScn Chip"
          0x0218 to "USBScn Chip"  // 添加你的扫码枪
      )
      ```
- 重编译，仍然不行。
- 再去查看
    - 设备过滤器配置不完整 。
    - 需要在`device_filter.xml` 中添加了缺失的vendor ID：
      ```

- ```
  <!-- USBScn Chip扫码枪 -->
  <usb-device vendor-id="536" />   <!-- 0x0218 -->
  ```
  - 设备过滤器作用： Android系统通过device_filter.xml来决定哪些USB设备可以被应用访问
- 仍然不行。

1.2 最小化的USB设备检测代码

- MyApplication中添加debugUsbDevices方法，直接使用系统USB管理器来检测和列出所有USB设备
  - ```
    /**
       * 简化的USB设备检测调试方法
       */
      private fun debugUsbDevices() {
          try {
              val usbManager = getSystemService(Context.USB_SERVICE) as
    UsbManager
              val deviceList = usbManager.deviceList

              Log.d("USB_DEBUG", "=== USB设备检测开始 ===")
              Log.d("USB_DEBUG", "检测到USB设备数量: ${deviceList.size}")

              if (deviceList.isEmpty()) {
                  Log.w("USB_DEBUG", "未检测到任何USB设备")

                  // 检查USB Host支持
                  val packageManager = packageManager
                  val hasUsbHost =
    packageManager.hasSystemFeature("android.hardware.usb.host")
                  Log.i("USB_DEBUG", "USB Host模式支持: $hasUsbHost")
                  return
              }

              // 已知的扫码枪厂商ID
              val knownScanners = mapOf(
                  0x05e0 to "Symbol/Zebra",
                  0x0536 to "Hand Held Products",
                  0x0c2e to "Metrologic",
                  0x1a86 to "QinHeng Electronics",
                  0x0483 to "STMicroelectronics",
                  0x04b4 to "Cypress",
                  0x0218 to "USBScn Chip"  // 目标设备
              )

              // 遍历所有USB设备
              for ((name, device) in deviceList) {
                  Log.d("USB_DEBUG", "\n=== USB设备详情 ===")
                  Log.d("USB_DEBUG", "设备名称: $name")
                  Log.d("USB_DEBUG", "设备ID: ${device.deviceId}")
                  Log.d("USB_DEBUG", "厂商ID: ${device.vendorId}
    (0x${device.vendorId.toString(16)})")
                  Log.d("USB_DEBUG", "产品ID: ${device.productId}
    (0x${device.productId.toString(16)})")
                  Log.d("USB_DEBUG", "设备类: ${device.deviceClass}")
                  Log.d("USB_DEBUG", "设备子类: ${device.deviceSubclass}")
    ```

```
                Log.d("USB_DEBUG", "设备协议: ${device.deviceProtocol}")
                Log.d("USB_DEBUG", "接口数量: ${device.interfaceCount}")

                // 检查是否为已知扫码枪
                val vendorName = knownScanners[device.vendorId]
                if (vendorName != null) {
                    Log.i("USB_DEBUG", "★★★ 发现已知扫码枪厂商:
$vendorName ★★★")
                }

                // 检查每个接口
                for (i in 0 until device.interfaceCount) {
                    val intf = device.getInterface(i)
                    Log.d("USB_DEBUG", "接口$i:")
                    Log.d("USB_DEBUG", "  - 类: ${intf.interfaceClass}
(${getUsbClassDescription(intf.interfaceClass)})")
                    Log.d("USB_DEBUG", "  - 子类:
${intf.interfaceSubclass}")
                    Log.d("USB_DEBUG", "  - 协议:
${intf.interfaceProtocol}")
                    Log.d("USB_DEBUG", "  - 端点数量:
${intf.endpointCount}")

                    // 特别标注HID接口
                    if (intf.interfaceClass ==
UsbConstants.USB_CLASS_HID) {
                        Log.i("USB_DEBUG", "  ★★★ HID接口发现! ★★★")

                        // 列出端点信息
                        for (j in 0 until intf.endpointCount) {
                            val endpoint = intf.getEndpoint(j)
                            val direction = if (endpoint.direction ==
UsbConstants.USB_DIR_IN) "IN" else "OUT"
                            val type = when (endpoint.type) {
                                UsbConstants.USB_ENDPOINT_XFER_CONTROL
-> "CONTROL"
                                UsbConstants.USB_ENDPOINT_XFER_ISOC ->
"ISOC"
                                UsbConstants.USB_ENDPOINT_XFER_BULK ->
"BULK"
                                UsbConstants.USB_ENDPOINT_XFER_INT ->
"INTERRUPT"
                                else -> "UNKNOWN"
                            }
                            Log.d("USB_DEBUG", "    端点$j: 地址
=0x${endpoint.address.toString(16)}, 方向=$direction, 类型=$type")
                        }
                    }
                }

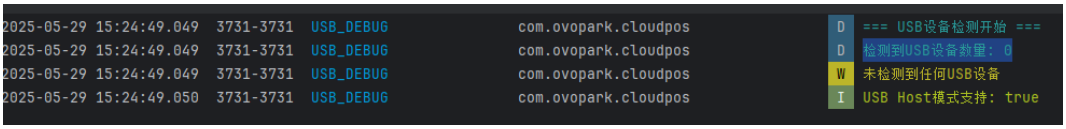
                // 检查权限
                val hasPermission = usbManager.hasPermission(device)
                Log.d("USB_DEBUG", "设备权限: $hasPermission")

                Log.d("USB_DEBUG", "==================")
            }

            Log.d("USB_DEBUG", "=== USB设备检测结束 ===")
```

```
        } catch (e: Exception) {
            Log.e("USB_DEBUG", "USB设备检测异常: ${e.message}", e)
        }
    }

    /**
     * 获取USB类描述
     */
    private fun getUsbClassDescription(usbClass: Int): String {
        return when (usbClass) {
            UsbConstants.USB_CLASS_APP_SPEC -> "Application Specific"
            UsbConstants.USB_CLASS_AUDIO -> "Audio"
            UsbConstants.USB_CLASS_CDC_DATA -> "CDC Data"
            UsbConstants.USB_CLASS_COMM -> "Communication"
            UsbConstants.USB_CLASS_CONTENT_SEC -> "Content Security"
            UsbConstants.USB_CLASS_CSCID -> "Smart Card"
            UsbConstants.USB_CLASS_HID -> "HID (Human Interface
Device)"
            UsbConstants.USB_CLASS_HUB -> "Hub"
            UsbConstants.USB_CLASS_MASS_STORAGE -> "Mass Storage"
            UsbConstants.USB_CLASS_MISC -> "Miscellaneous"
            UsbConstants.USB_CLASS_PER_INTERFACE -> "Per Interface"
            UsbConstants.USB_CLASS_PHYSICAL -> "Physical"
            UsbConstants.USB_CLASS_PRINTER -> "Printer"
            UsbConstants.USB_CLASS_STILL_IMAGE -> "Still Image"
            UsbConstants.USB_CLASS_VENDOR_SPEC -> "Vendor Specific"
            UsbConstants.USB_CLASS_VIDEO -> "Video"
            UsbConstants.USB_CLASS_WIRELESS_CONTROLLER -> "Wireless
Controller"
            else -> "Unknown ($usbClass)"
        }
    }
```

- 日志如下:
  - 

- 使用 adb shell dumpsys usb 命令查看系统级USB设备信息
  - ```
    panruiqi@SZ-PC-PDC-1204 MINGW64 /d/Develop/Android/CloudPos (master)
    $ adb shell dumpsys usb
    USB MANAGER STATE (dumpsys usb):
    {
      device_manager={
        handler={
          current_functions_applied=true
          screen_locked=false
          connected=true
          configured=true
          host_connected=false
          source_power=false
          sink_power=false
          usb_charging=false
          hide_usb_notification=false
          audio_accessory_connected=false
    ```
    ```

```
      kernel_state=CONFIGURED
    }
    USB Event Log=UsbDeviceManager activity
    USB Event=[
      05-29 14:35:30.224 USB intent: Intent {
act=android.hardware.usb.action.USB_STATE flg=0x31000000 (has extras) }
      05-29 14:36:47.463 USB UEVENT: {SUBSYSTEM=android_usb,
SEQNUM=3512, ACTION=change, USB_STATE=DISCONNECTED,
DEVPATH=/devices/virtual/android_usb/android0}
      05-29 14:36:47.551 USB intent: Intent {
act=android.hardware.usb.action.USB_STATE flg=0x31000000 (has extras) }
      05-29 14:36:49.175 USB UEVENT: {SUBSYSTEM=android_usb,
SEQNUM=3514, ACTION=change, USB_STATE=CONNECTED,
DEVPATH=/devices/virtual/android_usb/android0}
      05-29 14:36:49.184 USB intent: Intent {
act=android.hardware.usb.action.USB_STATE flg=0x31000000 (has extras) }
      05-29 14:36:49.217 USB UEVENT: {SUBSYSTEM=android_usb,
SEQNUM=3515, ACTION=change, USB_STATE=CONFIGURED,
DEVPATH=/devices/virtual/android_usb/android0}
      05-29 14:36:49.221 USB intent: Intent {
act=android.hardware.usb.action.USB_STATE flg=0x31000000 (has extras) }
      05-29 14:39:51.523 USB UEVENT: {SUBSYSTEM=android_usb,
SEQNUM=3516, ACTION=change, USB_STATE=DISCONNECTED,
DEVPATH=/devices/virtual/android_usb/android0}
      05-29 14:39:54.528 USB intent: Intent {
act=android.hardware.usb.action.USB_STATE flg=0x31000000 (has extras) }
      05-29 14:40:06.408 USB UEVENT: {SUBSYSTEM=android_usb,
SEQNUM=3535, ACTION=change, USB_STATE=CONNECTED,
DEVPATH=/devices/virtual/android_usb/android0}
      05-29 14:40:06.420 USB UEVENT: {SUBSYSTEM=android_usb,
SEQNUM=3536, ACTION=change, USB_STATE=CONFIGURED,
DEVPATH=/devices/virtual/android_usb/android0}
      05-29 14:40:06.424 USB intent: Intent {
act=android.hardware.usb.action.USB_STATE flg=0x31000000 (has extras) }
      05-29 14:40:06.430 USB intent: Intent {
act=android.hardware.usb.action.USB_STATE flg=0x31000000 (has extras) }
    ]
  }
  host_manager={
    num_connects=0
  }
  port_manager={
    is_simulation_active=false
    usb_hal_version=20
  }
  alsa_manager={
    cards_parser=-1
  }
  settings_manager={
    user_settings={
      user_id=0
      device_attached_activities=[
        {
          activity={
            package_name=com.android.gallery3d
            class_name=com.android.gallery3d.ingest.IngestActivity
          }
          filters={
```

```
            vendor_id=-1
            product_id=-1
            class=6
            subclass=1
            protocol=1
            manufacturer_name=null
            product_name=null
            serial_number=null
          }
        }
        {
          activity={
            package_name=com.android.mtp
            class_name=com.android.mtp.ReceiverActivity
          }
          filters=[
            {
              vendor_id=-1
              product_id=-1
              class=255
              subclass=255
              protocol=0
              manufacturer_name=null
              product_name=null
              serial_number=null
            }
            {
              vendor_id=-1
              product_id=-1
              class=6
              subclass=1
              protocol=1
              manufacturer_name=null
              product_name=null
              serial_number=null
            }
          ]
        }
        {
          activity={
            package_name=com.ovopark.cloudpos
            class_name=com.ovopark.cloudpos.ui.splash.SplashActivity
          }
          filters=[
            {
              vendor_id=-1
              product_id=-1
              class=3
              subclass=-1
              protocol=-1
              manufacturer_name=null
              product_name=null
              serial_number=null
            }
            {
              vendor_id=-1
              product_id=-1
              class=8
```

```
            subclass=-1
            protocol=-1
            manufacturer_name=null
            product_name=null
            serial_number=null
          }
          {
            vendor_id=-1
            product_id=-1
            class=2
            subclass=-1
            protocol=-1
            manufacturer_name=null
            product_name=null
            serial_number=null
          }
          {
            vendor_id=-1
            product_id=-1
            class=-1
            subclass=-1
            protocol=-1
            manufacturer_name=null
            product_name=null
            serial_number=null
          }
        ]
      }
      {
        activity={
          package_name=com.ovopark.cloudpos
          class_name=com.ovopark.cloudpos.ui.main.MainActivity
        }
        filters=[
          {
            vendor_id=-1
            product_id=-1
            class=3
            subclass=-1
            protocol=-1
            manufacturer_name=null
            product_name=null
            serial_number=null
          }
          {
            vendor_id=-1
            product_id=-1
            class=8
            subclass=-1
            protocol=-1
            manufacturer_name=null
            product_name=null
            serial_number=null
          }
          {
            vendor_id=-1
            product_id=-1
            class=2
```

```
              subclass=-1
              protocol=-1
              manufacturer_name=null
              product_name=null
              serial_number=null
            }
            {
              vendor_id=-1
              product_id=-1
              class=-1
              subclass=-1
              protocol=-1
              manufacturer_name=null
              product_name=null
              serial_number=null
            }
          ]
        }
      ]
    }
    profile_group_settings={
      parent_user_id=0
      USB Event Log=UsbProfileGroupSettingsManager activity
    }
  }
  permissions_manager={
    user_permissions={
      user_id=0
    }
  }
}
```

- 从 dumpsys usb 的输出可以看到：
  - USB Host连接状态：
  - ```
    host_connected=false
    host_manager={ num_connects=0 }
    ```
  - 这表明设备当前没有作为USB Host连接任何外部USB设备。
    - 应用已注册USB设备过滤器：
      可以看到 com.ovopark.cloudpos 应用已经正确注册了USB设备过滤器，包括：
      - class=3 (HID设备)
      - class=8 (Mass Storage)
      - class=2 (Communication)
      - class=-1 (所有设备)
  - USB连接记录：
    日志显示的都是 android_usb 相关的事件，这些是设备作为USB从设备（连接到电脑）的状态变化，而不是外部USB设备连接到Android设备的记录。
- 根本原因：
  - USB扫码枪没有物理连接到Android设备 ， 或者连接后没有被系统识别。关键证据：
  - host_connected=false - 没有USB Host连接
  - num_connects=0 - USB Host管理器显示0个连接
  - 没有任何外部USB设备的连接/断开事件记录
    ```