Maven集成（推荐）

前言

- **本文档介绍Android Studio提供的基于Maven的快速集成方案，配置简单、不容易出问题、后续更新维护方便，因此我们强烈推荐应用开发者根据本文档步骤进行个推集成。**
- 请参考 `Getui_SDK_Demo_AS_maven` Demo工程

1. 创建个推应用

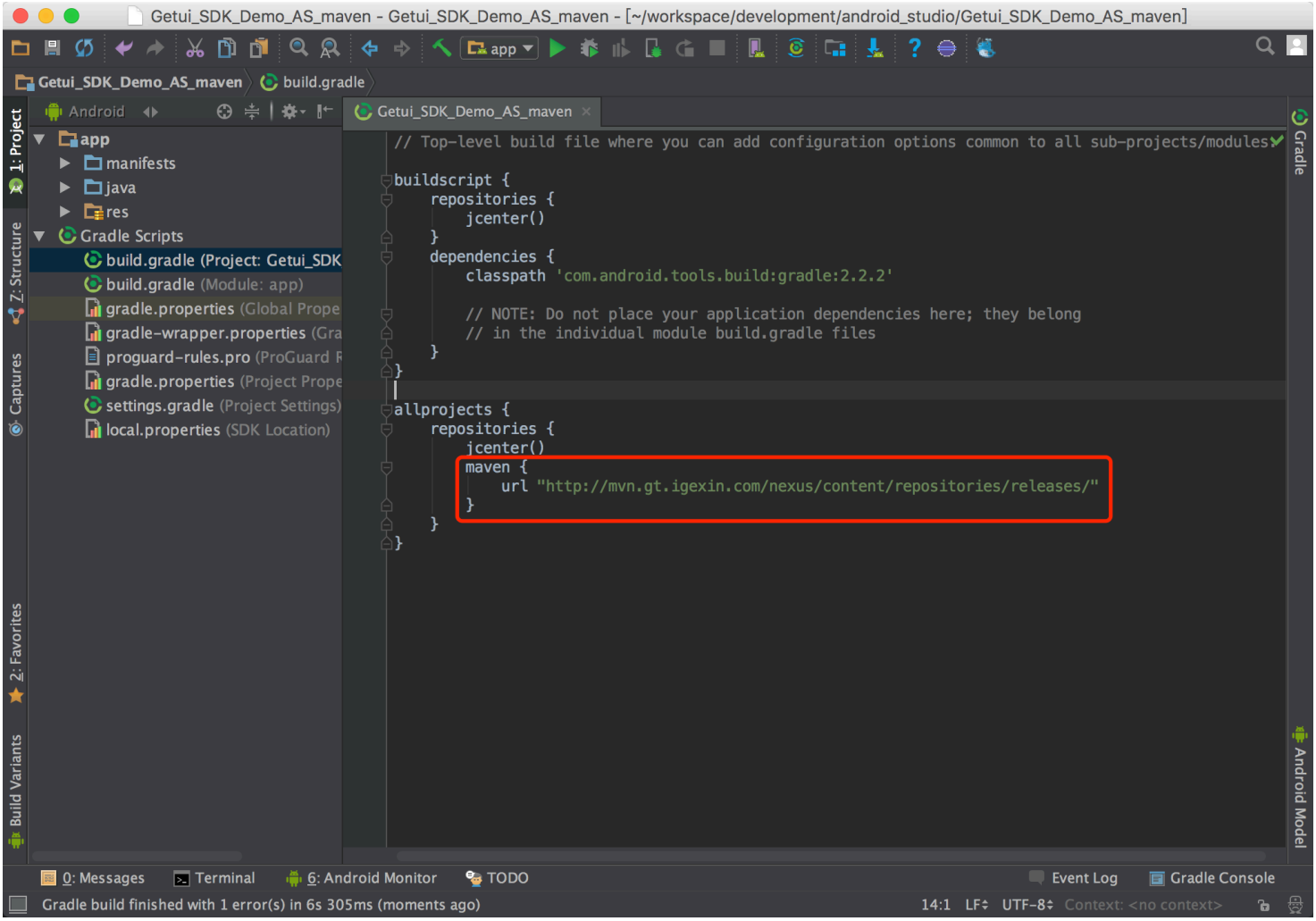
- 请参考 创建应用 获取相应的 `AppID` 、 `AppKey` 、 `AppSecret` 信息。

2. 添加个推相关配置

由手动集成切换到Maven，需删除原有集成配置再进行新集成

2.1 添加Maven库地址

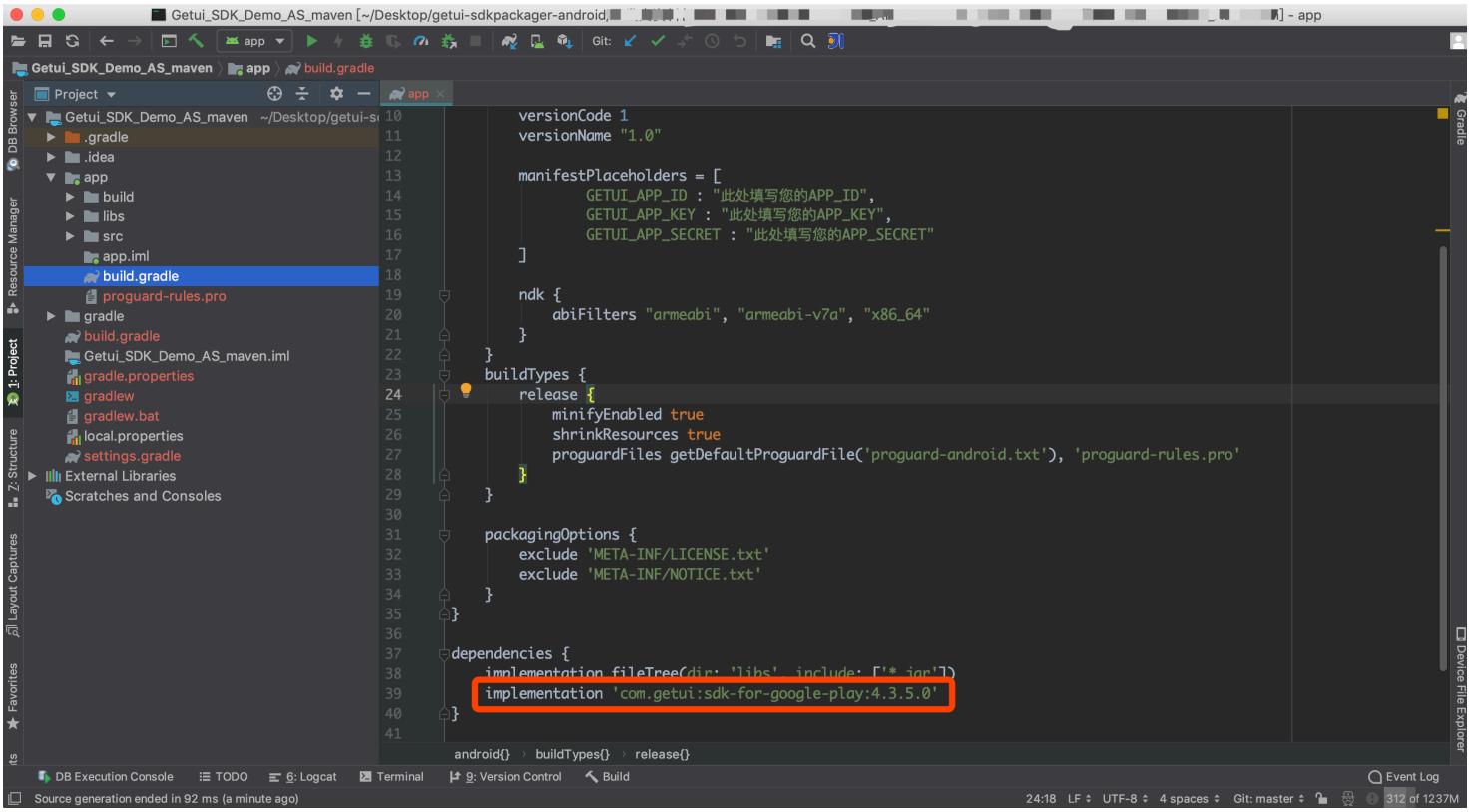
- 在项目根目录 `build.gradle` 文件中，添加个推maven库地址，如下所示：




```
maven {
    url "http://mvn.gt.igexin.com/nexus/content/repositories/releases/"
}
```

2.2 配置依赖

- 在 `app/build.gradle` 文件中引用个推SDK依赖库，如下图所示：



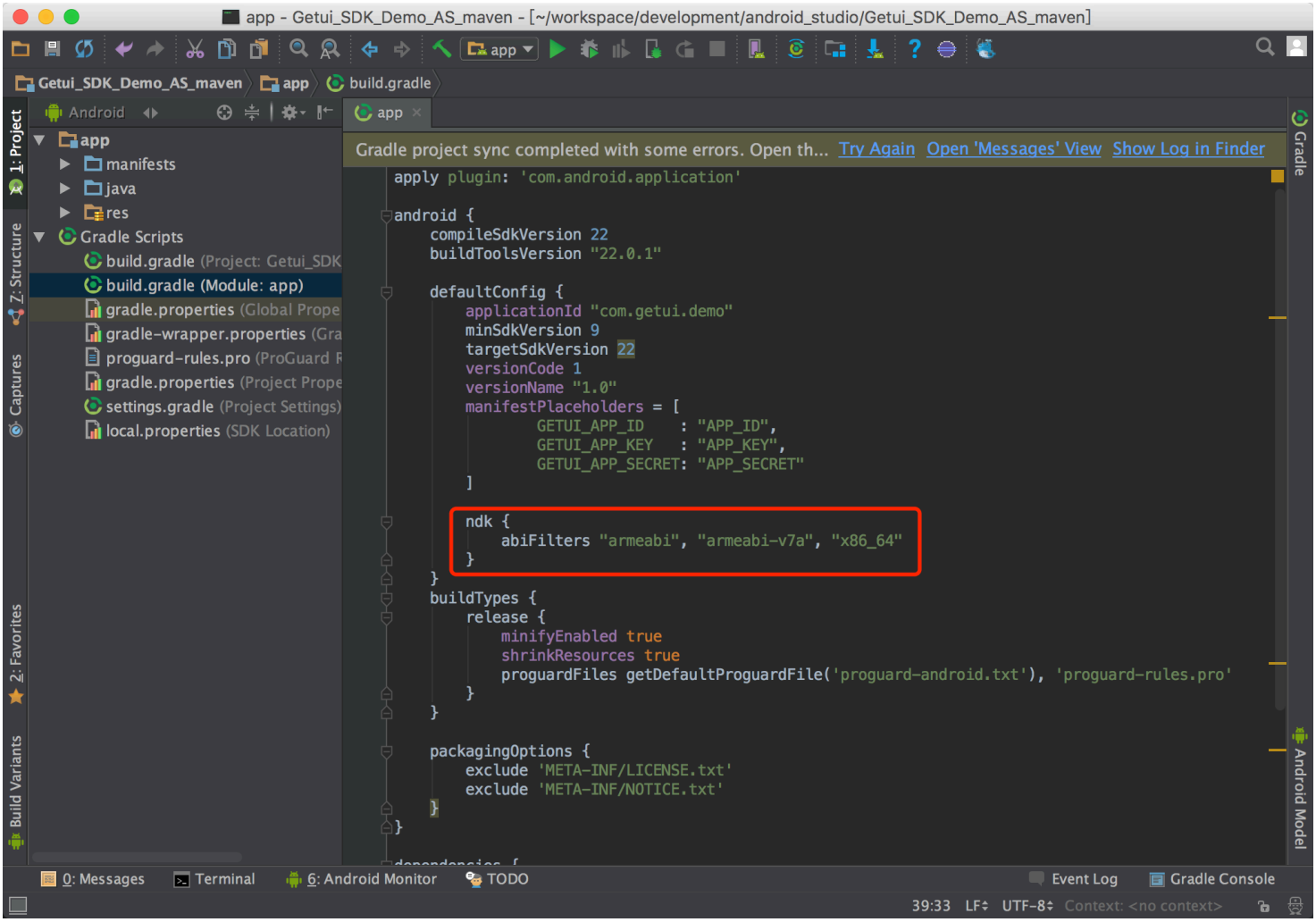

```
dependencies {
    compile fileTree(dir: 'libs', include: ['*.jar'])
    compile 'com.getui:sdk-for-google-play:4.3.8.0'
}
```

2.3 配置 so 库

目前个推SDK支持 `armeabi` 、 `armeabi-v7a` 、 `arm64-v8a` 、 `mips` 、 `mips64` 、 `x86` 、 `x86_64` 这几种 CPU 架构，请根据项目情况指定所需的架构。

如果项目中其他 so 库只支持其中某几种 cpu 架构，那么应该根据其他 so 库所支持的 CPU 架构的最小集来配置。否则如果在特定架构上未能支持所有 so 库，则很可能导致程序运行异常。切记！

- 在 `app/build.gradle` 文件中的 `android.defaultConfig` 下指定所需的 CPU 架构，如下图所示：




```
android {
  ...
  defaultConfig {
    ...
    ndk {
        // 添加项目所需CPU类型的最小集
      abiFilters "armeabi", "armeabi-v7a", "x86_64"
    }
  }
}
```

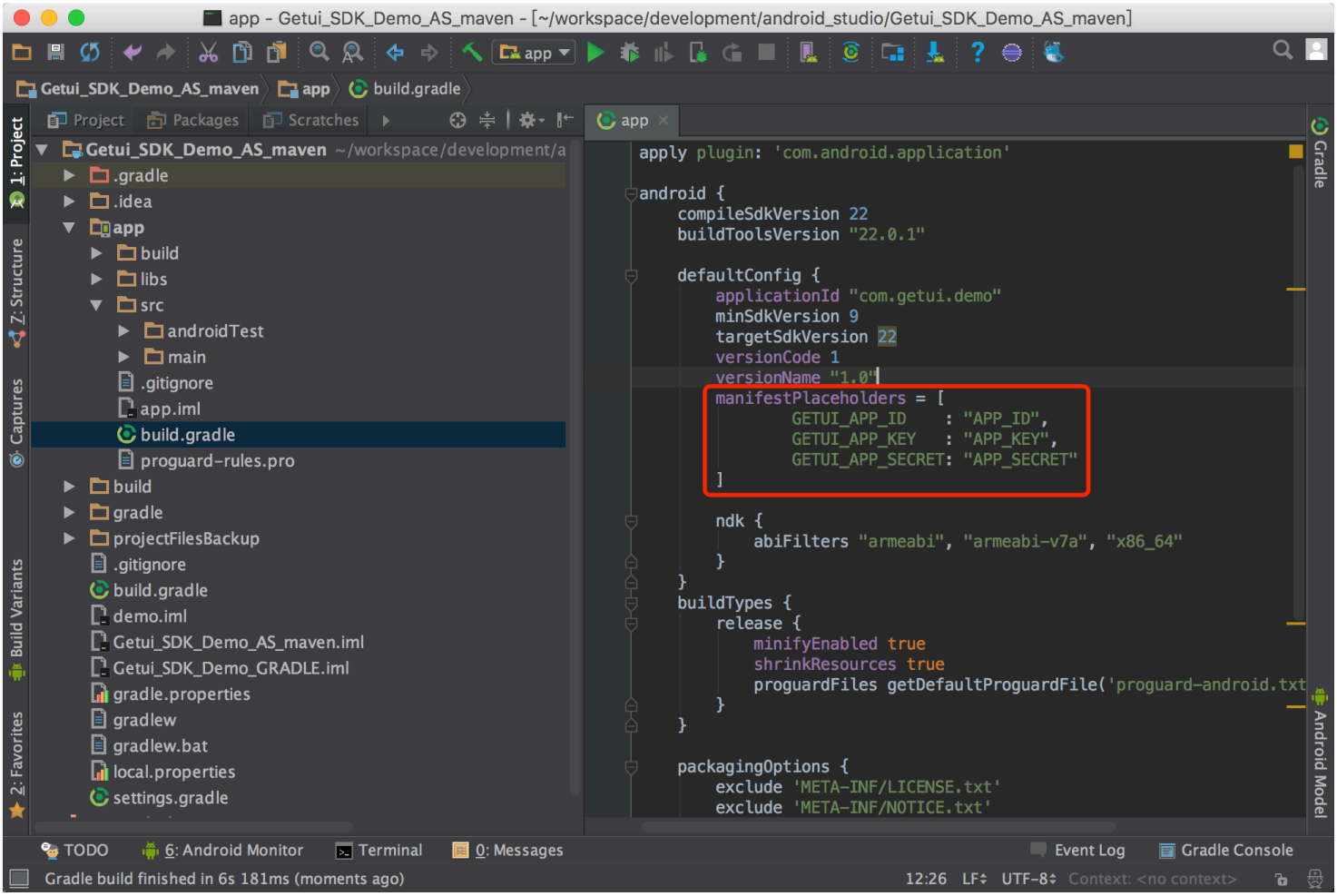
- 若AndroidStudio编译出现以下报错：

```
NDK integration is deprecated in the current plugin. Consider trying the new experimental plugin.
```

请在项目根目录 `gradle.properties` 文件中添加： `gradle android.useDeprecatedNdk=true`

2.4 配置个推应用参数

- 在 `app/build.gradle` 文件中的 `android.defaultConfig` 下添加 `manifestPlaceholders` ，配置个推相关的应用参数（参见【步骤1】），如下图所示：




```
manifestPlaceholders = [
    GETUI_APP_ID : "APP_ID",
    GETUI_APP_KEY : "APP_KEY",
    GETUI_APP_SECRET : "APP_SECRET"
]
```

- 请根据【步骤1】获取到的应用参数进行相应替换 `APP_ID` 、 `APP_KEY` 、 `APP_SECRET` 的值

2.5 适配Android P

- 在Android 9.0以上默认不支持http通信，为保证SDK正常使用，maven库中已添加如下属性（注：如遇工程报错找不到该属性，将 `app/build.gradle` 中的compileSdkVersion改为23以上即可）：

```
<application android:usesCleartextTraffic="true">
```

2.6 配置自定义推送服务

为了让推送服务在部分主流机型上更稳定运行，个推支持第三方应用配置使用自定义Service来作为推送服务运行的载体。

- 在项目源码中添加一个继承自com.igexin.sdk.PushService的自定义Service：

```
package com.getui.demo;

// 仅2.13.1.0及以上版本才能直接extends PushService，低于此版本请延用之前实现方式
public class DemoPushService extends com.igexin.sdk.PushService {

}
```

- 在AndroidManifest.xml中添加上述自定义Service (使用maven集成，process属性必须为 `pushservice` ，切勿更改）：

```
<service
  android:name="com.getui.demo.DemoPushService"
  android:exported="true"
  android:label="PushService"
  android:process=":pushservice">
</service>
```

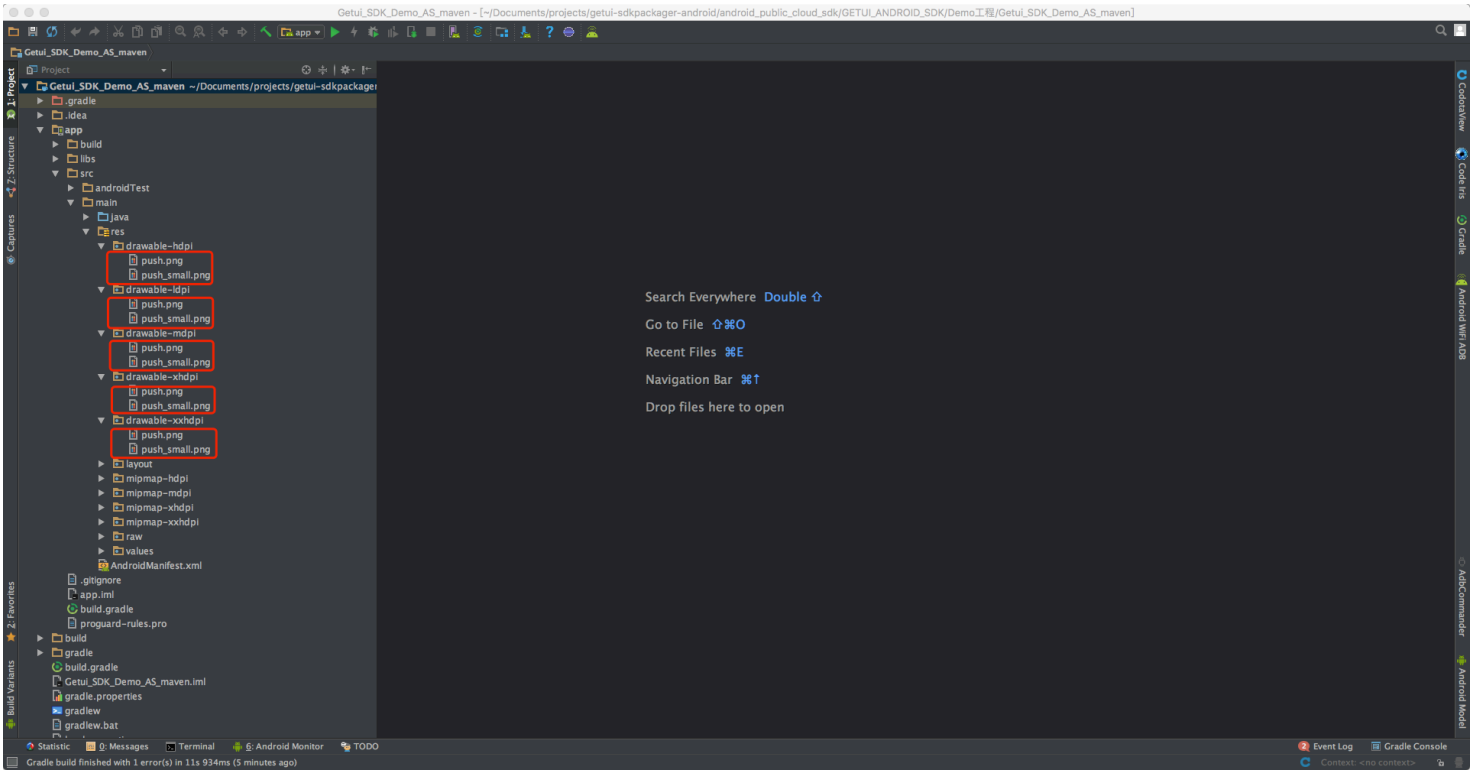
2.7 配置可选权限

- 上述接入方式已包含个推服务所需的所有必备权限。在此之外，您还可以配置以下可选权限，以便使用个推3.0提供的电子围栏功能。请在 `AndroidManifest.xml` 的 `<manifest>` 根标签下添加如下配置：

```
<!-- 个推3.0电子围栏功能所需权限 -->
<uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

2.8 导入通知栏图标

- 为了修改默认的通知图标以及通知栏顶部提示小图标，请在资源目录的 `res/drawable-ldpi/` 、 `res/drawable-mdpi/` 、 `res/drawable-hdpi/` 、 `res/drawable-xhdpi/` 、 `res/drawable-xxhdpi/` 等各分辨率目录下，放置相应尺寸的文件名为 `push.png` 和 `push_small.png` 图片，如图所示：



- 建议的 `push.png` 图片尺寸如下：

```
ldpi:    48*48
mdpi:    64*64
hdpi:    96*96
xhdpi:   128*128
xxhdpi:  192*192
```

- 该图标 `push.png` 将会作为通知图标，如下所示：

- 建议的 `push_small.png` 图片尺寸如下：

```
ldpi:    18*18
mdpi:    24*24
hdpi:    36*36
xhdpi:   48*48
xxhdpi:  72*72
xxxhdp:  96*96
```

- 该图标 `push_small.png` 将会作为通知图标展示在通知栏顶部，如下所示：

2.9 资源精简配置

2.9.1 shrinkResources

- 如果您的工程启用了资源精简，即如果在 `app/build.gradle` 的 `android.buildTypes.release` 下配置了 `shrinkResources true` ，为了避免个推SDK所需资源被错误精简导致功能异常，需要在项目资源目录 `res/raw` 中添加 `keep.xml` 文件
- 如果你的项目工程已经使用了 `keep.xml` ，则只需在 `tools:keep` 中增加如下声明：

```
<?xml version="1.0" encoding="utf-8"?>
<resources
    xmlns:tools="http://schemas.android.com/tools"
    tools:keep="@layout/other_res,...,
    @drawable/push,
    @drawable/push_small"/>
   <!-- 若您需要使用其他自定义推送图标，也需要在此处添加 -->
```

2.9.2 AndResGuard

- 如果您的工程使用了 `AndResGuard` 进行资源精简，为了避免个推SDK所需资源被错误精简导致功能异常，需要为个推添加白名单配置。

gradle集成 `AndResGuard` 的方式，需要您在 `andResGuard` 的 `whiteList` 节点下添加如下代码:

```
andResGuard {
    ...
    whiteList = [
           ...
           // for getui
           "R.drawable.push",
           "R.drawable.push_small",
           "R.id.getui_*"
           // 若您需要使用其他自定义推送图标，也需要在此处添加
    ]
    ...
}
```

命令行使用 `AndResGuard` 的方式，需要您在 `config.xml` 文件中的 `<issue id=whitelist>` 节点下添加如下代码

```
<issue id="whitelist" isactive="true">
    <path value="<your_package_name>.R.drawable.push"/>
    <path value="<your_package_name>.R.drawable.push_small"/>
    <path value="<your_package_name>.R.id.getui_*"/>
    <!-- 若您需要使用其他自定义推送图标，也需要在此处添加 -->
</issue>
```

3. 编写集成代码

3.1 初始化SDK

我们建议开发者在主进程的Application.onCreate()、Activity.onCreate()方法中初始化个推SDK。多次调用SDK初始化并无影响。

- 个推SDK初始化方法。如果使用了自定义推送服务，调用时还需要传入自定义推送服务名：

```
// DemoPushService 为【步骤2.6】自定义的推送服务
com.igexin.sdk.PushManager.getInstance().initialize(getApplicationContext(), com.getui.demo.DemoPushService.class);
```

3.2 接收推送服务事件

- 在项目源码中添加一个继承自com.igexin.sdk.GTIntentService的类，用于接收CID、透传消息以及其他推送服务事件。请参考下列代码实现各个事件回调方法：

```
package com.getui.demo;

import android.content.Context;
import android.os.Message;
import android.util.Log;

import com.igexin.sdk.GTIntentService;
import com.igexin.sdk.PushConsts;
import com.igexin.sdk.PushManager;
import com.igexin.sdk.message.FeedbackCmdMessage;
import com.igexin.sdk.message.GTCmdMessage;
import com.igexin.sdk.message.GTTransmitMessage;
import com.igexin.sdk.message.SetTagCmdMessage;

/**
 * 继承 GTIntentService 接收来自个推的消息，所有消息在线程中回调，如果注册了该服务，则务必要在 AndroidManifest中声明，否则无法接受消息<br>
 */
public class DemoIntentService extends GTIntentService {

    public DemoIntentService() {
    }

    @Override
    public void onReceiveServicePid(Context context, int pid) {
    }

    // 处理透传消息
    @Override
    public void onReceiveMessageData(Context context, GTTransmitMessage msg) {
        // 透传消息的处理方式，详看SDK demo
    }

    // 接收 cid
    @Override
    public void onReceiveClientId(Context context, String clientid) {
        Log.e(TAG, "onReceiveClientId -> " + "clientid = " + clientid);
    }

    // cid 离线上线通知
    @Override
    public void onReceiveOnlineState(Context context, boolean online) {
    }

    // 各种事件处理回执
    @Override
    public void onReceiveCommandResult(Context context, GTCmdMessage cmdMessage) {
    }

    // 通知到达，只有个推通道下发的通知会回调此方法
    @Override
    public void onNotificationMessageArrived(Context context, GTNotificationMessage msg) {
    }

    // 通知点击，只有个推通道下发的通知会回调此方法
    @Override
    public void onNotificationMessageClicked(Context context, GTNotificationMessage msg) {
    }
}
```

- 在 `AndroidManifest.xml` 中配置上述 IntentService 类：

```
<!-- permission属性在2.13.1.0版本后必须配置 -->
<service
    android:name="com.getui.demo.DemoIntentService"
    android:permission="android.permission.BIND_JOB_SERVICE"/>
```

- 在个推SDK初始化后，注册上述 IntentService 类：

```
// DemoIntentService 为第三方自定义的推送服务事件接收类
com.igexin.sdk.PushManager.getInstance().registerPushIntentService(getApplicationContext(), com.getui.demo.DemoInte
ntService.class);
```

3.3 授权隐私条款

- GooglePlay最新政策要求上架的应用必须弹框明示应用所需的权限列表以及数据获取行为，且经用户手动授权同意后才能使用。
- 基于以上政策，个推新增以下接口供开发者调用：

```
// enable表示用户是否授权
com.igexin.sdk.PushManager.getInstance().setPrivacyPolicyStrategy(Context context, boolean enable);
```

- 开发者可按照以下流程进行适配。

3.3.1 弹窗提示

- 首先，弹窗展示 `条款与声明` ，示例代码：

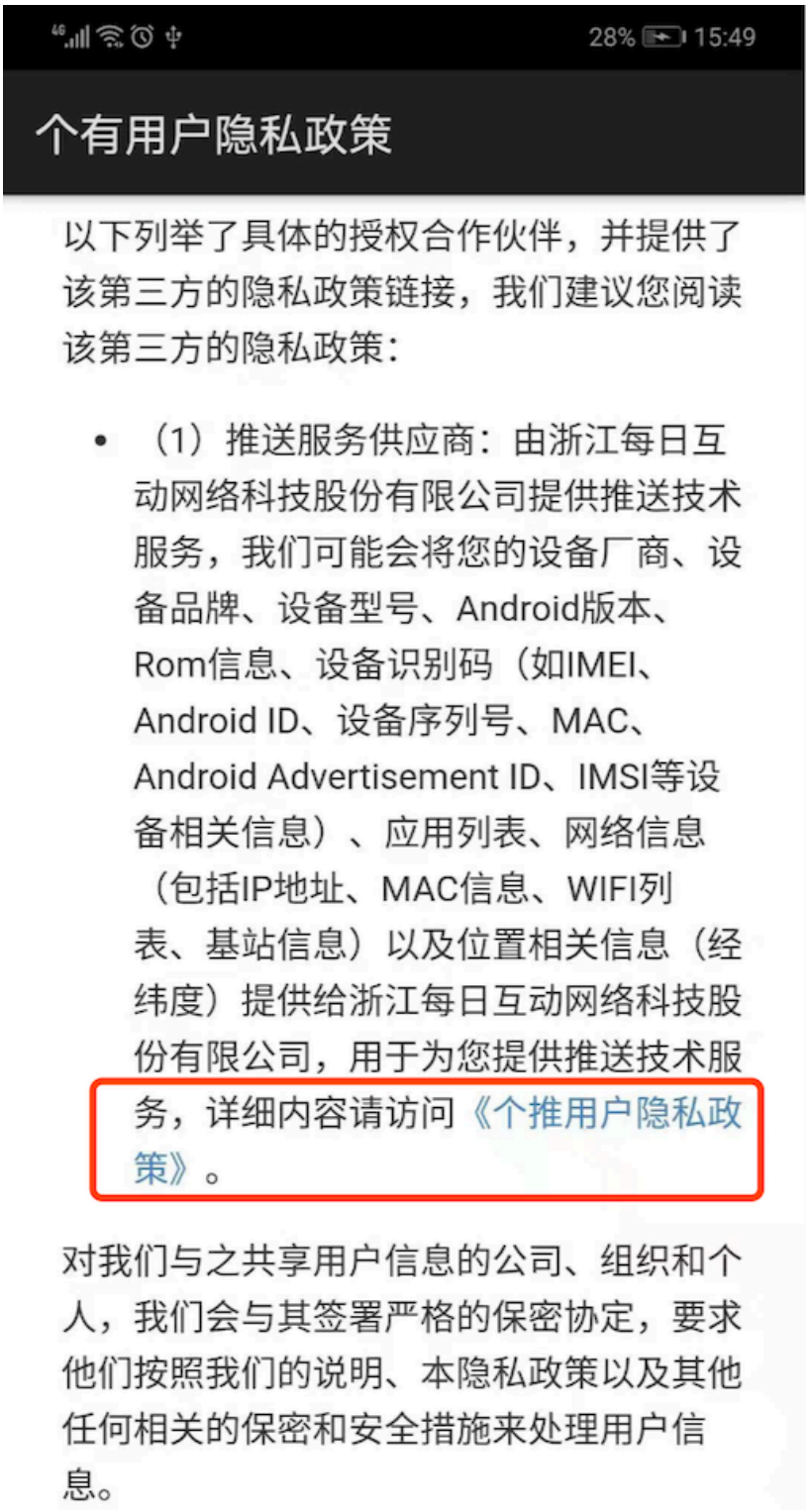
```
AlertDialog dialog = new AlertDialog.Builder(this).setCancelable(false)
                .setTitle("条款与声明")
                // layout为自定义布局，可参考Demo工程GetuiSdkDemoActivity类中的实现
                .setView(layout)
                .setPositiveButton("同意", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i) {
                        // 点击“同意”
                        boolean isPushEnabled = pushCB.isChecked();
                        if (isPushEnabled) {
                                // 勾选“消息推送”,选择框默认勾选
                            PushManager.getInstance().setPrivacyPolicyStrategy(context, true);
                        } else {
                                // 取消勾选“消息推送”
                            PushManager.getInstance().setPrivacyPolicyStrategy(context, false);
                        }
                    }
                }).setNegativeButton("拒绝", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i) {
                      // 点击“拒绝”
                        PushManager.getInstance().setPrivacyPolicyStrategy(context, false);
                    }
                }).create();
        dialog.setOnShowListener(new DialogInterface.OnShowListener() {
            @Override
            public void onShow(DialogInterface dialogInterface) {
                dialog.getButton(AlertDialog.BUTTON_NEGATIVE).setTextColor(Color.GRAY);
            }
        });

        dialog.show();
```

- 建议在用户点击“同意”后，调用 `PushManager.getInstance().setPrivacyPolicyStrategy(context, enabled)` 方法（enabled默认为true，除非用户取消“消息推送”勾选）
- 弹框展示效果如下：

- 上图文案模板在资源目录的 `隐私政策参考模板/弹窗文案.docx` 中，开发者需自行补全、修改文档中的黄色背景部分（切勿直接拷贝使用）！

3.3.2 应用隐私条款

- 点击弹框中的 `《xxx用户隐私政策》` 链接后，跳转至 `应用隐私条款` 页面（开发者自行实现）。在此页面中，需要对应用中的隐私数据进行详细说明。
- 应用示例：



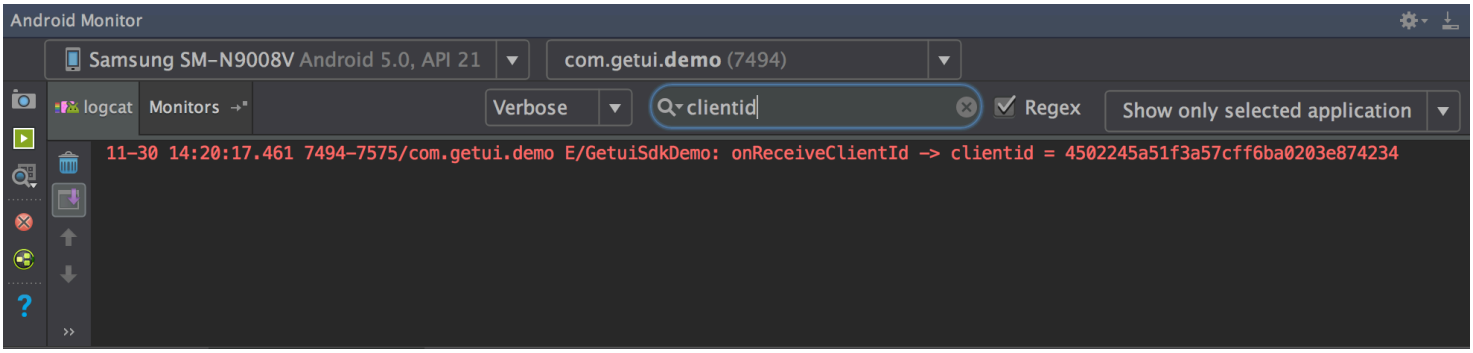
- 其中推送部分的说明，可使用个推提供的《个推用户隐私政策》。目前有中、英文版本（以后会支持更多语言版本），链接地址分别为：`https://legal.igexin.com/privacy_zh.html` 和 `https://legal.igexin.com/privacy_en.html`
- 上图文案模板在资源目录的 `隐私政策参考模板/用户隐私政策（模板）.md` 中，开发者需自行补全、修改文档中的“{}”及“待补充”部分（切勿直接拷贝使用）！也可将此文件转换成想要的html样式
- 至此，GooglePlay隐私政策适配完成。

4. 测试

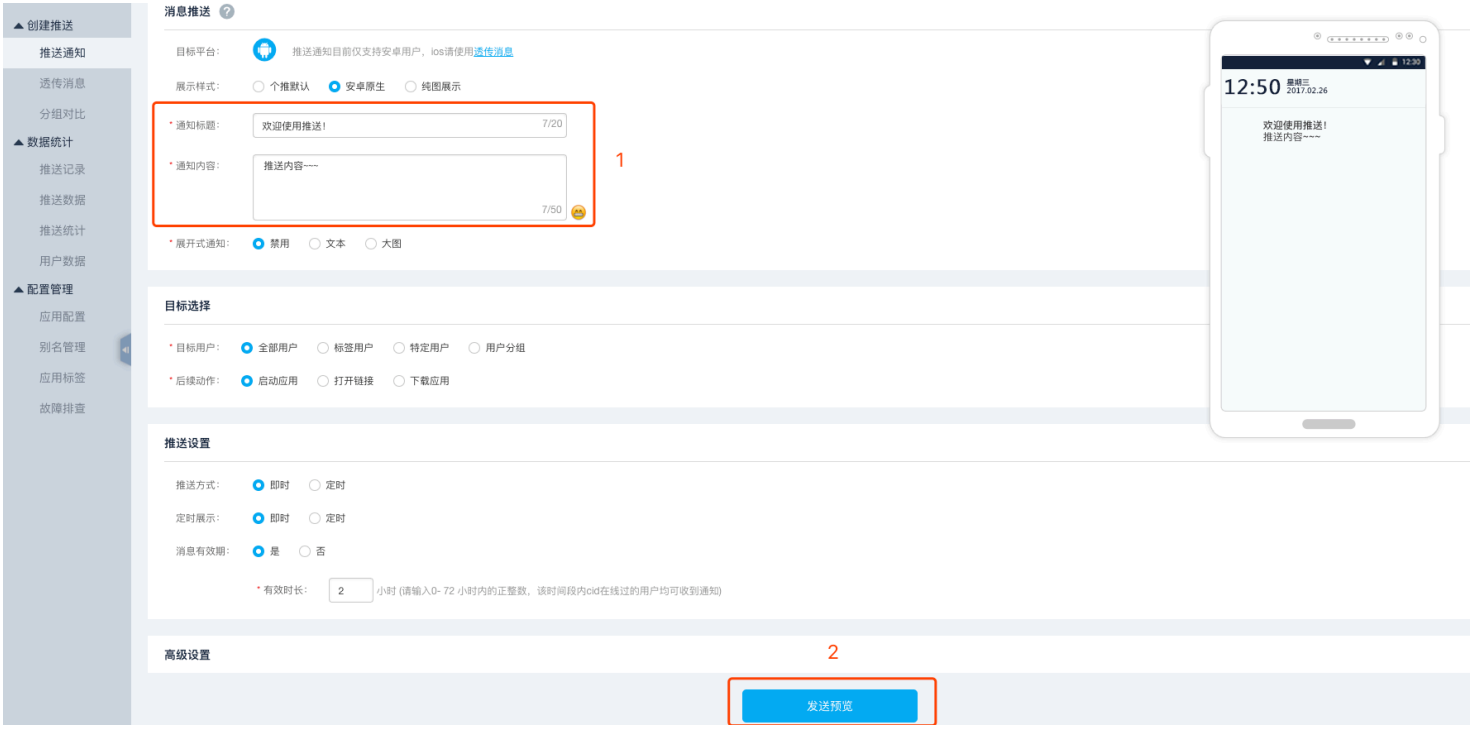
4.1 验证clientid和通知

- 连接手机或启动Android模拟器，编译运行你的工程，查看logcat信息。在搜索框中输入 `clientid` ，如果能显示 `clientid is xxx` 日志，

则说明个推SDK已经成功运行起来了：



- 登录 个推开发者中心，进入【个推·消息推送】产品，点击【创建推送】，进入待测试应用的推送通知界面：



- 依次填写 通知标题 和 通知内容，点击 发送 按钮即可向该推送应用名下所有CID推送通知消息。具体推送操作方法详见：创建推送通知
- 如果手机或模拟器收到消息，显示下图所示通知，那么恭喜您，个推SDK接入测试已经成功完成！

4.2 验证结果

- 若【步骤4.1】验证失败，建议重新按步骤仔细检查集成！！

- 可参考个推SDK集成Demo，资料包中的路径如下：

```
GETUI_ANDROID_SDK/
  | - Demo工程/
  |    |- Getui_SDK_Demo_AS_maven/ （AndroidStudio快速集成Demo工程）
```