程力行 2014.07.25

# 引言

## 编写目的

## 描述

## 参考文献

Android源码及网络资料。

# 系统关键异常监控总览

为了能否快速响应领导及用户投诉。系统在一些关键位置上报关键异常信息。

下面的异常类型均为在系统代码中触发。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 系统类异常 | 业务类异常 | 异常类型 | 详细说明 | 触发位置 |
| kernel\_crash | 内核死机 |  | 内核出现各种异常 | init.rc  cp\_dump\_debug\_info.c |
| system\_crash | System server进程异常 | systemserver | System server空指针等异常 | ActivitityMananerService.java |
| System server进程异常 | watchdog | Watchdog监控到卡死系统进程卡死 | Watchdog.java |
| native\_crash | 在native发生crash | native\_crash | 在native发生crash | BootReceiver.java |
| app\_crash | App进程异常 | app\_crash | app发生crash | ActivitityMananerService.java |
| anr\_crash | ANR | anr\_crash | app发生ANR | ActivitityMananerService.java |
| dumpstate\_crash | 内存不足 | system\_lowmemory | 系统剩余内存小于100M | DeviceStorageMonitorService.java |
| 频繁ANR | system\_frequencyANR | 短时间内发生多次ANR | ActivitityMananerService.java |
| 主动上报 | system\_activeuserpower | 用户按电源键主动触发 | GlobalActions.java |
| system\_critical | CPU占用率高 | system\_highcpu | 用户到界面主动触发，打印高CPU进程的调用栈 | 添加应用界面触发 |
|  | 内外置sdcard满 | system\_sdcardfull | 每隔一分中监控一次 | DeviceStorageMonitorService.java |
|  | 安装包损坏 | system\_badpackage |  | Packagemanagerservice.java |
|  | 包安装签名错误 | system\_mismatchassign |  | Packagemanagerservice.java |
|  | 包安装检查空间不足 |  |  | （可略,包安装可合并） |
|  | 卷挂载失败 | system\_mountfail |  | MountService.java |
|  | 灭屏长时间持有电源锁 | system\_holdwakelock |  | PowerManagerService.java |
|  |  |  |  |  |

# 死机监控

死机分为内核死机、开关机异常、虚拟机重启。

## 内核死机kernel\_crash

系统实现：

具体实现见\PLT\_SYS\_FRWK\frameworks\base\cmds\dump\_debug\_info.c， 编译到system/bin/dumpdebuginfo。

这个模块根据开机原因读取/proc/last\_kmsg到两个文件中

一份是读取到文件/data/yl\_debug\_mode\_stack.txt（追加方式读取，最大为20M）

另一份读取到/data/brs/kernel/debug\_info.txt（覆盖式读取，每次读取时将前面截短到0之后再写入）

last\_kmsg只能存储最近一次的死机信息，在死机时写的，原生的kernel机制，最后的死机现场，利用 dumpdebuginfo可以保存多次的内核死机日志，追加写到文件yl\_debug\_mode\_stack.txt，放在/data下，最大20M

debug\_info.txt是覆盖写入，即上次的死机日志，bugreport上报就是/data/brs/kernel/debug\_info.txt，文件较小。

如果上一次是死机，下次异常开机就会拉起死机日志拷贝。

启动过程在init.rc中实现如下：

# yulong begin, add

# Add for dump debug info to file, wangzhenhua, 2012.07.11

service dump\_debug /system/bin/dumpdebuginfo

    class main

    oneshot

    disabled

on property:ro.yulong.boot.reason=excep\_press\_poweron

    start dump\_debug

# yulong end

ro.yulong.boot.reason

excep\_press\_poweron

## 开关机异常

同内核死机

## System server进程异常

### system\_crash

#### watchdog子类型

watchdog监控到系统进程几个核心服务长时间没有返回、死锁。

if (YulongFeature.FEATURE\_BUGREPORT) {

BugReportTrigger but = new BugReportTrigger();

but.LaunchBugReportSystem("watchdog\_crash", null,

"system\_server", null);

}

上报traces.txt及logcat日志

#### systemserver子类型

ActivityManagerService.java (frameworks\base\services\java\com\android\server\am)

handleApplicationCrash函数

if (YulongFeature.FEATURE\_BUGREPORT) {

Slog.d(TAG, "handleApplicationCrash=>BRS report crash... ");

if (app != null)

but.LaunchBugReportSystem("app\_crash", processName, null,

crashInfo);

else

but.LaunchBugReportSystem("systemserver\_crash", processName,

null, crashInfo);

}

上报logcat日志

### native\_crash

BootReceiver.java (frameworks\base\services\java\com\android\server)

onEvent函数添加

if (YulongFeature.FEATURE\_BUGREPORT) {

Slog.d(TAG, "BootReceiver=> BugReportEnable...");

BugReportTrigger but = new BugReportTrigger();

but.LaunchBugReportSystem("native\_crash", headers,

filename, null);

}

# 应用崩溃监控

## app\_crash

ActivityManagerService.java (frameworks\base\services\java\com\android\server\am)

handleApplicationCrash函数

if (YulongFeature.FEATURE\_BUGREPORT) {

Slog.d(TAG, "handleApplicationCrash=>BRS report crash... ");

if (app != null)

but.LaunchBugReportSystem("app\_crash", processName, null,

crashInfo);

else

but.LaunchBugReportSystem("systemserver\_crash", processName,

null, crashInfo);

}

上报logcat日志

## native\_crash

同System server进程异常的native\_crash

# 系统慢监控

确定问题的根源

## ANR自动上报

ActivityManagerService.java (frameworks\base\services\java\com\android\server\am)

在appNotResponding函数

if (YulongFeature.FEATURE\_BUGREPORT) {

if(mBugreportAnrFrequentEnabled) {

long now = System.currentTimeMillis();

long diffTime = now - mLastReportTime;

if (diffTime > 4\*60\*1000) {

mLastReportTime = now;

mReportTimes = 0;

} else {

/\* if two anrs happened in four minutes \*/

mReportTimes ++;

}

}

if (mReportTimes > 2 && mBugreportAnrFrequentEnabled ) {

Slog.d(TAG, "appNotResponding=>Bug Report System report java crash, dumpstate\_crash, system\_frequencyANR ");

but.LaunchBugReportSystem("dumpstate\_crash", "system\_frequencyANR" + app.processName, null, null );

} else {

Slog.d(TAG, "appNotResponding=>Bug Report System report java crash... ");

if (activity == null) {

but.LaunchBugReportSystem("anr\_crash", app.processName,

null, null);

} else {

but.LaunchBugReportSystem("anr\_crash", app.processName,

activity.shortComponentName, null);

}

}

}

## 频繁ANR自动上报

同ANR自动上报

## 长时间CPU占用高上报（梳理中，目前可以主动上报）

参考实现

ProcessStats.java (frameworks\base\core\java\com\android\internal\os)

LoadAverageService.java (frameworks\base\packages\systemui\src\com\android\systemui)

DevelopmentSettings.java (packages\apps\settings\src\com\android\settings)

实现一个界面，导出当前进程的调用栈。

涉及应用、系统接口

ANR日志中也会重点体现。

System server有个专门的DebugCpuUsage

"DebugCpuUsage" prio=5 tid=18 TIMED\_WAIT

| group="main" sCount=1 dsCount=0 obj=0x415bbdd0 self=0x711929e8

| sysTid=636 nice=0 sched=0/0 cgrp=apps handle=1897049488

| state=S schedstat=( 258016427 167206831 600 ) utm=10 stm=15 core=0

at java.lang.VMThread.sleep(Native Method)

at java.lang.Thread.sleep(Thread.java:1031)

at java.lang.Thread.sleep(Thread.java:1013)

at android.os.SystemClock.sleep(SystemClock.java:115)

at com.android.server.am.ActivityManagerService$4.run(ActivityManagerService.java:1790)

在ActivityManagerService中，专门有个线程名为ProcessStats的线程在监控CPU

"ProcessStats" prio=5 tid=17 TIMED\_WAIT

| group="main" sCount=1 dsCount=0 obj=0x415bbb60 self=0x7116bf18

| sysTid=635 nice=0 sched=0/0 cgrp=apps handle=1897410176

| state=S schedstat=( 190750 43167 2 ) utm=0 stm=0 core=2

at java.lang.Object.wait(Native Method)

- waiting on <0x415bbb60> (a com.android.server.am.ActivityManagerService$3)

at java.lang.Object.wait(Object.java:401)

at com.android.server.am.ActivityManagerService$3.run(ActivityManagerService.java:1768)

参考《开发者选项界面分析》分析文档

# 存储空间监控

## 内存不足自动上报

DeviceStorageMonitorService.java (frameworks\base\services\java\com\android\server)添加了内存监控，目前默认阈值是100M

private void checkRAM() {

if (YulongFeature.FEATURE\_BUGREPORT) {

if ("0".equals(SystemProperties.get("persist.yulong.memoryreport", "0")))

{

// keep posting messages to itself periodically

postCheckRAMMsg(DEFAULT\_RAM\_CHECK\_INTERVAL);

return;

}

/\* only for vip customer \*/

long availMem = Process.getFreeMemory()/(1024\*1024);

String memStrThreshold = SystemProperties.get("persist.yulong.memoryvalue", "100");

long memThreshold = Integer.parseInt(memStrThreshold);

if (availMem > memThreshold) {

// keep posting messages to itself periodically

postCheckRAMMsg(DEFAULT\_RAM\_CHECK\_INTERVAL);

return;

}

Slog.d(TAG, "checkRAM, now is lowmemory, availMem is " + availMem);

long now = System.currentTimeMillis();

long diffTime = now - mLastReportTime;

if (diffTime > (2 \* 60 \* 60 \* 1000)) {

Slog.d(TAG, "checkRAM=>Bug Report System dumpstate\_crash, system\_lowmemory... ");

mBugReportTrigger.LaunchBugReportSystem("dumpstate\_crash", "system\_lowmemory",

memStrThreshold, null);

mLastReportTime = now;

}

// keep posting messages to itself periodically

postCheckRAMMsg(DEFAULT\_RAM\_CHECK\_INTERVAL);

}

}

ActivityManagerService.java (services\java\com\android\server\am)

参考handleMessage函数REPORT\_MEM\_USAGE分支的处理，这块代码，平台上有修改，在user版本中，也添加低内存日志的打印，如

[/data/system/dropbox/system\_server\_lowmem@946697822239.txt.gz](mailto:/data/system/dropbox/system_server_lowmem@946697822239.txt.gz)

压缩文件大概8KB。

在此处调用了dumpApplicationMemoryUsage函数，根据OOM的adj值排序输出

调用了procrank可执行文件。（没有权限执行）

调用了dumpProcessesLocked，输出了进程名及身份

调用了dumpServicesLocked，输出了所有的服务

调用了dumpActivitiesLocked，输出了所有的Activities。

REPORT\_MEM\_USAGE消息的触发，是应用被杀，appDiedLocked处理中，发现一个后台应用都没有，就认为是系统处于低内存了。

除了dropbox的日志，在离线日志中，

event中有I/am\_low\_memory( 325): 18（其中18表示当前的进程个数）

meminfo及procrank日志以Info级别打印在system日志中。

## 内外置存储空间不足自动上报

DeviceStorageMonitorService.java (frameworks\base\services\java\com\android\server)

执行du命令

# 包管理异常监控

## 包安装异常

包损坏

签名不一致

空间不足

Packagemanagerservice.java

# 挂载服务异常监控

内置Sdcard、外置sdcard挂载失败，上报挂载失败类型、内核日志、logcat

挂载成只读，上报内核日志、logcat（可能做不到，需要研究代码）

读写异常，上报异常类型、内核日志、logcat（可能做不到，需要研究代码）

MountService.java

# 电源管理异常监控（梳理中）

## 灭屏后，长时间持有锁

PowerManagerService.java (frameworks\base\services\java\com\android\server\power)

# 主动上报

## 日志收集

收集dumpstate日志

代码修改

GlobalActions.java (frameworks\base\policy\src\com\android\internal\policy\impl)

public void onPress() {

AlertDialog.Builder builder = new AlertDialog.Builder(mContext);

builder.setTitle(com.android.internal.R.string.bugreport\_title);

builder.setMessage(com.android.internal.R.string.yulong\_bugreport\_message);

builder.setNegativeButton(com.android.internal.R.string.cancel, null);

builder.setPositiveButton(com.android.internal.R.string.report,

new DialogInterface.OnClickListener() {

@Override

public void onClick(DialogInterface dialog, int which) {

// Add a little delay before executing, to give the

// dialog a chance to go away before it takes a

// screenshot.

mHandler.postDelayed(new Runnable() {

@Override public void run() {

if (YulongFeature.FEATURE\_BUGREPORT) {

launchBugReportSystem();

} else {

try {

ActivityManagerNative.getDefault()

.requestBugReport();

} catch (RemoteException e) {

}

}

}

}, 500);

}

});

AlertDialog dialog = builder.create();

dialog.getWindow().setType(WindowManager.LayoutParams.TYPE\_KEYGUARD\_DIALOG);

dialog.show();

/\* yulong begin, add \*/

/\* add for power bug report menu dialog dismiss,wzh,20130301\*/

mPwrBugDlg= dialog;

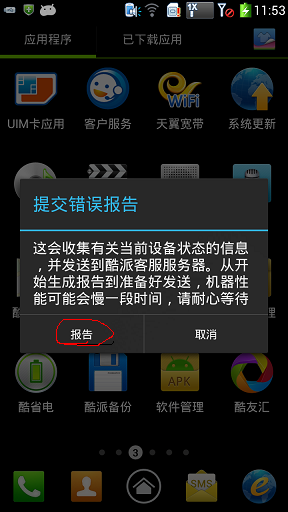
/\* yulong end \*/

}

## 操作步骤

操作步骤



# DropBox错误类型上报

DropBox是一个记录错误的服务，默认路径记录在/data/system/dropbox

# bugreport服务

bugreport服务设计参考<http://172.16.2.11/webpage/ControlPanel.html>，

系统的日志监控参考 系统关键异常监控总览一小节。

这里另外要特别说明的是如何抓取到root权限才能抓取到的日志。

参考一些已有的实现，以及如何更好实现。

## root权限参考

1. Am.java (frameworks\base\cmds\am\src\com\android\commands\am)

am bug-report

(2) ActivityManagerService.java (services\java\com\android\server\am)

public void requestBugReport() {

// No permission check because this can't do anything harmful --

// it will just eventually cause the user to be presented with

// a UI to select where the bug report goes.

SystemProperties.set("ctl.start", "bugreport");

}

用adb开启服务

setprop  ctl.start  $SERVICE\_NAME      #启动某个服务  
    setprop  ctl.stop  $SERVICE\_NAME      #停止某个服务

NVIDIA原生

# bugreport is triggered by the VOLUME-DOWN and VOLUME-UP keys

service bugreport /system/bin/dumpstate -d -v -o /data/storage/bugreports/bugreport

    class main

    disabled

    oneshot

    keycodes 115 114

设置setprop  ctl.start bugreport启动服务：

root@android:/data/storage/bugreports # ll

-rw------- shell shell 4039025 2013-06-17 11:56 bugreport-2013-06-17-11-5

5-53.txt

## 设计

系统接口先封装一个接口

调用下setprop ctl.start "bugreport:-d -v -o /sdcard/dumpstate.txt" 试下。

或者setprop ctl.start "bugreport:-dvz -o /data/storage/bugreports/dumpstate.txt"

root@android:/data/storage/bugreports # ll

-rw------- shell shell 462547 2013-06-17 14:51 dumpstate.txt-2013-06-17-

14-51-12.txt.gz

setprop ctl.start "bugreport:-dvz -o /data/brs/temp/dumpstate"

setprop ctl.start ylbugreport

setprop ctl.start " ylbugreport:-dvz -o /data/brs/temp/yldumpstate "

setprop ctl.start "ylbugreport:-dvz -o /data/brs/temp/yldumpstate"

setprop ctl.start " ylbugreport:-dvz -o /sdcard/dumpstate.txt"

setprop ctl.start " ylbugreport:-dv -o /sdcard/dumpstate.txt"

## 读写文件

文件通知

# 系统状态日志输出优化

去除不必要的日志打印，增加必要的日志。系统服务梳理

系统接口，增加日志触发

CP\_MemoryReport.apk，优化界面

## 早期VIP用户的开关控制

核心是要导出系统的dumpsys状态日志，procrank、librank日志

定义类型dumpstate\_crash

消息Intent定义：

com.yulong.android.system.brs.control

1、当内存小时触发

memoryswitch:1开启，0关闭，当内存小到一定程度时，可以

memorythreshold:100，默认是100M，

根据不同机型配置，可以填写不同值

当内存低于100M触发。

2、当CPU忙、系统忙时触发

（1）ANR

anrswitch:1，1开启，0关闭,

当系统应用发生ANR时，触发。

（2）丢事件

dropevent:1 ，1开启，0关闭

当系统连续丢弃100个触屏或按键消息时，触发。

3、用户主动上报

useractive:1 ，1开启，0关闭

参考《长时间使用优化专项修改点》

### 主动上报完善

adb shell am broadcast -a com.android.chrome.GPU\_PROFILER\_START -e file /sdcard/Download/trace.txt

adb shell am broadcast -a com.android.test --es test\_string "this is test string" --ei test\_int 100 --ez test\_boolean true

用命令验证：

adb shell am broadcast -a com.yulong.android.system.brs.control -e BroadcastValue {"useractive":1}

Broadcasting: Intent { act=com.yulong.android.system.brs.control (has extras) }

Broadcast completed: result=0

adb shell am broadcast -a com.yulong.android.system.brs.control -e BroadcastValue {"vipcustomer":33}

Broadcasting: Intent { act=com.yulong.android.system.brs.control (has extras) }

Broadcast completed: result=0

adb shell am broadcast -a com.yulong.android.system.brs.control

–e BroadcastValue {\"memoryswitch\":1,\"memorythreshold\":50,\"dropevent\":1,\"useractive\":1}"

在发送广播消息要注意，

adb shell am broadcast -a com.yulong.android.system.brs.control -e BroadcastValue {" vipcustomer":

最后不全，在系统中处理就会导致systemserver空指针。

getprop |busybox grep "persist.yulong"查看属性变化。

adb shell am broadcast -a com.yulong.android.bugreportclient.brs.control -e BroadcastValue {" switchcontrol ":1}

BugReportClient\src\com\yulong\android\bugreport\client\menu\BootReciver.java

## VIP用户标记优化

使用一个系统属性标记VIP用户

persist.yulong.vipcustomer

1：（1）表示vip用户

11：（2，3）表示内存低要上报

101：（4，5）表示内外置存储空间低要上报

1001：（8, 9）表示频繁ANR要上报

10001：（16，17）表示包安装异常要上报

100001:(32，33)表示服务器主动获取

setprop persist.yulong.vipcustomer 1

vipcustomer

## 上报触发点标记

由于触发日志需要触发yldumpstate

要标记触发子类型，则要使用一个属性来标记触发点：

persist.yulong.reportbugpos

setprop persist.yulong.reportbugpos 1

上报日志验证

在/data/brs目录下，bugreportclient会压缩成log.zip

log.zip包含Accessories.txt、logcat.txt，Accessories.txt修改成yldumpstate.txt.gz，解压即可。

## User版本新增集成内容

集成Android\_filesystem\_config.h (system\core\include\private)

修改 #if FEATURE\_YULONG == 1

{ 00755, AID\_ROOT, AID\_ROOT, "system/xbin/tcpdump" },

{ 00755, AID\_ROOT, AID\_ROOT, "system/xbin/librank" },

{ 00755, AID\_ROOT, AID\_ROOT, "system/xbin/procrank" },

{ 00755, AID\_ROOT, AID\_ROOT, "system/xbin/procmem" },

#else

{ 06755, AID\_ROOT, AID\_ROOT, "system/xbin/tcpdump" },

{ 06755, AID\_ROOT, AID\_ROOT, "system/xbin/librank" },

{ 06755, AID\_ROOT, AID\_ROOT, "system/xbin/procrank" },

{ 06755, AID\_ROOT, AID\_ROOT, "system/xbin/procmem" },

{ 06755, AID\_ROOT, AID\_ROOT, "system/xbin/su" },

#endif

\build\_script\coolpad.mk新增集成文件

libpagemap \

librank \

procmem \

procrank \

showmap

# 系统关键异常监控移植推广

系统关键异常日志详细参考上面章节

涉及模块、发布形式、负责人

|  |  |  |  |
| --- | --- | --- | --- |
| 模块/文件 | 说明 | 发布形式 | 责任人 |
| \build\_script\coolpad.mk  添加 libpagemap \  librank \  procmem \  procrank \  showmap | User版本集成一些工具库 | 系统集成 | 系统集成人员 |
| Android\_filesystem\_config.h | 修改权限，避免CTS测试suid不允许 | 系统集成 | 系统集成人员 |
| CP\_SystemInterface.jar | Bugreport服务 | Hudson | 刘惠盛 |
| yldumpstate  放置在  \frameworks\native\cmds\yldumpstate目录下  涉及Android.mk  yldumpstate.c  yldumpstate.h  ylutils.c四个文件 | 抓取内存详细信息日志 | 系统集成 | 系统集成人员 |
| snaplog.sh | 抓取内存详细信息日志到sdcard上 | 系统集成 | 系统集成人员 |
| init.rc  添加  # yulong begin, add  # bugreport is triggered, chenglixing 20130617  service ylbugreport /system/bin/yldumpstate -a -d -z -v -o /data/brs/temp/yldumpstate      class main      disabled      oneshot  # yulong, end  # yulong begin, add  # bugreport is triggered  service snaplog /system/bin/sh /system/xbin/ylsnaplog.sh      class main      user root      group shell      disabled      oneshot  # yulong, end | 进程启动 | 系统集成 | 各产品系统人员 |
| ActivityManagerService.java | ANR、进程crash、远程控制 | 系统集成 | 各产品系统人员 |
| Watchdog.java | 卡死监控  watchdog\_crash类型 | 系统集成 | 各产品系统人员 |
| BootReceiver.java | 监控native的crash  native\_crash类型  即tombstone日志 | 系统集成 | 各产品系统人员 |
| GlobalActions.java | 用户主动按电源键 | 系统集成 | 各产品系统人员 |
| PackageManagerService.java | 包安装失败上报 | 系统集成 | 各产品系统人员 |
| DeviceStorageMonitorService.java | 存储空间监控上报 | 系统集成 | 各产品系统人员 |
| CP\_MemoryReport.apk  需要platform签名  在工程模式中调用 | 内存监控APK  此APK直接集成 | 系统集成  邮件直接发送  或平台上取 | 系统集成人员  马鸿耀 |
| busybox | busybox工具直接集成 | 系统集成 | 系统集成人员 |

## CP\_MemoryReport.apk说明

CP\_MemoryReport.apk

1、CP\_MemoryReport.apk是一个提取内存信息的APK，。集成是为了分析内存问题，默认在launcher界面上不能显示。只在工程模式里提供入口进入查看。

2、请集成CP\_MemoryReport.apk，放入到项目中：PLT\_PUB\_Vendor\vendor\yulong\ThirdApp

平台已经放置了，可一个直接取

PLT\_PUB\PLT\_PUB\_Vendor\vendor\yulong\ThirdApp取一个最新的。

工程模式会动态判断CP\_MemoryReport是否被安装。如果集成了，会在工程模式-系统类设置中显示内存监控。

3、工程模式中拉应用的方式

Intent intent = new Intent();

                intent.setClassName("com.yulong.android.memoryreport", "com.yulong.android.memoryreport.MemoryWatchActivity");

         startActivity(intent);

# 日志间接上报

目前BugReportService存在缺陷，要压缩成一个文件，才能取出完整的日志。

BugReportService监控了/data/brs/indirect

sBRS\_INDIRECT\_Observer = new FileObserver(INDIRECT\_TRANSFER\_DIR.getPath(), FileObserver.CLOSE\_WRITE)

Thread [<18> FileObserver] (Suspended)

BugReportService.setBugType(String, String, String, String, String, String, int) line: 638

BugReportService$4$1.onEvent(int, String) line: 914

FileObserver$ObserverThread.onEvent(int, int, String) line: 125

FileObserver$ObserverThread.observe(int) line: not available [native method]

FileObserver$ObserverThread.run() line: 88

发送MSG\_BUG\_REPORT消息，在处理了MSG\_BUG\_REPORT消息时，拷贝了文件。

Thread [<52> BugReportService] (Suspended)

BugReportService.reportInDirectCrash(String) line: 1222

BugReportService.Handle\_BugTpye(int) line: 384

BugReportService.access$200(BugReportService, int) line: 34

BugReportService$BugReportHandler.handleMessage(Message) line: 266

BugReportService$BugReportHandler(Handler).dispatchMessage(Message) line: 99

Looper.loop() line: 137

BugReportService$BugReportThread.run() line: 529

reportInDirectCrash只是拷贝了一个文件

private void reportInDirectCrash(String filename) {

BufferedReader buffer = null;

try {

File files = new File(filename);

if (files != null && files.exists()) {

InputStream inputstream = new FileInputStream(files);

buffer = new BufferedReader(new InputStreamReader(inputstream));

String line = buffer.readLine(); // 只读了一行

inputstream.close();

if (line != null && line.contains(DIRECT\_TRANSFER\_PATH)) {

Slog.d(TAG, "Copy file path: " + line);

File\_Copy(false, line, DATA\_BRS\_PATH + "/Accessories.txt");

IntentPara(clientIntent, BUG\_INFO[2], line, false);

File\_Delete(line);

} else {

Slog.w(TAG, "Copy file path is invalid : " + line);

}

}

}

发送MSG\_LAUNCH\_CLIENT消息

Thread [<52> BugReportService] (Suspended (breakpoint at line 218 in BugReportService$BugReportHandler))

BugReportService$BugReportHandler.handleMessage(Message) line: 218

BugReportService$BugReportHandler(Handler).dispatchMessage(Message) line: 99

Looper.loop() line: 137

BugReportService$BugReportThread.run() line: 529

在处理MSG\_LAUNCH\_CLIENT，发送yulong.intent.action.BUGREPORT启动应用服务BugReportClient

# 测试用例

测试用例如下，更详细的测试用例补充

1、进入工程模式—>系统类设置—>内存监控，点击导出系统状态日志详情，过段时间后（如1分钟后），会进入到snaplog目录下，看到anr等日志。

2、进入工程模式—>系统类设置—>内存监控，点击上报当前系统状态，服务器<http://172.16.2.11/webpage/ControlPanel.html>上能看到对应的日志。类型为system\_critical，需要在异常类型输入框中手动输入system\_critical

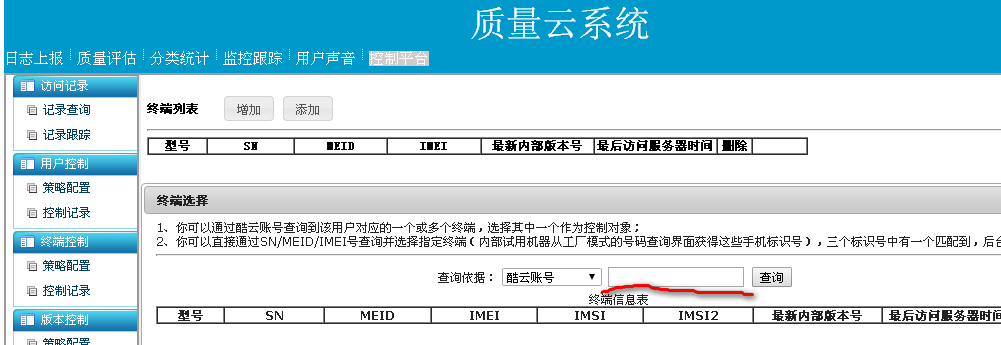
服务

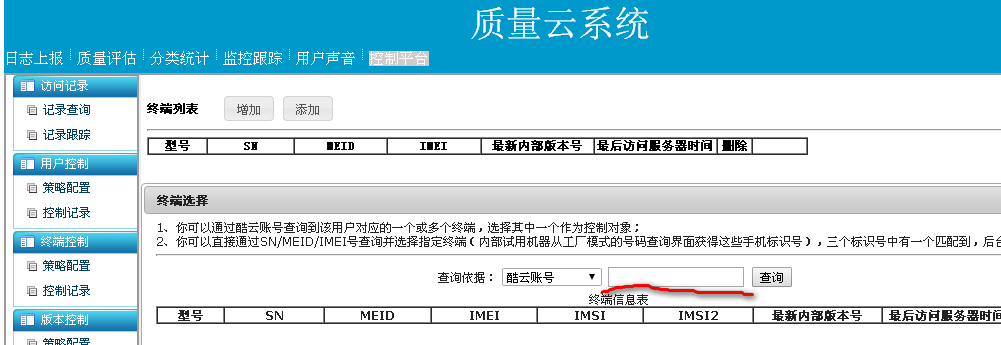
3、后台主动抓日志，可针对某台机器进行抓取，点击控制平台



用账户密码登录（咨询具体开发人员，后台管理员）

通过具体酷云账号，查到对应的机器，或直接IMEI号找到。





参考上面的<VIP用户标记优化>小节，输入33，如果bugreport客户端与服务器交互，则会触发上报，后台服务器就应该能看到对应日止，也是system\_critical类型。

注意：

log.zip日志解压后，为Accessories.txt，打开是乱码，则将后缀改为gz，再解压，打开，查看对应日志是否全面满足。