Android PUSH-SDK 集成指南

注意：

1. **vivo 推送服务 SDK 支持的最低 android 版本为 Android 6.0。**
2. **由于项目架构变动，可能会导致您在更新 sdk 版本时类的路径错误，重新导入类的路径即可。**

1. 集成 SDK

导入推送 aar 包

将解压后的 libs 文件夹中 vivo_pushsdk-VERSION.aar(vivo_pushsdk-VERSION.aar 为集成的 jar 包名字，VERSION 为版本名称)拷贝到您的工程的 libs 文件夹中。

引用 aar 包到工程

在 android 项目 app 目录下的 build.gradle 中添加 aar 依赖。

```
1. dependencies {
2.     implementation fileTree(include: ['*.jar'], dir: 'libs')
3.     implementation 'androidx.appcompat:appcompat:1.0.2'
4.     implementation 'androidx.constraintlayout:constraintlayout:1.1.3'
5.     // 添加 aar 依赖到工程
6.     implementation files("libs/vivo_pushsdk-2.5.3.1.aar")
7. }
```

2. 配置信息

添加权限

Vivo Push 集成只需要配置网络权限，请在当前工程 AndroidManifest.xml 中的 manifest 节点下添加以下代码：

```
<!—Vivo Push 需要的权限-->
<uses-permission android:name="android.permission.INTERNET"/>
```

配置 appid 、api key 等信息

Vivo Push 集成需要配置对应的 activity、appid 、app key 信息，其中 appid 和 app key 是在开发者平台中申请的，详见 vivo push 操作手册。
请在当前工程 AndroidManifest.xml 中的 Application 节点下添加以下代码（**建议复制粘贴防止出错**）：

```
<!--Vivo Push 需要配置的 service、activity-->
<service
    android:name="com.vivo.push.sdk.service.CommandClientService"
    android:exported="true"/>
<!--Vivo Push 开放平台中应用的 appid 和 api key-->
<meta-data
    android:name="com.vivo.push.api_key"
    android:value="xxxxxxxx"/>
<meta-data
    android:name="com.vivo.push.app_id"
    android:value="xxxx"/>
```

3. 启动推送

添加启动推送代码

在工程的 Application 中，添加以下代码，用来启动打开 push 开关，成功后即可在通知消息到达时收到通知。

```
// 在当前工程入口函数，建议在 Application 的 onCreate 函数中，添加以下代码
PushClient.getInstance(getApplicationContext()).initialize ();
// 打开 push 开关，关闭为 turnOffPush，详见 api 接入文档
PushClient.getInstance(getApplicationContext()).turnOnPush(new IPushActionListener() {
        @Override
        public void onStateChanged(int state) {
            // TODO: 开关状态处理， 0 代表成功
        }
```

```
9.      });
```

自定义通知回调类

在当前工程中新建一个类 PushMessageReceiverImpl 继承 OpenClientPushMessageReceiver 并重载实现相关方法。并在当前工程的 AndroidManifest.xml 文件中，添加自定义 Receiver 信息，代码如下：

```
1.  <!-- push 应用定义消息 receiver 声明 -->
2.  <receiver android:name="xxx.xxx.xxx.PushMessageReceiverImpl" >
3.          <intent-filter>
4.              <!-- 接收 push 消息 -->
5.              <action android:name="com.vivo.pushclient.action.RECEIVE" />
6.          </intent-filter>
7.  </receiver>
```

4. 处理推送消息

通知点击消息

当设备接收到通知消息后，查看手机的通知栏，可以看到通知栏内的该新通知展示。当点击通知时，回调 PushMessageReceiverImpl#onNotificationMessageClicked 方法。

示例代码：

```
1.  public class PushMessageReceiverImpl extends OpenClientPushMessageReceiver {
2.
3.          /***
4.           * 当通知被点击时回调此方法
5.           * @param context 应用上下文
6.           * @param msg 通知详情，详细信息见 API 接入文档
7.           */
8.          @Override
9.          public void onNotificationMessageClicked(Context context,
    UPSNotificationMessage msg) {
10.
11.         }
12.
13.         /***
```

```
14.          * 当首次 turnOnPush 成功或 regId 发生改变时，回调此方法
15.          * 如需获取 regId，请使用 PushClient.getInstance(context).getRegId()
16.          * @param context 应用上下文
17.          * @param regId 注册 id
18.          */
19.         @Override
20.         public void onReceiveRegId(Context context, String regId) {
21.
22.         }
23.     }
```

5. 混淆说明

若需要混淆 app，请在混淆文件中添加以下说明，防止 SDK 内容被二次混淆，自定义回调类切勿混淆。

```
1.  -dontwarn com.vivo.push.**
2.  -keep class com.vivo.push.**{*; }
3.  -keep class com.vivo.vms.**{*; }
4.  -keep class xxx.xxx.xxx.PushMessageReceiverImpl{*;}
```

6. 统一推送联盟接入

说明：请完成上述 ‘1.集成 SDK’ 和 ‘2.配置信息’ 两个步骤再开始统一推送联盟的接入。

打开 push 开关

这里只是做了相应的初始化操作，建议用户在自己应用的 Application 中的 onCreate()方法中调用 turnOnPush 操作。

示例代码：

```
1.  VUpsManager.getInstance().turnOnPush(this, new UPSTurnCallback() {
2.              @Override
3.              public void onResult(CodeResult codeResult) {
4.                  if(codeResult.getReturnCode() == 0){
5.                      Log.d(TAG, "初始化成功");
```

```
6.                }else {
7.                    Log.d(TAG, "初始化失败");
8.                }
9.            }
10.       });
```

注册 push

注册 push，获取申请的 regId

示例代码：

```
1.  VUpsManager.getInstance().registerToken(this, "XXX", "XXX", "XXX", new
    UPSRegisterCallback() {
2.            @Override
3.            public void onResult(TokenResult tokenResult) {
4.                if (tokenResult.getReturnCode() == 0) {
5.                    Log.d(TAG, "注册成功 regID = " + tokenResult.getToken());
6.                } else {
7.                    Log.d(TAG, "注册失败");
8.                }
9.            }
10.       });
```