Android Logging System和一种独立的Logger App实现方式

guangli.han@tendcloud.com
2017/5/8

TalkingData

Background

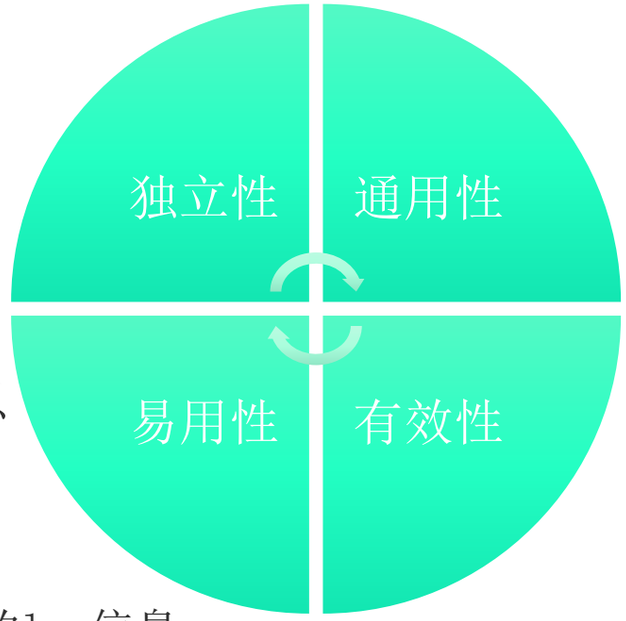
- 两则故事
  - 开发人员调试的故事
  - 测试人员和开发人员之间的故事

Background

- 上面log查看和存储方式问题
  - 查看log：
    - **需要USB有线连接或者WiFi无线连接**，存在连接不稳定情况。
  - 存储log：
    - 1. Host方式，**需要USB有线连接或者WiFi无线连接**
    - 2. App内部存储方式：**影响App性能，增加额外的维护成本，不方便测试人员使用**

Background

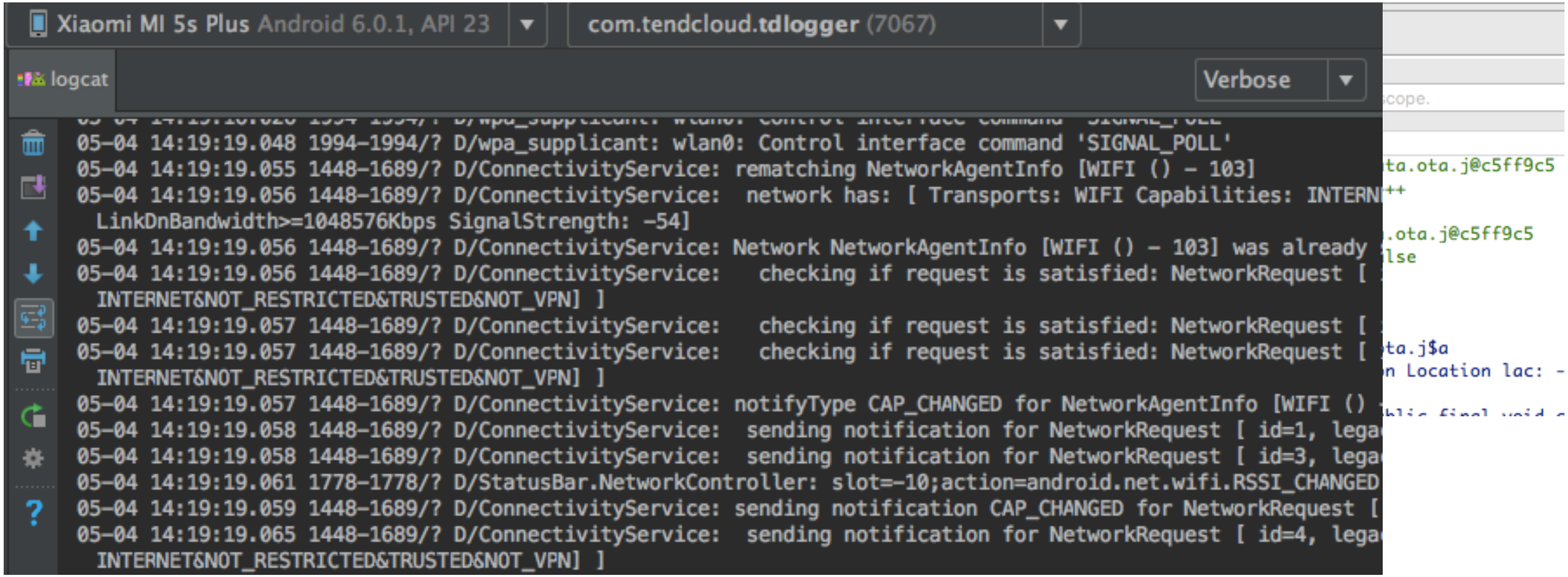
- 查看和存储log需求:
  - 1. 独立性，与开发者App 代码独立
  - 2. 通用性，适用Android不同App
  - 3. 易用性，方便开发者和测试人员使用
  - 4. 有效性，log信息能够完整记录App log信息



- 如何实现这一需求？
  - 一个独立的App：LoggerApp，能够记录被测试App的log信息
    - Android log机制
    - 两个App之间能够共享数据

Background

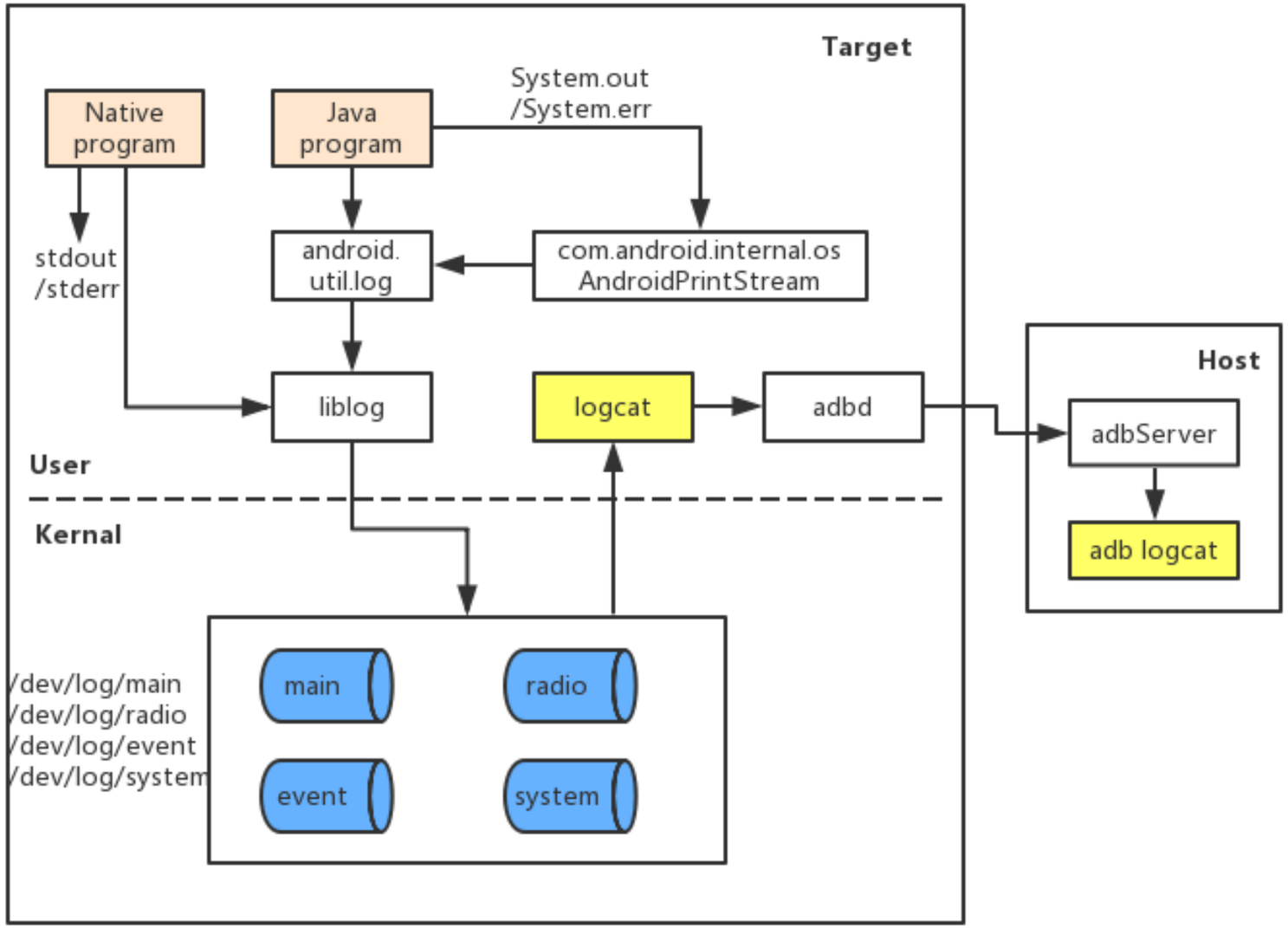
- Android App开发者常用的log查看和存储方式：
  - 查看log
    - Eclipse、Android Studio、DDMS：Logcat



Background

- Android App开发者常用的log查看和存储方式：
  - 存储log
    - 1. Host端存储方式：DDMS/ 手机USB 连接logcat cmd/…
    - 2. App内部存储方式：APP中实现log保存到文件的功能模块，存储log文件

Android Logging System



Logging interface

VERBOSE	DEBUG	INFO	WARN	ERROR

- Java applications
  - Class Name: android.util.Log
  - General method: Log.v()/Log.d()/Log.i()/Log.w()/ Log.e()
- Native applications
  - Header File: #include <cutils/log.h>

    LOGV()/LOGD()/LOGI()/ LOGW()/ LOGE()
- 小技巧:
  - 控制打印某个log level以下的日志
  - adb shell setprop log.tag.<YOUR_LOG_TAG> <LEVEL>
    - adb shell setprop log.tag.TDLog ERROR
    - 仅本次开机有效

Logging interface

- private static final String TA
- Log.v(TAG, “ This is log stri
- 测试代码和实际效果

```
/* 1.去掉TAG: 我们可以用类名来作为TAG的内容,获取方法名和行数: */

private static void getMethodNames(StackTraceElement[] sElements){
    className = sElements[1].getFileName();
    methodName = sElements[1].getMethodName();
    lineNumber = sElements[1].getLineNumber();
}
/* 2.重新定义写Log的方法: */

public static void v(String message){

    getMethodNames(new Throwable().getStackTrace());
    Log.v(className, createLog(message));
}
/* 3.封装成方法 */

private static String createLog( String log ) {
    StringBuffer buffer = new StringBuffer();
    buffer.append(methodName);
    buffer.append("(").append(className).append(":").append(lineNumber).append(")");
    buffer.append(log);
    return buffer.toString();
}
```

```
v("This is log string");
```

```
05-05 11:14:03.871 5837-5837/? V/MainActivity.java: onCreate(MainActivity.java:13)This is log string
05-05 11:14:03.871 5837-5837/? V/MainActivity: This is log string
```

logcat

参数	描述
-b <buffer>	加载一个可使用的日志缓冲区供查看，比如event和radio。默认值是main
-c	清除缓冲区中的全部日志并退出（清除完后可以使用-g查看缓冲区）
-d	将缓冲区的log转存到屏幕中然后退出
-f <filename>	将log输出到指定的文件中<文件名>.默认为标准输出（stdout）
-g	打印日志缓冲区的大小并退出
-n <count>	设置日志的最大数目<count>，默认值是4，需要和-r选项一起使用
-r <kbytes>	没<kbytes>时输出日志，默认值是16，需要和-f选项一起使用
-s	设置过滤器
-v <format>	设置输出格式的日志消息。默认是短暂的格式。支持的格式列表

- Refer:
  - http://developer.android.com/guide/developing/tools/adb.html

logcat -v <format>

格式
brief
process
tag
raw
time
threadtime
long

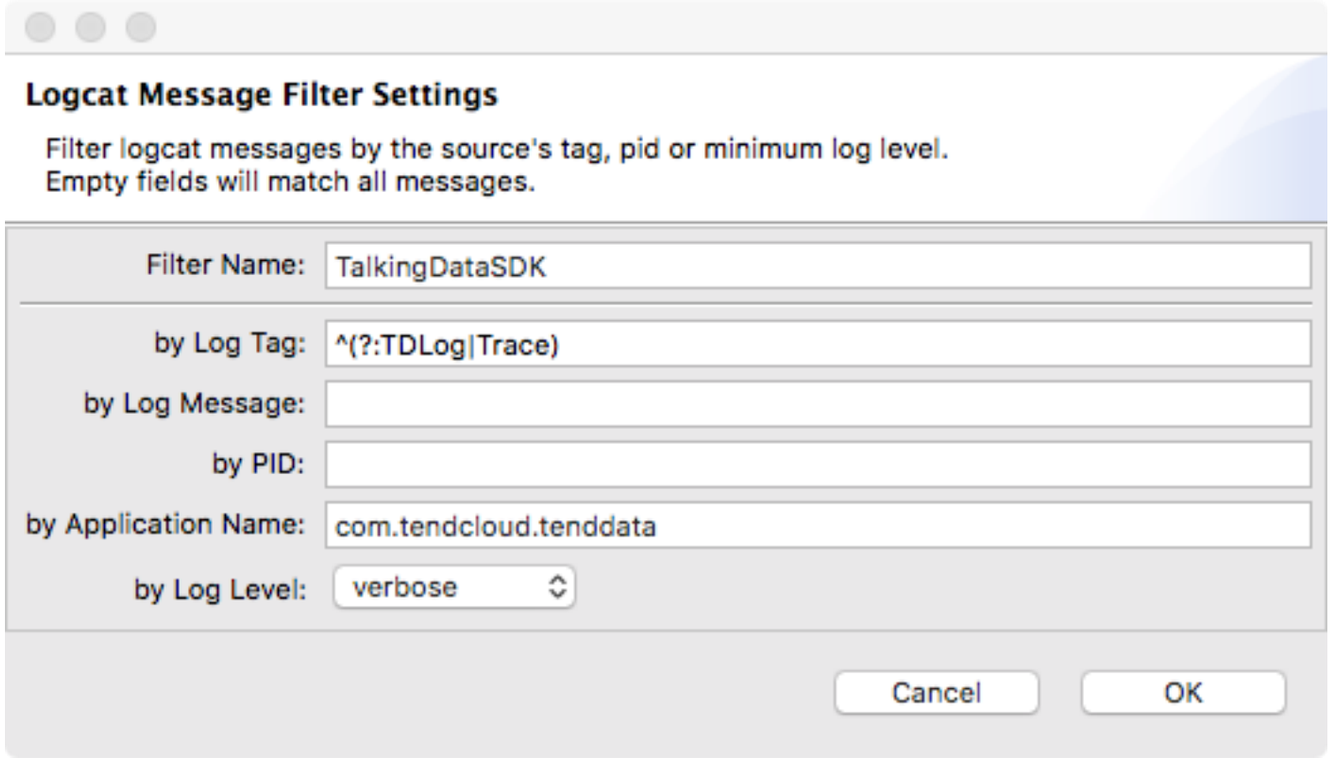
■brief	D/Security_PermControlService(3806): handleCheckCase notifyType=1
■process	V(3806) sending alarm {195b28a8 type 1 *alarm*:agoo_action_heart} (AlarmManager)
■tag	V/AlarmManager: sending alarm {2b1740ce type 2 *walarm*:ALARM_ACTION(14286)}
■raw	sending alarm {2ae58171 type 0 *walarm*:cn.wps.moffice.readlater.PushAction}
■time	12-01 17:14:25.268 V/AlarmManager(3806): sending alarm {2d2c41df type 0 *walarm*:alarm.
■threadtime	12-01 17:15:30.758 3806 5150 V AlarmManager: sending alarm {27cd6f46 type 0 *walarm*:c
■long	[12-01 17:17:28.557 3806: 3806 V/AlarmManager] done {eade8ed, *walarm*:wns.heartbeat} [20ms]

将设备main log信息存储到PC端文件：log.txt

```
adb logcat -b main -v threadtime>log.txt
```

Logcat使用技巧

- Logcat + grep 提高调试查看log效率
  - adb logcat | grep --invert-match 'notshownmatchpattern'
  - 过滤出指定tag的日志信息
    - ^(?:tag1|tag2|tag3)
  - 忽略指定tag的日志信息
    - ^(?!tag1|tag2|tag3)



Logger App

- 实现方式
  - logcat + android:shareUserId
    - Runtime.*getRuntime*().exec(“logcat -f file -b main -v threadtime”)
    - android:shareUserId ： 使Logger App与被测试App独立
      - Logger APK 和开发者APP manifest中添加 相同的sharedUserId

Logger App

AndroidManifest.xml

manifest application

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.tendcloud.tdlogger"
    android:sharedUserId="com.tendcloud.sdk" >
```

被测试 App

AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.tendcloud.demo"
    android:versionCode="1"
    android:versionName="1.0"
    android:sharedUserId="com.tendcloud.sdk" >
```

android:shareUserId

- 获取另外程序的context

  Context ctx = **this**.createPackageContext(“com.example.shareusertesta”,

  Context.CONTEXT_IGNORE_SECURITY);

  - **利用ShareUserID共享数据**

Logger App

- 对App安全性：
  - APK的签名必须相同
  - android:shareUserId的值必须相同

    - User IDs
    - ✓ Each Android package (.apk) file installed on the device is given its own unique Linux user ID, creating a sandbox for it and preventing it from touching other applications (or other applications from touching it).
    - ✓ This user ID is assigned to it when the application is installed on the device, and remains constant for the duration of its life on that device.
    - ✓ You can use the sharedUserId attribute in the AndroidManifest.xml's manifest tag of each package to have them assigned the same user ID.

Note: In order to retain security, only two applications signed with the same signature (and requesting the same sharedUserId) will be given the same user ID.

AppLogger UI

- GitHub source code:
  - https://github.com/GuangliHan/AndroidLogge

谢谢！

TalkingData