一. AndroidManifest.xml 文件配置
打开 Androidmanifest.xml，添加以下配置（建议参考下载包的 Demo 修改），其中 YOUR_ACCESS_ID 和
YOUR_ACCESS_KEY 替换为 APP 对应的 accessId 和 accessKey,请确保按照要求配置，否则可能导致服务不能正常使用

```
<application
<!-- APP 项目的其它配置... -->
<!-- 【必须】 (2.30 及以上版新增)展示通知的 activity -->
<activity
android:name="com.tencent.android.tpush.XGPushActivity"
android:theme="@android:style/Theme.Translucent"
android:exported="false" >
<intent-filter>
<!-- 若使用 AndroidStudio，请设置 android:name="android.intent.action"-->
<action android:name="" />
</intent-filter>
</activity>
<!-- 【必须】 信鸽 receiver 广播接收 -->
<receiver
android:name="com.tencent.android.tpush.XGPushReceiver"
android:process=":xg_service_v3" >
<intent-filter android:priority="0x7fffffff" >
<!-- 【必须】 信鸽 SDK 的内部广播 -->
<action android:name="com.tencent.android.tpush.action.SDK" />
<action android:name="com.tencent.android.tpush.action.INTERNAL_PUSH_MESSAGE" />
<!-- 【必须】 系统广播：网络切换 -->
<action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
<!-- 【可选】 系统广播：开屏 -->
<action android:name="android.intent.action.USER_PRESENT" />
<!-- 【可选】 一些常用的系统广播，增强信鸽 service 的复活机会，请根据需要选择。当然，你也可以添加 APP 自定义的一
些广播让启动 service -->
<action android:name="android.bluetooth.adapter.action.STATE_CHANGED" />
<action android:name="android.intent.action.ACTION_POWER_CONNECTED" />
<action android:name="android.intent.action.ACTION_POWER_DISCONNECTED" />
</intent-filter>
<!-- 【可选】 usb 相关的系统广播，增强信鸽 service 的复活机会，请根据需要添加 -->
<intent-filter android:priority="0x7fffffff" >
<action android:name="android.intent.action.MEDIA_UNMOUNTED" />
<action android:name="android.intent.action.MEDIA_REMOVED" />
<action android:name="android.intent.action.MEDIA_CHECKING" />
<action android:name="android.intent.action.MEDIA_EJECT" />
<data android:scheme="file" />
</intent-filter>
</receiver>
<!-- 【必须】 信鸽 service -->
<service
android:name="com.tencent.android.tpush.service.XGPushServiceV3"
android:exported="true"
android:persistent="true"
android:process=":xg_service_v3" />
```

```
 <!-- 【必须】 提高 Service 的存活率 -->
 <service android:name="com.tencent.android.tpush.rpc.XGRemoteService"
                  android:exported="true" >
                  <intent-filter>
<!-- 【必须】 请修改为当前 APP 名包.PUSH_ACTION，如 demo 的包名为：com.qq.xgdemo -->
                        <action android:name="您的包名.PUSH_ACTION" />
                  </intent-filter>
            </service>

<!-- 【必须】增强 xg_service 存活率 -->
<service
android:name="com.tencent.android.tpush.service.XGDaemonService"
android:process=":xg_service_v3" />
<!-- 【可选】APP 实现的 Receiver，用于接收消息透传和操作结果的回调，请根据需要添加 -->
<!-- YOUR_PACKAGE_PATH.CustomPushReceiver 需要改为自己的 Receiver： -->
<receiver android:name="com.qq.xgdemo.receiver.MessageReceiver"
android:exported="true" >
<intent-filter>
<!-- 接收消息透传 -->
<action android:name="com.tencent.android.tpush.action.PUSH_MESSAGE" />
<!-- 监听注册、反注册、设置/删除标签、通知被点击等处理结果 -->
<action android:name="com.tencent.android.tpush.action.FEEDBACK" />
</intent-filter>
</receiver>
<!-- 【必须】 【注意】authorities 修改为 包名.AUTH_XGPUSH, 如 demo 的包名为：
com.qq.xgdemo-->
<provider
android:name="com.tencent.android.tpush.XGPushProvider"
android:authorities="com.qq.xgdemo.AUTH_XGPUSH"
android:exported="true"
/>
<!-- 【必须】 【注意】authorities 修改为 包名.TPUSH_PROVIDER, 如 demo 的包名为：
com.chace.secondgcm-->
<provider
android:name="com.tencent.android.tpush.SettingsContentProvider"
android:authorities="com.qq.xgdemo.TPUSH_PROVIDER"
android:exported="false"
/>
<!-- 【必须】 【注意】authorities 修改为 包名.TENCENT.MID.V3, 如 demo 的包名为：
com.qq.xgdemo-->
<provider
android:name="com.tencent.mid.api.MidProvider"
android:authorities="com.qq.xgdemo.TENCENT.MID.V3"
android:exported="true" >
</provider>
<!-- 【必须】 请修改为 APP 的 AccessId，“21”开头的 10 位数字，中间没空格 -->
<meta-data
android:name="XG_V2_ACCESS_ID"
android:value="2100047428" />
<!-- 【必须】 请修改为 APP 的 AccessKey，“A”开头的 12 位字符串，中间没空格 -->
<meta-data
```

```
android:name="XG_V2_ACCESS_KEY"
android:value="AKH3A5485ETM" />
</application>
<!-- 【必须】 信鸽 SDK 所需权限 3.0 版本可选 WRITE_SETTINGS 权限-->
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.VIBRATE" />
<!-- 【常用】 信鸽 SDK 所需权限 -->
<uses-permission android:name="android.permission.RECEIVE_USER_PRESENT" />
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<!-- 【可选】 信鸽 SDK 所需权限 -->
<uses-permission android:name="android.permission.RESTART_PACKAGES" />
<uses-permission android:name="android.permission.BROADCAST_STICKY" />
<uses-permission android:name="android.permission.KILL_BACKGROUND_PROCESSES" />
<uses-permission android:name="android.permission.GET_TASKS" />
<uses-permission android:name="android.permission.READ_LOGS" />
<uses-permission android:name="android.permission.BLUETOOTH" />
<uses-permission android:name="android.permission.BATTERY_STATS" />
```

AndroidManifest.xml 和信鸽 2.x 版本的区别

AndroidManifest.xml 配置和 2.x 版本的信鸽是有不少变化的,请确保新版本配置和上面所说的一致, 变化点有如下:

XGPushService 修改为 XGPushServiceV3

所以进程名 android:process=":xg_service_v2"修改为 android:process=":xg_service_v3"

删除 com.tencent.android.tpush.rpc.XGRemoteService"节点

增加 com.tencent.android.tpush.XGPushProvider 节点, 注意 authorities 为 包名.AUTH_XGPUSH, 如 demo 的包名为：

com.qq.xgdemo, 那这里配置为 com.qq.xgdemo.AUTH_XGPUSH

增加 com.tencent.android.tpush.SettingsContentProvider 节点，注意 authorities 为 包名.TPUSH_PROVIDER, 如

demo 的包名为：com.qq.xgdemo, 那这里配置为 com.qq.xgdemo.TPUSH_PROVIDER

增加 com.tencent.mid.api.MidProvider 节点, 注意 authorities 为 包名.TENCENT.MID.V3, 如 demo 的包名为：

com.qq.xgdemo, 那这里配置为 com.qq.xgdemo.TENCENT.MID.V3

增加 com.tencent.android.tpush.service.XGDaemonService 节点(之前是可选的)

WRITE_SETTINGS 权限去掉

二. 启动并注册 APP

完成工程配置后，打开工程的主 Activity，在其 onCreate(Bundle savedInstanceState)重载方法内，添加以下代码，完成信鸽服

务的启动与 APP 注册过程。

```
// 开启 logcat 输出，方便 debug，发布时请关闭
// XGPushConfig.enableDebug(this, true);
// 如果需要知道注册是否成功，请使用 registerPush(getApplicationContext(), XGIOperateCallback) 带 callback 版本
// 如果需要绑定账号，请使用 registerPush(getApplicationContext(),account)版本
// 具体可参考详细的开发指南
```

// 传递的参数为 ApplicationContext
Context context = getApplicationContext();
XGPushManager.registerPush(context);
// 其它常用的 API：
// 绑定账号（别名）注册： registerPush(context,account) 或 registerPush(context,account, XGIOperateCallback)，其
中 account 为 APP 账号，可以为任意字符串（qq、openid 或任意第三方），业务方一定要注意终端与后台保持一致。
// 取消绑定账号（别名）：registerPush(context,"*")，即 account="*"为取消绑定，解绑后，该针对该账号的推送将失效
// 反注册（不再接收消息）：unregisterPush(context)
// 设置标签：setTag(context, tagName)
// 删除标签：deleteTag(context, tagName)