**Android端百度Crab的集成**

1. **Crab简介**

Crab 是百度旗下的APP监控平台，能够实现崩溃日志收集解析、问题分析定位以及线上质量统计和报警。平台主页 <http://crab.baidu.com/>

1. **Crab的集成细节**
2. **集成简介**

集成分为两大类，分别是：

1. 自动集成   －  Bintray Maven仓库 的方式
2. 手动集成   －  直接下载SDK，jar包导入的方式
3. **集成步骤**

**注：**由于项目全部使用AS开发，所以选择自动集成较为方便，也是Crab推荐的集成方案，所以这里只介绍自动集成，手动集成请参考 http://crab.baidu.com/wiki/sdk

1. **添加Bintray Maven仓库地址**

为了从Bintray Maven仓库获取Crab，需要在Module的build.gradle文件中添加如下Bintray Maven仓库地址

repositories {

maven {

url 'https://dl.bintray.com/crab/baidu'

}

}

1. **集成sdk和ndk，ndk选择性集成，但是集成了ndk就必须集成sdk**

在Module的build.gradle文件中添加依赖和属性配置：

dependencies {

//其中latest.release指代最新版的Crab SDK。您也可以指定特定版本的Crab SDK，如5.1.0

compile 'crab.baidu.crabsdk:crabsdk:latest.release@aar'

//其中latest.release指代最新版的Crab NDK。您也可以指定特定版本的Crab NDK，如2.2.1

compile 'crab.baidu.crabndk:crabndk:latest.release@aar'

}

1. **在项目中添加必须的代码**
2. 在AndroidManifes.xml中添加权限

<uses-permission android:name="android.permission.INTERNET"/>

<uses-permission android:name="android.permission.ACCESS\_NETWORK\_STATE"/>

<uses-permission android:name="android.permission.ACCESS\_WIFI\_STATE"/>

<uses-permission android:name="android.permission.READ\_PHONE\_STATE"/>

<!-- 仅debug模式下dump OOM文件需要以下权限 -->

<uses-permission android:name="android.permission.WRITE\_EXTERNAL\_STORAGE"/>

<!-- 记录logcat日志功能 所需的权限 -->

<uses-permission android:name="android.permission.READ\_LOGS" />

1. 在自定义Application中添加初始化代码

public class MyApplication extends Application {

public void onCreate() {

super.onCreate();

CrabSDK.init(this,appKey);

}

}

1. **按需求添加项目中需要的功能**
2. 添加日活代码（需要在每个Activity的生命周期方法中添加代码，建议在项目基类BaseActivity中添加）

@Override

protected void onPause() {

CrabSDK.onPause(this);

super.onPause();

}

@Override

protected void onResume() {

super.onResume();

CrabSDK.onResume(this);

}

1. 信息收集接口------异常数据收集接口，catch块中调用**【在初始化后调用】**

**try {**

**// YOUR CODE**

**} catch (Exception ex) {**

**CrabSDK.uploadException(ex);**

**Log.e("TAG", "ERR\_MSG", ex);**

**}**

1. 信息收集接口------ Native Crash收集接口（在CrabSDK.init后调用）

// 以下两个接口切勿同时调用，产品线应根据自身需求择一调用

// 开启native功能，过滤纯系统库堆栈，如libc.so

CrabSDK.openNativeCrashHandler();

// 开启native功能，不做任何过滤。

CrabSDK.openNativeCrashHandlerWithSysCatched()

注意: 需要将libcrab\_native.so分别拷入工程中对应的armeabi、armeabi-v7a、x86、x86\_64、arm64-v8a文件夹。如需捕获native crash的同时获取java层堆栈，需在native函数前后调用如下方法：

CrabSDK.callBeforeNativeMethod(); // native 函数之前调用

stringFromJNI(); // 用户自己的native函数

CrabSDK.callAfterNativeMethod(); // native 函数之后调用

注1：callBeforeNativeMethod()与callAfterNativeMethod()要成对使用，放在native函数前后。

注2：不调用不会影响native crash的捕获，只是拿不到java层的堆栈。

1. 信息收集接口------ ANR数据收集接口**【默认开启】**

* **ANR数据收集默认在CrabSDK.init时开启，调用CrabSDK.closeAnrHandler可关闭**
* **ANR数据收集功能支持单独开启，直接调用CrabSDK.initAnrCollector(this, appKey)即可**

1. 信息收集接口------Block（卡顿）数据收集接口**【仅支持Android4.1及以上系统】**

* **调用CrabSDK.enableBlockCatch(int...)可打开卡顿搜集功能，int值代表卡顿上传次数的最大值，-1为不限制上传次数。**
* **调用CrabSDK.disableBlockCatch可关闭。**
* **卡顿数据收集功能支持设置自定义阈值，默认阈值为2000毫秒。设置接口为CrabSDK.setBlockThreshold(2000)**
* **Crab卡顿收集原理是设置Looper.getMainLooper.setMessageLogging(..) 以及设置Choreographer类的FrameCallback 请确保本地程序未对该函数进行操作，以免卡顿收集功能受影响**

**(注：卡顿阈值设置不宜过小，建议至少大于1000ms)**

1. 信息收集接口------跳转路径收集接口[最多保存10条]【仅限4.0(不含)以下系统需要添加】

@Override

protected void onStart() {

super.onStart();

CrabSDK.doActivityStart(this);

}

@Override

protected void onStop() {

CrabSDK.doActivityStop(this);

super.onStop();

}

1. 信息收集接口------添加行为记录功能

* 第一步：设置记录最大条数

注：不设置情况下，默认是5条记录

CrabSDK.setBehaviorRecordLimit(5);

* 第二步：建议在项目基类BaseActivity中添加，同日活代码添加方法

@Override

public boolean dispatchTouchEvent(MonitorEvent event) {

CrabSDK.behaviorRecordEvent(event, this);

return super.dispatchTouchEvent(event);

}

1. 信息收集接口------添加WebView记录功能。（若app未用到WebView可忽略该功能）

* 第一步：设置记录最大条数

注：不设置情况下，默认是10条记录

CrabSDK.setUrlRecordLimit(10);

* 第二步：建议在项目基类BaseActivity中添加，同日活代码添加方法。或者仅在包含WebView的Activity的dispatchTouchEvent中单独添加

@Override

public boolean dispatchTouchEvent(MonitorEvent event) {

CrabSDK.urlRecordEvent(event, this);

return super.dispatchTouchEvent(event);

}

可配置参数（未表明的就是初始化函数之后调用）

1. 信息收集接口------收集应用截图，默认为false

CrabSDK.setCollectScreenshot(boolean);

1. 信息收集接口------设置SDK是否为debug模式，默认为false

CrabSDK.setDebugMode(boolean);

1. 信息收集接口------是否发送设备信息数据，默认为是

CrabSDK.setSendPrivacyInformation(boolean);

1. 信息收集接口------是否仅wifi环境下上传，默认为false

CrabSDK.setUploadCrashOnlyWifi(boolean);

1. 信息收集接口------设置app用户名，一般用于需要登录的app

CrabSDK.setUserName(String);

1. 信息收集接口------设置渠道号，仅第一次调用有效

CrabSDK.setChannel(String);

1. 信息收集接口------设置用户自定义uid，用于统计Crash影响用户数和日活跃用户数

CrabSDK.setUid(String uid);

1. 信息收集接口------设置单日单设备同类Crash上传次数上限，默认为10次上限，设置为-1则代表无上限限制

CrabSDK.setUploadLimitOfSameCrashInOneday(int limit)

1. 信息收集接口------设置单日单设备Crash上传次数上限，默认为30次上限，设置为-1则代表无限制

CrabSDK.setUploadLimitOfCrashInOneday(int limit)

1. 信息收集接口------设置单日单设备同类anr上传次数上限，默认为10次上限，设置为-1则代表无限制

CrabSDK.setUploadLimitOfAnrInOneday(int limit)

1. 信息收集接口------添加自定义字段

CrabSDK.setUsersCustomKV(HashMap<tring, String>);

CrabSDK.setUsersCustomKV(String, String);

1. 信息收集接口------关闭ANR

CrabSDK.closeAnrHandler();

1. 信息收集接口------设置Crash和ANR发生时的回调接口

CrabSDK.setOnAnrCrashListener(new OnAnrCrashListener() {

@Override

public void onAnrStarted(Map<String, Object> anrTraceMap) {}

@Override

public void onCrashStarted(Thread thread, Throwable ex) {}

@Override

public void onNativeCrashStarted(Error e, String backTrace, int signo) {}

});

1. 信息收集接口------设置同一Crash连续发生N次，下次启动时回调接口

注：要在crabSDK.init()之前设置，默认是-1，代表关闭

CrabSDK.setConstantSameCrashExceedLimit(3);

CrabSDK.setOnCrashExceedListener(new OnCrashExceedListener() {

@Override

public void onCrashExceedCallback() {

Log.d("^^^^^^^^^^", "同一Crash连续发生，回调此方法");

}

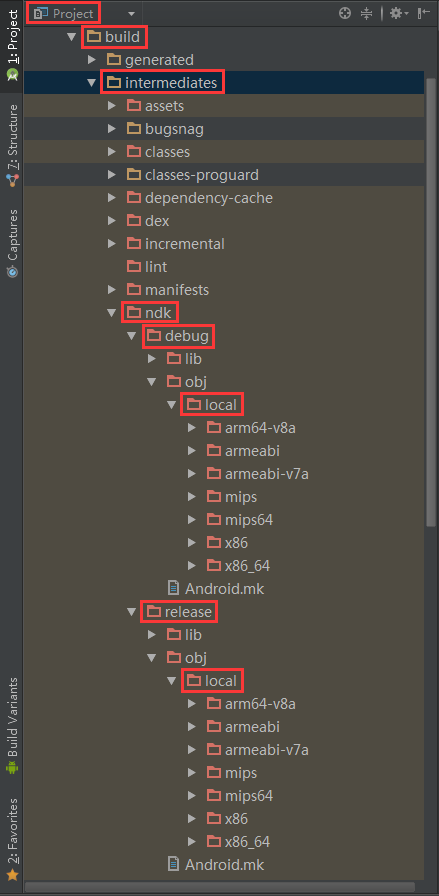
});

1. **上传Native符号表**

* 对于已集成crab native crash捕获功能的产品线，需要上传编译so文件时生成的符号表文件。后台会根据符号表对app发生的native crash的程序堆栈进行解析和还原。

注：这里只介绍IDE是AndroidStudio的。

* IDE如果使用Android Sutdio+NDK，默认情况下，Debug编译的符号表文件将位于： <项目文件夹>/<Module>/build/intermediates/ndk/debug/obj/local<架构>/。而Release编译的符号表文件将位于： <项目文件夹>/<Module>/build/intermediates/ndk/release/obj/local<架构>/。如下图：



1. **编译发布**

完成了上面的步骤，编译时可能有一些 import 问题，补上即可，SDK jar包是混淆过的，请在proguard-project.tx中添加白名单：

-keep public class com.baidu.crabsdk.\*\*

-keepclassmembers public class com.baidu.crabsdk.\*{

\*;

}

#Native Crash收集请加入如下配置

-keepclassmembers public class com.baidu.crabsdk.sender.NativeCrashHandler{

\*;

}

1. **相关说明**

* Crash发生时，SDK会尝试上传，若上传失败，会在app下次启动时上传，同一个Crash重试上传次数最多为三次。
* Crash和Block数据流量，无截图Crash数据为2.3k左右，带截图Crash数据大小为2.3K加额外截图大小，图片大小随手机分辨率和DPI而变化，480P – 1080P的分辨率对应的图片大小在10K – 30K之间。ANR一次上传数据量依系统运行情况在10k至几十k不等。
* Crab数据产生的24小时内所有上传操作默认仅在wifi条件下进行（可配置移动网络自动上传）。

1. **预装地图集成的功能**

* 10.1.0

1. 百度地图 10.1.0 集成了所有功能。

CrabController.java 当中做的处理。主要是收集ANR信息，同时设置了cuid、版本号，渠道号以及统计了日活量。另外在此类中提供了注册云控开关的入口。

1. 在MapsActivity.java中注册云控开关的回调监听器，此监听器在类CloudController.java中做的处理。目的是通过传过去的指令，判断是否收到服务器端数据。