

北京基调网络股份有限公司

# 听云平台部署说明

听云 APP Android SDK Gradle 部署说明

Chengmy  
2018/5/25



TINGYUN.COM

# 前言

## 产品版本

与本手册相对应的产品版本如下所示。

产品名称	产品版本	手册版本
Android 探针	2.4.4	V1.0
Android 探针	2.5.0	V2.0
Android 探针	2.5.5	V2.1
Android 探针	2.5.7	V2.2
Android 探针	2.5.9	V2.3
Android 探针	2.7.0	V2.4
Android 探针	2.7.1	V2.5
Android 探针	2.8.0	V2.6
Android 探针	2.8.1	V2.7
Android 探针	2.9.0	V2.8
Android 探针	2.9.1	V2.9

## 内容介绍

本手册主要介绍了 Android SDK 探针的 Gradle 部署方法及步骤。

## 读者对象

本手册适用于以下人员：

- Android 开发工程师
- Android 测试工程师
- 网络监控工程师
- 系统维护工程师

## 约定

### 符号约定

在本文中可能出现下列标志，它们所代表的含义如下。

符号	说明
	以本标志开始的文本表示有潜在风险, 如果忽视这些文本, 可能导致设备损坏、数据丢失、设备性能降低或不可预知的结果。
	以本标志开始的文本是正文的附加信息, 是对正文的强调和补充。

## 修改记录

修订记录累积了每次文档更新的说明。最新版本的文档包含以前所有版本的更新内容。

### 版本 V2.9 (2018-05-25)

- 修改

位置	描述
简介	更新内容, 新增性能消耗及兼容性说明
使用 Gradle 构建	“点选” 更名为 “可视化操作命名”
配置应用权限	增加一个权限
插入初始化探针代码	新增 WebView 嵌码相关内容
配置本地文件	去掉 WebView 嵌码相关内容
FAQ	“其他注意事项” 更名为 “FAQ” 章节, 并更新内容

### 版本 V2.8 (2018-04-25)

- 修改

位置	描述
简介	OKHttp 版本支持 >2.4.0 版本
配置应用权限	新增悬浮窗权限
Gradle 方式部署	新增点选功能配置
配置本地文件	更新配置 AuthKey 内容, 移除项目嵌码控制开关
其他注意事项	更新注意事项

## 版本 V2.7 (2018-01-29)

- 修改

位置	描述
简介	探针支持安卓版本改为 Android4.0 - Android8.0
插入初始化探针代码	新增只采集主进程的初始化方法
其他注意事项	更新注意事项

## 版本 V2.6 (2018-01-12)

- 修改

位置	描述
Android SDK 体积增量	增加体积大小改为 600KB 左右
其他注意事项	新增其他注意事项章节

## 版本 V2.5 (2017-11-27)

- 修改

位置	描述
使用 Gradle 方式构建	移除 2.1.5 在 MainActivity 中初始化 SDK
使用 Gradle 方式构建	新增 Cordova 及 React Native 工程相关依赖构建

## 版本 V2.4 (2017-09-25)

- 修改

位置	描述
配置本地文件	移除配置 OkHttp 开关相关内容，增加项目嵌码控制开关

## 版本 V2.3 (2017-07-27)

- 修改

位置	描述
配置 OKHttp 控制开关	OKHttp2.0 系列请使用 2.1.0 以上版本
配置 WebView 控制开关	通过调用接口嵌码 WebView

## 版本 V2.2 (2017-06-22)

- 新增

位置	描述
高级功能	控制台版本提示

## 版本 V2.1 (2017-05-25)

- 修改

位置	描述
安装探针	无需添加 SO 库即可收集 TCP 数据
高级功能	用户自定义 ID 修改接口限制

## 版本 V2.0 (2016-10-27)

- 新增

位置	描述
简介	简介
高级功能	高级功能

## 版本 V1.0 (2014-06-18)

手册初次发行



# 目录

前言.....	1
产品版本.....	1
内容介绍.....	1
读者对象.....	1
约定.....	1
符号约定.....	1
修改记录.....	2
版本 V2.9 (2018-05-25) .....	2
版本 V2.8 (2018-04-25) .....	2
版本 V2.7 (2018-01-29) .....	3
版本 V2.6 (2018-01-12) .....	3
版本 V2.5 (2017-11-27) .....	3
版本 V2.4 (2017-09-25) .....	3
版本 V2.3 (2017-07-27) .....	3
版本 V2.2 (2017-06-22) .....	4
版本 V2.1 (2017-05-25) .....	4
版本 V2.0 (2016-10-27) .....	4
版本 V1.0 (2014-06-18) .....	4
目录.....	5
<b>1 简介.....</b>	<b>7</b>
1.1 ANDROID SDK 目录结构.....	8
1.2 ANDROID SDK THEORY.....	8
1.3 ANDROID SDK 体积增量.....	9
<b>2 GRADLE 方式部署.....</b>	<b>10</b>
2.1 使用 GRADLE 构建.....	10
2.1.1 Android Studio 工程相关依赖构建.....	10
2.1.2 Cordova 工程相关依赖构建.....	12
2.1.3 React Native 工程相关依赖构建.....	14
2.1.4 配置应用权限.....	15
2.1.5 可视化操作命名功能配置(选配).....	16
2.1.6 插入初始化探针代码.....	17
2.1.7 使用 Gradle 命令打包编译.....	18
2.1.8 配置混淆.....	18
2.2 嵌码完整性校验.....	18
<b>3 高级功能.....</b>	<b>19</b>



3.1 用户自定义 ID.....	19
3.2 面包屑.....	19
3.3 自定义 EVENT.....	20
3.4 自定义 TRACE.....	20
3.5 自定义 LOG.....	21
3.6 自定义附加信息.....	21
3.7 控制台版本提示.....	22
<b>4 设置本地配置文件.....</b>	<b>22</b>
4.1 新建 TINGYUN.PROPERTIES 配置文件.....	22
4.2 配置 CRASH 反混淆 (MAPPING) .....	22
4.2.1 通过报表上传 mapping 文件.....	23
4.2.2 通过 <i>tingyun.properties</i> 文件自动上传 mapping 文件.....	23
<b>5 FAQ.....</b>	<b>24</b>

# 1 简介

---

Android SDK 探针支持如下数据采集：

- Http/Https 协议数据收集
- 崩溃数据收集（java 异常）
- ANR 数据收集（安卓 5.0 及以下）
- 事件性能数据收集
- 进程内存和 CPU 使用率

Android SDK 探针支持如下协议类库：

- HttpURLConnection
- Android HttpClient 4.0
- Apache HttpClient (> 4.0)
- Volley+OkHttpClient
- OkHttp (> 2.4.0)
- Retrofit
- WebView（原生，支持 Android4.4 及以上版本）

Android SDK 探针支持如下系统版本：

- Android 4.0 版本~Android8.0 版本

用户体验分析采集应用启动及用户操作数据。

应用启动数据采集需实现自定义的 Application 类。

- 启动耗时
- 启动次数
- 启动期间的崩溃、事件性能数据

用户操作数据采集点击事件（onClick、onItemClick、ItemSelected、PageSelected 等方法）。

- 操作耗时
- 阻塞耗时
- 操作次数
- 操作期间的崩溃、事件性能数据

Android SDK 采集应用运行时发生的异常。

- 崩溃
- 自定义异常上传

Android SDK 采集应用运行时发生的 ANR。

- ANR
- 卡顿

Android SDK 采集应用的网络数据。

- 应用响应时间
- DNS 解析时间
- TCP 建连时间
- SSL 握手时间
- 首包时间
- 剩余包时间
- 访问量

Android SDK 探针也可以采集线程堆栈、数据库、自定义参数和 HTTP 请求参数。

## 1.1 Android SDK 目录结构

- nbs-newlens-android-sdk
  - nbs.newlens.agent.jar
  - nbs.newlens.class.rewriter.jar

## 1.2 Android SDK Theory

```
protected void onCreate(Bundle bundle) {
    APM.startTracing(getClass().getName());
    try {
        APM.enterMethod(this._nbs_trace, "Activity#onCreate", null);
    } catch (NoSuchFieldError e) {
        while (true) {
            APM.enterMethod(null, "Activity#onCreate", null);
        }
    }
    super.onCreate(bundle);
    this.detector = new GestureDetector((GestureDetector.OnGestureListener) this);
    APM.exitMethod();
}
```

探针开始, 进行数据采集

探针结束, 完成数据采集

听云 SDK 通过虚拟机技术在应用打包编译过程中对应用采样点（Http 标准协议和 Https 标准协议）进行嵌码操作，该操作会在协议类库方法前后部署听云 SDK 探针，该过程不会影响用户代码逻辑。

每当 App 启动时，听云 Agent 开始工作。应用有网络请求时，通过之前部署的听云 SDK 探针以一定的采集频率来采集数据，并对采集的数据进行汇总后，上传到服务器（报表展现）。

应用退出到后台或用户关闭 App 时，听云 Agent 停止工作，以便减少不必要的流量消耗。



## 1.3 Android SDK 体积增量

- 应用 App 嵌码后体积增量为 600KB 左右。

## 2 Gradle 方式部署

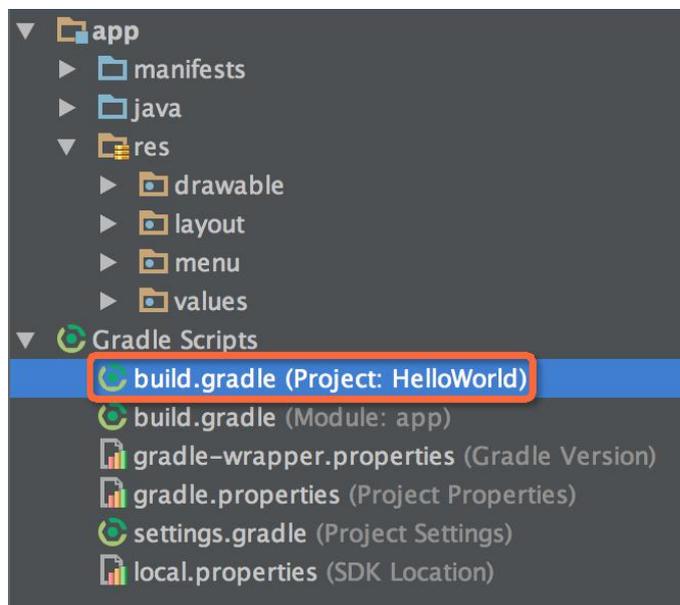
### 2.1 使用 Gradle 构建

#### 2.1.1 Android Studio 工程相关依赖构建



操作员需要确保已经安装了 Gradle 构建环境和 AS 开发环境

1. 打开项目根目录下的 build.gradle (Project) 文件



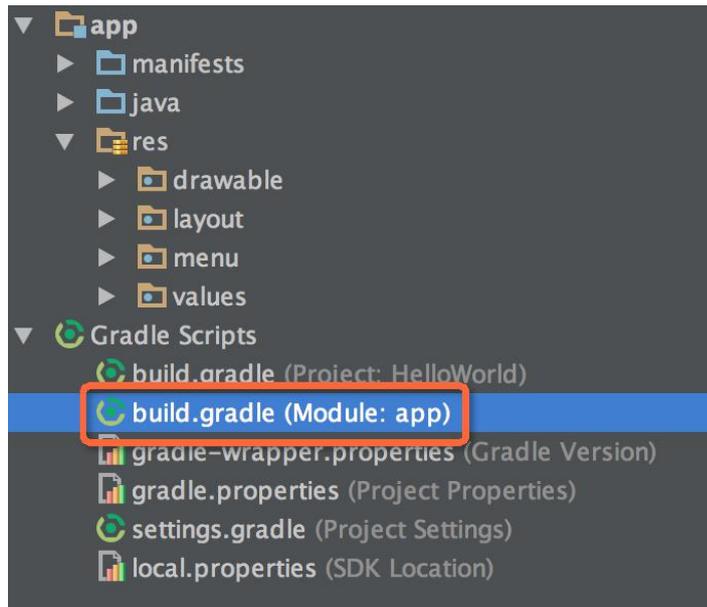
2. 在 buildscript 模块中加入代码

```
classpath
'com.networkbench.newlens.agent.android:agent-gradle-plugin:TingYun_Version
'//TingYun_Version 为当前听云 APP 版本号

buildscript {
    repositories {
        jcenter()
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:1.0.0'
        classpath "com.networkbench.newlens.agent.android:agent-gradle-plugin:2.2.5"
    }
}

allprojects {
    repositories {
        jcenter()
    }
}
```

3. 打开项目工程主模块下的 build.gradle (Module) 文件



4. 在文件中引入 mavenCentral()

```
repositories {
    mavenCentral()
}
```



```
repositories {
    mavenCentral()
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:21.0.3'
}
```

5. 在 dependencies 模块中添加代码

```
compile
"com.networkbench.newlens.agent.android:nbs.newlens.agent:TingYun_Version"
// TingYun_Version 为当前听云 APP 版本号
```



```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:21.0.3'
    compile "com.networkbench.newlens.agent.android:nbs.newlens.agent:2.2.5"
}
```

6. 添加听云符号表插件

```
apply plugin: 'newlens'
```

```

apply plugin: 'android'
apply plugin: 'newlens'

android {
    compileSdkVersion 21
    buildToolsVersion "21.1.2"

    defaultConfig {
        applicationId "com.hello.nbs.helloworld"
        minSdkVersion 14
        targetSdkVersion 21
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'), 'proguard-rules.pro'
        }
    }
}

repositories {
    mavenCentral()
}

dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.android.support:appcompat-v7:21.0.3'
    compile "com.networkbench.newLens.agent.android:nbs.newLens.agent:2.2.5"
}
    
```

## 2.1.2 Cordova 工程相关依赖构建

1. 打开项目 android 目录下的 build.gradle 文件

```

16     specific language governing permissions and limitations
17     under the License.
18     */
19
20     apply plugin: 'com.android.application'
21     apply plugin: 'newlens'
22
23     buildscript {
24         repositories {
25             jcenter()
26             maven {
27                 url "https://maven.google.com"
28             }
29         }
30     }
31
32     // Switch the Android Gradle plugin version requirement depending on the
33     // installed version of Gradle. This dependency is documented at
34     // http://tools.android.com/tech-docs/new-build-system/version-compatibility
35     // and https://issues.apache.org/jira/browse/CB-8143
36     dependencies {
37         classpath 'com.android.tools.build:gradle:2.2.3'
38         classpath 'com.networkbench.newLens.agent.android:agent-gradle-plugin:2.7.0'
39     }
40 }
41
42 // Allow plugins to declare Maven dependencies via build-extras.gradle.
43 allprojects {
44     repositories {
45         jcenter()
46         maven {
47             url "https://maven.google.com"
48         }
49     }
50 }
51
52 task wrapper(type: Wrapper) {
53     gradleVersion = '2.14.1'
54 }
55
    
```

2. 添加代码

```
apply plugin: 'newlens'
```

```
apply plugin: 'com.android.application'  
apply plugin: 'newlens'  
  
buildscript {  
    repositories {  
        jcenter()  
        maven {  
            url "https://maven.google.com"  
        }  
    }  
}
```

3. 在 buildscript 中添加以下代码

```
classpath  
'com.networkbench.newlens.agent.android:agent-gradle-plugin:TingYun_Version'  
//TingYun_Version 为当前听云 APP 版本号
```

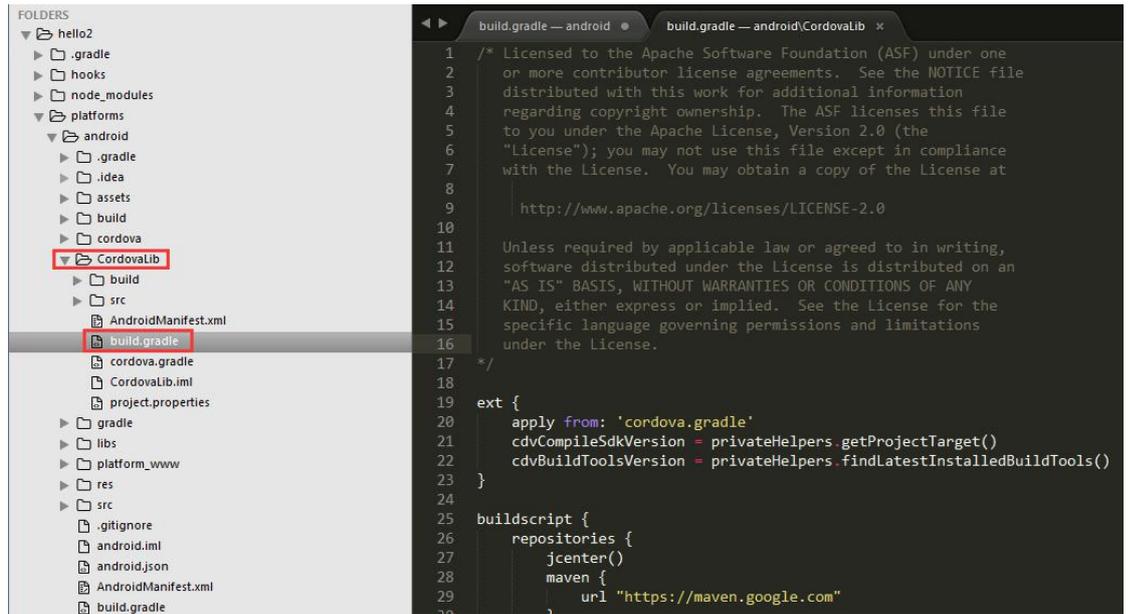
```
buildscript {  
    repositories {  
        jcenter()  
        maven {  
            url "https://maven.google.com"  
        }  
    }  
  
    dependencies {  
        classpath 'com.android.tools.build:gradle:2.2.3'  
        classpath 'com.networkbench.newlens.agent.android:agent-gradle-plugin:2.7.0'  
    }  
}
```

4. 在 dependencies 中添加以下代码

```
compile  
'com.networkbench.newlens.agent.android:nbs.newlens.agent:TingYun_Version' /  
/TingYun_Version 为当前听云 APP 版本号
```

```
dependencies {  
    compile 'com.networkbench.newlens.agent.android:nbs.newlens.agent:2.7.0'  
    compile fileTree(dir: 'libs', include: '*.jar')  
    // SUB-PROJECT DEPENDENCIES START  
    debugCompile(project(path: "CordovaLib", configuration: "debug"))  
    releaseCompile(project(path: "CordovaLib", configuration: "release"))  
    // SUB-PROJECT DEPENDENCIES END  
}
```

5. 打开 CordovaLib 的 build.gradle 文件



## 6. 添加以下代码

```
apply plugin: 'newlens'

allprojects {
    repositories {
        mavenCentral()
        jcenter()
    }
    // mavenCentral()和jcenter()配置任意一个均可
}

dependencies {
    provided
    'com.networkbench.newlens.agent.android:nbs.newlens.agent:TingYun_Version'
    //此处为provided依赖，TingYun_Version为当前听云APP版本号
}
```

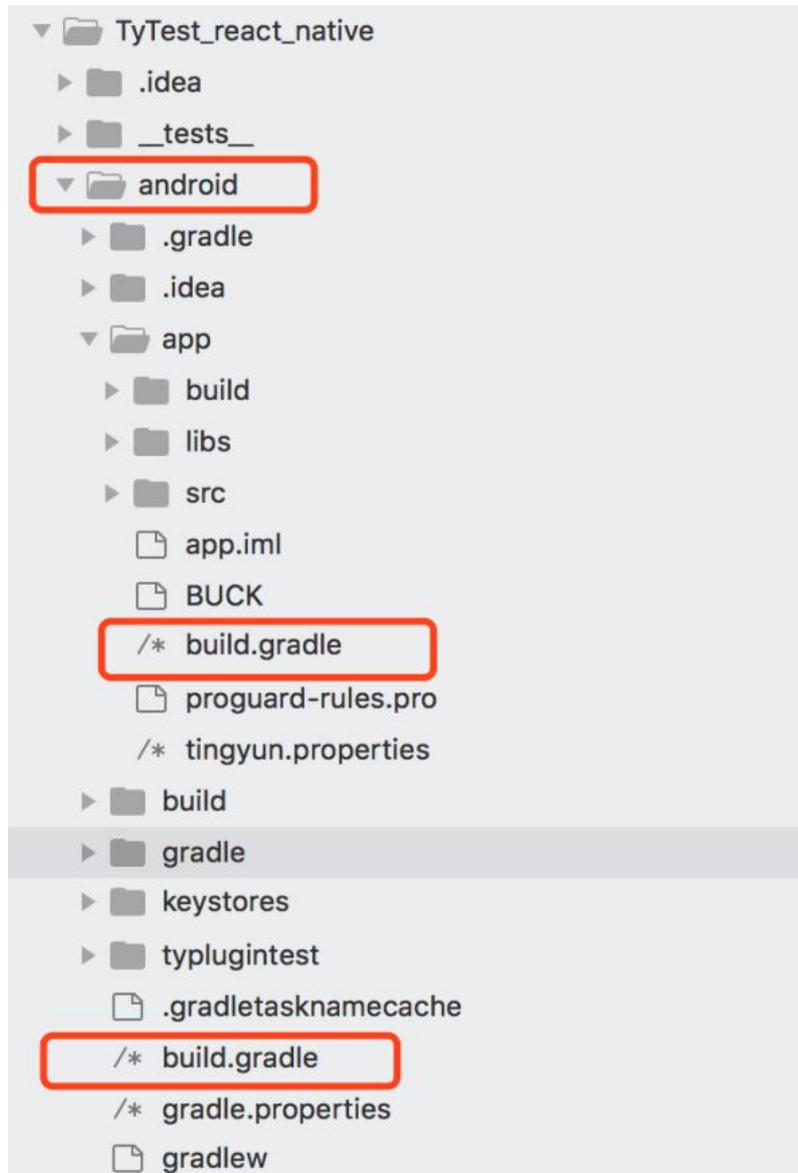
```
apply plugin: 'com.android.library'
apply plugin: 'com.github.dcendents.android-maven'
apply plugin: 'com.jfrog.bintray'
apply plugin: 'newlens'

allprojects {
    repositories {
        mavenCentral()
        jcenter()
    }
}

dependencies {
    provided 'com.networkbench.newlens.agent.android:nbs.newlens.agent:2.7.0'
}
```

## 2.1.3 React Native 工程相关依赖构建

React Native 工程嵌码和普通的 android 项目一样的，项目目录如下：



可以参照 2.1.1 Android Studio 工程相关依赖构建进行配置。

## 2.1.4 配置应用权限



由于听云 SDK 嵌码会解析 AndroidManifest.xml 文件，请确保文件中不要存在非 UTF-8 字符（例如，注释中的中文引号“”），否则可能导致 SDK 无法对 Activity 嵌码。

构建完成后，请在待监测的 App 工程的 AndroidManifest.xml 文件中增加以下的权限

```
<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_PHONE_STATE"/>
```

```
<uses-permission android:name="android.permission.READ_LOGS" />
<uses-permission android:name="android.permission.INTERNET" />
<!--使用可视化操作命名功能需配置悬浮窗权限(选配)-->
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
<uses-permission android:name="android.permission.SYSTEM_OVERLAY_WINDOW" />
```

## 2.1.5 可视化操作命名功能配置(选配)

### 1. 获取 URL Scheme

当您在听云报表“新建 App”时，您可以看到 URL Scheme。



您也可以随时进入“修改配置”页面中找到您的应用对应的 URL Scheme。



### 2. 可视化操作命名需要在 AndroidManifest.xml 文件的启动 Activity 增加 scheme 配置。

示例如下：

```
<activity android:name=".MainActivity">
  <intent-filter>
    <action android:name="android.intent.action.MAIN" />
    <category android:name="android.intent.category.LAUNCHER" />
  </intent-filter>
  <!--请添加这里的整个 intent-filter 区块，并确保其中只有一个 data 字段-->
  <intent-filter>
    <data android:scheme="tingyun.xxxx" />
    <!--请将 scheme 中的 "tingyun.xxxx" 替换为您应用的 URL Scheme-->
    <action android:name="android.intent.action.VIEW"/>
    <category android:name="android.intent.category.DEFAULT"/>
    <category android:name="android.intent.category.BROWSABLE"/>
  </intent-filter>
  <!--请添加这里的整个 intent-filter 区块，并确保其中只有一个 data 字段-->
</activity>
```

### 3. 设置控件 ID

听云 SDK 会采集用户操作的控件 ID，建议您在 Layout 文件中添加控件 ID。对于动态生成的控件，可以使用 `setViewId()` 方法对它设置唯一的 ID。

示例如下：

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    LinearLayout layout = findViewById(R.id.layout);
    final Button button = new Button(this);
    button.setLayoutParams(new
ViewGroup.LayoutParams(ViewGroup.LayoutParams.WRAP_CONTENT,
ViewGroup.LayoutParams.WRAP_CONTENT));
    button.setText("Login");
    //viewId 最多包含 32 个字符,支持英文、数字、下划线
    NBSAgent.setViewId(button, "bt_login");
}
```

## 2.1.6 插入初始化探针代码

1. 在嵌码项目工程的“Application”中 import NBSAppAgent 类

```
import com.networkbench.agent.impl.NBSAppAgent;
```

2. 在“Application”中的 `onCreate()` 方法（如未找到该方法请新增 `onCreate()`）中初始化 Android SDK

```
NBSAppAgent.setLicenseKey("AppKey").withLocationServiceEnabled(true).start(
this.getApplicationContext()); //Appkey 请从官网获取
```

 注：

1. 若无需采集地理位置，请使用以下配置

```
NBSAppAgent.setLicenseKey("AppKey").start(this.getApplicationContext());
//Appkey 请从官网获取
```

2. SDK 默认采集所有进程的数据，若只想采集主进程数据，请使用以下配置

```
NBSAppAgent.setLicenseKey("AppKey").withOnlyMainProcEnabled(true).start(
this.getApplicationContext());
//Appkey 请从官网获取
```

3. 采集 WebView 数据需调用 `setWebViewClient` 方法，如嵌码 App 中未调用该方法，请添加如下内容

```
webview.setWebViewClient(new WebViewClient());
```

4. 采集 WebView 数据需在 `WebChromeClient` 的 `onProgressChanged` 函数中调用接口：`NBSWebChromeClient.initJSMonitor(view, newProgress)`;例子如下：

```
webview.setWebChromeClient(new WebChromeClient() {
    @Override
    public void onProgressChanged(WebView view, int newProgress) {
        NBSWebChromeClient.initJSMonitor(view, newProgress);
        super.onProgressChanged(view, newProgress);
    }
});
```

## 2.1.7 使用 Gradle 命令打包编译

```
gradle clean build
```

## 2.1.8 配置混淆

1. 发布前请在 **proguard** 混淆配置文件中增加以下内容，以免 **tingyunSDK** 不可用

```
# ProGuard configurations for NetworkBench Lens  
-keep class com.networkbench.** { *; }  
-dontwarn com.networkbench.**  
-keepattributes Exceptions, Signature, InnerClasses  
# End NetworkBench Lens
```

2. 若需要保留行号信息，请在 **proguard.cfg** 中添加以下内容

```
-keepattributes SourceFile,LineNumberTable
```

## 2.2 嵌码完整性校验

1. 数据收集服务器校验
2. 嵌码完成后可通过“LogCat”查看听云 SDK 日志输出结果，用以进行数据收集服务器校验 TAG 为 **NBSAgent**，标准日志输出结果如下所示：

```
NBSAgent start  
NBSAgent enabled  
NBSAgent V "TingYun_Version" //TingYun_Version 为当前 SDK 的版本号  
connect success
```

3. 数据功能完整性校验  
嵌码完成后可通过“LogCat”查看听云 SDK 日志输出结果，用以进行数据功能完整性校验 TAG 为 **TingYun**，标准日志输出结果如下所示：

```
D/TingYun: Crash switch is true  
D/TingYun: webView switch is true  
D/TingYun: ANR monitor switch is true  
D/TingYun: UserAction Switch is true  
D/TingYun: cdnSwitch Switch is true
```

## 3 高级功能

### 3.1 用户自定义 ID

1. 功能说明  
用户自定义 ID 为当前用户设置唯一标示码，在任意位置均可设置 UserID。
2. 相关接口

```
//UserID 最多包含 64 个字符，支持中文、英文、数字、下划线，但不能包含空格或其他的转义字符  
NBSAppAgent.setUserIdentifier("userIdentifier");
```

3. 代码示例

```
public class MainActivity extends Activity {  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        String userIdentifier = getUserID();  
        NBSAppAgent.setLicenseKey("AppKey").withLocationServiceEnabled(true).  
start(this.getApplicationContext());  
        NBSAppAgent.setUserIdentifier(userIdentifier);  
    }  
}
```

### 3.2 面包屑

1. 功能说明  
面包屑能够更好的协助用户调查崩溃发生的原因，可以知晓用户发生崩溃之前的代码逻辑与崩溃轨迹结合使用能够更好的复现用户崩溃场景。
2. 相关接口

```
//最多包含 100 个字符，支持中文、英文、数字、下划线  
NBSAppAgent.leaveBreadcrumb("keyPressed");  
NBSAppAgent.leaveBreadcrumb("loginDone");
```

3. 代码示例

```
public MyActivity extends Activity {  
    public void onCreate(Bundle savedInstanceState) {  
        super.onCreate(savedInstanceState);  
        NBSAppAgent.setLicenseKey("AppKey").withLocationServiceEnabled(  
true).start(this.getApplicationContext());  
        NBSAppAgent.leaveBreadcrumb("login MyActivity onCreate");  
    }  
    public void onResume() {  
        super.onResume();  
        NBSAppAgent.leaveBreadcrumb("login MyActivity onResume");  
    }  
    public void loginPressed(View view) {  
        NBSAppAgent.leaveBreadcrumb("login MyActivity loginPressed");  
        new LoginAsyncTask.execute();  
    }  
}
```

```
public void onStop() {
    super.onStop();
    NBSAppAgent.leaveBreadcrumb("login MyActivity onStop");
}
}
```

### 3.3 自定义 Event

#### 1. 功能说明

自定义事件用于统计 App 中的任意事件，开发者可以在 SDK 初始化后的任意位置添加自定义事件，并设置对应上传参数。如：真实用户操作时候点击某个功能按钮或触发了某个功能事件等。

#### 2. 相关接口

```
//EVENT_ID 最多包含 32 个字符，支持中文、英文、数字、下划线，但不能包含空格或其他的转义字符
NBSAppAgent.onEvent(String EVENT_ID);
```

#### 3. 代码示例

```
@Override
public void onClick(View v) {
    .....
    NBSAppAgent.onEvent("添加购物车");
    .....
}
```

### 3.4 自定义 Trace



由于自定义 Trace 是成对出现的，请勿跨方法、跨进程以及在异步加载和递归调用中使用该接口。

#### 1. 功能说明

听云 SDK 默认采集系统类和方法的性能数据，无法采集开发者自定义类和方法的性能数据。使用“自定义 Trace”接口就可以帮助开发者时刻了解所写代码的健壮性及其性能数据。如：开发者想要了解某个自定义方法的初始化耗时及性能消耗情况，就可以在该自定义方法前后添加“自定义 Trace”接口即可。

#### 2. 相关接口

```
//Name 为当前方法所在方法名或自定义名称，支持中文、英文、数字、下划线，但不能包含空格或其他的转义字符
NBSAppAgent.beginTracer("String Name");
NBSAppAgent.endTracer("String Name");
```

#### 3. 代码示例

```
//用户可以在 SDK 初始化后的任意方法前后添加自定义 Trace
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    init();
}
```

```
private void init() {
    //方法开始前添加 beginTracer
    NBSAppAgent.beginTracer("这是 Init 方法");
    try {
        .....
    } catch (NameNotFoundException e) {
        e.printStackTrace();
    }
    //方法结束后添加 endTracer
    NBSAppAgent.endTracer("这是 Init 方法");
}
```

## 3.5 自定义 Log



收集 LogCat 信息,需要使用 READ\_LOGS 权限,请将下列代码添加到应用程序的 AndroidManifest.xml 文件中,默认收集 50 行,最多收集 100 行日志。

```
< uses-permission android:name="android.permission.READ_LOGS" >
```

### 1. 功能说明

开发者可通过 Android 日志系统的 LogCat,来收集和查看系统调试输出的信息,通过打印输出的 Log 信息来调查 Bug 发生时的应用程序信息,并通过听云 SDK 上传自定义 Log 日志。

### 2. 控制开关

```
NBSAppAgent.setLicenseKey("AppKey").withLocationServiceEnabled(true).enableLogging(true).start(this.getApplicationContext());
```

### 3. 相关接口

```
NBSAppAgent.setLogging(int lineNumber);
NBSAppAgent.setLogging(String filter);
NBSAppAgent.setLogging(int lineNumber,String filter);
```

### 4. 代码示例

```
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    // Enable logging
    NBSAppAgent.setLicenseKey("AppKey").withLocationServiceEnabled(true).enableLogging(true).start(this.getApplicationContext());
    // Log last 100 messages
    NBSAppAgent.setLogging(100);
}
```

## 3.6 自定义附加信息

### 1. 功能说明

用户可以在初始化之后任意位置配置该接口,最多可添加 10 条附加信息,每条附加信息最大支持 100 个字节随崩溃上传。

## 2. 相关接口

```
NBSAppAgent.setUserCrashMessage(String key,String value);
```

## 3. 代码示例

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main);  
    NBSAppAgent.setLicenseKey("AppKey").withLocationServiceEnabled(true).start(this.getApplicationContext());  
    //初始化后的任意位置插入自定义附加信息  
    NBSAppAgent.setUserCrashMessage("张三","13700001234");  
}
```

# 3.7 控制台版本提示

## 1. 功能说明

用户调试时会在控制台输出版本更新提示，当开启调试模式且当前使用 SDK 版本低于线上最新版本时，App 运行后打印输出。



注：项目工程只有 debug 模式才会开启此功能。

可通过在 AndroidManifest.xml 文件中的 application 标签添加 debuggable 属性控制是否开启调试模式。

```
<application android:debuggable="true">
```

## 2. 相关接口

默认开启，若无需提示，可关闭控制台提示功能。

```
NBSAppAgent.closeLogForUpdateHint();
```

## 3. 代码示例

```
public void onCreate(Bundle savedInstanceState) {  
    super.onCreate(savedInstanceState);  
    setContentView(R.layout.main);  
    NBSAppAgent.setLicenseKey("AppKey").withLocationServiceEnabled(true).closeLogForUpdateHint().start(this.getApplicationContext());  
    //初始化时调用接口关闭版本提示功能  
}  
}
```

# 4 设置本地配置文件

## 4.1 新建 tingyun.properties 配置文件

在项目工程根目录及 Module 目录下，新建 tingyun.properties 配置文件即可

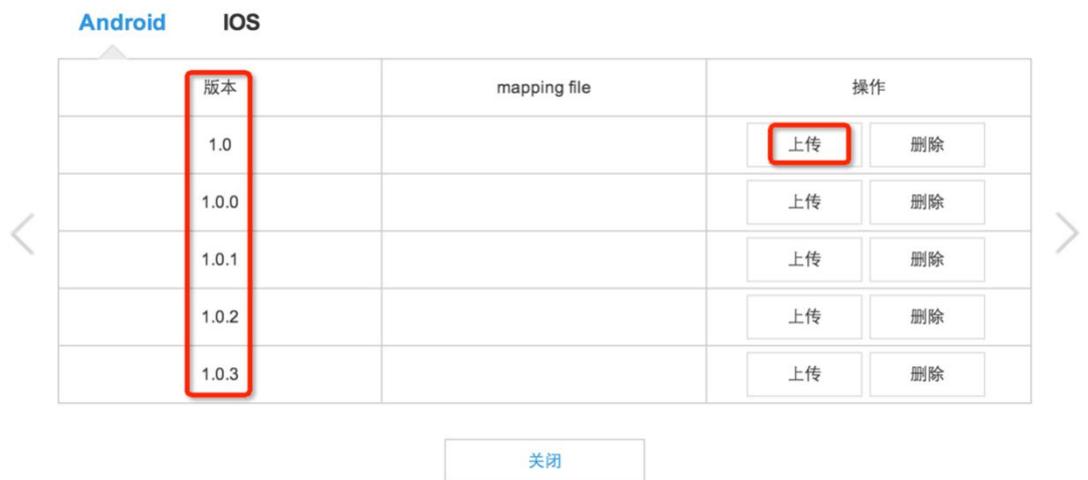
## 4.2 配置 Crash 反混淆（mapping）

听云 SDK 提供了两种方式进行反混淆设置

## 4.2.1 通过报表上传 mapping 文件



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”



2. 找到所需版本，并上传本地 mapping 文件即可

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

1. 添加如下配置到 `tingyun.properties` 文件

```
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. 配置 Authkey 授权码  
您可以在报表如下位置获取授权码。



### 3. 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。

```
webview.setWebChromeClient(new WebChromeClient() {
    @Override
    public void onProgressChanged(WebView view, int newProgress) {
        NBSWebChromeClient.initJSMonitor(view, newProgress);
        super.onProgressChanged(view, newProgress);
    }
});
```

## 5 FAQ

### 1. 应用启动数据未采集或缺失生命周期方法

1. 检查 `AndroidManifest.xml` 文件中存在非 UTF-8 字符（例如，注释中的中文引号 “”），如存在需删除。
2. 若 `AndroidManifest.xml` 文件中注册了自定义的 `MyBaseActivity`，请删除它，或对其配置过滤嵌码。需要在 `tingyun.properties` 文件中添加以下内容：  
#等号 “=” 后替换为需要过滤听云嵌码的类的路径，如有多个路径，使用英文逗号隔开  
`noInstrumentedClasses=com/nbs/ui/base/MyBaseActivity`
3. 应用需实现自定义的 `Application` 类

4. ant 或 eclipse 编译, 需在 `tingyun.properties` 文件中配置以下内容:  
#等号 "=" 后替换为需要过滤听云嵌码的类的路径, 如有多个路径, 使用英文逗号隔开  
`applicationName=com/nbs/app/MyApp`

## 2. 用户操作数据采集少

听云 SDK 编译时对点击事件 (如 `onClick`、`onItemClickListener`、`ItemSelected`、`PageSelected` 等方法) 嵌码, 仅支持点击数据采集, 暂不支持触摸、长按、滑动等操作。

## 3. 未采集到崩溃或采集的崩溃数量偏少

1. 听云 SDK 只采集 java 层崩溃, 不支持 Native 层异常采集。
2. 听云 SDK 初始化前的崩溃无法采集。
3. 检查报表的崩溃采集开关是否开启。
4. 若应用中存在应用自身异常捕获或第三方崩溃采集逻辑, 需放在听云 SDK 之前初始化。

## 4. OKHttp 网络数据未采集

不能对 OKHttp 混淆。在项目的 `proguard-rules.pro` 文件中, 添加

```
#OKHttp3 配置
-keep class okhttp3.** { *;}
-dontwarn okhttp3.**
#OKHttp2 配置
-keep class com.squareup.okhttp.**
```

## 5. TCP、首包数据为 0

1. 听云 SDK 需在应用自身网络请求发起之前初始化
2. 暂不支持安卓 9.0 TCP 数据采集。
3. 使用 Http 代理时无法获取 TCP 数据。
4. WebView 跨域资源响应头未配置 `Timing-Allow-Origin: *`

## 6. WebView 数据未采集或只采集到了网络错误、Http 错误

1. 未在 `WebChromeClient` 的 `onProgressChanged` 方法中调用  
`NBSWebChromeClient.initJSMonitor(view, newProgress);`
2. 不支持腾讯 X5 框架
3. 不支持 SPA 单页面应用数据采集。
4. 不支持使用 `loadDataWithBaseURL` 加载 html 代码

## 7. URL 过滤、错误过滤未生效

1. 检查报表上配置是否正确, 可以通过在线正则工具检查, 需注意报表配置正则的前后需要加斜杠 "/"。
2. 过滤配置是 SDK 连接听云服务器成功后下发的, 再此之前的网络数据无法过滤。

## 8. 如项目使用 MultiDex 分包, 安卓 5.0 以下设备可能会出现 `java.lang.NoClassDefFoundError` 异常

需将报错的类分配至主 dex 中, 可参考 Google 提供的解决方案  
<https://developer.android.com/studio/build/multidex.html?hl=zh-cn>

## 9. Sophix 补丁工具生成补丁失败

听云 SDK 为了采集应用启动数据, 会在编译时对 `Application` 类的 `attachBaseContext()` 及 `onCreate()` 方法嵌码。而阿里 Hotfix 补丁工具自 3.2.1 版本开始, 要求在 `initialize` 之前, 不能调用非 Android 官方 API 类, 导致生成补丁失败。  
解决方案请参阅听云帮助中心的《与 Sophix 冲突问题解决方法》。

## 10. 编译时出现 `java.util.zip.ZipException` 异常



```
SDK jar 包重复引用，检查 libs 目录及 build.gradle 文件，确保只有一份  
nbs.newlens.agent.jar 引用。
```

## 11. 项目使用 Lambda 时编译失败

将听云插件放在 lambda 后面

```
apply plugin: 'me.tatarka.retrolambda'  
apply plugin: 'newlens'
```