

Bugly (Android) SDK 快速接入向导

Bugly文档资料

一、库文件导入

如果您使用Gradle编译Apk，我们强烈推荐您使用自动接入方式配置库文件。Bugly支持[JCenter仓库](#)和[Maven仓库](#)。

方式1. 自动导入 (推荐)

为了实现更加灵活的配置，Bugly SDK (2.1.5及以上版本) 和NDK (SO库) 目前已经分开成两个独立的仓库：

- SDK : com.tencent.bugly:crashreport
- NDK : com.tencent.bugly:nativecrashreport

其中，集成Bugly NDK时，需要同时集成Bugly SDK。

单独集成Bugly SDK

在Module的build.gradle文件中添加依赖和属性配置：

```
dependencies {  
    compile 'com.tencent.bugly:crashreport:latest.release' //其中latest.release指代最新版本号，也可以指定明确的版本号，例如2.1.5  
}
```

同时集成Bugly SDK和NDK

在Module的build.gradle文件中添加依赖和属性配置：

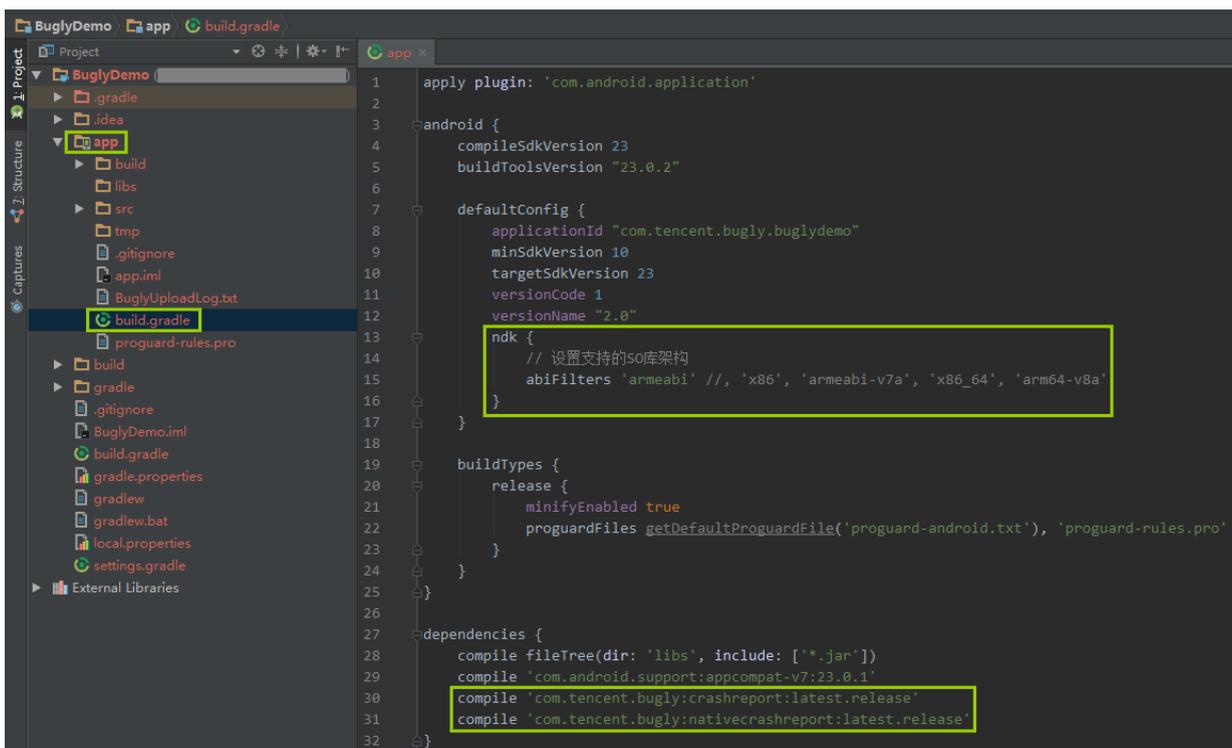
```

android {
    defaultConfig {
        ndk {
            // 设置支持的SO库架构
            abiFilters 'armeabi' //, 'x86', 'armeabi-v7a', 'x86_64', 'arm64-v8a'
        }
    }
}

dependencies {
    compile 'com.tencent.bugly:crashreport:latest.release' //其中latest.release指代最新版本号，也可以指定明确的版本号，例如2.1.5
    compile 'com.tencent.bugly:nativecrashreport:latest.release' //其中latest.release指代最新版本号，也可以指定明确的版本号，例如2.2.0
}

```

同时集成Bugly SDK和NDK的配置如下图所示，后续更新Bugly SDK和NDK时，只需变更配置脚本中的版本号即可。



注意：自动集成时会自动包含Bugly SO库，建议在Module的build.gradle文件中使用NDK的“abiFilter”配置，设置支持的SO库架构。

如果在添加“abiFilter”之后Android Studio出现以下提示：

NDK integration is deprecated in the current plugin. Consider trying the new experimental plugin.

则在项目根目录的gradle.properties文件中添加：

```
android.useDeprecatedNdk=true
```

方式2. 手动导入

如果您不采用上述自动导入方式，也可以手动集成Bugly SDK。

下载Bugly库文件

- 下载Bugly的[Android SDK包](#)；
- 如果您的工程有Native代码（C/C++）或者集成了其他第三方SO库，建议下载Bugly的[NDK动态库](#)。

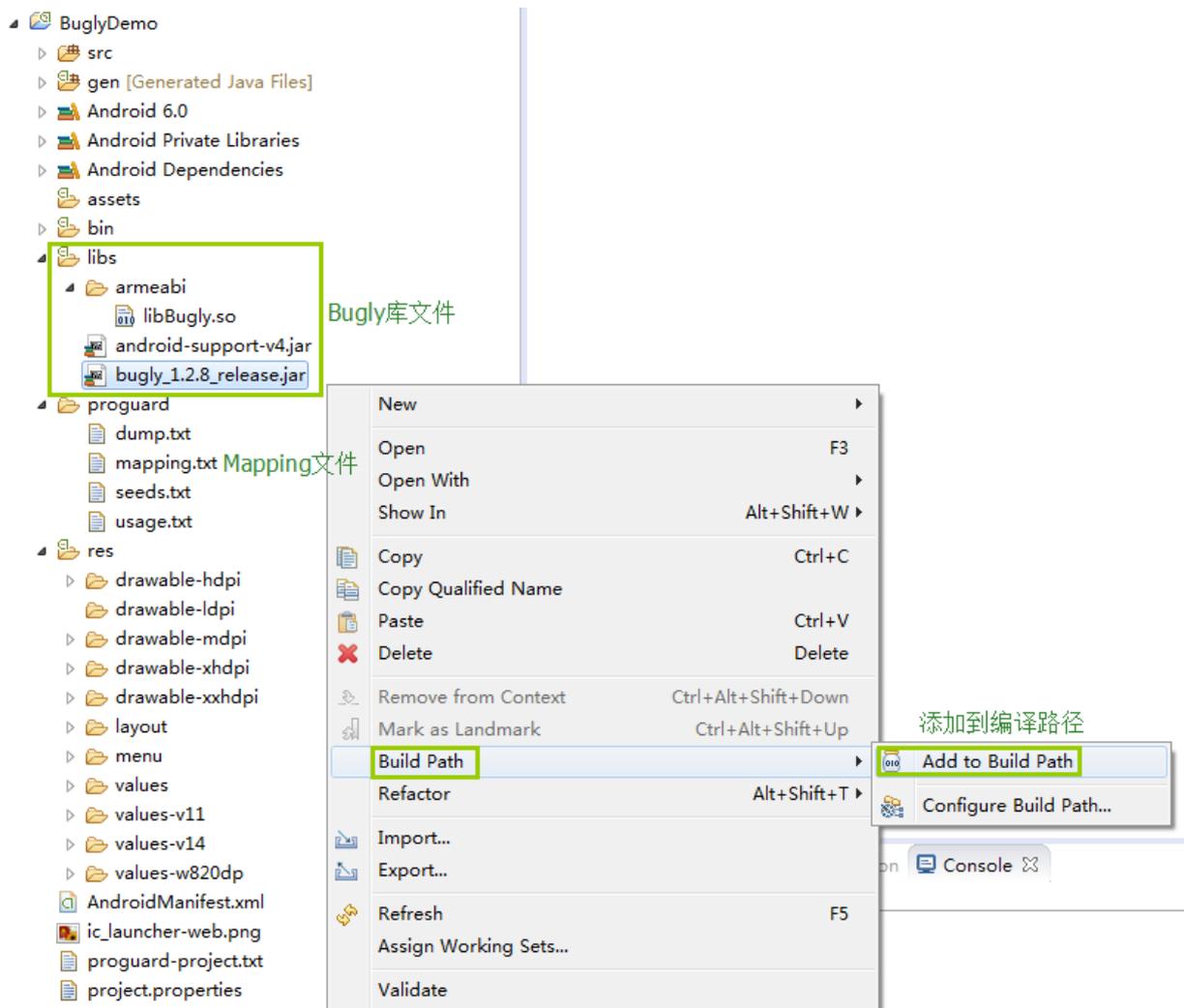
Bugly NDK包含多个架构的SO库：

- armeabi
- armeabi-v7a
- arm64-v8a
- x86
- x86_64

在集成Bugly SO库时，请注意只保留支持的架构SO库。

Eclipse工程

- 将Bugly库文件复制到工程的libs目录下；
- Refresh一下工程；
- 添加工程依赖：鼠标右键点击Bugly的JAR文件，添加到编译路径中。

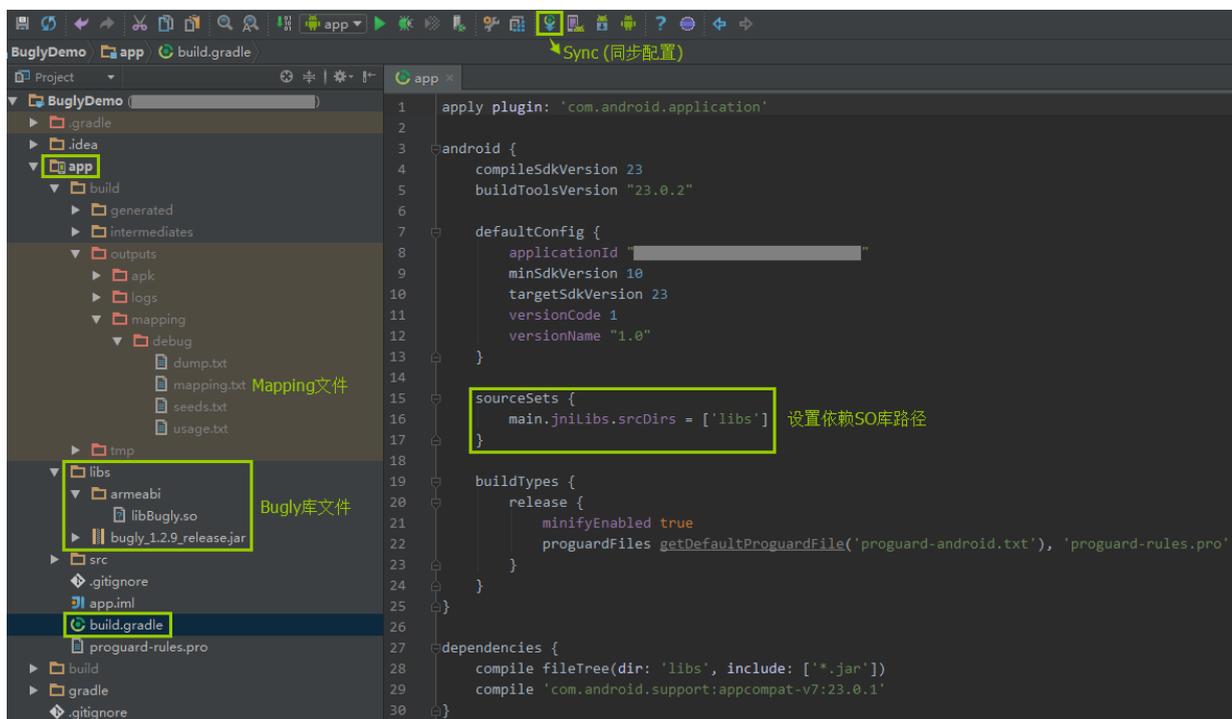


Android Studio工程

- 将Bugly库文件复制到工程的libs目录下；
- 如果集成Bugly NDK，则在Module的build.gradle文件中添加SO库目录配置：

```
android {  
    sourceSets {  
        main.jniLibs.srcDirs = ['libs']  
    }  
}
```

- 点击Sync，同步配置。



为了使APP Crash堆栈的可读性更高，建议您配置符号表文件，更准确地定位问题：

- 纯Java代码的工程：只需要配置混淆后生成的Mapping文件即可；
- 含有Native代码的工程：建议配置符号表工具从Debug SO中提取的Symbol符号表文件。

Bugly支持手动和自动配置两种方式，具体的配置方法请参考：[《Bugly Android符号表配置》](#)

二、参数配置

- 在AndroidManifest.xml中添加权限：

```
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<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_LOGS" />
```

- 请避免混淆Bugly，在Proguard混淆文件中增加以下配置：

```
-dontwarn com.tencent.bugly.**  
-keep public class com.tencent.bugly.**{*;}}
```

三、最简单的初始化

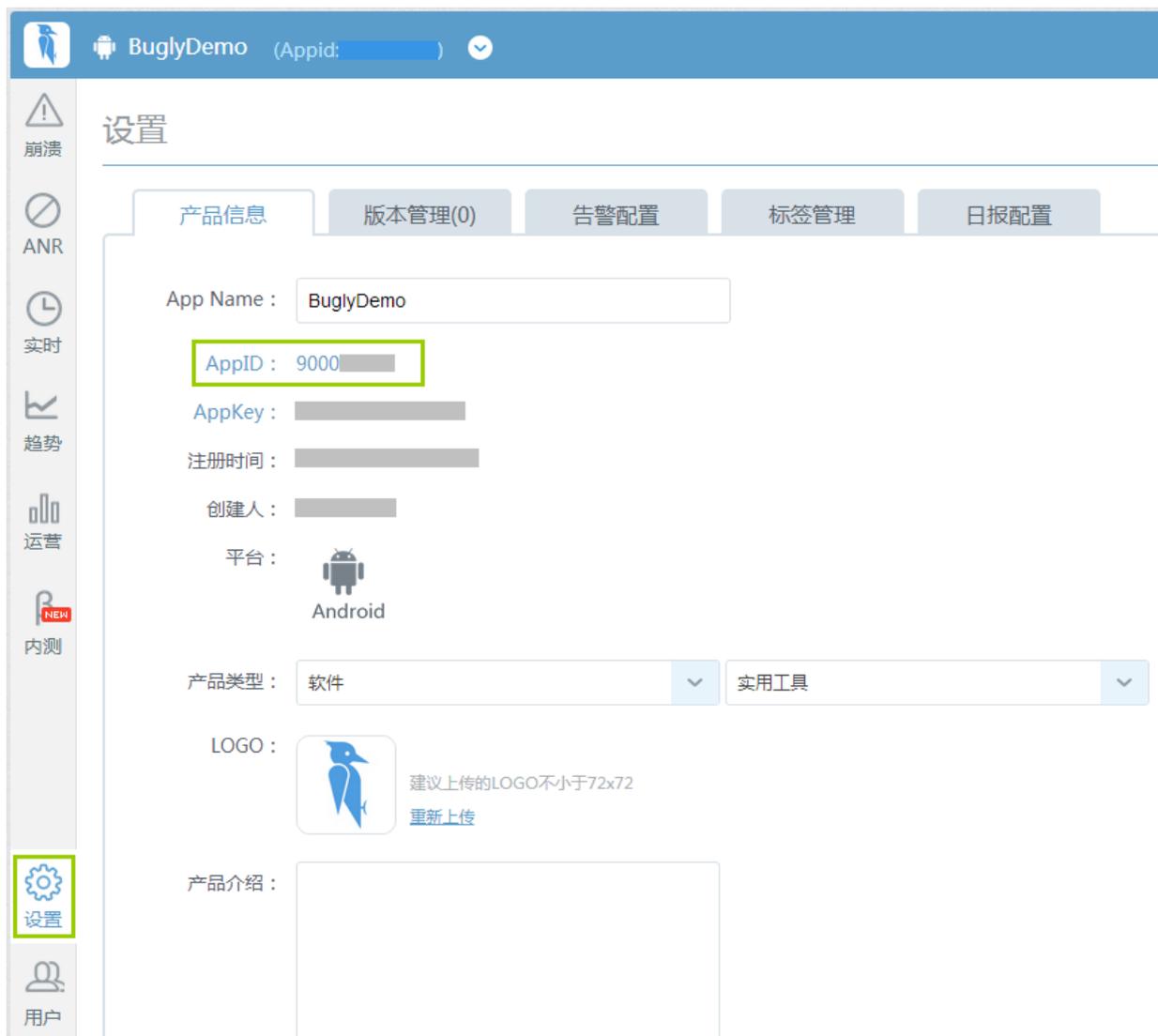
获取APP ID并将以下代码复制到项目Application类onCreate()中，Bugly会为自动检测环境并完成配置：

```
CrashReport.initCrashReport(getApplicationContext(), "注册时申请的APP  
ID", false);
```

第三个参数为SDK调试模式开关，调试模式的行为特性如下：

- 输出详细的Bugly SDK的Log
- 每一条Crash都会被立即上报
- 自定义日志将会在Logcat中输出

建议在测试阶段建议设置成true，发布时设置为false。



此外，Bugly2.0及以上版本还支持通过“AndroidManifest.xml”来配置APP信息。如果同时又通过代码中配置了APP信息，则最终以代码配置的信息为准。

在“AndroidManifest.xml”的“Application”中增加“meta-data”配置项：

```
<application
  <!-- 配置APP ID -->
  <meta-data
    android:name="BUGLY_APPID"
    android:value="<APP ID>" />
  <!-- 配置APP版本号 -->
  <meta-data
    android:name="BUGLY_APP_VERSION"
    android:value="<APP Version>" />
  <!-- 配置APP渠道号 -->
  <meta-data
    android:name="BUGLY_APP_CHANNEL"
    android:value="<APP Channel>" />
  <!-- 配置Bugly调试模式（true或者false） -->
  <meta-data
    android:name="BUGLY_ENABLE_DEBUG"
    android:value="<isDebug>" />
</application>
```

不同于“android:versionName”，“BUGLY_APP_VERSION”配置的是Bugly平台的APP版本号。

通过“AndroidManifest.xml”配置后的初始化方法如下：

```
CrashReport.initCrashReport(getApplicationContext());
```

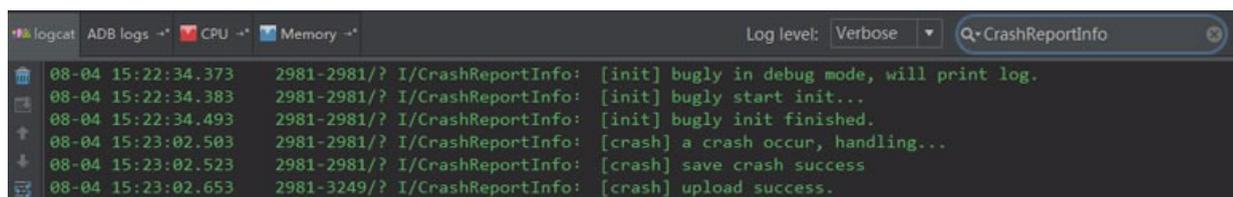
Bugly默认从“AndroidManifest.xml”文件中读取“VersionName”作为版本号，自定义设置请使用参考[“高级设置”](#)

四、测试

现在您可以制造一个Crash（建议通过“按键”来触发），来体验Bugly的能力了。在初始化Bugly的之后，调用Bugly测Java Crash接口。

```
CrashReport.testJavaCrash();
```

执行到这段代码时会发生一个Crash，Logcat的TAG=CrashReportInfo中输出为：



```
08-04 15:22:34.373 2981-2981/? I/CrashReportInfo: [init] bugly in debug mode, will print log.
08-04 15:22:34.383 2981-2981/? I/CrashReportInfo: [init] bugly start init...
08-04 15:22:34.493 2981-2981/? I/CrashReportInfo: [init] bugly init finished.
08-04 15:23:02.503 2981-2981/? I/CrashReportInfo: [crash] a crash occur, handling...
08-04 15:23:02.523 2981-2981/? I/CrashReportInfo: [crash] save crash success
08-04 15:23:02.653 2981-3249/? I/CrashReportInfo: [crash] upload success.
```

现在您已经可以在“崩溃”页面看到刚才触发的Crash issue了（延迟一般在10s以内）。

