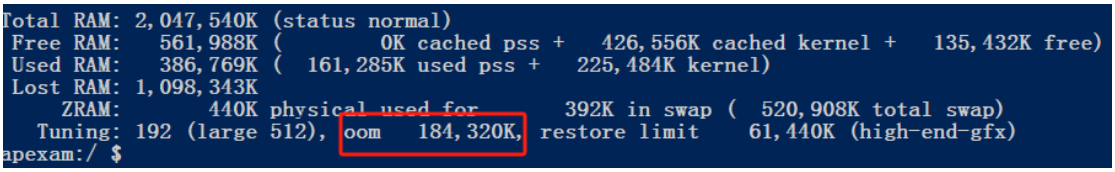
RB-A3288FSL APK 重启问题分析报告（二次）

研发部：凌周，洪丰

1、上电后进入系统，自启动 ipos order contrlo apk,静置观察，没有复现现象；

2、客户问题视频在关闭 WiFi 的情况下测试，但目前有无连接 VPN 网络，均没有复现现象；

3、系统 OOM 参数设为 184,320K，当系统内存低于 184,320K 时，系统可能会开始关闭占用内存过高的进程。ipos apk 有可能在空余内存偏低的状况下被关闭。令系统内存被占用到低于 184,320K，观察一段时间，没有复现现象。内存低于 OOM 预设时，看到 swap 分区被调用；（内部 3288fsl 的主板+早期客户的 BSP，问题 BSP 版本待验证）；



4、客户几个 BSP adb 内存信息与 setting 内的内存信息不对等；带有 GMS 包的 BSP 内存占用常规都在 1.5G 以上，对于 2G 内存的主板，显然空间紧凑；

5、监视 ipos apk 每次启动时会调用到 com.google.android.packageinstaller 包安装器，相关原因有待研究。

6、当打开 com.ipos.controllerorder 这个应用并且观察到 com.google.android.packageinstaller 被调用时，可能意味着 com.ipos.controllerorder 正在尝试安装或更新另一个应用。这是通过 Android 设备上提供的系统包安装器来完成的。com.google.android.packageinstaller 负责管理 Android 设备上的应用安装和更新。当像 com.ipos.controllerorder 这样的应用请求安装或更新另一个应用时，包安装器组件就会

处理这个过程，包括权限检查、兼容性检查以及用户交互等。因此，没有看到明显的安装提示，但是发现这些组件之间的这种交互，可能是因为 com.ipos.controllerorder 在后台自动尝试进行某些应用的更新或安装操作。看来与 reboot 问题没有关联。

7、内存泄露的可能性

首先从最初的客户提供的 dump 内存信息上观察，发现内存状态异常，如下：

```
Total RAM:         0K (status normal)
 Free RAM:         0K (        0K cached pss +         0K cached kernel +         0K free)
 Used RAM:    87,193K (   87,193K used pss +         0K kernel)
 Lost RAM:   -87,193K
   Tuning: 192 (large 512), oom   184,320K, restore limit    61,440K (high-end-gfx)
```

正常情况下，我们应该能看到总内存数、可用内存和已使用内存的具体数值，而不是全为 0K。这样的输出不符合正常的系统状态，确实可能与内存泄露有关，尤其是当系统的 Home 键和 Recent 键功能失效时。内存泄露可能导致系统资源耗尽，进而影响到系统的稳定性和响应能力。长时间运行内存泄露的应用程序会逐渐消耗掉更多的内存，直到没有足够的资源供其他应用或系统进程使用，导致系统状态出现 issue。我们通过 log 观察，发现在 Home 键和 Recent 键失效的情况下按下 home 键，出现如下 log:

```
04-26 09:36:51.950   705   741 I WindowManager: Not starting activity because user setup is in progress: Intent { act=an
droid.intent.action.MAIN cat=[android.intent.category.HOME] flg=0x10200000 (has extras) }
```

04-26 09:36:51.950 705 741 I WindowManager: Not starting activity because user setup is in progress: Intent {act=android.intent.action.MAINcat=[android.intent.category.HOME] flg=0x10200000 (has extras) }

这表明系统在尝试启动主屏幕（HOME）时因为设备的设置（用户设置）尚未完成而终止了操作。使用命令查看数据库设置的值：

settings get secure user_setup_complete

发现结果是 0

```
apexam:/ $ settings get secure user_setup_complete
0
apexam:/ $
```

cat /data/system/users/0/settings_secure.xml | grep user_setup

settings_secure.xml 需要 root 权限才可查看，找到公司内部状态正常的主板，可以看到输出结果：

```
rk3288:/ # cat /data/system/users/0/settings_secure.xml | grep user_setup
  <setting id="93" name="user_setup_complete" value="1" package="root" />
rk3288:/ #
```

可以看到，常规状态下，user_setup_complete 的值在设备启动的首次设置过程完成后被设置为 1，以标记用户完成了设备的初始设置，使得所有功能都可以正常使用。在正常的主板上用命令将该值手动设置为 0，模拟问题整机主板出现的状态，发现也出现了 home 和 recents 功能异常的问题。反过来，在客户的主板上将 user_setup_complete 设置为 1

settings put secure user_setup_complete 1

发现客户主板 home 键功能恢复，顶部状态栏在 ipos app 打开的状态下也可以正常使用。不过 recents

按键依旧失效，应该在其它地方有些问题，只是 recents 键没有 log 输出，无法深究。到这里可以看出，user_setup_complete 被设置为 0 显然是导致 Home 键和 Recent 键以及状态栏功能异常的主要原因。设备可能因为一个系统层面的错误或 bug，在更新、重启或因其他原因过程中将此标志重置。或者某个应用或服务可能错误地修改了这一设置，尽管这听起来比较少见，但并不排除应用在内存泄露的情况下发生相关 bug。当然，这个不是我们当前主要探讨的 issue，只是说这与内存泄露可能有较多的关联性。

回到最初的问题，从客户主板运行 ipos order control apk 出现了 reboot 现象。注意，这里应该是 apk reboot，而非系统 reboot。这个现象更像是 app 被终止了，只是 ipos apk 本身自带了自启动。所以主要问题应该是什么原因导致 apk 的进程被终止掉。回顾前面提到的，客户的 BSP 本身带了 GMS 包，我们简单看一下内存信息：

```
apexam:/ $ free -h
              total        used        free      shared     buffers
Mem:           1.9G        1.3G        628M         52M         1.6M
-/+ buffers/cache:         1.3G        630M
Swap:          509M           0        509M
apexam:/ $ ps
```

而 apk 跑起来后如下所示

```
apexam:/ $ free -h
              total        used        free      shared     buffers
Mem:           1.9G        1.7G        252M         90M         1.9M
-/+ buffers/cache:         1.7G        254M
Swap:          509M           0        509M
```

在没有 GMS 的 BSP 中，ipos 运行不会占用过大的内存，但在 google 服务下运行的 ipos 会同时于后台启用部分的 GMS 服务，导致内存被迅速占用，进入一个紧凑的状态。如果我们继续压低内存，例如多开几个应用，到达一定的低内存状态后系统则会开始调用 swap 分区，这表明内存压力较大，因为交换空间的读写速度远低于物理内存，会进一步减慢系统性能。

```
apexam:/ $ free -h
              total        used        free      shared     buffers
Mem:           1.9G        1.8G         97M        160M         416K
-/+ buffers/cache:         1.8G         97M
Swap:          509M         61M        448M
```

在这种情况下，可能导致系统为了释放更多内存而进行频繁的垃圾回收（GC）操作，影响应用和系统的响应速度。这时我们用 dumpsys meminfo 观察，发现 free RAM 的值会再低于 200M 的情况下出现回升，并且保持一定范围的波动。

```
Total RAM: 2,047,540K (status normal)
 Free RAM:    156,908K (        0K cached pss +    74,112K cached kernel +    82,796K free)
 Used RAM:    462,875K (  144,819K used pss +   318,056K kernel)
 Lost RAM: 1,419,196K
     ZRAM:     16,508K physical used for    51,752K in swap (  520,908K total swap)
   Tuning: 192 (large 512), oom   184,320K, restore limit    61,440K (high-end-gfx)
```

```
Total RAM: 2,047,540K (status normal)
 Free RAM:    284,452K (        0K cached pss +   129,372K cached kernel +   155,080K free)
 Used RAM:    459,438K (  146,206K used pss +   313,232K kernel)
 Lost RAM: 1,292,621K
     ZRAM:     19,116K physical used for    57,876K in swap (  520,908K total swap)
   Tuning: 192 (large 512), oom   184,320K, restore limit    61,440K (high-end-gfx)
```

可以看到 lost ram 高达 1.2 个 G，而空余的空间很少。尝试把 ipos 关闭，发现从 lost ram 回收的内存不多，lost ram 仍保持一个较高的状态，说明系统中存在一定程度的内存泄露问题，即部分内存资源长期占用不释放，导致"Lost RAM"的累积。鉴于出现问题的场景为 ipos 持续使用的状态，有很大可能性为 com.ipos.controllerorder 本身存在内存泄露的问题，加之应用在内存仅为 2G 的低内存设备，在终端客户长周期运行的状态下一定的几率出现了系统崩溃现象，app 卡死闪退，导致客诉 issue。在

这种情况下，只有重刷系统才能恢复正常的状态；但是心病还需心药医，内存不足最根本还是要从内存入手，增大内存才是治标治本之法。

当然，以上仅为一个可能性较大的猜测。由于内存泄露需要通过专业工具库结合 ipos apk 源码进行分析，同时也需要具备一定的 Android 系统权限，条件限制下无法继续深究，故而此观点仅供保留参考，后续有待观察。

8、新的发现

回头查看客户提供 logcat 信息记录，发现部分 log 如下：

```
01-03 09:33:50.618 13088 13269 E GnpSdk  :       at com.google.android.gms.auth.m.g(PG:11)
01-03 09:33:50.618 13088 13269 E GnpSdk  :       ... 16 more
01-03 09:33:50.669 13593 13603 W art     : Suspending all threads took: 10.085ms
01-03 09:33:50.727   201   201 I lowmemorykiller: ActivityManager disconnected
01-03 09:33:50.727   201   201 I lowmemorykiller: Closing Activity Manager data connection
01-03 09:33:50.729 29763 29763 E          : eof
01-03 09:33:50.730 29763 29763 E          : failed to read size
01-03 09:33:50.730 29763 29763 I          : closing connection
01-03 09:33:50.730   202   202 I ServiceManager: service 'telecom' died
01-03 09:33:50.730   202   202 I ServiceManager: service 'notification' died
01-03 09:33:50.730   202   202 I ServiceManager: service 'devicestoragemonitor' died
01-03 09:33:50.730   202   202 I ServiceManager: service 'barcode' died
```

产生 error 之前系统出现了 lowmemorykiller 消息。lowmemorykiller 低内存管理器因为系统内存不足，需要释放资源，断开了活动管理器(Activity Manager)的连接，紧接着关闭与活动管理器的数据连接，随后陆续关闭各种服务，包括从通讯、壁纸、音频到设备管理等多个系统核心功能。低内存状态下系统出现了崩溃。顺着 ActivityManager 往下看，发现日志：

```
01-02 13:56:20.762  5146  5148 E sdcard  : [0] someone stole our marbles!
01-02 13:56:20.764  7549  7561 I ActivityManager: user 0 is still locked. Cannot load recents
01-02 13:56:20.769  7376  7638 D CommandListener: Clearing all IP addresses on wlan0
```

找到了导致任务键无法正常启用的原因：Android 系统中的主用户（User 0）尚未完全解锁，因此系统无法加载最近使用的应用列表或其他与用户相关的数据。从目前的情况上看，设备不存在密码、图案等设备加密方式，较大的可能性是系统软件问题或错误导致用户数据异常。

```
-02 13:56:15.205  7374  7374 I art     : Starting a blocking GC Explicit
-02 13:56:15.222  7375  7375 I ServiceManager: Waiting for service media.camera.proxy...
-02 13:56:15.227  7374  7374 I art     : Explicit concurrent mark sweep GC freed 38991(5MB) AllocSpace objects, 133(2MB) LOS objects, 39% free, 2MB/4MB,
used 174us total 21.943ms
-02 13:56:15.230  7374  7374 I art     : Starting a blocking GC Explicit
-02 13:56:15.240  7374  7374 I art     : Explicit concurrent mark sweep GC freed 4869(197KB) AllocSpace objects, 0(0B) LOS objects, 40% free, 2MB/3MB,
used 158us total 10.022ms
-02 13:56:15.267  7374  7449 I art     : Starting a blocking GC HeapTrim
-02 13:56:15.269  7374  7374 I art     : Starting a blocking GC Background
```

这是一段 GC（垃圾回收）操作的日志。在应用程序运行过程中执行多个 GC 操作，包括显示的阻塞式回收和后台回收，目的是释放不在使用的内存资源。但在 log 中发现存在多处类似的 GC 回收操作，两分钟的日志中大约出现了 200 个 GC 相关记录。高频 GC，这是内存泄露的一个标志，当对象不在被需要却仍然被引用，它们不会被垃圾回收器回收，随着时间的推移，这些对象积累会占用越来越多的内存，结果，垃圾回收器必须更频繁地运行，以尝试释放可用内存。从目前没有复现客户现象的设备上看，logcat 中没有出现 GC 操作，也没有触发 lowmemorykiller。

9、阶段性结论

目前暂时认为是内存泄露所导致的客诉问题。由应用程序中的代码错误或设计缺陷引起，如果不遵循良好的编程规范或忽视资源管理，很容易造成内存泄露。在长期运行在内存泄露的状态下，加之

2G 的小内存，导致 lowmemorykiller 触发后关闭 ActivityManager 连接和各类服务，从而出现 apk reboot 的现象。

10、综上

我们当前的对策是

1 用 4G 内存应该就可以解决

2 如果当前客户指定使用的 APK，不需要 GMS，使用不带 GMS 包的 BSP，也可以解决当前市场问题。