

魅族开放平台推送 SDK 客户端接入文档

1. 概述

本文档旨在指导开发者如何将魅族开放平台 PushSDK 集成到其 Android 应用中，以实现消息推送功能。为了简化接入流程，PushSDK 已预先处理通用的权限配置和混淆配置，使得接入过程更为简便，开发者只需按以下步骤操作即可。

2. 环境准备

在开始集成之前，请确保以下准备工作已完成：

- **注册开发者账号**：前往 [魅族Flyme推送平台](#) 注册开发者账号并创建应用，流程可见《[魅族推送接入指南](#)》。
- **获取应用信息**：在推送平台上创建应用并获取其 **AppId** 和 **AppKey**，这些信息将在集成过程中使用。

提示：AppId 和 AppKey 用于应用身份识别及推送网络请求的凭证，为确保安全性，请务必妥善保管 AppId 和 AppKey，避免直接在 AndroidManifest.xml 中明文配置。

3. 集成步骤

3.1 添加依赖

从【魅族开放平台】-【文档】-【[魅族推送 SDK 下载](#)】下载【魅族推送 Android 客户端 SDK】，并将解压后的 aar 文件放至你的项目 `libs` 文件夹中，接着在你的 `app/build.gradle` 文件中，在 `dependencies` 块中加入以下代码引入 PushSDK：

代码块

```
1 dependencies {  
2     .....  
3     implementation fileTree(include: ['MzPushSdk.aar'], dir: 'libs')  
4 }
```

3.2 声明消息接收的广播

在你的 Android 项目的 **AndroidManifest.xml** 文件中声明消息接收广播：

代码块

```

1 <receiver android:name="【替换你的类完整路径】.MyPushMsgReceiver"
2 android:exported="true"
3 android:permission="com.meizu.cloud.push.permission.MESSAGE">
4     <intent-filter>
5         <action android:name="com.meizu.flyme.push.intent.MESSAGE" />
6         <action android:name="com.meizu.flyme.push.intent.REGISTER.FEEDBACK" />
7         <action
8             android:name="com.meizu.flyme.push.intent.UNREGISTER.FEEDBACK"/>
9         <category android:name="【替换你的包名】" />
10    </intent-filter>
11 </receiver>

```

注意：请将以上的【替换你的类完整路径】和【替换你的包名】替换为你自己工程相对应的值。

3.3 实现消息接收的广播

代码块

```

1 public class MyPushMsgReceiver extends MzPushMessageReceiver {
2
3     /**
4      * 调用订阅方法后，会在此方法回调结果
5      * 订阅方法：PushManager.register(context, appId, appKey)
6      *
7      * @param context
8      * @param registerStatus
9      */
10    @Override
11    public void onRegisterStatus(Context context, RegisterStatus
12        registerStatus) {
13    }
14
15    /**
16     * 调用取消订阅方法后，会在此方法回调结果
17     * 取消订阅方法：PushManager.unregister(context, appId, appKey)
18     *
19     * @param context
20     * @param unregisterStatus
21     */
22    @Override
23    public void onUnRegisterStatus(Context context, UnRegisterStatus
24        unregisterStatus) {
25    }
26

```

```

27     /**
28      * 调用开关转换或检查开关状态方法后，会在此方法回调开关状态
29      * 通知栏开关转换方法：PushManager.switchPush(context, appId, appKey, pushId,
pushType, switcher)
30      * 检查开关状态方法：PushManager.checkPush(context, appId, appKey, pushId)
31      *
32      * @param context
33      * @param pushSwitchStatus
34      */
35     @Override
36     public void onPushStatus(Context context, PushSwitchStatus
pushSwitchStatus) {
37
38     }
39
40     /**
41      * 调用标签订阅、取消标签订阅、取消所有标签订阅和获取标签列表方法后，会在此方法回调标
签相关信息
42      * 标签订阅方法：PushManager.subScribeTags(context, appId, appKey, pushId,
tags)
43      * 取消标签订阅方法：PushManager.unSubScribeTags(context, appId, appKey,
pushId, tags)
44      * 取消所有标签订阅方法：PushManager.unSubScribeAllTags(context, appId,
appKey, pushId)
45      * 获取标签列表方法：PushManager.checkSubScribeTags(context, appId, appKey,
pushId)
46      *
47      * @param context
48      * @param subTagsStatus
49      */
50     @Override
51     public void onSubTagsStatus(Context context, SubTagsStatus subTagsStatus) {
52
53     }
54
55     /**
56      * 调用别名订阅、取消别名订阅和获取别名方法后，会在此方法回调别名相关信息
57      * 别名订阅方法：PushManager.subScribeAlias(context, appId, appKey, pushId,
alias)
58      * 取消别名订阅方法：PushManager.unSubScribeAlias(context, appId, appKey,
pushId, alias)
59      * 获取别名方法：PushManager.checkSubScribeAlias(context, appId, appKey,
pushId)
60      *
61      * @param context
62      * @param subAliasStatus
63      */

```

```

64     @Override
65     public void onSubAliasStatus(Context context, SubAliasStatus
subAliasStatus) {
66
67     }
68
69     /**
70      * 当用户点击通知栏消息后会在此方法回调
71      *
72      * @param context
73      * @param mzPushMessage
74      */
75     @Override
76     public void onNotificationClicked(Context context, MzPushMessage
mzPushMessage) {
77
78     }
79
80     /**
81      * 当推送的通知栏消息展示后且应用进程存在时会在此方法回调
82      *
83      * @param context
84      * @param mzPushMessage
85      */
86     @Override
87     public void onNotificationArrived(Context context, MzPushMessage
mzPushMessage) {
88
89     }
90 }

```

注意：如果你在自定义广播类中声明了构造方法，请务必增加 `super();`。

3.4 执行消息订阅

接下来需要在代码的某个适当位置（例如在 *Application* 的 *onCreate* 方法中）触发消息订阅，调用：`PushManager.register(this, AppId, AppKey);` 确保你的消息接收广播 `Receiver` 类中能够在 `onRegisterStatus(RegisterStatus registerStatus)` 方法中正确接收回调，这样表示接入成功。

3.5 动态申请权限（适用于 Android T 及以上）

如果你的应用运行在 Android T 及以上版本，需要在适当的位置动态申请 `android.Manifest.permission.POST_NOTIFICATIONS` 通知权限，以确保能正常弹出通知。

3.6 测试与问题排查

至此，PushSDK 的集成工作完成。接下来，登录到 [魅族Flyme推送平台](#)，找到你创建的应用，在【配置管理】-【问题排查】中进行消息推送测试。

4. 功能说明

4.1 PushManager 方法说明

方法名称	方法说明	使用建议	对应回调
void register(context, appld, appKey)	消息订阅	建议在应用启动后尽早调用，只有进行过订阅后才能接收到推送消息。	onRegisterStatus(context,registerStatus)
void unRegister(context, appld, appKey)	取消消息订阅	用于停用所有推送功能，请慎用。 - 若应用进行撤回隐私政策操作，建议同时调用此方法进行取消消息订阅； - 若应用只是临时关闭推送消息，建议使用下面的 switchPush 方法即可。	onUnRegisterStatus(context,unRegisterStatus)
void subScribeTags(context, appld, appKey, pushId, tags)	标签订阅	无	onSubTagsStatus(context,subTagsStatus)
void unSubScribeTags(context, appld, appKey, pushId, tags)	取消标签订阅	无	onSubTagsStatus(context,subTagsStatus)
void unSubScribeAllTags(context, appld, appKey, pushId)	取消所有标签订阅	无	onSubTagsStatus(context,subTagsStatus)
void checkSubScribeTags(context)	获取标签列表	无	onSubTagsStatus(context,subTagsStatus)

ext, appld, appKey, pushId)			
void subscribeAlias(context, appld, appKey, pushId, alias)	别名订阅	无	onSubAliasStatus(context, subAliasStatus)
void unsubscribeAlias(context, appld, appKey, pushId, alias)	取消别名订阅	无	onSubAliasStatus(context, subAliasStatus)
void checkSubscribeAlias(context, appld, appKey, pushId)	获取别名	无	onSubAliasStatus(context, subAliasStatus)
void switchPush(context, appld, appKey, pushId, switcher)	通知消息和透传消息开关同时转换	透传功能已停用，请忽略	onPushStatus(context, pushSwitchStatus)
void switchPush(context, appld, appKey, pushId, pushType, switcher)	通知消息或透传消息开关单独转换	透传功能已停用，请忽略	onPushStatus(context, pushSwitchStatus)
void checkPush(context, appld, appKey, pushId)	检查通知消息和透传消息开关状态	透传功能已停用，请忽略。此方法在有无网络下都能成功返回	onPushStatus(context, pushSwitchStatus)
void clearNotification(context)	清除应用弹出的所有通知栏消息	无	无
void clearNotification(context, notifyId)	清除应用弹出的指定notifyId的通知栏消息	无	无
String getPushId(context)	获取已订阅后的PushId 缓存，若应用清除过本地数据会返回空	无	无
String getSdkVersion()	获取 PushSDK 版本	无	无
boolean isBrandMeizu()	是否魅族设备	无	无

参数说明：

参数名称	说明
appId	魅族Flyme推送平台 申请的应用id。
appKey	魅族Flyme推送平台 申请的应用key。
pushId	订阅凭证ID，可在订阅接口的回调方法 <code>onRegisterStatus</code> 中，通过： <code>registerStatus.getPushId()</code> ；获取。
tags	标签名称，多个逗号隔离，每个标签不能超过 20 个字符，限100个。
alias	别名名称，长度不能超过 20 个字符，每一个应用用户仅能设置一个别名。
pushType	消息类型，0：通知栏消息 1：透传消息。
switcher	开关状态
notifyId	由服务端生成的通知Id，在通知展示的回调方法 <code>onNotificationArrived</code> 中，通过： <code>mzPushMessage.getNotifyId()</code> ；获取。

4.2 MzPushMessageReceiver 回调方法说明

方法名称	方法说明	使用建议
<code>onRegisterStatus(context, registerStatus)</code>	消息订阅回调	无
<code>onUnRegisterStatus(context, unRegisterStatus)</code>	取消订阅回调	无
<code>onPushStatus(context, pushSwitchStatus)</code>	通知消息和透传消息开关状态回调	无
<code>onSubTagsStatus(context, subTagsStatus)</code>	标签状态回调	无
<code>onSubAliasStatus(context, subAliasStatus)</code>	别名状态回调	无
<code>onNotificationClicked(context, mzPushMessage)</code>	通知点击回调	无
<code>onNotificationArrived(context, mzPushMessage)</code>	通知展示回调	只有在应用进程存在时才会在此方法回调
<code>onMessage(context, message, platformExtra)</code>	透传消息回调	透传功能已停用，请忽略

5. 通知点击选择

通知栏中的消息支持四种点击动作，分别是：打开应用主页、打开应用内页面、打开URI页面 以及 应用客户端自定义，如图。

Flyme 推送平台

首页 创建推送

推送通知

① 任务备注

请选择

通知栏样式

☒ 安卓原生

☐ 图片

标题

请输入标题

内容

请输入内容

展开方式

☒ 标准

☐ 文本

☐ 大图

☐ ACT文件

① 点击动作

请输入调起的应用包名

☒ 打开应用

☐ 打开应用页面

☐ 打开URI页面

☐ 应用客户端自定义 ①

输入参数

输入值

—

添加参数

存为参数模板

使用参数模板

18:30

中国移动

9月2日 周二

WiFi

数据

蓝牙

便携热点

亮度

此处显示标题文字

此处为内容说明文字显示一行文字说明

×

实时预览

当用户点击了通知栏消息后，四种方式都会从 `MzPushMessageReceiver` 的 `onNotificationClicked` 收到点击的回调，其中前三种打开方式还会调起相应的Activity。

5.1 打开应用主页 和 打开应用内页面

第二种点击动作：打开应用内页面，需要另外配置完整的 Activity 名称，比如：`com.meizu.pushdemo.TestActivity`。

参数传递

打开应用主页 和 打开应用内页面 都支持附加参数。在消息创建时，会解析了平台传递的消息点击类型和参数列表，通过 `intent.putString("key","value")` 的方式在构建 Intent 时进行添加。

参数获取

当跳转到目标 Activity 时，可以通过如下方式获取在平台填写的参数值：`String value = getIntent().getStringExtra("key")`

点击通知栏消息的时候，除了获取用户自定义的参数，还可以获取平台 taskid 等参数，此参数为 SDK 传递的默认参数，不需在平台配置，如果需要 taskId 相关参数可以通过如下方式获取：`String platfromExtra = getIntent().getStringExtra("platform_extra");`，该参数的格式如：`{"task_id":"123456"}`

5.2 打开URI

打开URI可支持配置 https/http 的网页地址，也可支持您应用内部自定义的URI。使用时要特别注意您应用是否全版本支持该 URI，以免造成在旧版本中点击通知栏无效的情况。

5.3 应用客户端自定义（从 Android S 起不支持）

该方式不会触发打开 Activity 的逻辑，而只是会在 `MzPushMessageReceiver` 的 `onNotificationClicked` 回调方法的 `MzPushMessage` 参数中附带平台上配置的自定义内容，代码如下：

代码块

```
1  @Override
2  public void onNotificationClicked(Context context, MzPushMessage mzPushMessage)
3  {
4      String selfDefineContentString =
        mzPushMessage.getSelfDefineContentString();
5  }
```

注意：从 Android S（通知 trampoline 限制）起，该类通知点击类型不再支持。

6. 常见问题

问题1：为什么订阅回调提示AppID不合法？

1. 请先确定你接入的SDK是魅族官方推送SDK，以及保证跟 [魅族Flyme推送平台](#) 上【配置管理】-【应用配置】页面中，“应用形态”是普通应用 以及 AppID、AppKey 要和代码中一致（注意前后空格）。
2. 是否存在过在同一个工程中直接修改过 AppID 和 AppKey 进行订阅操作情况？因为应用包名、AppID 和 AppKey 三者是唯一绑定关系，如果出现过错误的订阅会导致系统中记录了错误的缓存。解决办法可以在手机【系统设置】-【应用管理】-【所有应用】点击右上角【显示系统服务应用】找到【推送服务】，如下图，对其进行进行“清除数据”，然后重启手机，待手机启动后再次执行一次正确的订阅操作。



问题2：为什么执行了订阅后一直没有收到广播回调？

1. 请检查您手机网络是否设置了代理、是否稳定畅通，尝试切换网络后重试。
2. MzPushMessageReceiver 广播中的回调方法里 onRegister 方法已经废除，正常情况下会在 onRegisterStatus 方法中回调，请检查是否使用错误。
3. 请检查接入PushSDK 的过程是否存在错误（可参考文档中“快速接入”），特别要保证 AndroidManifest.xml 中广播的注册要将【替换您的类完整路径】和【替换您的包名】替换正确 和广播必须继承 MzPushMessageReceiver。
4. 不要在你的App中去实现多个 MzPushMessageReceiver，因为只会回调其中一个。
5. 如果你在自定义广播类中声明了构造方法，请务必增加 super();。
6. 尝试重启手机，待手机启动后再次执行一次订阅操作。

问题3：为什么一直无法收到消息，该如何定位？

1. 先检查消息是否被放进了通知栏右上角收纳盒子里，如下图红圈位置。一般地当App多次消息到达到都没有对其进行点击，消息就会自动收进收纳盒里。若要恢复，可在收纳盒里长按消息选择“不再收纳”。



2. 请检查应用的“通知消息”权限是否关闭了，步骤如：【设置】-【通知管理】，在应用列表中找到你自己的App，勾上“允许通知”选项。
3. 请检查手机日期和时间是否正确，错误的时间会影响消息的展示逻辑。
4. 在 [魅族Flyme推送平台](#) 【配置管理】-【问题排查】中按从上往下步骤进行相应的状态查询，如下图所示。

Flyme 推送平台

首页创建推送数据统计配置管理

应用配置

问题排查

设备对应关系查询

IMEI查询PushId8667[redacted]查询

8667[redacted]

设备是否在线查询

8667[redacted]查询

在线状态: 在线

设备是否订阅查询

8667[redacted]AppId:[redacted]是否: [redacted]是否订阅: 已订阅通知栏开关: 开透传开关: 开系统通知栏开关: 开查询

推送测试

8667[redacted]推送

已推送msgId : NS20190715[redacted]

- a. 【设备对应关系查询】输入手机的设备ID点击“查询”按钮获得相应的PushId，若提示“PushId未注册”，请执行PushSDK中 `PushManager.register(context, appId, appKey)` 进行推送订阅；
 - b. 【设备是否在线查询】输入刚才获得的PushId，点击“查询”按钮查询状态，若手机处理离线状态，解决方法请见“问题4”；
 - c. 【设备是否订阅查询】输入刚才获得的PushId，点击“查询”按钮查询订阅信息：“是否已经订阅”，若未订阅请执行PushSDK中 `PushManager.register(context, appId, appKey)` 方法进行推送订阅；
“通知栏开关”，若为关，请执行PushSDK中 `PushManager.switchPush(context, appId, appKey, pushId, pushType, switcher)` 方法进行打开；
“系统通知栏开关”，若为关，解决方法请见“问题5”；
 - d. 【推送测试】中输入刚才获得的PushId，并点击“推送”按钮，下方显示“已推送 msgId: xxx”代表已经成功发送测试通知。
5. 弹出消息时是否存在 **Invalid notification (no valid small icon)** 异常日志输出？这是设置通知栏图标异常，如果开启了像AndResGuard之类的资源路径混淆，尝试在whiteList中添加：
`R.drawable.stat_sys_third_app_notify`。
 6. 在Flyme5或较老系统也出现 **Invalid notification (no valid small icon)** 异常的话，还可以在drawable不同分辨率文件夹下放置一张名为 `mz_push_notification_small_icon` 的图片，并在 `MzPushMessageReceiver` 的 `onUpdateNotificationBuilder` 方法中按文档说明进行设置通知栏小图标。

问题4：为什么手机一直连着网络，但还是显示处于离线状态？

1. 手机离线状态并不是指没连网络，而是手机上推送服务(系统进程)跟推送服务器无法建立长连接，常见于网络不稳或者开发过程中非正式网络环境下，可按以下步骤进行修复。
 - a. 请检查您手机网络是否设置了代理、是否稳定畅通，尝试断开网络再进行重连或者移动网络和Wi-Fi网络互相切换一下，再重试。
 - b. 再次执行一次订阅操作，再重试。
 - c. 重启手机，再重试。
 - d. 查看您手机Flyme版本，对于Flyme5或以下较老的系统若进行以上操作后还是处于离线状态，那么等待几分钟后再重试，同时建议对手机系统进行升级。

问题5：问题排查中，“系统通知栏开关”是关闭状态，该如何打开？

1. 打开你手机中【设置】-【通知管理】，找到你的App，把“允许通知”勾上

2. 然后在手机【设置】-【应用管理】-【所有应用】点击右上角【显示系统服务应用】找到【推送服务】，如下图，对其进行进行“清除数据”，然后重启手机，待手机启动后再次执行一次订阅操作，这样便会触发系统通知栏开关状态的上传，完成操作后再到问题排查中查看状态是否发生变化。



问题6：为什么点击消息后，不能打开应用页面？

1. 先查看输出日志中是否存在：`Click message StartActivity error` 或者 `android.content.ActivityNotFoundException` 异常。消息推送是支持打开内部非对外的Activity，只要在平台上配置好完整的名称即可，同时请确保名称拼写正确以及名称前后不能含有空格等特殊字符。
2. 再查看输出日志中是否存在：`invalid push message` 错误信息。该情况一般是在点击消息时当前网络环境异常或者App当前是debug版本，请更换到release版本试试，而且要保证点击时手机网络是正常。

问题7：Debug版本正常但在Release版本中报异常是什么原因？

1. 常见于数据反序列化过程中异常，请检查Release中是否对APK进行了加固或者字符串的混淆之类的操作，如果是请对 `com.meizu.cloud` 进行过滤。
2. 如果您正在使用 Android Studio 3.4 或以上版本出现该问题，那可能是因为 Android Studio 默认开启了R8混淆导致的。解决方法：
 - a. 请接入PushSDK4.1.0或以上版本，并检查您的系统应用“推送服务”是否7.1.4或以上版本（查看方法见问题5的图），如不是请进行升级 [点击下载](#)。
 - b. 若a方法仍不能解决，可以尝试在 `gradle.properties` 中添加 `android.enableR8 = false` 进行关闭R8。

问题8：魅族PushSDK有离线包吗？

前往【魅族开放平台】 - 【文档】 - 【[魅族推送 SDK 下载](#)】页面下载 aar 包进行集成。

问题9：PushID会在什么场景下发生变化？

卸载了App或者App不活跃一个月。

问题10：收到的推送消息可以对其进行删除吗？

不支持使用常规通知栏方法`NotificationManager.cancel()`删除，但可以使用
`PushManager.clearNotification(context, notifyId)` 进行删除。

问题11：通知到达后，会收到回调吗？

不一定，当通知到达后且App进程存在的情况下才会收到 `MzPushMessageReceiver` 的 `onNotificationArrived` 回调。

问题12：接入PushSDK后，APK会增大多少？

APK会比原来增加几十K。

问题13：使用透传功能提示超出了限制是什么原因？

已不再支持透传功能。

问题14：推送支持角标设置吗？

不支持，但可以前往【魅族开放平台】 - 【文档】 - 【ROM适配】 - 【桌面图标角标】进行相应的处理。

问题15：推送消息支持自定义提示音吗？

不支持。

问题16：为什么会出现熄屏状态下无法收到通知栏消息？

该异常会在较老的Flyme系统中发生，建议对手机系统进行升级。

7. SDK 更新日志

[2025-07-14] V5.0.5

- 适配 Flyme 12
- 优化日志记录与异常处理逻辑
- 优化订阅逻辑

- 提升回调结果的安全性校验机制
- 修复若干 BUG

[2025-01-09] V5.0.3

- 修复 BUG

[2024-07-24] V5.0.2

- 适配 Flyme 11
- 增加支持实况通知
- 优化代码和减小包体积
- 修复若干 BUG

[2023-11-17] V4.3.0

- 适配统一推送联盟接口

[2023-09-22] V4.2.7

- 解决混淆 a.a.a 冲突的问题
- 修复 BUG

[2023-09-08] V4.2.6

- 适配 Android U
- 修复一些安全漏洞和若干 BUG
- 移除用于兼容 Flyme5 老固件时读取 IMEI 的逻辑
- 移除读取运营商信息

[2023-02-14] V4.2.3

- 适配 Android T
- 适配新设备
- 移除 READ_EXTERNAL_STORAGE 和 WRITE_EXTERNAL_STORAGE 权限的声明
- 移除读取正在运行应用进程
- 修复若干 BUG

[2021-09-02] V4.1.4

- 适配新设备

- 修复可删除相对路径文件的安全漏洞

[2021-06-18] V4.1.0

- 解决 R8 兼容问题
- 修复某情况下通知栏消息不振动的BUG
- 移除不必要的异常打印

[2021-01-04] V4.0.4

- 适配新设备
- 移除读取 BssId、运营商信息、Wi-Fi列表、Mac地址、地位信息

[2020-11-13] V4.0.2

- 全面适配 AndroidR
- 修复一些安全漏洞和若干 BUG
- 优化一些性能问题

[2020-06-16] V3.9.7

- 优化推送逻辑
- 修复若干 BUG

[2020-03-09] V3.9.0

- 优化点击通知消息逻辑
- 修复若干 BUG
- 全面适配 AndroidQ
- 移除读取应用列表

[2019-11-27] V3.8.7

- 移除 READ_PHONE_STATE 权限声明
- 移除读取 IMEI、IMSI

[2019-11-18] V3.8.6

- 修复若干BUG
- 移除读取手机号

[2019-08-21] V3.8.4

- 移除 SUBSTITUTE_NOTIFICATION_APP_NAME 权限声明
- 移除旧版订阅和取消订阅回调方法的声明

[2019-07-01] V3.8.3

- 修复一些安全漏洞和若干BUG
- 优化内部推送逻辑

[2018-08-30] V3.8.1

- 修复一些安全漏洞

[2018-08-01] V3.7.4

- 优化非魅族手机订阅逻辑

[2018-07-12] V3.7.3

- 优化一些小细节

[2018-06-28] V3.7.1

- 修复zip文件目录遍历的漏洞

[2018-06-03] V3.7.0

- 修复部分第三方机型可能出现的兼容性报错问题

[2018-05-03] V3.6.7

- 修复三方机型的兼容性问题

[2018-02-08] V3.6.3

- 修改可能出现的空指针问题

[2017-12-04] V3.6.0

- 启用代码proguard,减少包大小
- 去除获取地址位置权限声明

[2017-12-01] V3.5.2

- 通知栏删除接口增加一次删除多个NotifyID的功能一次可传入多个notifyId

[2017-11-13] V3.5.0

- 增加通知栏清除功能,通知栏消息聚合功能
- MzPushMessageReceiver 接口重大变更
- 优化数据上报逻辑,提升数据上报准确度
- 一些已知问题的修改

[2017-08-18] V3.4.2

- 解决使用换机助手时,应用无法更新pushId
- 应用没有设置状态栏图标,默认设置flyme第三方风格的状态栏图标

[2017-06-01] V3.3.170601

- 增加应用拉活功能

[2017-05-18] V3.3.170518

- 增加内存缓存数据上报策略,解决SQLite数据库无法读取的错误

[2017-04-26] V3.3.170505

- 优化PushManager逻辑
- 增加打开第三方应用的功能
- 统一PushSDK内外版本,artifactId为:push-internal,完整配置如下:compile 'com.meizu.flyme.internet:push-internal:3.3.170505@aar'
- 增加点击通知栏和透传消息传递平台参数的功能

[2017-03-29] V3.3.170329

- 外部应用设置状态栏图标也能正确显示
- 优化log输出逻辑,日志按天输出
- 修复intent.parseUri的安全漏洞,但是打开应用某个界面必须填写Activity的全路径
- 增加统一修改通知栏和透传消息开关的接口
- 增加取消所有标签接口
- 优化一些性能问题

[2017-01-18] V3.3.170112

- 删除https无用代码
- 解决deviceId无法获取导致无法订阅的问题
- 删除无用权限声明

[2017-01-10] V3.3.170110

- 解决Jar包集成无法找到R类的问题
- 解决数据上报安全性问题

[2017-01-04] V3.3.170103

- 精简sdk代码,权限,不再依赖第三方库
- 加入通知栏动态视频功能，如需要使用，需要向平台申请开通权限

[2016-12-26] V3.3.161226

- Https 加入安全校验
- 通知栏功能仅支持 API 16以上的android版本
- 通知栏兼容至API 11

[2016-12-23] V3.3.161222

- pushsdk 去除第三方依赖，不再依赖其他任何第三方库

8. 更多技术支持

推送交流QQ群：488591713

推送交流邮箱：push_support@xjmz.com