



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 React Native 部署说明

孙凯  
2019/1/28



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	2.8.1	V2.7
Android 探针	2.9.0	V2.8
Android 探针	2.9.1	V2.9
Android 探针	2.10.0	V2.10
Android 探针	2.10.2	V2.11
Android 探针	2.11.0	V2.12
Android 探针	2.11.1	V2.13
Android 探针	2.12.0	V2.14

## 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。

符号	说明
	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 V2.14 （2019-01-28）

- 修改

位置	描述
混淆配置	修正混淆配置代码

### 版本 V2.13 （2018-11-16）

- 修改

位置	描述
使用 Crodova 构建	新增采集 X5WebView 配置

### 版本 V2.12 （2018-11-06）

- 修改

位置	描述
使用 React Native 构建	新增采集 Native Crash 配置

### 版本 V2.11 （2018-09-06）

- 修改

位置	描述
初始化探针代码	修正示例代码
可视化操作命名功能配置	修正示例代码

位置	描述
嵌码完整性校验	更新 log 信息

## 版本 V2.10 （2018-07-24）

- 修改

位置	描述
简介	应用 App 嵌码后体积增量改为 370KB 左右
使用 Gardle 构建	更新内容，新增听云仓库配置
FAQ	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 Gardle 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

位置	描述
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



# 目录

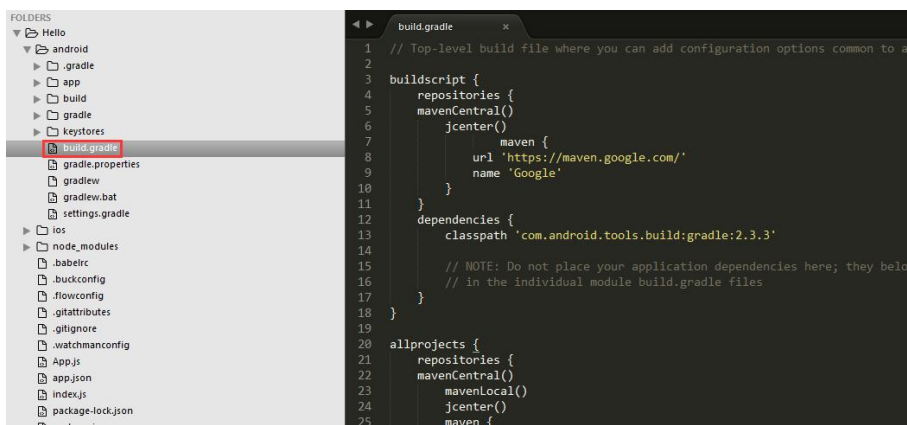
目录.....	5
1 REACT NATIVE 方式部署.....	6
1.1 使用 REACT NATIVE 构建.....	6
1.1.1 React Native 工程相关依赖构建.....	6
1.1.2 配置应用权限.....	8
1.1.3 可视化操作命名功能配置(选配).....	10
1.1.4 插入初始化探针代码.....	8
1.1.5 采集 WebView 数据配置.....	9
1.1.6 采集 X5WebView 数据配置.....	9
1.1.7 使用 React Native 命令打包编译.....	11
1.1.8 配置混淆.....	11
1.2 嵌码完整性校验.....	11

# 1 React Native 方式部署

## 1.1 使用 React Native 构建

### 1.1.1 React Native 工程相关依赖构建

1. 打开项目 android 目录下的 build.gradle 文件



2. 在 buildscript 的 repositories 中添加以下代码

```
maven { url
"http://nexus2.tingyun.com/nexus/content/repositories/snapshots/" }
```

```
buildscript {
  repositories {
    mavenCentral()
    jcenter()
    maven {
      url 'https://maven.google.com/'
      name 'Google'
    }
    maven { url "http://nexus2.tingyun.com/nexus/content/repositories/snapshots/" }
  }
  dependencies {
    classpath 'com.android.tools.build:gradle:2.3.3'
    classpath 'com.networkbench.newlens.agent.android:agent-gradle-plugin:2.10.1'
  }
}
```

3. 在 buildscript 的 dependencies 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 {
        mavenCentral()
        jcenter()
        maven {
            url 'https://maven.google.com/'
            name 'Google'
        }
        maven { url "http://nexus2.tingyun.com/nexus/content/repositories/snapshots/" }
    }
    dependencies {
        classpath 'com.android.tools.build:gradle:2.3.3'
        classpath 'com.networkbench.newlens.agent.android:agent-gradle-plugin:2.10.1'
    }
}
```

#### 4. 在 allprojects 的 repositories 中添加以下代码

```
maven { url
"http://nexus2.tingyun.com/nexus/content/repositories/snapshots/" }
```

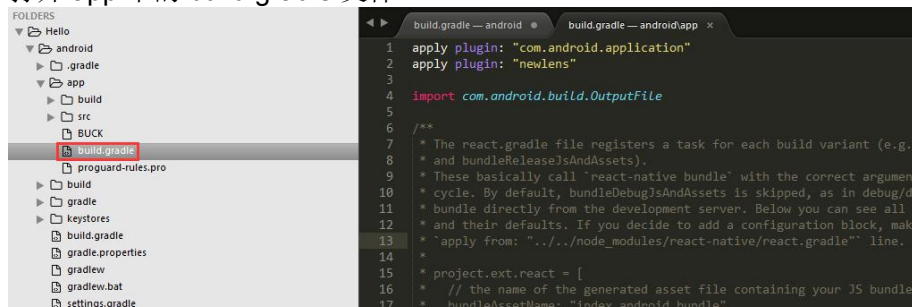
```
allprojects {
    repositories {
        mavenCentral()
        mavenLocal()
        jcenter()
        maven {
            url "$rootDir/../node_modules/react-native/android"
        }
        maven {
            url 'https://maven.google.com/'
            name 'Google'
        }
        maven { url "http://nexus2.tingyun.com/nexus/content/repositories/snapshots/" }
    }
}
```

#### 5. 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 fileTree(dir: 'libs', include: '*.jar')
    compile 'com.networkbench.newlens.agent.android:nbs.newlens.agent:2.10.1'
    debugCompile(project(path: "CordovaLib", configuration: "debug"))
    releaseCompile(project(path: "CordovaLib", configuration: "release"))
}
```

#### 6. 打开 app 下的 build.gradle 文件



```
1 apply plugin: "com.android.application"
2 apply plugin: "newlens"
3
4 import com.android.build.OutputFile
5
6 /**
7  * The react.gradle file registers a task for each build variant (e.g.
8  * and bundleReleaseJsAndAssets).
9  * These basically call 'react-native bundle' with the correct arguments
10  * cycle. By default, bundleDebugJsAndAssets is skipped, as in debug/de
11  * bundle directly from the development server. Below you can see all t
12  * and their defaults. If you decide to add a configuration block, make
13  * * apply from: "../node_modules/react-native/react.gradle" line.
14  *
15  * project.ext.react = [
16  *     // the name of the generated asset file containing your JS bundle
17  *     bundleAssetName: "index.android.bundle",
```



## 7. 添加代码

```
apply plugin: 'newlens'

apply plugin: "com.android.application"
apply plugin: "newlens"

import com.android.build.OutputFile
```

## 8. 在 dependencies 中添加以下代码

```
dependencies {
    compile
    '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
}
```

```
dependencies {
    compile fileTree(dir: "libs", include: ["*.jar"])
    compile "com.android.support:appcompat-v7:${rootProject.ext.supportLibVersion}"
    compile "com.facebook.react:react-native:+" // From node modules
    compile 'com.networkbench.newlens.agent.android:nbs.newlens.agent:2.10.1'
}
```

## 9. 采集 Native Crash, 需在 dependencies 中添加以下代码

```
compile
'com.networkbench.newlens.agent.android:nbs.newlens.nativecrash:1.0.0'
//此处为 provided 依赖, TingYun_Version 为当前听云 APP 版本号
```

```
dependencies {
    compile fileTree(dir: 'libs', include: '*.jar')
    compile 'com.networkbench.newlens.agent.android:nbs.newlens.agent:2.11.0'
    compile 'com.networkbench.newlens.agent.android:nbs.newlens.nativecrash:1.0.0'
    // SUB-PROJECT DEPENDENCIES START
    debugCompile(project(path: "CordovaLib", configuration: "debug"))
    releaseCompile(project(path: "CordovaLib", configuration: "release"))
    // SUB-PROJECT DEPENDENCIES END
}
```


## 1.1.2 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").withLocationServiceEnabled(false).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.  
s.getApplicationContext());  
//Appkey 请从官网获取
```

### 1.1.3 采集 WebView 数据配置

1. 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(){});
```

2. 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);  
    }  
});
```

### 1.1.4 采集 X5WebView 数据配置

1. 采集 X5WebView 数据需调用 `addWebViewBridge` 方法，如嵌码 App 中未调用该方法，请添加如下内容

```
X5WebView x5WebView = new X5WebView(this, null);  
NBSWebChromeX5Client.addWebViewBridge(x5WebView);
```

2. 采集 X5WebView 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(){});
```

3. 采集 X5WebView 数据需在 `WebChromeClient` 的 `onProgressChanged` 函数中调用接口：`NBSWebChromeX5Client.initJSMonitorX5(view, newProgress)`;例子如下：

```
x5webview.setWebChromeClient(new WebChromeClient() {  
    @Override  
    public void onProgressChanged(WebView view, int newProgress) {  
        NBSWebChromeX5Client.initJSMonitorX5(view, newProgress);  
        super.onProgressChanged(view, newProgress);  
    }  
});
```

### 1.1.5 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-->
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW"/>
<uses-permission android:name="android.permission.SYSTEM_OVERLAY_WINDOW" />
```

## 1.1.6 可视化操作命名功能配置

### 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"/>
    </intent-filter>
```

```
<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
    NBSAppAgent.setViewId(button, "bt_login");
}
```

## 1.1.7 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，请在混淆配置文件中添加以下内容

```
-keepattributes SourceFile,LineNumberTable
```

## 1.1.8 使用 React Native 命令打包编译

```
react-native run-android
```

## 1.2 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: Network Switch is true  
D/TingYun: UI Switch is true  
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
```