

符号表工具Android版-使用指南

- 符号表工具Android版-使用指南
 - 1. 介绍
 - 1.1 环境要求
 - 1.2 符号表提取要求
 - 1.3 配置文件
 - 1.3 上传功能
 - 1.5 如何获取App ID和App Key
 - 2. 提取符号表文件的方法
 - 2.1 工具使用方法
 - 2.2 工具选项
 - 2.3 例子
 - 2.3.1 Debug SO的符号表生成和上传
 - 2.3.2 Mapping文件的上传
 - 3. Debug SO文件
 - 3.1 什么是Debug SO文件?
 - 3.2 如何定位Debug SO文件?
 - 3.2.1 IDE: Eclipse
 - 3.2.2 IDE: Android Sutdio
 - 3.3 如何判断是否与Crash匹配?
 - 3.4 如何查看Debug SO文件的UUID?
 - 3.5 找不到Crash对应的Debug SO文件?

1. 介绍

为了能快速并准确地定位用户App发生**Crash**的代码位置，Bugly使用符号表文件对App发生Crash的程序堆栈进行解析和还原。

举一个例子：

还原前堆栈

```
#00 pc 0021cdf4 /lib/libgame.so  
#01 pc 002abe36 /lib/libgame.so  
#02 pc 003aebec /lib/libgame.so
```

还原后堆栈

```
#00 pc 0021cdf4 _ZNK14CAnimationNode13getCurEquipIdEv (CAnimationNode.h:51)  
#01 pc 0021cdf4 _ZNSt6vectorI15NDKCallbackNodeSaISO_EE5beginEv (NDKHelper.cpp:312)  
#02 pc 0021cdf4 _ZNK6b2Vec26LengthEv (b2Math.h:121)
```

而符号表工具，正是Bugly提供给开发者提取符号表文件（.symbol）的工具。另外，Bugly提供了自动上传符号表文件的方法，请参考《[Bugly Android 符号表配置](#)》，建议使用自动上传的方式。

如果项目工程中没有Native代码，但使用了代码混淆Proguard，那么只需要上传Proguard生成的Mapping文件。该符号表工具也支持Mapping文件的上传，具体方法请参看下文。

1.1 环境要求

符号表工具的运行需要[Java运行环境](#)（Java SE Runtime Environment），JRE或JDK版本需要 ≥ 1.6 。

1.2 符号表提取要求

提取符号表需要[符号表工具](#)和

Debug SO文件（具有调试信息的SO文件，可参考下文的第三部分：“[3. Debug SO文件](#)”）。

1.3 配置文件

Bugly Android符号表工具2.5.0及以上版本增加了配置文件的解析功能，工具包中提供了一个与工具JAR同目录的默认配置文件（settings.txt）。

可以通过配置文件设置以下信息：

- Debug：调试模式开关（打印更多Log）
- Upload：上传开关
- ID：Bugly平台的App ID
- Key：Bugly平台的App key

1.3 上传功能

Bugly Android符号表工具2.5.0及以上版本增加了上传功能，并支持Mapping文件的上传。

使用上传功能时，需要指定以下信息：

- App ID（可通过配置文件指定）
- App key（可通过配置文件指定）
- App版本
- App包名

目前脚本不支持以上信息的指定，因此需要通过直接执行JAR包来使用上传功能。

1.5 如何获取App ID和App Key

- Bugly 1.0


崩溃


ANR


实时


趋势


运营


内测分发


设置

设置

产品信息 版本管理(39) 告警配置

App Name : BuglyDemo

AppID : App ID

AppKey : App Key

注册时间 : 2016-01-31 16:47:11

创建人 : 605352494

平台 : 
Android

产品类型 : 软件

LOGO :  建议上传的LOGO不小于72x72
[重新上传](#)

产品介绍 :

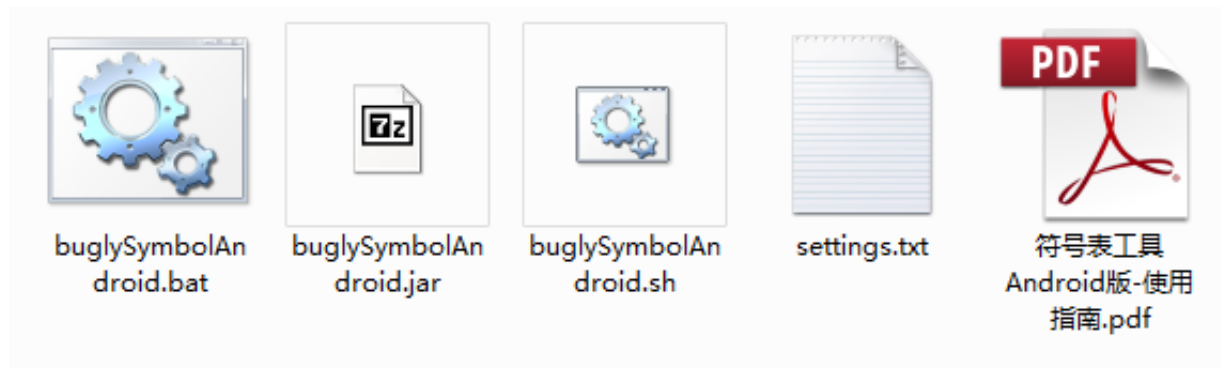
• Bugly 2.0



2. 提取符号表文件的方法

Android版符号表工具支持Windows、Linux和Mac三个平台，同时提供了JAR包、各平台脚本和符号表配置文件：

- 符号表工具JAR包 (buglySymbolAndroid.jar)
- Windows的脚本 (buglySymbolAndroid.bat)
- Shell脚本 (buglySymbolAndroid.sh)
- 默认符号表配置文件 (settings.txt)



使用脚本时，请保证脚本和jar包在同个目录下！

2.1 工具使用方法

```
1 java -jar <JAR包> <选项>
2
3 <Bat脚本> <选项>
4
5 <Shell脚本> <选项>
```

2.2 工具选项

选项	说明
-i	指定文件路径，可指定目录
-o	输出的符号表zip文件的路径，必须是zip文件
-d	调试模式开关（默认关闭）
-s	指定配置文件（默认读取JAR目录下的“settings.txt”文件）
-u	上传开关
-id	App ID
-key	App key
-package	App包名
-version	App版本
-channel	App渠道（可选）
-mapping	Mapping文件
-symbol	生成Symbol文件

注意该版本的符号表工具默认不会生成Symbol文件，如果需要请指定“-symbol”参数运行符号表工具。

2.3 例子

注意不要直接复制例子中的命令运行，需要根据自己的具体情况更改下命令。

2.3.1 Debug SO的符号表生成和上传

环境和用户信息

- 系统: **Windows**
- Android工程目录: **E:\Projects\Demo**
- 符号表工具（已解压）所在目录: **D:\Downloads\buglySymbolAndroid**
- Debug SO所在目录: **E:\Projects\Demo\obj**
- App ID: **900012345**
- App key: **abcdefghijk**
- App包名: **com.batman.demo**
- App版本: **2.3.1**

生成符号表文件

使用符号表工具的JAR包生成符号表文件的命令如下:

```
1 | cd D:\Downloads\buglySymbolAndroid
2 |
3 | java -jar buglySymbolAndroid.jar -i E:\Projects\Demo\obj
```

生成的符号表文件位于: E:\Projects\Demo\

生成符号表文件并自动上传

使用符号表工具的JAR包生成符号表文件, 并自动上传的命令如下:

```
1 | cd D:\Downloads\buglySymbolAndroid
2 |
3 | java -jar buglySymbolAndroid.jar -i E:\Projects\Demo\obj -u -id 900012345
   | -key abcdefghijk -package com.batman.demo -version 2.3.1
```

2.3.2 Mapping文件的上传

环境和用户信息

- 系统: **Windows**
- Android工程目录: **E:\Projects\Demo**
- 符号表工具（已解压）所在目录: **D:\Downloads\buglySymbolAndroid**
- Mapping文件路径: **E:\Projects\Demo\Mapping\mapping.txt**
- App ID: **900012345**

- App key: **abcdefghijk**
- App包名: **com.batman.demo**
- App渠道: **tencent**
- App版本: **2.3.1**

使用符号表工具上传Mapping文件的命令如下:

```
1 | cd D:\Downloads\buglySymbolAndroid
2 |
3 | java -jar buglySymbolAndroid.jar -mapping E:\Projects\Demo\Mapping\mapping.txt -u -id 900012345 -key abcdefghijk -package com.batman.demo -version 2.3.1 -channel tencent
```

3. Debug SO文件

3.1 什么是Debug SO文件?

Android平台中, 目标文件对应的是SO文件。Debug SO文件是指具有调试信息的SO文件。

为了方便找回Crash对应的Debug SO文件和还原堆栈, 建议每次构建或者发布App版本的时候, 备份好Debug SO文件。

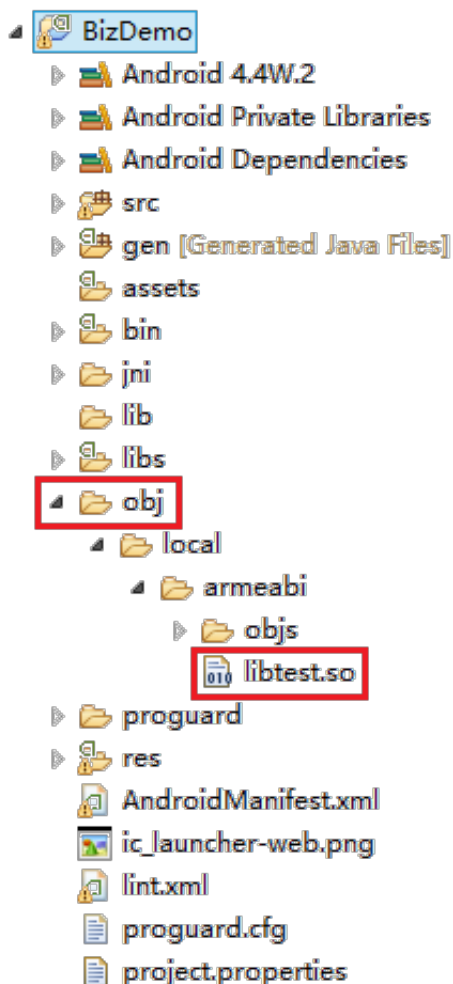
3.2 如何定位Debug SO文件?

3.2.1 IDE: Eclipse

IDE如果使用Eclipse+NDK, 默认情况下, Debug SO文件将位于:

```
<项目文件夹>/obj/local/<架构(Architecture)>/
```

如下图所示:



3.2.2 IDE: Android Sutdio

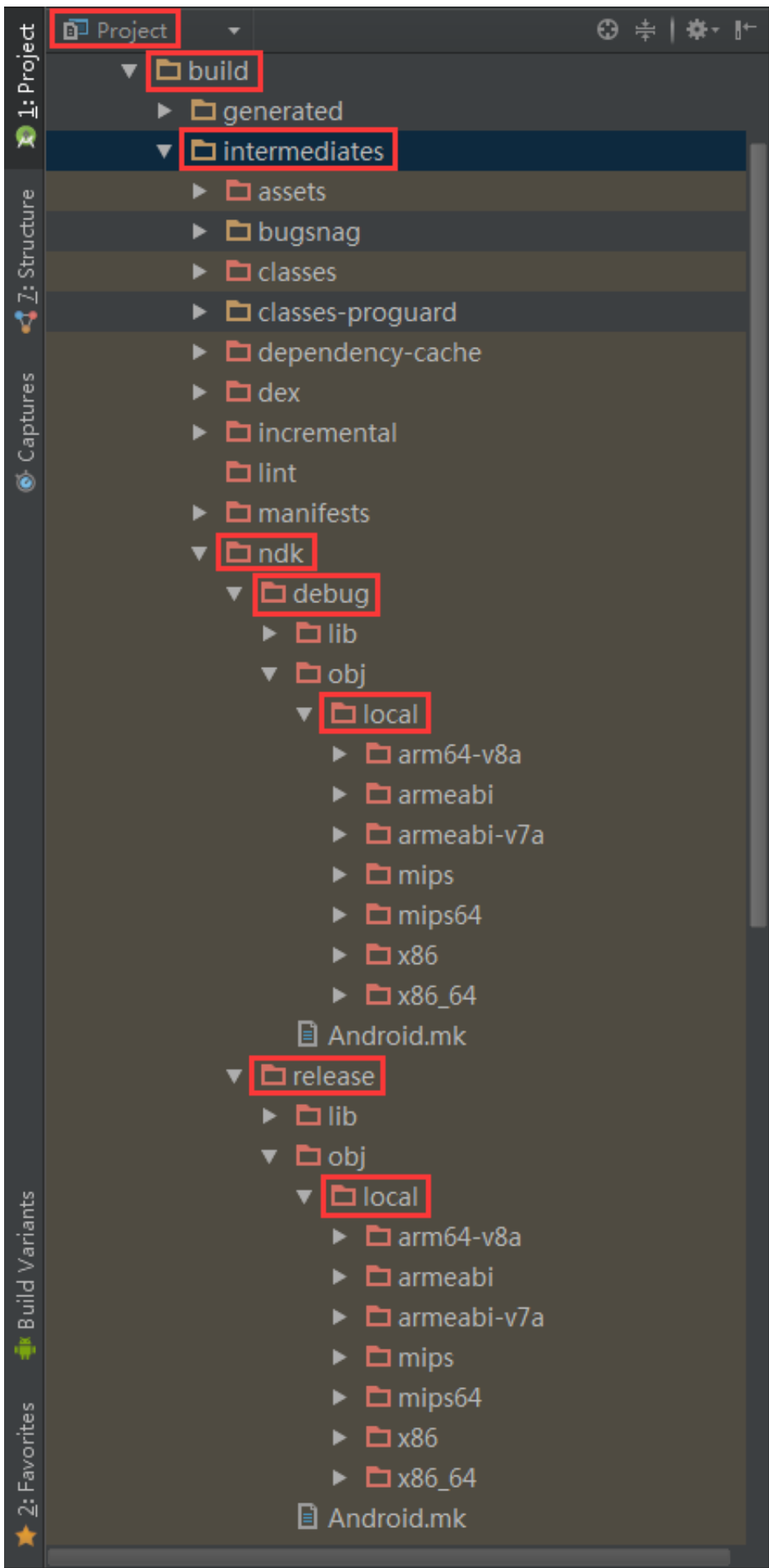
IDE如果使用Android Sutdio+NDK，默认情况下，**Debug**编译的Debug SO文件将位于：

<项目文件夹>/<Module>/build/intermediates/ndk/debug/obj/local<架构>/

而**Release**编译的Debug SO文件将位于：

<项目文件夹>/<Module>/build/intermediates/ndk/release/obj/local<架构>/

如下图所示：



3.3 如何判断是否与Crash匹配？

Bugly v2.0页面

Bugly v1.0还原Crash堆栈时，根据App包名、App版本号、SO文件名和架构来匹配符号表文件。

- 使用工具上传方式
 - 检查生成符号表文件时输入的SO文件名和架构是否和Crash堆栈的SO文件匹配；
 - 上传时指定的App包名、版本号是否和Crash对应的App包名和版本号匹配。
- 手动上传方式
 - 检查生成符号表文件时输入的SO文件名和架构是否和Crash堆栈的SO文件匹配；
 - 在版本管理页面下，检查符号表文件是否是在Crash对应App版本（包名和版本）下上传的。

Bugly v2.0页面

Bugly v2.0还原Crash堆栈时，需要根据UUID来匹配符号表文件，因此只有上传的符号表文件的UUID与Crash堆栈的SO文件的UUID一致时，才能准确地对堆栈进行还原。

- 查看符号表文件的UUID（“3.4 如何查看Debug SO文件的UUID？”）
- 查看Crash对应的App的UUID：

崩溃分析 ---> Crash issue ---> 符号表 ---> UUID

出错堆栈

跟踪数据

跟踪日志

符号表

Java符号表文件  解析规则 部分so符号表文件未上传，堆栈中的源代码类名、行号等信息可能无法正常显示

Version: 1.2.9 Channel: all 待上传

so符号表文件

libBugly.so

UUID:d79b71c72ee24eaf893d1acde2f29c17

 Arch:armeabi-v7a) 待上传

上传符号表文件

支持.zip类型文件，上限100M,超过限制请使用[符号表工具](#)上传

3.4 如何查看Debug SO文件的UUID?

```
1 | java -jar buglySymbolAndroid.jar -uuid -i <SO文件路径或目录>
```

例子:

```
[→ Downloads java -jar buglySymbolAndroid.jar -uuid -i libtest.so  
/Users/jalen/Downloads/libtest.so [armeabi-v7a] 00adc858155c5cbe7d0dfa1d2d90a21f
```

3.5 找不到Crash对应的Debug SO文件?

如果本地已经无法找到Crash对应的符号表文件或者Debug SO文件了，但还能找回Crash对应的App版本的Native工程代码，建议尝试重新用NDK编译出Debug SO文件并用符号表工具生成符号表文件。

如果连Native工程代码也无法找回了，那就真的无法还原这个Crash堆栈了。

为了防止出现这种情况，建议每次构建或者发布App版本的时候，一定要备份好Debug SO文件!