

# Android SDK 开发指南

## 1. 使用说明

- 本文是 DMHub Android SDK 标准的开发指南文档，用以指导 SDK 的集成和使用，默认读者已经具备一定的 Android 开发能力。
- 本篇指南匹配的 DMHub Android SDK 版本为：`v0.2.0`。
- DMHub Android SDK 0.2.0 要求 `Java >= 1.7` & `Android API >= 9`。

## 2. 开发准备

### 2.1 创建应用

集成 DMHub SDK 之前，您首先需要到 DM Hub 平台创建应用。

#### 2.1.1 进入应用设置页面

点击 DM Hub 平台首页右上角的齿轮图标，选择 `开放与集成` 选项，进入应用设置页面。



## 2.1.2 新建应用

在应用设置页面点击右上角的 **+ 新建** 按钮，在弹出的创建应用弹出框中填写应用名称和描述后保存。



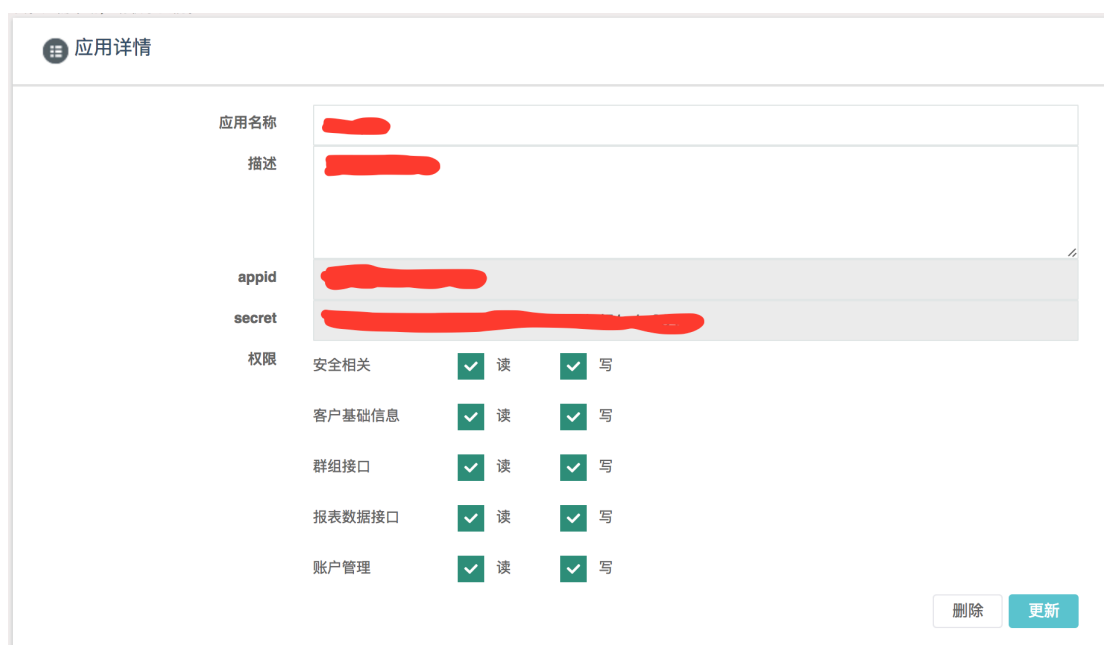
创建应用

应用名称

描述

## 2.1.3 更新权限设置

创建应用成功之后，即可获得集成 SDK 所需的 **appid** 和 **secret** 信息。根据开发需求进行权限设置后，点击右下角的 **更新** 按钮（注：即使没有更改权限设置，也要进行更新），完成应用创建。



应用详情

应用名称

描述

appid

secret

权限	安全相关	读	写
客户基础信息	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
群组接口	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
报表数据接口	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>
账户管理	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>

## 2.2 环境搭建

- 集成 `OkHttp3` 或 `Retrofit2` 。
- 集成 `极光推送` 或 `个推推送` 。

## 3. 导入 SDK

---

### 3.1 复制 aar 包

复制 `libs` 目录下的 [dmhubsdk-android-0.2.0.aar](#) 文件到工程主 `module` 的 `libs` 目录下。

### 3.2 修改 gradle 配置

打开工程主 `module` 的 `build.gradle` 配置文件，添加配置：

```
android {  
    .....  
}  
  
repositories {  
    flatDir {  
        dirs 'libs'  
    }  
}  
  
dependencies {  
    .....  
    compile(name: 'dmhubsdk-android-0.2.0', ext: 'aar')  
    // 集成 OkHttp3 或 Retrofit2，具体版本请根据开发需求选择  
    compile 'com.squareup.okhttp3:okhttp:3.x.x'  
    compile 'com.squareup.retrofit2:retrofit:2.x.x'  
    .....  
}
```

### 3.3 配置 AndroidManifest.xml

在 AndroidManifest.xml 中的 `<application></application>` 标签内配置 SDK 所需组件和参数：

```
<receiver android:name="com.convertlab.dmhubsdk.NetReceiver">
    <intent-filter>
        <action android:name="android.net.conn.CONNECTIVITY_CHANGE" />
    </intent-filter>
</receiver>

<meta-data
    android:name="DMHubSDKAppId"
    android:value="在 DM Hub 平台获得的 appid" />
<meta-data
    android:name="DMHubSDKSecret"
    android:value="在 DM Hub 平台获得的 secret" />
<!--
    您所采用的推送平台，支持极光和个推。
    如果您采用极光推送，请填写 'jpush' (默认，可以不配置)；
    如果您采用个推推送，请填写 'getui'；
    如果您同时采用极光推送和个推推送，请填写 'jpush&getui'。
-->
<meta-data
    android:name="DMHubPushChannel"
    android:value="采用的推送平台" />
```

如果您是在测试账号下创建的应用，则需要添加配置：

```
<meta-data
    android:name="DMHubServer"
    android:value="http://api.convertwork.cn" />
```

## 4. 初始化

---

在自定义的 `Application` 中的 `onCreate` 方法中调用初始化方法：

```
public class DMHubApp extends Application{

    @Override
    public void onCreate() {
        super.onCreate();

        // 初始化 DMHubSDK
        DMHubSDK.sharedInstance().init(this);
    }
}
```

注：在整个应用程序全局，只需要进行一次初始化。

## 5. 创建初始客户和客户身份

---

在 App 首次接收到推送平台分配的 `设备 ID` 时，以 `设备 ID` 作为客户身份创建未知客户和客户身份。

- 如果您采用了极光推送，请参照下面的代码创建未知客户和客户 JPush 身份：

```

public class YourJPushReceiver extends BroadcastReceiver {

    @Override

    public void onReceive(Context context, Intent intent) {

        if

(JPushInterface.ACTION_REGISTRATION_ID.equals(intent.getAction())) {

            // 接收到 JPush Registration Id

            String jPushId =

intent.getExtras().getString(JPushInterface.EXTRA_REGISTRATION_ID);

            // 创建未知客户和客户 JPush 身份

            if (jPushId != null && <App 首次接收到 JPushId>) {

                /**

                 * 创建未知客户和客户 JPush 身份

                 *

                 * @param unknownName 创建未知客户时指定的初始用户名，默认为

'unknown'

                 * @param dmHubAppName 在 DM Hub 创建应用时设置的应用名称

                 * @param jPushAppKey JPush 的 AppKey

                 * @param jPushId JPush SDK 向 JPush Server 注册所得到的注

册 ID

                 * @param source 显示在 DM Hub 客户时间轴上的客户来源，建

议使用 App 名称

                 */

                DMHubSDK.sharedInstance().createUnknownCustomerWithJPushIdentity(unknownNam

e, dmHubAppName, jPushAppKey, jPushId, source);

                // 或者

                DMHubSDK.sharedInstance().createUnknownCustomerWithJPushIdentity(dmHubAppNa

me, jPushAppKey, jPushId, source);

            }

        }

    }

}

```

- 如果您采用了个推推送，请参照下面的代码创建未知客户和客户 GeTui 身份：

```
public class YourGeTuiIntentService extends GTIntentService {

    @Override
    public void onReceiveClientId(Context context, String clientId) {
        // 接收到 GeTui clientId, 创建未知客户和客户 GeTui 身份
        if (clientId != null && <App 首次接收到 clientId>) {
            /**
             * 创建未知客户和客户 GeTui 身份
             *
             * @param unknownName 创建未知客户时指定的初始用户名，默认为
            'unknown'
             * @param dmHubAppName 在 DM Hub 创建应用时设置的应用名称
             * @param geTuiAppKey GeTui 的 AppKey
             * @param geTuiId      GeTui SDK 向 GeTui Server 注册所得到的
            clientId
             * @param source      显示在 DM Hub 客户时间轴上的客户来源，建议使
            用 App 名称
             */

            DMHubSDK.sharedInstance().createUnknownCustomerWithGeTuiIdentity(unknownName, dmHubAppName, geTuiAppKey, geTuiId, source);

            // 或者

            DMHubSDK.sharedInstance().createUnknownCustomerWithGeTuiIdentity(dmHubAppName, geTuiAppKey, geTuiId, source);
        }
    }
}
```

## 6. 跟踪客户事件

跟踪客户事件是 DMHubSDK 最核心的功能。开发人员可以根据实际需求，通过调用 SDK 提供的方法，对客户事件进行跟踪，将客户在手机原生应用中产生的有价值行为，记录到 DM Hub 平台的客户时间轴上。

而 DM Hub 平台则会以跟踪到的客户事件为数据基础，对海量客户进行智能筛选和高效互动，从而实现精准营销。

### 6.1 跟踪预置客户事件

为了开发人员能够更方便的调用，DM Hub 平台预置了几种常见的客户事件。

- 跟踪客户打开应用事件

```
/**
 * @param appName    应用名称，会在客户时间轴上显示
 * @param targetId    消息推送服务开放平台分配的 AppKey，如果既采用了极光推送又采用了个推推送，请选择 jPush AppKey
 * @param appVersion 应用版本号
 * @param properties 事件的自定义属性
 */
DMHubSDK.sharedInstance().openApp(appName, targetId, appVersion,
properties);
// 或者
DMHubSDK.sharedInstance().openApp(appName, targetId);
```



- 跟踪客户退出应用事件

```
/**
 * @param appName    应用名称，会在客户时间轴上显示
 * @param targetId    消息推送服务开放平台分配的 AppKey，如果既采用了极光推送又采用了个推推送，请选择 jPush AppKey
 * @param appVersion 应用版本号
 * @param properties 事件的自定义属性
 */
DMHubSDK.sharedInstance().exitApp(appName, targetId, appVersion,
properties);
// 或者
DMHubSDK.sharedInstance().exitApp(appName, targetId);
```

- 跟踪客户进入页面事件

```
/**
 * @param viewName    页面的名称，可以使用类名或自定义名称，会在客户时间轴上显示
 * @param viewId       页面的 id，可以使用 hashCode
 * @param appName      应用名称
 * @param appVersion   应用版本号
 * @param properties   事件的自定义属性
 * 注：页面可以是 Activity、Fragment...
 */
DMHubSDK.sharedInstance().openView(viewName, viewId, appName, appVersion,
properties);
// 或者
DMHubSDK.sharedInstance().openView(viewName, viewId);
```

- 跟踪客户离开页面事件

```
/**
 * @param viewName 页面的名称，可以使用类名或自定义名称，会在客户时间轴上显示
 * @param viewId 页面的 id，可以使用 hashCode
 * @param appName 应用名称
 * @param appVersion 应用版本号
 * @param properties 事件的自定义属性
 * 注：页面可以是 Activity、Fragment...
 */
DMHubSDK.sharedInstance().exitView(viewName, viewId, appName, appVersion,
properties);
// 或者
DMHubSDK.sharedInstance().exitView(viewName, viewId);
```

- 跟踪客户点击通知事件

```
/**
 * @param notiTitle 通知标题，会在客户时间轴上显示
 * @param notiId 通知 id
 * @param appName 应用名称
 * @param appVersion 应用版本号
 * @param properties 事件的自定义属性
 */
DMHubSDK.sharedInstance().clickNotification(notiTitle, notiId, appName,
appVersion, properties);
// 或者
DMHubSDK.sharedInstance().clickNotification(notiTitle, notiId);
```

## 6.2 跟踪自定义客户事件

通过自定义客户事件，可以更灵活的对客户产生的事件进行跟踪。

在 DM Hub 平台新建自定义事件后，可以通过下面的方法对自定义事件进行跟踪：

```
/**
 * @param eventId    与 DM Hub 中自定义的事件对应的事件 ID
 * @param targetName 对于自定义事件，客户时间轴上只会显示 targetName，相当于事件
标题
 * @param targetId   客户产生该事件对应的目标(如按钮)的 Id
 * @param appName    应用名称
 * @param appVersion 应用版本号
 * @param properties 事件的自定义属性
 * 注：事件产生的时间信息 SDK 已进行处理，不需要在通过 properties 参数传递
 */
DMHubSDK.sharedInstance().track(eventId, targetName, targetId, appName,
appVersion, properties);
// 或者
DMHubSDK.sharedInstance().track(eventId, targetName, targetId, properties);
```

## 7. 技术支持

---

- 在线客服：在 DM Hub 平台右下角进行客服咨询
- 电子邮件：[support@convertlab.com](mailto:support@convertlab.com)