Bugly（Android）更多配置和FAQ

这里介绍了Bugly为您提供的一些扩展接口，以及一些您可能关心的FAQ。

更多的Bugly参数控制

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", false, strategy);
```

UserStrategy主要API：

1、设置App版本、渠道：
Bugly默认读取AndroidManifest.xml文件中VersionName作为App版本名。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
```

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
```

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：

CrashReport主要API：

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");  //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.postCatchedException(thr);  // bugly会将这个throwable上报
}
```

更准确全面的堆栈信息

注：如果您采用Gradle自动接入Bugly方式，我们的插件已经帮您自动配置了符号表，可以直接跳过本节内容。

为了快速、准确地定位App崩溃堆栈的位置，Bugly使用**符号表**对程序堆栈做解析并还原。

未配置堆栈示例：

```
C++堆栈: #00 pc 0021cdf4 /lib/libgame.so
Java堆栈: com.tencent.bizdemo.a.b()
```

配置后堆栈示例：

```
C++堆栈: #00 pc 0021cdf4 _ZNK14CAnimationNode13getCurEquipIdEv (CAnimationNode.h:51)
Java堆栈: com.tencent.bizdemo.BizActivity$4.onClick(BizActivity.java:55)
```

1、对于纯Java代码项目，只需上传混淆后生成的Mapping.txt文件即可；
2、对于含So的Native项目，除了Mapping.txt，还需要通过so手动提取Symbol符号表文件。我们提供了一个符号表生成工具，请在生成完成后将其上传至Bugly产品对应版本页面。

下载符号表工具

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是否设置正确（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，可能触发了Bugly的流量保护，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？
如果您采用Gradle自动接入Bugly方式，我们的插件已经帮您自动配置了符号表，无须任何操作。
如果您是手动接入Bugly，那么每个App版本都需要对应一份符号表（Java的Mapping文件及SO的Symbol文件），配置只对对应App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