此文章将会根据不同厂商的SDK的接入方式逐步分析，进而梳理出最为精简的接入方式，同时说明各个接入配置的细节问题

一 接入准备工作

关于组件的基本配置将会全部打包到aar中的AndroidManifest中，用户只需要手动配置一些与包名相关的权限配置

1.1 小米

混淆配置

基于aar整体打包的方式

AppID和APPKey本地配置

```
   <meta-data
      android:name="XIAOMI_APP_ID"
      android:value="${XIAOMI_APP_ID}"/>

   <meta-data
      android:name="XIAOMI_APP_KEY"
      android:value="${XIAOMI_APP_KEY}"/>
```

Android Manifest 配置

```
    <!-- 小米个性化配置 需要到应用的主manifest中去配置-->
    <!-- the following 2 ${your packageName} should be changed to your package
name -->
    <permission
    android:name="${your packageName}.permission.MIPUSH_RECEIVE"
    android:protectionLevel="signature" />
    <uses-permission android:name="${your packageName}.permission.MIPUSH_RECEIVE"
/>
```

UpsPushReceiver 配置

1.2 魅族

AppID和APPKey本地配置

```
   <meta-data
        android:name="MEIZU_APP_ID"
        android:value="${MEIZU_APP_ID}"/>
```

```
  <meta-data
      android:name="MEIZU_APP_KEY"
      android:value="${MEIZU_APP_KEY}"/>
```

Android Manifest 配置

```
    <!-- 魅族个性化配置 需要到应用的主manifest中去配置-->
    <uses-permission android:name="com.meizu.flyme.push.permission.RECEIVE">
</uses-permission>
    <permission android:name="${PACKAGE_NAME}.push.permission.MESSAGE"
android:protectionLevel="signature"/>
    <uses-permission android:name="${PACKAGE_NAME}.push.permission.MESSAGE">
</uses-permission>

    <uses-permission android:name="com.meizu.c2dm.permission.RECEIVE" />
    <permission android:name="${PACKAGE_NAME}.permission.C2D_MESSAGE"
                android:protectionLevel="signature"></permission>
    <uses-permission android:name="${PACKAGE_NAME}.permission.C2D_MESSAGE"/>
```

UpsPushReceiver 配置

1.3 华为

AppID和APPKey本地配置

```
<!-- APPID 华为移动服务配置 请将value替换成实际的appid -->
   <meta-data
        android:name="com.huawei.hms.client.appid"
        android:value="此处为华为开发者联盟为开发者应用分配的appid" />
```

Android Manifest 配置

```
    <!-- 华为个性化配置 需要到应用的主manifest中去配置-->
    <!-- the following 2 ${your packageName} should be changed to your package
name -->
    <permission
    android:name="${your packageName}.permission.MIPUSH_RECEIVE"
    android:protectionLevel="signature" />
    <uses-permission android:name="${your packageName}.permission.MIPUSH_RECEIVE"
/>


    <!-- 华为移动服务配置,将xxx替换为实际包名 -->
    <provider
       android:name="com.huawei.hms.update.provider.UpdateProvider"
```

```
        android:authorities="com.meizu.upspushdemo.hms.update.provider"
        android:exported="false"
        android:grantUriPermissions="true" >
    </provider>
```

UpsPushReceiver 配置

二 统一推送配置

AndroidManifest 配置

```
        <!-- 统一推送配置 -->
        <receiver android:name=".UpsReceiver">
            <intent-filter>
            <!-- 接收push消息 -->
            <action android:name="com.meizu.ups.push.intent.MESSAGE" />
            </intent-filter>
        </receiver>
```

三 通知栏消息点击行为分析

现在各个厂商目前支持以下四种类型，具体使用方式详见各个平台的服务端参考文档

- 魅族,提供JavaSDK
- 小米,提供JavaSDK
- 华为,只提供api接口参数需要自己组装

NOTE: 目前我们已经根据各个平台提供的策略，重新设计魅族统一推送平台的通知栏消息点击策略，目前有以下五种，对应各个平台的策略如下表：

通知栏行为	**魅族**	**小米**	华为	**OPPO**	**说明**
打开应用	支持	支持	支持	支持	无
打开应用页面	支持	支持	支持	支持	魅族只需传递Activity名称,小米需要传递转换后的intentUri
打开Web页面	支持	支持	支持	支持	只需填写web url地址即可
打开自定义URI	支持	支持	支持	不支持	魅族可通过打开URL功能实现,小米可通过打开应用内页面传递转换的intentURI,华为通过自定义点击行为
应用客户端自定义	支持	支持	不支持	不支持	魅族支持客户端自定义功能，小米可通过不指定notifyEffect实现

统一传参方式

传参的方式目前有两种方式

- 当执行打开应用，打开应用内页面时，将参数通过intent的方式传递给Activity，开发者通过 `getIntent().getStringExtra("key")`方式获取 因此只需要在组建intent uri时将参数值以key-value方式拼装即可，如下：

```
intent:#Intent;component=com.meizu.upspushdemo/.TestActivity;S.key=value;end
```

- 当点击通知栏时，将参数回调给开发者，开发者在只需在 `onNotificationArrived`,`onNotificationClicked`时接收参数，在应用客户端自定义时，这种方法较为适用 开发者。在接收到该自定义参数时，自行决定后续动作

3.1 打开应用

- 魅族
  - 点击动作：打开应用
- 小米
  - 点击动作：打开应用
- 华为
  - 点击通知：打开应用
  - 后续动作：直接打开应用
- OPPO
  - 点击动作：打开应用

3.2 打开应用内页面

- 魅族
  - 点击动作：打开应用页面
  - 页面名称： 打开页面只需要填写应用页面的全路径名称，例如 `com.meizu.upspushdemo.TestActivty`
- 小米
  - 点击动作：打开应用内指定页面
  - 页面地址：intent Uri

通过如下代码打开页面需要获取Intent uri，具体获取方法如下：

```
Intent intent = new Intent(this,TestActivity.class);
intent.putExtra("key","value");
UpsLogger.i(this,"intent uri "+intent.toUri(Intent.URI_INTENT_SCHEME));
```

得到Intent uri字符传如下:

```
intent:#Intent;component=com.meizu.upspushdemo/.TestActivity;S.key=value;end
```

- 华为
    - 根据activity名称与参数组装intent,原理同上
- OPPO
    - 点击动作：打开应用内页
    - 页面地址：Activity对应的Intent Action/或者是Activity名称

3.3 打开web页面

- 魅族
    - 点击动作：打开URI页面
    - Url: 标准URI格式如下：`https://www.baidu.com`
- 小米
    - 点击动作：打开网页
    - Url: 标准URI格式如下：`https://www.baidu.com`
- 华为
    - 点击通知：打开网页
    - 输入网址: 标准URI格式如下：`https://www.baidu.com`
- OPPO
    - 点击动作：打开网页地址
    - 输入网址: 标准URI格式如下：`https://www.baidu.com`

3.4 应用客户端自定义

目前仅仅魅族与小米支持

- 魅族
    - 点击动作：应用客户端自定义
    - 自定义内容：完全由用户填写
- 小米
    - 点击动作：由应用客户端自定义
    - 传输数据：由用户自己填写

NOTE: 由于小米不提供远程仓库支持，ups_pushsdk不会将`MiPush_SDK_Client_3_4_5.jar`包含进最终的aar包中，此时需要开发者自己手动将此jar包引入到自己的工程中

- 小米服务端客户端自定义API使用方式

在推送时只要不指定notify_effect，即是代表自定义动作 服务端推送代码如下

```
    /**
     * 创建自定义消息内容的格式
```

```
     * **/
    private static Message buildCustomMessage() throws Exception {
        //自定义消息体
        String messagePayload = "This is a message";
        String title = "notification title";
        String description = "notification description";
        Message message = new Message.Builder()
                .title(title)
                .description(description)
                .payload(messagePayload)
                .restrictedPackageName(MY_PACKAGE_NAME)
                .passThrough(0)  //消息使用通知栏方式
                .notifyType(1)
                .build();
        return message;
    }
```

3.5 打开自定义URI

- 魅族
  - 点击动作：打开URI页面
  - Url: 标准URI格式如下：https://www.baidu.com
- 小米
  - 点击动作：打开应用内指定页面
  - 页面地址: 标准URI经过Intent.toURI得出
- 华为
  - 点击通知：打开应用
  - 后续行为->自定义动作: 标准URI经过Intent.toURI得出

3.5.1 URI转换方式

- URI 原始格式如下

```
upspushscheme://com.meizu.upspush/notify_detail?title=ups title&content=ups
content
```

- Uri 经过如下代码

```
Intent intent = new
Intent(Intent.ACTION_VIEW,Uri.parse("upspushscheme://com.meizu.upspush/notify_deta
il?title=ups title&content=ups content"));
intent.addFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
UpsLogger.e(this,intent.toUri(Intent.URI_INTENT_SCHEME));
```

转换后格式如下

```
intent://com.meizu.upspush/notify_detail?title=ups title&content=ups
content#Intent;scheme=upspushscheme;launchFlags=0x10000000;end
```

- Android Manifest TestActivity 必须如下配置如下

```
<activity android:name=".TestActivity">
 <intent-filter>
      <action android:name="android.intent.action.VIEW"/>
      <category android:name="android.intent.category.DEFAULT"/>
      <data android:scheme="upspushscheme" //scheme类型
            android:host="com.meizu.upspush" //host全称
            android:path="/notify_detail"/>  //path路径
 </intent-filter>
</activity>
```

极光推送

- 极光推送接入文档

极光推送由于只支持通知栏自定义点击动作，因此需要根据此方案，适配现有点击动作，目前仅仅适配打开URI 只需要在推送通知栏消息时传递如下extra字段名称即可 即在构建payload时，附加如下字段CLICK_TYPE-4,CLICK_TYPE_WEB_URI=""

```
 public static PushPayload buildPushObject_ios_audienceMore_messageWithExtras() {
        return PushPayload.newBuilder()
                .setPlatform(Platform.android_ios())
                .setAudience(Audience.newBuilder()
                        .addAudienceTarget(AudienceTarget.tag("tag1", "tag2"))
                        .addAudienceTarget(AudienceTarget.alias("alias1",
"alias2"))
                        .build())
                .setMessage(Message.newBuilder()
                        .setMsgContent(MSG_CONTENT)
                        .addExtra("CLICK_TYPE", 4) //点击类型，沿用通知栏点击类型定义
                        .addExtra("CLICK_TYPE_WEB_URI",
"upspushscheme://com.meizu.upspush/notify_detail?title=ups title&content=ups
content")// 打开uri时填写uri
                        .build())
                .build();
    }
```