

# 听云 App 符号化使用指南

## 一、Native 符号表工具使用及上传

### 1. 介绍

为了能快速并准确地定位应用发生 Native Crash 的代码位置，听云使用符号表文件对应用崩溃堆栈进行解析和还原。

例如：

原堆栈

崩溃线程main

格式化

```
#00 pc 0000000000dcdc /data/app/com.caterwind.jnidemo-1/lib/arm64/libcater-lib.so (Java_com_caterwind_jnidemo_MainActivity_testCppCrash+28)
#01 pc 00000000006b7094 /data/app/com.caterwind.jnidemo-1/oat/arm64/base.odex (oatexec+2629780)
at com.caterwind.jnidemo.MainActivity.testCppCrash(Native Method)
at com.caterwind.jnidemo.MainActivity$14.onClick(MainActivity.java:199)
at android.view.View.performClick(View.java:5205)
at android.view.View$PerformClick.run(View.java:21176)
at android.os.Handler.handleCallback(Handler.java:739)
at android.os.Handler.dispatchMessage(Handler.java:95)
at android.os.Looper.loop(Looper.java:171)
at android.app.ActivityThread.main(ActivityThread.java:5682)
at java.lang.reflect.Method.invoke(Native Method)
at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:732)
at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:622)

binaryImages:
| file name | sha-1 | cpu arch | type |
libcater-lib.so c5f03a44158510b1d17304fe72845ca9 arm64-v8a 1
```

还原后堆栈

崩溃线程main

格式化

```
#00 pc 0000000000dcdc libcater-lib.so Java_com_caterwind_jnidemo_MainActivity_testCppCrash+28 (E:\osapp201807_JNIDemo\app\src\main\cpp\native-lib.cpp:18)
m64-v8a]
#01 pc 00000000006b7094 /data/app/com.caterwind.jnidemo-1/oat/arm64/base.odex (oatexec+2629780)
at com.caterwind.jnidemo.MainActivity.testCppCrash(Native Method)
at com.caterwind.jnidemo.MainActivity$14.onClick(MainActivity.java:199)
at android.view.View.performClick(View.java:5205)
at android.view.View$PerformClick.run(View.java:21176)
at android.os.Handler.handleCallback(Handler.java:739)
at android.os.Handler.dispatchMessage(Handler.java:95)
at android.os.Looper.loop(Looper.java:171)
at android.app.ActivityThread.main(ActivityThread.java:5682)
at java.lang.reflect.Method.invoke(Native Method)
at com.android.internal.os.ZygoteInit$MethodAndArgsCaller.run(ZygoteInit.java:732)
at com.android.internal.os.ZygoteInit.main(ZygoteInit.java:622)

binaryImages:
| file name | sha-1 | cpu arch | type |
libcater-lib.so a798aee6832fe0d58e3b28308e903733 arm64-v8a 1
```

符号表工具 nbs.newlens.so.parser.jar，是听云 App 提供给开发者提取符号表文件的工具。

### 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）。

#### 2.1 配置文件

符号表工具使用 tinglyun.properties 文件作为配置文件，需要配置以下信息：

```
authKey=*听云 API 账号授权 Key，由报表系统生成*
appKey=*听云 AppKey*
```

#### 2.2 工具选项

选项	说明
-i	指定 so 文件夹路径
-s	指定配置文件（默认读取 JAR 包目录下的 tinglyun.properties

	文件)
-u	上传开关

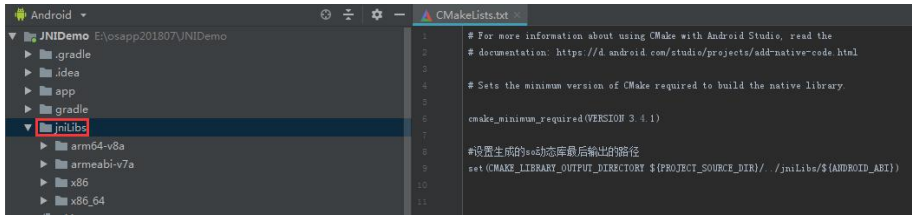
2.3 生成符号表文件 NewlensSymbol.zip

使用符号表工具的 JAR 包生成符号表文件的命令如下：

```
java -jar nbs.newlens.so.parser.jar -i E:\JNIDemo\jniLibs
```

生成的符号表文件 NewlensSymbol.zip 位于符号表工具目录下。

注：听云上传符号表文件仅支持 zip 格式，-i 参数请指定生成 so 文件夹的路径。该路径一般默认为 app\build\intermediates\cmake\debug\obj，也可以通过 CMakeLists.txt 文件配置输出到其他目录。



3. 上传符号表

听云提供了三种方式上传符号表文件。

3.1 使用符号表工具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：

```
java -jar nbs.newlens.so.parser.jar -i E:\JNIDemo\jniLibs -u -s E:\JNIDemo\tingyun.properties
```

3.2 使用报表上传符号表文件

3.2.1 报表中选择“崩溃”模块，在“崩溃历史记录列表”打开“Native symbol 文件管理”

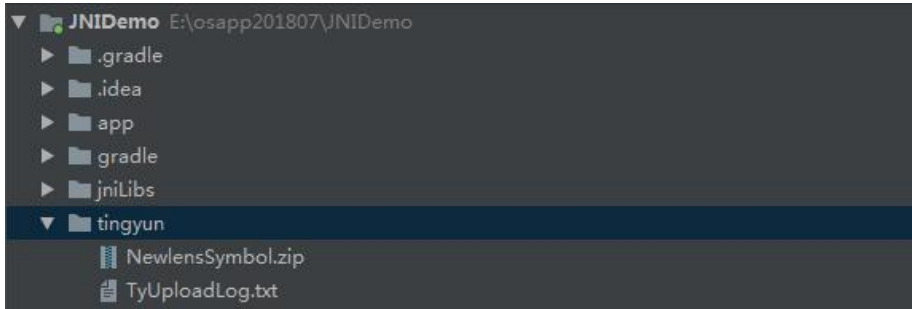


3.2.2 上传 NewlensSymbol.zip 文件即可。



3.3 自动上传符号表文件（需 2.11.1 以上版本）

3.3.1 听云默认在 release 编译时于创建 tingyun 目录生成符号表并自动上传



3.3.2 若需要在 debug 时生成符号表并上传，需在项目 app 目录下的 build.gradle 文件中，添加配置

```
newlensExt{
    isDebug = true
}
```

3.3.3 若不希望听云上传符号表，需在项目 app 目录下的 build.gradle 文件中，添加配置

```
newlensExt{
    //默认开启，若置为 false，则不上传 native 符号表及 mapping 文件
    mapUploadEnabled = false
}
```

## 二、上传 mapping 文件

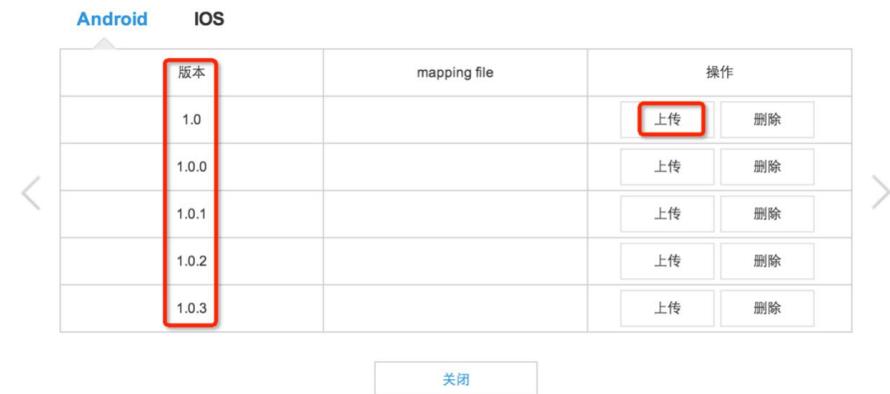
听云 SDK 提供了两种方式上传 mapping 文件

### 1. 通过报表上传 mapping 文件

1.1 报表中选择“崩溃”模块，在“崩溃历史记录列表”打开“dSYM/Mapping 文件管理”



1.2 找到所需版本，并上传本地 mapping 文件即可



### 2. 通过 tingyun.properties 文件自动上传 mapping 文件

注：若您的项目不存在 tingyun.properties 文件，需在项目 app 目录及根目录下新建该文件

2.1 在 tingyun.properties 文件中配置

```
authKey=*听云 API 账号授权 Key，由报表系统生成*
appKey=*听云 AppKey*
mapping_file_auto_upload=true
```

2.2 开启混淆器

mapping\_file\_auto\_upload 控制开关只有在启用混淆器的时候才会生效，开启控制开关后，听云 SDK 会将本地目录下的 mapping 文件自动上传到听云服务器。若未开启混淆器则该配置项不生效。