# 华为Push SDK集成说明

**（Android 版）**

发布日期： 2016年1月4日

华为开发者中心

（版权所有 翻版必究）

# 目录

# 1 华为 Push 服务简介

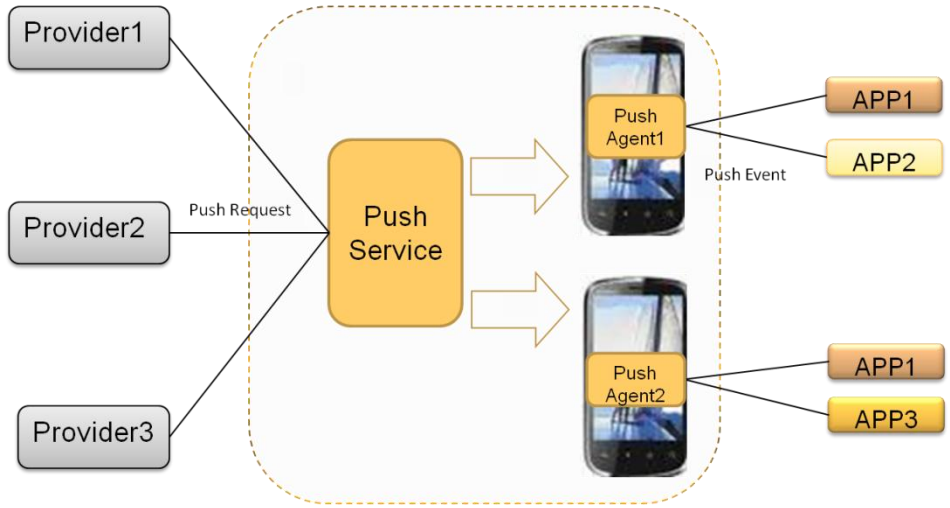
华为Push服务Android SDK是华为推出的基于Android平台的SDK，提供给Android开发者简单接口，轻松集成华为推送服务。

Push 服务以后台service方式运行。如果一款手机安装了多个集成Push SDK的应用，不会每个应用都开启一个service，而是只有一个service实例运行，与服务器建立长连接通道。这样能够减少手机系统运行的进程数，减少内存使用，降低功耗并能减少网络流量开销。

Push service 运行在一个独立进程里，主程序不需要常驻内存。当Push service接收到push消息后会以广播方式通知主进程，触发相关回调接口。

本SDK主要功能：与服务器建立稳定push通道，上报标签、LBS信息，通知推送（含富媒体）。

云基础服务如下所示，分服务器侧和终端侧，通过华为云通道进行通讯。




# 2 集成相关文件

## 2.1 集成 jar 包

将libs里面的HwPush_SDK_Vxxxx.jar拷贝到您的工程中的lib文件夹中，并在库中加入这些sdk。具体操作如下：(两种方式)

方式一：右击jar文件，选择Build Path->Add to Build Path

方式二：右击工程目录，选择Build Path->Add Library,选中user library->next,点击user libraries,单击New,输入一个名字->ok,接着单击Add Jars->选择这些jar文件在电脑上的存放目录，确定即可。

### 2.2 集成 res 文件

将res目录中的values、layout、drawable等所有文件夹拷贝到自己的工程中，注意目录名和文件名不可改变。

### 2.3 集成 Androidmanifest.xml

将AndroidManifest.xml文件中的所有activity，receiver，service，meta-data拷贝至自己的AndroidManifest.xml文件中。根据业务需要增加相关权限，可参考附录。其中com. huawei.pushtest.receiver.MyReceiver是com.huawei.android.pushagent.api.PushEventReceiver的子类，应用需要使用自己的类替换。

**注意：本SDK相对于之前开发者通过华为开发者联盟下载集成的SDK有一定的变更，包括接口、AndroidManifest.xml文件配置，集成文件等等都可能有改变，请开发者务必仔细查看该集成文档，然后修改相关接口调用和配置。**

### 2.4 混淆打包

开发者编译APK时请不要混淆本SDK，避免功能异常。在配置文件中加入：

-keep class com.huawei.android.pushagent.**{*;}

-keep class com.huawei.android. pushselfshow.**{*;}

-keep class com.huawei.android. microkernel.**{*;}

-keep class com.baidu.mapapi.**{*;}

仅供参考。

# 3 App 使用 SDK 进行开发

### 3.1 主要接口说明

目前由com.huawei.android.pushagent.api.PushManager类提供提供如下几个

方法供其他业务主动调用获取对应的信息。

### 3.1.1 Token 申请

接口描述：向Push服务请求应用的唯一标示Token，触发启动Push服务，token申请成功后，结果会通过回调传给应用。应用接入Push后，必须调用该方法来获取token。

接口原型：

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

参数说明：

| 参数名称 | 描述 | 备注 |
| --- | --- | --- |
| context | 业务上下文环境 | |

示例：

```
PushManager.requestToken(PustDemoActivity.this);
```

### 3.1.2 设置标签信息，标签以 key-value 的 Map 提供

接口描述：设置标签信息。应用可以调用该接口来设置标签，然后根据指定的标签信息来下发Push消息。

接口原型：

```
public static synchronized void setTags(Context context,
Map<String, String> tags) throws PushException
```

参数说明：

| 参数名称 | 描述 | 备注 |
| --- | --- | --- |
| context | 上下文 | |
| tags | 设置的标签键值对map | 以key-value方式集合提供 |

异常信息：接口调用失败时抛出PushException异常，业务必须捕捉做处理。

示例：

```
String key = "name";
String value = "Black";
HashMap<String, String> map = new HashMap<String, String>();
map.put(key, value);
try {
    PushManager.setTags(TagReportActivity.this, map);
} catch (PushException e) {
}
```

### 3.1.3 获取当前设备当前应用已经设置的标签信息

接口描述：获取应用已经设置的所有标签信息。

接口原型：

```
public static synchronized Map<String, String>
getTags(Context context) throws PushException
```

参数说明：

| 参数名称 | 描述 | 备注 |
| --- | --- | --- |
| context | 上下文 | |

异常信息：接口调用失败时抛出PushException异常，业务必须捕捉做处理。

示例：

```
Map<String, String> tags;
try {
    tags = PushManager.getTags(TagReportActivity.this);
    if (null != tags && !tags.isEmpty()) {
        for (Map.Entry<String, String> mapEntry : tags.entrySet()) {
            String key = mapEntry.getKey();
            String value = mapEntry.getValue();
        }
    }
} catch (PushException e) {
}
```

### 3.1.4 删除当前设备当前应用的标签信息

接口描述：删除指定的标签信息。

接口原型：

```
public static synchronized void deleteTags(Context
```

```
context, List<String> keyList) throws PushException
```

参数说明

| 参数名称 | 描述 | 备注 |
| --- | --- | --- |
| context | 上下文 | |
| keyList | 所删除标签key集 | |

异常信息：接口调用失败时抛出PushException异常，业务必须捕捉做处理。

示例：

```
List<String> list = new ArrayList<String>();
list.add("name");
try {
    PushManager.deleteTags(TagReportActivity.this, list);
} catch (PushException e) {
}
```

## 3.2 回调接口说明

回调接口由业务继承com.huawei.android.pushagent.api.PushEventReceiver，实现必要的回调方法。各回调方法都是在新开启的线程中处理。

### 3.2.1 实现 Token 返回给应用

接口描述：pushToken申请成功后，会自动回调该方法，应用可以通过该接口中获取token。本接口必须被实现。

接口原型：

```
public void onToken(Context context, String token, Bundle extras);
```

参数说明

| 参数名称 | 描述 | 备注 |
| --- | --- | --- |
| context | 上下文 | |
| token | 应用唯一标示token | |
| extras | 扩展信息 | |

示例：

```
@Override
public void onToken(Context context, String token, Bundle extras){
    String content = "获取token成功，token = " + token;
}
```

### 3.2.2 实现消息透传给应用

接口描述：供子类继承实现后，推送消息下来时会自动回调onPushMsg方法实现应用透传消息处理。本接口必须被实现。

接口原型：

```
public boolean onPushMsg(Context context, byte[] msgBytes,
Bundle extras);
```

参数说明：

| 参数名称 | 描述 | 备注 |
| --- | --- | --- |
| context | 上下文 | |
| msgBytes | 透传消息字节数组 | |
| extras | 扩展信息 | 暂时不启用 |

示例：

```
@Override
public boolean onPushMsg(Context context, byte[] msg, Bundle bundle) {
    try {
        String content = "收到一条Push消息： " + new String(msg, "UTF-8");
    } catch (Exception e) {
    }
    return false;
}
```

### 3.2.3 实现业务事件的回调

接口描述：供子类继承实现，实现业务事件。该方法会在设置标签、LBS信息之后、点击打开通知栏消息、点击通知栏上的按钮之后被调用。由业务决定是否调用该函数。

接口原型：

```
public void onEvent(Context context, Event event, Bundle extras)
```

参数说明：

| 参数名称 | 描述 | 备注 |
|---|---|---|
| context | 上下文 | |
| event | 事件类型 | event为枚举类型，事件定义如下 |
| extras | 扩展信息 | |

Event定义事件如下：

```
public static enum Event
{
  NOTIFICATION_OPENED,     //通知栏中的通知被点击打开
  NOTIFICATION_CLICK_BTN,     //通知栏中通知上的按钮被点击
  PLUGINRSP,              //标签上报回应
}
```

extras为携带的参数，应用可根据不同事件获取不同参数作处理。

| event | extras参数 | 描述 |
|---|---|---|
| NOTIFICATION_OPENED | BOUND_KEY.pushMsgKey | 附加信息，由业务自行处理。需要在华为开发者联盟上推送消息时添加自定义键值对才回调。 |
| NOTIFICATION_CLICK_BTN | BOUND_KEY.pushMsgKey | 匹配该字符串以决定点击按钮后处理何种事件 |
| | BOUND_KEY.pushNotifyId | 通知栏id，点击通知栏上的按钮后，业务可根据需要清除掉该通知栏或保留；默认为保留。 |
| PLUGINRSP | BOUND_KEY.PLUGINREPORTTYPE | 上报类型，1为LBS(当前暂不支持)，2为标签。 |
| | BOUND_KEY.PLUGINREPORTRESULT | 上报结果，成功则为true，默认为false。 |

示例：

```
public void onEvent(Context context, Event event, Bundle extras) {
    if (Event.NOTIFICATION_OPENED.equals(event) || Event.NOTIFICATION_CLICK_BTN.equals(event)) {
        int notifyId = extras.getInt(BOUND_KEY.pushNotifyId, 0);
        if (0 != notifyId) {
            NotificationManager manager = (NotificationManager) context.getSystemService(Context.NOTIFICATION_SERVICE);
            manager.cancel(notifyId);
        }
        String content = "收到通知附加消息： " + extras.getString(BOUND_KEY.pushMsgKey);
        Log.d(PustDemoActivity.TAG, content);
    } else if (Event.PLUGINRSP.equals(event)) {
        int reportType = extras.getInt(BOUND_KEY.PLUGINREPORTTYPE, -1);
        boolean isSuccess = extras.getBoolean(BOUND_KEY.PLUGINREPORTRESULT, false);
    }
    super.onEvent(context, event, extras);
}
```

# 4 注意事项

**1）建议不要在继承的PushEventReceiver中做过多的逻辑处理，如启线程、启动hander等。**

**2）push业务逻辑需要将push相关的Receiver和Service 在一个进程中，请勿修改下面部分：**

**<receiver android:name=" com.huawei.android.pushagent.PushEventReceiver "**

**android:process=":pushservice">**

**<receiver android:name="com.huawei.android.pushagent.PushBootReceiver"**

**android:process=":pushservice" >**

**<service android:name=" com.huawei.android.pushagent.PushService "**

**android:process=":pushservice">**

**3)因为APP涉及到多进程，每个进程启动时，会默认初始化Application，所以为了防止标签重复上报，建议不要在自定义的Application类里面调用标签上报等操作。**

**4)在某些手机上申请token不成功，可能是被手机系统或其他安全性软件限制，建议在手机中设置允许应用开机自启动再尝试。**

# 附录 1：

```
<!-- 必选，声明允许程序连接网络 -->
```

```
  <uses-permission android:name="android.permission.INTERNET" />
  <!-- 必选，声明允许程序程序访问网络信息 -->
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
  <!-- 必选，允许程序访问Wi-Fi网络状态信息 -->
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
  <!-- 必选，声明允许程序访问电话状态 -->
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
  <!-- 必选，声明允许程序在手机屏幕关闭后后台进程仍然运行 -->
<uses-permission android:name="android.permission.WAKE_LOCK" />
  <!-- 可选，声明允许程序写外部存储权限，如果不声明，部分自呈现消息不可用 -->
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<!-- 可选，创建桌面快捷方式权限，没有该权限不能创建富媒体的桌面快捷方式-->
<uses-permission
android:name="com.android.launcher.permission.INSTALL_SHORTCUT" />
<!—可选，根据地理位置推送消息需要事先上报地理位置信息，需要如下权限，不上报则不需要， -->
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE"/>
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
```

# 附录 2：

```
        <!-- 第三方相关 :接收Push消息（注册、Push消息、Push连接状态、标签，LBS上报
结果）广播 -->
        <receiver android:name="com.huawei.pushtest.receiver.MyReceiver" >
            <intent-filter>
                <!-- 必须,用于接收token-->
                <action
android:name="com.huawei.android.push.intent.REGISTRATION" />
                <!-- 必须，用于接收消息-->
                <action android:name="com.huawei.android.push.intent.RECEIVE"
/>
                <!-- 可选，用于点击通知栏或通知栏上的按钮后触发onEvent回调-->
                <action android:name="com.huawei.android.push.intent.CLICK" />
                <!-- 可选，查看push通道是否连接，不查看则不需要-->
```

```
                <action android:name="com.huawei.intent.action.PUSH_STATE" />
                <!-- 可选，标签、地理位置上报回应，不上报则不需要 -->
                <action
android:name="com.huawei.android.push.plugin.RESPONSE" />
            </intent-filter>
            <meta-data android:name="CS_cloud_ablitity"
android:value="@string/hwpush_ability_value"/>
        </receiver>

        <!-- 备注：Push相关的android组件需要添加到业务的AndroidManifest.xml，
         Push相关android组件运行在另外一个进程是为了防止Push服务异常而影响主业务
-->

        <!-- PushSDK:PushSDK接收外部请求事件入口 -->
        <receiver
            android:name="com.huawei.android.pushagent.PushEventReceiver"
            android:process=":pushservice" >
            <intent-filter>
                <action
android:name="com.huawei.android.push.intent.REFRESH_PUSH_CHANNEL" />
                <action android:name="com.huawei.intent.action.PUSH" />
                <action android:name="com.huawei.intent.action.PUSH_ON" />
                <action android:name="com.huawei.android.push.PLUGIN" />
            </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>
        <receiver
            android:name="com.huawei.android.pushagent.PushBootReceiver"
            android:process=":pushservice" >
            <intent-filter>
                <action
android:name="com.huawei.android.push.intent.REGISTER" />
                <action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
            </intent-filter>
            <meta-data
                android:name="CS_cloud_version"
                android:value="\u0032\u0037\u0030\u0034" />
```

```
        </receiver>

        <!-- PushSDK:Push服务 -->
        <service
            android:name="com.huawei.android.pushagent.PushService"
            android:process=":pushservice" >
        </service>

        <!-- PushSDK:富媒体呈现页面，用于呈现服务器下发的富媒体消息 -->
        <!-- locale|layoutDirection 切换语言后不重新创建activity -->
        <activity

android:name="com.huawei.android.pushselfshow.richpush.RichPushActivity"
            android:process=":pushservice"
            android:theme="@style/hwpush_NoActionBar"

android:configChanges="orientation|screenSize|locale|layoutDirection"
            android:screenOrientation="portrait">
            <meta-data android:name="hwc-theme"
                       android:value="androidhwext:style/Theme.Emui"/>
            <intent-filter>
                <action
android:name="com.huawei.android.push.intent.RICHPUSH" />

                <category android:name="android.intent.category.DEFAULT" />
            </intent-filter>
        </activity>

        <activity

android:name="com.huawei.android.pushselfshow.permission.RequestPermissions
Activity"

android:theme="@android:style/Theme.DeviceDefault.Light.Dialog.NoActionBar"
            android:launchMode="singleTop"
            android:screenOrientation="portrait"

android:configChanges="orientation|screenSize|locale|layoutDirection"
            android:exported="false">
        </activity>
```