Android SDK 集成指南

本文内容

本文面向 Android开发者，提供简单、快速的 JPush Android SDK 集成指导。

如果您在集成过程中遇到问题，还可以参考另外更具体的教程：

- Android SDK 集成详解 （配图详细具体的步骤）
- Android SDK 调试指南 （集成过程中可能遇到的问题）

如果您还没有下载SDK，请访问SDK下载页面下载。

产品功能说明

极光推送是一个端到端的 Push
平台，使得服务器端消息能够及时地推送终端用户手机上，让开发者积极地保持与用户的连接，从而提高用户留存率，提高活跃度。

主要功能

- 随时 Push 通知栏提示给用户
- Push 自定义消息，开发者应用程序通过接口接收到消息

主要特点

- 客户端维持连接占用资源少、耗电低
- SDK丰富的接口，可定制通知栏提示样式
- 服务器大容量、稳定

jpush-sdk_v1.x.y.zip 集成压缩包内容

- AndoridManifest.xml
  - 客户端嵌入SDK参考的配置文件
- libs/jpush-sdk-release.jar

    - SDK Java 开发包
- libs/armeabi/libpushprotocol.so
    - SDK native 开发包
- example
    - 是一个完整的 Android 项目，通过这个演示了 JPush SDK 的基本用法，可以用来做参考。

Android SDK 版本

目前SDK只支持Android 2.1或以上版本的手机系统

SDK集成步骤

1、导入 SDK 开发包到你自己的应用程序项目

- 解压缩 jpush-sdk_v1.x.y.zip 集成压缩包
- 复制 libs/jpush-sdk-release.jar 到工程 libs/ 目录下
- 复制 libs/armeabi/libpushprotocol.so 到工程 libs/armeabi 目录下

2、配置 AndroidManifest.xml

根据 SDK 压缩包里的 AndroidManifest.xml 样例文件，来配置应用程序项目的 AndroidManifest.xml 。

主要步骤为：

1. 复制备注为 "Required" 的部分
2. 将备注为替换包名的部分，替换为当前应用程序的包名
3. 将AppKey替换为在Portal上注册该应用的的Key,例如（9fed5bcb7b9b87413678c407）

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="Your Package"
    android:versionCode="100"
    android:versionName="1.0.0"
    >

    <!-- Required -->
    <permission android:name="You Package.permission.JPUSH_MESSAGE"
android:protectionLevel="signature" />

    <!-- Required -->
    <uses-permission android:name="You Package.permission.JPUSH_MESSAGE"
/>
    <uses-permission
android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
    <uses-permission
android:name="android.permission.RECEIVE_USER_PRESENT" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WAKE_LOCK" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission
android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    <uses-permission
```

```
android:name="android.permission.READ_EXTERNAL_STORAGE" />
    <uses-permission android:name="android.permission.VIBRATE" />
    <uses-permission
android:name="android.permission.MOUNT_UNMOUNT_FILESYSTEMS" />
    <uses-permission
android:name="android.permission.ACCESS_NETWORK_STATE" />

    <!-- Optional. Required for location feature -->
    <uses-permission
android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission
android:name="android.permission.ACCESS_COARSE_UPDATES" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"
/>
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE"
/>
    <uses-permission
android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission
android:name="android.permission.ACCESS_LOCATION_EXTRA_COMMANDS" />
    <uses-permission
android:name="android.permission.CHANGE_NETWORK_STATE" />


    <application
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:name="Your Application">

        <!-- Required -->
        <service
            android:name="cn.jpush.android.service.PushService"
            android:enabled="true"
            android:exported="false" >
            <intent-filter>
                <action android:name="cn.jpush.android.intent.REGISTER" />
                <action android:name="cn.jpush.android.intent.REPORT" />
                <action android:name="cn.jpush.android.intent.PushService"
/>
            </intent-filter>
        </service>

        <!-- Required -->
        <receiver
            android:name="cn.jpush.android.service.PushReceiver"
            android:enabled="true" >
            <intent-filter>
                <action
android:name="android.intent.action.BOOT_COMPLETED" />
                <action android:name="android.intent.action.USER_PRESENT"
/>
                <action
android:name="android.net.conn.CONNECTIVITY_CHANGE" />
```

```
            </intent-filter>
            <intent-filter>
                <action android:name="android.intent.action.PACKAGE_ADDED"
/>
                <action
android:name="android.intent.action.PACKAGE_REMOVED" />
                <data android:scheme="package" />
            </intent-filter>
        </receiver>
   <!-- Required SDK:RICH PUSH-->
        <activity
            android:name="cn.jpush.android.ui.PushActivity"
            android:theme="@android:style/Theme.Translucent.NoTitleBar"
            android:configChanges="orientation|keyboardHidden" >
            <intent-filter>
                <action android:name="cn.jpush.android.ui.PushActivity" />
                <category android:name="android.intent.category.DEFAULT"
/>
                <category android:name="Your Package" />
            </intent-filter>
        </activity>
        <!-- Required SDK:RICH PUSH-->
        <service
            android:name="cn.jpush.android.service.DownloadService"
            android:enabled="true"
            android:exported="false" >
        </service>
        <!-- Required SDK-->
        <receiver android:name="cn.jpush.android.service.AlarmReceiver" />

        <!-- Required. For publish channel feature -->
        <meta-data android:name="JPUSH_CHANNEL"
android:value="developer-default"/>
        <!-- Required. AppKey copied from Portal -->
        <meta-data android:name="JPUSH_APPKEY" android:value="Your
AppKey"/>
    </application>
</manifest>
```

3、添加代码

JPush SDK 提供的 API 接口，都主要集中在 cn.jpush.android.api.JPushInterface 类里。

基础 API

- init 初始化SDK

```
public static void init(Context context)
```

- setDebugMode 设置调试模式

```
// You can enable debug mode in developing state. You should close debug
mode when release.
public static void setDebugMode(boolean debugEnalbed)
```

调用示例代码（参考 example 项目）

- init 只需要在应用程序启动时调用一次该 API 即可。
- Application AndoridManifest.xml AndroidManifest.xml example

```
public class ExampleApplication extends Application {
     @Override
     public void onCreate() {
         super.onCreate();
         JPushInterface.setDebugMode(true);
         JPushInterface.init(this);
     }
}
```

4、测试确认

1. 确认所需的权限都已经添加。如果必须的权限未添加，日志会提示错误。
2. 确认 AppKey（在Portal上生成的）已经正确的写入 Androidmanifest.xml 。
3. 确认在程序启动时候调用了init(context) 接口
4. 确认测试手机（或者模拟器）已成功连入网络
    - 客户端调用 init 后不久，如果一切正常，应有登录成功的日志信息
5. 启动应用程序，在 Portal 上向应用程序发送自定义消息或者通知栏提示。详情请参考管理Portal。
    - 在几秒内，客户端应可收到下发的通知或者正定义消息

如果 SDK 工作正常，则日志信息会如下图所示：

com.quentin	JPush	[JPushInterface] action:init
com.quentin	JPush	[JPush] metadata: appKey - 72d7bd7b7a896deb0175e9ff
com.quentin	JPush	[JPush] metadata: channel - developer-default
com.quentin	JPush	[PushService] action:checkValidManifest
com.quentin	JPush	[PushService] Login succeed!

如图所示，客户端启动分为 4 步：

1. 检查 metadata 的 appKey 和 channel ，如果不存在，则启动失败
2. 初始化 JPush SDK，检查 JNI 等库文件的有效性，如果库文件无效，则启动失败
3. 检查 Androidmanifest.xml，如果有 Required 的权限不存在，则启动失败
4. 连接服务器登录，如果存在网络问题，或者前面三步有问题，则登陆失败

高级功能

请参考：

统计 API

标签与别名API

通知栏样式定制API

接收推送消息

技术支持

邮件联系： support@jpush.cn

技术QQ群：132992583