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
2016/10/27

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

内容介绍

本手册主要介绍了 Android SDK 探针的 Ant 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 V2.0 （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 1
版本 V2.0 （2016-10-27） *2*
版本 V1.0 （2014-06-18） *2*

目录 **3**

1 简介 **4**
1.1 ANDROID SDK 目录结构 5
1.2 ANDROID SDK THEORY 5
1.3 ANDROID SDK 体积增量 5

2 GRADLE 方式部署 **6**
2.1 使用 GRADLE 构建 6
2.1.1 工程相关依赖构建 *6*
2.2 嵌码完整性校验 10

3 高级功能 **10**
3.1 自定义 ID 10
3.2 面包屑 11
3.3 自定义 EVENT 11
3.4 自定义 TRACE 12
3.5 自定义 LOG 13
3.6 自定义附加信息 13

4 设置本地配置文件 **14**
4.1 新建 TINGYUN.PROPERTIES 配置文件 14
4.2 配置 CRASH 反混淆（MAPPING） 14
4.2.1 通过报表上传 mapping 文件 *14*
4.2.2 通过 tingyun.properties 文件自动上传 mapping 文件 *14*
4.3 配置 OKHTTP 控制开关 16
4.4 配置 WEBVIEW 控制开关 16

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/ANR 异常数据收集
- 事件性能数据收集
- 视图性能数据收集
- 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
- Retrofit
- WebView（原生/腾讯 X5）

Android SDK 探针支持如下协议类库：

- Android 2.0 版本~Android6.0 版本

Android SDK 通过在指定的方法中嵌码去采集下列方法中的数据。可以通过报表服务器查询各种方法的数据和调用次数。

- 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
- 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（751KB）
  - nbs.newlens.class.rewriter.jar（2.1MB）
- nbs-newlens-android-ndk
  - arm64-v8a
    - libnbsdc.so（682KB）
  - armeabi
    - libnbsdc.so（399KB）
  - armeabi-v7a
    - libnbsdc.so（382KB）

1.2 Android SDK Theory

```
protected void onCreate(Bundle bundle) {
    APM.startTracing(getClass().getName());   // 探针开始，进行数据采集
    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（默认一分钟，可设置）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

- 未添加 SO 文件，应用 App 嵌码后体积增量为 250KB 左右。
- 添加 SO 文件后，应用 App 嵌码后体积增量为 250KB 加对应 SO 文件大小。

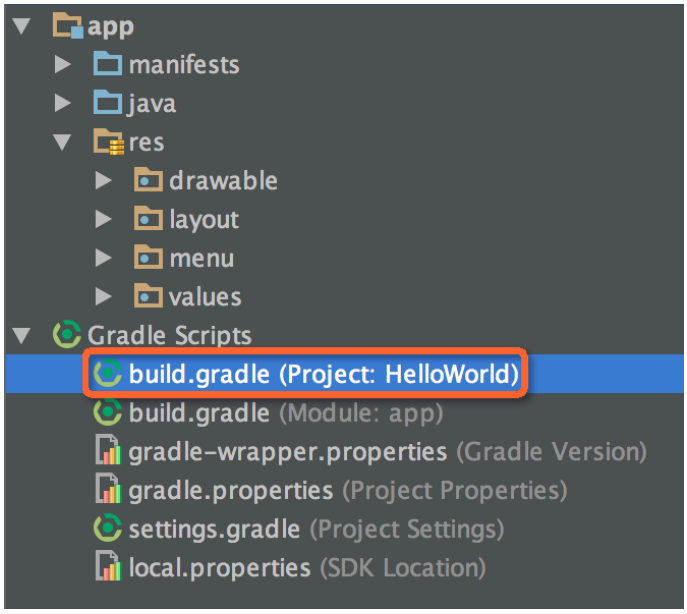
2 Gradle 方式部署

2.1 使用 Gradle 构建

2.1.1 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



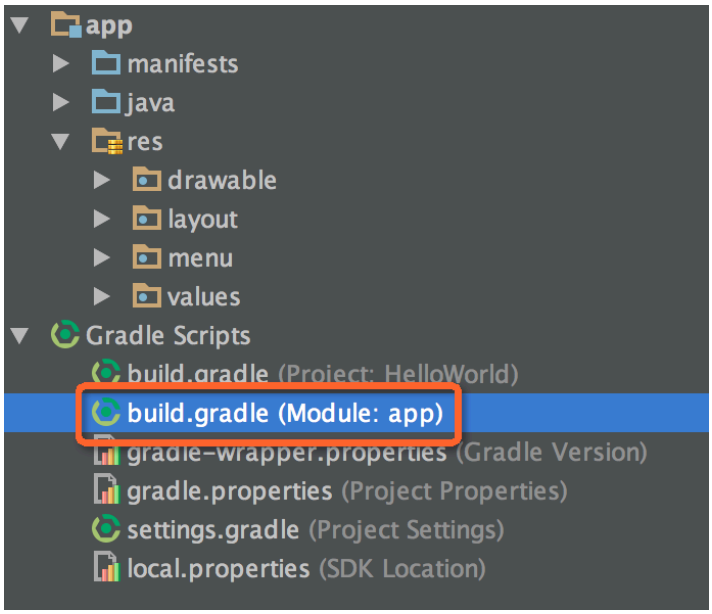
2. 在 buildscript 模块中加入代码

```
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()
}
```



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
```



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"

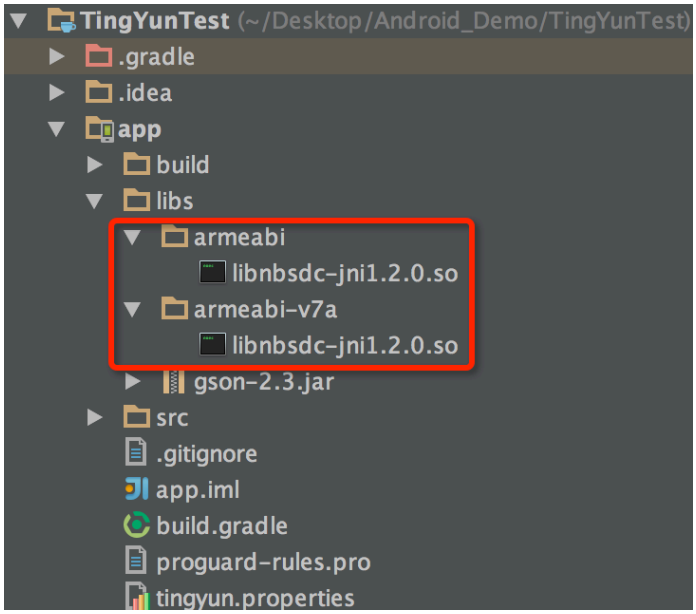
    defaultConfig {
        applicationId "com.hello.nbs.helloworld"
        minSdkVersion 14
        targetSdkVersion 21
        versionCode 1
        versionName "1.0"
    }
    buildTypes {
        release {
            minifyEnabled false
            proguardFiles getDefaultProguardFile('proguard-android.txt'),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

7. 若需采集 TCP 性能数据（主机 IP、DNS 时间、TCP 建联时间），请将 SO 文件添加到 libs 目录下的相应文件夹中，如下图所示：



SO 库下载地址为：https://report.tingyun.com/mobile/download/sdk

同时添加以下代码到 build.gradle（Module）文件中：

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



2.1.2 配置应用权限

构建完成后，请在待监测的 App 工程的 AndroidMainfest.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" />
```

2.1.3 插入初始化探针代码

1. 在嵌码项目工程的启动页面“MainActivity”中 import NBSAppAgent 类

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

2. 在“MainActivity”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
```

3. 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).startInApplication(this.getApplicationContext());//Appkey 请从官网获取
```

4. 启用腾讯 X5 框架初始化方法

```
NBSAppAgent.setLicenseKey("AppKey").withLocationServiceEnabled(true).setX5Enable().start(this.getApplicationContext());//Appkey 请从官网获取
```

5. 禁用地理位置信息采集

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

2.1.4 使用 Gradle 命令打包编译

```
gradle clean build
```

2.1.5 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：

```
webView switch is true
okhttp switch is true
```

3 高级功能

3.1 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 MainActivity 的 onCreate 里用初始化方法传入 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.setLicenseKey("AppKey").withLocationServiceEnabled(true).setUse
rIdentifier("userIdentifier").start(this.getApplicationContext());
```

3. 代码示例

```
public class MainActivity extends Activity {
```

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

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，但不能包含空格或其他的转义字符
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 logginPressed(View view) {
             NBSAppAgent.leaveBreadcrumb("login MyActivity logginPressed");
             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).enable
Logging(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
   用户可以再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).s
tart(this.getApplicationContext());
      //初始化后的任意位置插入自定义附加信息
      NBSAppAgent.setUserCrashMessage(String “张三”,String “13700001234”);
}
```

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

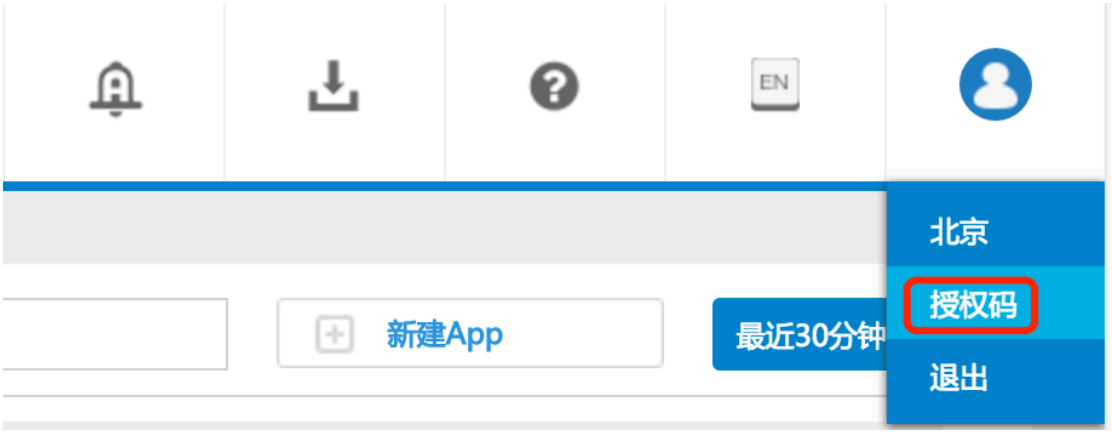
4.2.2 通过 tingyun.properties 文件自动上传 mapping 文件

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



a. 登陆报表后选择“授权码”



b. 复制当前授权码并添加到 tingyun.properties 即可

3. 配置 AppKey



tingyun.properties 配置文件中的 AppKey 相同，在“修改设置”中获取即可

4. 配置 mapping_file_auto_upload 控制开关

```
mapping_file_auto_upload=true
```

5. 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。

4.3 配置 OkHttp 控制开关

听云 SDK 默认对 OkHttp2.0 系列版本进行嵌码

请选择对应版本进行配置，不可设置多个版本

1. 若当前 App 使用的是 OkHttp2.0 系列版本请忽略该设置
2. 若当前 App 使用的是 OkHttp3.0 系列版本请添加如下配置到 tingyun.properties 配置文件中

```
okhttp3.0=true
```

3. 若当前 App 使用的是 OkHttp1.0 系列版本请添加如下配置到 tingyun.properties 配置文件中

```
okhttp2.0=false
```

4.4 配置 WebView 控制开关

WebView（H5）页面的性能数据是通过 JS 回调方式采集的，请开启 JS 回调开关：WebView.setWebContentsDebuggingEnabled(**true**);

注：允许 JS 回调在 Android4.0 系统上会有一些系统级别的安全漏洞，如有较高安全需求，则可在 Android5.0 及其以上版本使用或忽略该配置

1. 请将如下配置添加到 tingyun.properties 配置文件中

```
webview=true
```

2. 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(){});
```