

Bugly (Android) SDK 高级配置

Bugly文档资料

更多的Bugly行为控制

我们提供了UserStrategy类作为Bugly的初始化扩展，在这里您可以修改本次初始化Bugly数据的版本、渠道及部分初始化行为。通过以下方式传入：

```
UserStrategy strategy = new UserStrategy(appContext);  
//...在这里设置strategy的属性，在bugly初始化时传入  
//...  
CrashReport.initCrashReport(appContext, APPID, true, strategy);
```

如果通过UserStrategy设置了版本号和渠道号，则会覆盖“AndroidManifest.xml”里面配置的版本号和渠道。

UserStrategy主要API：

1、设置App版本、渠道、包名：

Bugly默认读取AndroidManifest.xml文件中VersionName、Package信息。若您有自己的版本或渠道设定需求，可通过该接口修改。

```
strategy.setAppChannel("myChannel"); //设置渠道  
strategy.setAppVersion("1.0.1"); //App的版本  
strategy.setAppPackageName("com.tencent.xx"); //App的包名
```

2、设置Bugly初始化延迟：

Bugly会在启动10s后联网同步数据。若您有特别需求，可以修改这个时间。

```
strategy.setAppReportDelay(20000); //改为20s
```

3、设置标签：

自定义标签，用于标明App的某个“场景”。在发生Crash时会显示该Crash所在的“场景”，以最后设置的标签为准，标签id需大于0。例：当用户进入界面A时，打上9527的标签：

```
CrashReport.setUserSceneTag(context, 9527); // 上报后的Crash会显示该标签
```

注意：打标签之前，需要在Bugly产品页配置中添加标签，取得标签ID后在代码中上报。

4、设置自定义Map参数

自定义Map参数可以保存发生Crash时的一些自定义的环境信息。在发生Crash时会随着异常信息一起上报并在页面展示。

```
CrashReport.putUserData(context, "userkey", "uservalue");
```

注意：最多可以有9对自定义的key-value（超过则添加失败），key限长50字节、value限长200字节，过长截断。

Javascript的异常捕获功能

Bugly Android SDK 1.2.8及以上版本提供了Javascript的异常捕获和上报能力，以便开发者可以感知到 WebView中发生的Javascript异常。

```
/**
 * 设置Javascript的异常监控
 *
 * @param webView 指定被监控的webView
 * @param autoInject 是否自动注入Bugly.js文件
 * @return true 设置成功; false 设置失败
 */
CrashReport.setJavascriptMonitor(Webview webView, boolean autoInject)
```

- Bugly.js文件在Bugly SDK包中，可以在HTML手动嵌入；
- 如果使用自动集成SDK方式，且需要在HTML手动嵌入Bugly.js，请下载[Bugly SDK包](#)；
- 由于Android 4.4以下版本存在反射漏洞，接口默认只对Android 4.4及以上版本有效；
- 接口不会设置webView的WebViewClient和Listener；
- 接口默认会开启webView的JS执行能力；

1、自动集成

建议在WebChromeClient的onProgressChanged函数中调用接口：

```
CrashReport.setJavascriptMonitor(webView, true);
```

例子如下：

```
WebView webView = new WebView(this);
// 设置WebChromeClient
webView.setWebChromeClient(new WebChromeClient() {
    @Override
    public void onProgressChanged(WebView webView, int progress) {
        // 增加Javascript异常监控
        CrashReport.setJavascriptMonitor(webView, true);
        super.onProgressChanged(webView, progress);
    }
});
// 加载HTML
webView.loadUrl(url);
```

2、手动集成

- 下载Bugly.js文件并添加到需要监控Javascript异常的HTML中：

```
<html>
  <script src="bugly.js" ></script>
<body>
  ...
</body>
</html>
```

- 在WebView加载完该HTML后设置Javascript的异常捕获功能：

```
WebView webView = new WebView(this);
// 加载HTML
webView.loadUrl(url);
// 增加Javascript异常监控
CrashReport.setJavascriptMonitor(webView, false);
```

在Bugly Android SDK捕获到Javascript异常后，默认会上报以下信息：

- Android设备的相关信息；
- Javascript异常堆栈和其他信息；
- Java堆栈；
- WebView的信息，目前只包括ContentDescription。

更多的Bugly日志附加信息

我们提供了一些信息记录API供您补充额外的内容。这些信息会随着异常一起上报。例如App环境、用户属性等等。主要包含以下接口：

1、设置用户ID

您可能会希望能精确定位到某个用户的异常，我们提供了用户ID记录接口。

例：网游用户登录后，通过该接口记录用户ID，在页面上可以精确定位到每个用户发生Crash的情况。

```
CrashReport.setUserId("9527"); //该用户本次启动后的异常日志用户ID都将是9527
```

2、主动上报开发者Catch的异常

您可能会关注某些重要异常的Catch情况。我们提供了上报这类异常的接口。

例：统计某个重要的数据库读写问题比例。

```
try {  
    //...  
} catch (Throwable thr) {  
    CrashReport.postCaughtException(thr); // bugly会将这个throwable  
    上报  
}
```

3、自定义日志功能

我们提供了自定义Log的接口，用于记录一些开发者关心的调试日志，可以更全面地反应App异常时的前后文环境。使用方式与android.util.Log一致。用户传入TAG和日志内容。该日志将在Logcat输出，并在发生异常时上报。有如下接口：

```
BuglyLog.v(tag, log)  
BuglyLog.d(tag, log)  
BuglyLog.i(tag, log)  
BuglyLog.w(tag, log)  
BuglyLog.e(tag, log)
```

注意：

1) 使用BuglyLog接口时，为了减少磁盘IO次数，我们会先将日志缓存在内存中。当缓存大于一定阈值（默认10K），会将它持久化至文件。您可以通过setCache(int byteSize)接口设置缓存大小，范围为0-30K。例：

BuglyLog.setCache(12 * 1024) //将Cache设置为12K

2) 如果您没有使用BuglyLog接口, 且初始化Bugly时isDebug参数设置为false, 该Log功能将不会有新的资源占用;

3) 为了方便开发者调试, 当初始化Bugly的isDebug参数为true时, 异常日志同时还会记录Bugly本身的日志。请在App发布时将其设置为false;

4) 上报Log最大30K。

添加额外的SO文件信息

为了更好得区分不同构建或者版本的SO文件以方便地管理Native代码, 建议给SO文件加上独立的版本号或者UUID。

1、添加SO文件的版本号

在任意一个源码文件 (建议是专门控制版本相关信息的源码文件) 中加入一行:

```
extern "C" const char SO_FILE_VERSION[] __attribute__((section
(".bugly_version"))) = "<SO文件版本号>";
```

此后, NDK编译的SO文件将带有一个具有版本信息的段 (.bugly_version)。之所以把段名定义为“.bugly_version”, 是为了Bugly的NDK解析的统一性。查看SO文件的版本号的一个方法如下 (需要readelf或类似工具):

```
readelf -p .bugly_version libxxx.so
```

2、添加SO文件的UUID

在“<NDK目录>/build/core/import-locals.mk”文件的最后加上一行:

```
LOCAL_LDFLAGS += -Xlinker --build-id
```

此后, NDK构建的SO文件将带有一个段 (.note.gnu.buildid) 专门存放构建的UUID。查看SO文件的UUID的一个方法如下 (需要readelf或类似工具):

```
readelf -x .note.gnu.build-id libxxx.so
```

其中“readelf”是GNU Binary Utilities一个工具, 用于解析ELF格式文件 (SO文件属于ELF格式文件)。Linux下默认安装了; Windows下可使用NDK提供的readelf (<NDK路径>\toolchains\xxx\prebuilt\windowsxxx\bin\xxxreadelf.exe) 或者安装MinGW或者Cygwin; Mac下如果没有该工具, 可从GNU Binary Utilities官网下载安装或者使用NDK提供的readelf。

Native堆栈例子如下图 (UUID不在堆栈中显示):

出错线程	全部线程	系统日志	自定义日志
SIGSEGV(SEGV MAPERR)			
#00 pc 00000cb4 libhelloNDK.so search_EIT_table (/s/ndk-toolchain/src/build/./gcc/gcc-4.8/libgcc/unwind-arm-common.inc:172) [armeabi-v7a] <u>[1.2.5]</u>			
#01 pc 0000118b /data/dalvik-cache/arm/data@app@bugly.tencent.com.simpletimer-1@base.apk@classes.dex			
java:			
.MainActivity\$1.onClick(MainActivity.java:127)			
android.view.View.performClick(View.java:4815)			
android.view.View\$PerformClick.run(View.java:19875)			

FAQ

1、开发过程中怎样查看Bugly的Logcat日志？

开启Bugly的Logcat日志需要在初始化时，isDebug参数设为true。

TAG为CrashReportInfo，是Bugly主要操作日志，包括初始化、日志上报信息；

TAG为CrashReport，是Bugly调试日志，若Bugly使用中有问题，可以将该日志信息反馈给客服人员。

2、为什么相同的用户一天上报了几百条Crash？会消耗用户流量吗？

系统在进程发生Crash后，有可能会再次将它拉起（系统的、代码逻辑的），导致不停地Crash。例如Service、Recevier等都比较容易出现自动拉起的情况。

Bugly有自己的流量保护机制，在保证数据准确性的前提下，会尽量减少用户流量消耗。

3、Bugly上报Crash的时机是？

Bugly会在发生Crash时尝试尽量上报。如果失败，会在下次启动选择合适的时机上报。

4、为什么我完成了Bugly集成，在页面还是看不到日志？

请检查：

- 1) AppId是否正确（若设置错误，请更正后卸载App重试）；
- 2) 初始化SDK是否在Crash之前完成；
- 3) 网络是否可用；
- 4) 测试时如果之前有上报突然不上报了，可能是因为Bugly有流量保护，请卸载App后再试（它并不影响真实用户Crash准确度）；

以上检查OK，仍然没有上报？请直接联系我们的客服人员。

5、每个版本都要配置符号表吗？

是的。每个App版本都需要对应一份符号表（Java的Mapping文件及SO的Symbol文件，apple++的dsym文件），配置只对设置的版本有效，重复配置将会覆盖。

6、不配置还原符号表会影响异常上报吗？会有什么影响？

不会影响异常上报！没有符号表，堆栈无法还原成代码中的类或方法及源文件行号，会对异常合并存在一定影响。

7、配置了还原符号表，为什么显示日志仍然没有行号信息？

您的行号信息有可能在编译或混淆apk的时候已经丢失了。符号表中是没有行号信息的。详情请参考：<http://bugly.qq.com/blog/?p=110#>

8、用了Bugly的库应用启动不了，提示发生UnsatisfiedLinkError异常？

通常是因为安装包中各CPU架构目录下所需要的动态库缺少导致的。举个例子：
libs目录下存在：

```
armeabi\ libA.so , libBugly.so  
armeabi-v7a\ libBugly.so
```

如果App运行在armv7的设备上，系统发现有armeabi-v7a目录，会把armeabi-v7a目录下的SO安装到系统，此时程序运行会发现找不到libA.so，就会上报UnsatisfiedLinkError了。

解决办法：

提供armv7编译的libA.so放入armeabi-v7a下，补齐各架构所需要的SO。

把armeabi-v7a目录去掉，默认armv7下也能使用armeabi下的SO。

9、Bugly收集了设备哪些信息？有用户隐私吗？

Bugly收集的信息都是为了更真实地为开发者还原Crash场景服务的，并不涉及用户隐私信息：
Crash环境：Crash信息及线程堆栈，ROM/RAM/SD卡容量、网络/语言等状态
App信息：包名、版本、所属进程名
设备信息：IMEI等设备识别，用于判断Crash设备统计。