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 高级功能说明

孙凯

2019/4/19

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
Android 探针	2.13.0	V2.15

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

符号	说明
	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 V2.15 （2019-04-19）

- 修改

位置	描述
高级功能	新增首次启动功能开关接口 新增自定义启动接口

版本 V2.14 （2019-01-28）

- 修改

位置	描述
高级功能	新增页面加载埋点接口

版本 V2.13 （2018-11-16）

- 修改

位置	描述
使用 Gradle 构建	新增 X5WebView 配置
设置本地配置文件	移除该章节

版本 V2.12 （2018-11-06）

- 修改

位置	描述
使用 Crodova 构	新增 Native Crash 配置

位置	描述
建	

版本 V2.11 （2018-09-06）

- 修改

位置	描述
全文	拆分文档

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
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

1 高级功能

1.1 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。

1. 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");
```

2. 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);
      }
 }
```

1.2 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 logginPressed(View view) {
```

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

1.3 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");
    ……
}
```

1.4 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");
    }
```

1.5 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);
    }
```

1.6 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).s
tart(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");}
```

1.7 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 AndroidMainfest.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
}
```

1.8 页面加载埋点

1. 功能说明

在页面加载分析模块，SDK 默认以 IdleHandler 的 queueIdle()回调时间点作为页面内容加载完成的时间点。用户可以通过 setPageLoadingEndTime()方法手动埋点，设置页面结束时间点。

2. 相关接口

```
NBSAppAgent.setPageLoadingEndTime(Class clazz);
```

3. 代码示例

```
//用户可以在页面加载回调中（或其他合适的位置），调用 setPageLoadingEndTime()方法
private void showSuccess() {
    networkStateView.showSuccess();
    //传入当前 Activity.class 或当前 Fragment.class
    NBSAppAgent.setPageLoadingEndTime(MainActivity.class);
}
```

1.9 首次启动功能开关

1. 功能说明

   首次启动应用时，SDK 默认只开启崩溃数据采集（之后的启动按连接服务器下发的配置开启相应功能）。用户可以通过接口自定义首次启动 SDK 的功能开关。

2. 相关接口

```
NBSAppAgent.setStartOption(int option);
//SDK 定义了功能开关如下：
//网络数据采集
NBSAppAgent.HTTP_NETWORK_ENABLED = 1;
//UI 数据采集（启动、页面、操作数据）
NBSAppAgent.UI_ENABLED = 2;
//崩溃数据采集
NBSAppAgent.CRASH_ENABLED = 4;
//WebView 数据采集
NBSAppAgent.WEBVIEW_ENABLED = 8;
//Socket Hook
NBSAppAgent.SOCKET_DATA_ENABLED = 16;
//跨应用功能
NBSAppAgent.CROSS_APP_ENABLED = 32;
//卡顿数据采集
NBSAppAgent.ANR_ENABLED = 64;
//行为数据采集
NBSAppAgent.USER_ACTION_ENABLED = 128;
//CND 数据采集
NBSAppAgent.CDN_ENDBLED = 256;
```

3. 代码示例

```
public void onCreate() {
    NBSAppAgent.setLicenseKey("AppKey")
    .setStartOption(NBSAppAgent.HTTP_NETWORK_ENABLED |
NBSAppAgent.UI_ENABLED | NBSAppAgent.CRASH_ENABLED)
    //首次启动开启网络、UI、崩溃数据采集
    .start(this.getApplicationContext());
```

1.10 自定义启动

1. 功能说明

   听云 SDK 采集冷启动(首次启动)数据，以第一个启动的 Activity 的 onResume() 方法结束作为启动时间结束点。用户也可以自己定义启动结束时间点。

2. 相关接口

```
NBSAppAgent.isCustomAppStart(boolean b);
//默认关闭，如需开启，设置为 true
NBSAppAgent.setCustomOnResumeEndIns(String className);
//传入启动的第一个 Activity 的 class.getName
```

3. 代码示例

```
public class MyApplication extends Application {
   @Override
   public void onCreate() {public void onCreate() {
    NBSAppAgent.setLicenseKey("AppKey")
    .isCustomAppStart(true)//在初始化 SDK 时调用，开启自定义启动时间功能
    .start(this.getApplicationContext());
}

public class SplashActivity extends Activity {
//启动的第一个 Activity
}

public class MainActivity extends Activity {
//启动的第二个 Activity
   @Override
   public void onResume() {
        super.onResume();
        NBSAppAgent.setCustomOnResumeEndIns(SplashActivity.class.getName())
;
        //以第二个 Activity 的 onResume()方法作为启动结束时间
   }
}
```