DuCaller SDK for Android

接入手册

DU Caller SDK V1.2.5

目录

前提:

DU Caller SDK V1.2.5 目前需要依赖 DU Ad Platform_SDK HW1.0.9.8 或 CW1.0.9.7（含）以上版本。

如需要关联 Facebook 广告位，请确认 Facebook Audience SDK 版本为 V4.23 以上。

1. 获取身份

请参照 HW 或 CW 版 DUADplatform SDK 文档 第 3 章;

申请 DuCaller 广告位 ID 时 ，需申请的广告位类型为「来电提醒」。

广告卡片样式控制类型，推荐选则全屏卡片，半屏卡片在 Google 政策上可能存在风险；

2. 加载与配置

2.1 加载 DU Ad Platform_SDK 压缩包

请参照 HW 或 CW 版 DUADplatform SDK 文档 4.1 章节 加载 sdk。

如接入 DuCaller 则额外需要进行如下操作：

1) 将 DuCallerSDK-Vx.x.aar 加入工程 libs 文件夹下：
2) 配置工程 build.gradle：

```
repositories {
    flatDir {
        dirs 'libs'
    }
}
dependencies {
    compile fileTree(include: ['*.jar'], dir: 'libs')
    compile(name: 'DuappsAd-xW-xxx-release', ext: 'aar')    //
DUcallerSDK 必须依赖 DAP 广告 SDK
    compile(name: 'DuCallerSDK-Vx.x', ext:'aar')
}
```

2.2 配置 AndroidManifest.xml

请参照 HW 或 CW 版 DUADplatform SDK 文档 4.2 章节配置 AndroidManifest.xml。

A. 添加权限。如接入 DuCaller，除基本网络权限外，

```
<uses-permission android:name= "android.permission.INTERNET" />
<uses-permission android:name= "android.permission.ACCESS_NETWORK_STATE" />
```

额外需要的权限如下：

权限名称	含义	对广告变现影响	是否必备
READ_PHONE_STATE	允许手机访问手机状态信息。	无法弹出通话场景	是
PROCESS_OUTGOING_CALLS	获取去电状态	无法弹出去电场景广告（占总场景 50%）	否
READ_CONTACTS	允许程序读取用户联系人数据	无法甄别号码是否为联系人,	否
RECEIVE_BOOT_COMPLETED	允许程序开机启动服务（离线推送服务）	进程保活，影响监听率	否
CALL_PHONE	允许拨打电话（用于黑名单拦截功能）	拦截功能开关不显示	否

*注：READ_PHONE_STATE 为必备权限；

PROCESS_OUTGOING_CALLS 建议保留，会影响 50%变现；

READ_CONTACTS 如没有，不影响 SDK 变现，但会影响 56%的用户的体验；

- 代码示例：

```
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission
android:name="android.permission.PROCESS_OUTGOING_CALLS" />
<uses-permission android:name="android.permission.READ_CONTACTS" />
<uses-permission
android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
```

B. 在 app_license 的 value 中填入已申请的 APP_ID。APP_ID 的申请 ,参见 HW 或 CW 版 DUADplatform SDK 文档 3.1。

```
<application
    android:name=" [your&package&name].MobulaApplication"
```

```
        android:label="@string/app_name"
        . . . >
        <meta-data
            android:name="app_license"
            android:value="@string/Your_DAP_APP_ID" />
    </application>
```

C. 声明 *com.duapps.ad.stats.DuAdCacheProvider.* 用你的真实包名替换这里的 *Your_packagename.* 如包名和 du ad platform 后台注册的包名不一致，会导致无法获取广告。

```
<provider
    android:name="com.duapps.ad.stats.DuAdCacheProvider"
    android:authorities="[your&package&name].DuAdCacheProvider"
    android:exported="false">
</provider>
```

D. 注册 APP 安装广播监听

静态在 AndroidManifest.xml 中注册安装广播监听

```
    <receiver android:name="com.duapps.ad.base.PackageAddReceiver" >
    <intent-filter>
    <action android:name="android.intent.action.PACKAGE_ADDED"/>
        <data android:scheme="package"  />
    </intent-filter>
</receiver>
```

E. 添加 DU Caller SDK provider

```
<provider
    android:name="com.ducaller.fsdk.provider.FsdkContentProvider"
    android:authorities="[your&package&name].FsdkContentProvider"
    android:exported="false" >
```

```
</provider>
```

2.3 混淆代码

A. 把 DU Caller Ad Platform SDK 中的类排除在混淆之外：

```
-keep class com.duapps.ad.**{*;}
-dontwarn com.ducaller.fsdk.**
```

B. 请参照 HW 或 CW 版 DUADplatform SDK 文档 4.3 章节，及 DU Caller SDK Demo 混淆代码。

```
-keep class com.dianxinos.DXStatService.stat.TokenManager {
public static java.lang.String getToken(android.content.Context);
}
-keep public class * extends android.content.BroadcastReceiver
-keep public class * extends android.content.ContentProvider

-keepnames @com.google.android.gms.common.annotation.KeepName class *
-keepclassmembernames class * {
       @com.google.android.gms.common.annotation.KeepName *;}
-keep class com.google.android.gms.common.GooglePlayServicesUtil {
     public <methods>;}

-keep class com.google.android.gms.ads.identifier.AdvertisingIdClient {
     public <methods>;}
-keep class
com.google.android.gms.ads.identifier.AdvertisingIdClient$Info {
     public <methods>;}
```

*注：混淆方法参见 Android 官方混淆文档：${ android-sdk }/tools/proguard/

C. 如接入 Facebook 广告，须将下类可以添加到 proguard 配置：

```
-keep class com.facebook.ads.NativeAd
```

3. 初始化

请参照 HW 或 CW 版 DUADplatform SDK 文档 第 5 章初始化。

如需接入 DuCaller 广告，请在 json 中添加如下标签，并传入相应广告位 ID.

```
{
    "ducaller": [
      {
       "pid": "YOUR_DAP_PLACEMENT_ID",
       "fbids": [ "YOUR_FACEBOOK_PLACEMENT_ID"]
       }
    ]
}
```

- 方法：

  在 application 的 OnCreate 方法中使用 DuCallerSDKHelper.init()。

  *注：请按上述要求使用该接口.否则可能导致初始化无效。

- 接口说明：

  public static void init(Context context，String pidsJson);

参数	说明
Context context	ACTIVITY CONTEXT
String pidsJson	Placement_ID 与广告位 ID 的对应关系

- 代码示例：

```
/**
 * 初始化 Ducaller SDK
 * @param context Application Context
 * @param pidsJson 广告配置 json
 */

DuCallerSDKHelper.init(Context context, String pidsJson)
```

4. 其他个性化设置

4.1 设置提示应用名称

- DuCallerSDKHelper.displayAppLabel(String appLabel)
  设置显示在 DuCaller 通话提醒卡片上的应用名称(@param appLabel)，如果未设置，则通话提醒卡片上不显示应用名称（@param appLabel）。

4.2 功能开关

- DuCallerSDKUtils.setSDKEnable(Boolean enable)
  开发者设置开启/禁止 DuCaller 功能

- DuCallerSDKHelper.disable()
  用户禁用 DuCaller 功能（用户行为优先级更高）

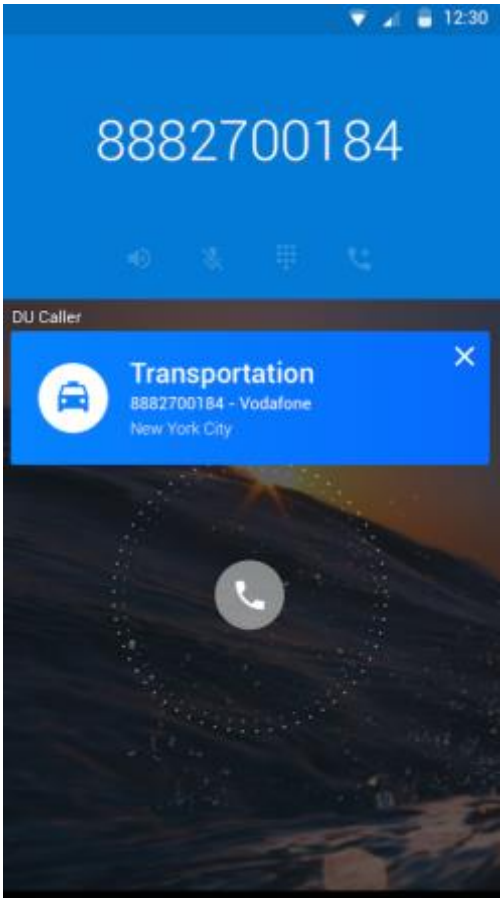
- DuCallerSDKHelper.enable()
  用户开启 DuCaller 功能（用户行为优先级更高）

4.3 判断 SDK 是否可用

- DuCallerSDKHelper.isEnabled()
  判断当前 SDK 是否可用，@return true 当前 SDK 功能可用， false 不可用，仅返回调用 disable 或者 endable 的设置结果
  返回 false 时，原因可能为以下情况：
  1. DuCallerSDKHelper.init() 没有调用或填写错误
  2. DuCallerSDKUtils.setSDKEnable(Boolean enable)参数为 false
  3. DuCallerSDKHelper.disable()被执行

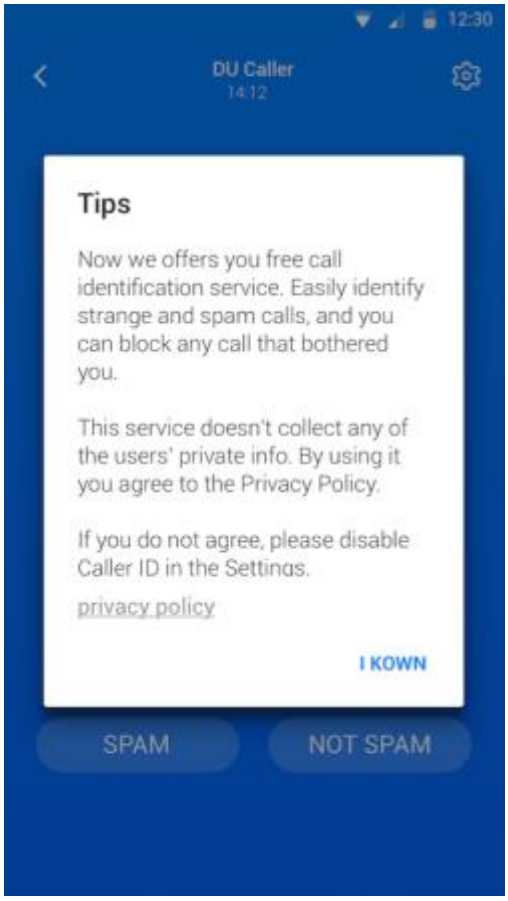
4.4 设置悬浮框是否显示

- public static void setFlowViewDisable(boolean disable)
  开发者设置悬浮框是否显示，@param disable。true :不显示来去电悬浮框； false 显示来去电悬浮框。如不调用该方法，默认显示来去电悬浮框



~~4.5 设置隐私条款状态（接口废弃）~~

- ~~public static void setPrivacyPolicyState(boolean enable)~~
  ~~设置隐私条款状态，@param enable。true 为可用状态，SDK 会根据状态自动弹出，提醒用户； false:为不可用状态，SDK 不再弹隐私条款。如不调用该方法，默认不显示隐私声明。隐私条款弹窗如图所示:~~



4.6 设置引导功能

- public static void needSDKGuide(boolean need)
  如设置，则代表需要通过引导，开启 ducaller SDK 功能。如调用该方法，只有在 @ isUserAccepted 方法返回为 true 的时候，DU Caller SDK 卡片及广告功能才正常展示
  @param need true 需要 ducaller SDK 进行功能开启引导，false 不需要 ducaller SDK 处理功能开启引导（即：合入宿主进行引导），如未调用，则默认不需要宿主及 SDK 进行功能开启引导

- public static void setUserAccepted()
  当第三方宿主需要自己进行引导时，需要设置用户引导成功，即用户通过引导同意使用该功能。
  必须通过调用{@link com.ducaller.fsdk.main DuCallerSDKUtils}设置是否需要 SDK 引导

如未调用则默认不需要第三方及 SDK 进行用户引导，该值设置将无意义。

- public static boolean isUserAccepted()
  判断是否引导成功，即 用户是否通过引导，同意使用该功能。如通过引导，则触发场景；如为通过引导，则继续引导逻辑。
  @return true 用户已经同意， false 用户尚未同意 （第三方引导以及 SDK 引导状态都可以通过）

4.7 悬浮窗权限引导功能

如需要开启来去电悬浮窗功能,则可调用 CallerSDK 提供的接口,进行引导

- DuCallerSDKHelper.checkFloatWindowPermission
  判断悬浮窗权限接口

- DuCallerSDKHelper.applyFloatWindowPermission
  请求悬浮窗权限接口: