蚂蚁科技产品手册

社交分享

产品版本： V20210430

文档版本： V20210430

蚂蚁科技技术文档

商标声明

及其他蚂蚁科技服务相关的商标均为蚂蚁科技所有。
本文档涉及的第三方的注册商标，依法由权利人所有。

免责声明

由于产品版本升级、调整或其他原因，本文档内容有可能变更。蚂蚁科技保留在没有任何通知或者提示下对本文档的内容进行修改的权利，并在蚂蚁科技授权通道中不时发布更新后的用户文档。您应当实时关注用户文档的版本变更并通过蚂蚁科技授权渠道下载、获取最新版的用户文档。如因文档使用不当造成的直接或间接损失，本公司不承担任何责任。

目录

1 社交分享简介

mPaaS 的社交分享组件提供微博、微信、支付宝、QQ、短信等渠道的分享功能，提供给开发者统一的接口，无须处理各 SDK 的接口差异性。

2 接入 Android

2.1 快速开始

关于此任务

社交分享组件提供微博、微信、支付宝、QQ、钉钉、短信等渠道的分享功能，提供给开发者统一的接口，无需处理各 SDK 的接口差异性。要将分享组件接入 Android 客户端，您需要配置工程确定基础框架，并添加 share 组件的 SDK。

前置条件

在接入各渠道之前，必须在分享渠道的官方网站申请账号。例如以下分享渠道的官方网站：

- 微博： http://open.weibo.com/
- 微信： https://open.weixin.qq.com/
- QQ： http://open.qq.com/
- 支付宝： http://open.alipay.com/index.htm
- 钉钉：https://www.dingtalk.com/

社交分享支持 **原生 AAR 接入**、**mPaaS Inside 接入** 和 **组件化接入** 三种接入方式。

- 若采用原生 AAR 方式接入，需先完成 将 mPaaS 添加到您的项目中 的前提条件和后续相关步骤。
- 若采用 mPaaS Inside 方式接入，需先完成 mPaaS Inside 接入流程 。
- 若采用组件化方式接入，需先完成 组件化接入流程 。

添加 SDK

原生 AAR 方式

参考 AAR 组件管理 ，通过 **组件管理（AAR）** 在工程中安装 **分享** 组件。

mPaaS Inside 方式

在工程中通过 **组件管理** 安装 **分享** 组件。
更多信息，参考 管理组件依赖 。

组件化方式

在 Portal 和 Bundle 工程中通过 **组件管理** 安装 **分享** 组件。
更多信息，参考 管理组件依赖 。

初始化 mPaaS

如果使用原生 AAR 方式 / mPaaS Inside 方式，需要初始化 mPaaS。

请在 Application 中添加以下代码：

```
public class MyApplication extends Application {

@Override
protected void attachBaseContext(Context base) {
super.attachBaseContext(base);
// mPaaS 初始化回调设置
QuinoxlessFramework.setup(this, new IInitCallback() {
@Override
public void onPostInit() {
// 此回调表示 mPaaS 已经初始化完成，mPaaS 相关调用可在这个回调里进行
}
});
}

@Override
public void onCreate() {
super.onCreate();
// mPaaS 初始化
QuinoxlessFramework.init();
}
}
```

各个平台分享 SDK 使用

本文将结合 社交分享 官方 Demo 介绍如何在 10.1.32 及以上版本的基线中使用社交分享 SDK。

微信分享

您需要手动生成一个特定路径和名称的 Activity 用来接收微信分享的回调事件。这个 Activity 继承自 DefaultWXEntryActivity，路径为 package_name.wxapi.WXEntryActivity。其中，package_name 为应用的包名。

说明：路径和 Activity 名称必须准确，否则将无法收到回调。

查看以下示例，其中包名为 com.mpaas.demo：

```
package com.mpaas.demo.wxapi;
import com.alipay.android.shareassist.DefaultWXEntryActivity;
public class WXEntryActivity extends DefaultWXEntryActivity {
}
```

在 AndroidManifest.xml 中对该 Activity 进行注册：

```
<application>
...
<activity android:name="com.mpaas.demo.wxapi.WXEntryActivity"
android:exported="true"
android:launchMode="singleTop"></activity>
...
</application>
```

说明：设置分享图标时，确保图标的大小不超过 32 KB，否则可能会引起微信分享失败。目前在 Android 端 SDK 做了校验，图标大小超过 32 KB 时会用默认的支付宝图标代替。

QQ、QZone 分享

您需要在 AndroidManifest.xml 中，对 QQ 分享所需要的 Activity 进行注册，否则无法正常使用 QQ、QZone 的分享和回调功能。

说明：

- 若您在 AndroidManifest.xml 中填写的 QQ 分享 ID 和在代码中注册的 QQ 分享 ID 不一致时，会导致 QQ 分享回调错乱的异常，即使分享成功也会回调 onException，请务必仔细检查。
- 在 data android:scheme 中要填写对应的 QQ 分享 ID，格式为 tencent+QQID (+ 号请忽略)。该 ID 需到 腾讯开放平台 中自行申请。查看以下示例，其中 QQ ID 为 1104122330。

```xml
<application>
...
<activity
android:name="com.tencent.connect.common.AssistActivity"
android:configChanges="orientation|keyboardHidden|screenSize"
android:screenOrientation="portrait"
android:theme="@android:style/Theme.Translucent.NoTitleBar"/>
<activity
android:name="com.tencent.tauth.AuthActivity"
android:launchMode="singleTask"
android:noHistory="true">
<intent-filter>
<action android:name="android.intent.action.VIEW"/>
<category android:name="android.intent.category.DEFAULT"/>
<category android:name="android.intent.category.BROWSABLE"/>
<data android:scheme="tencent1104122330"/>
</intent-filter>
</activity>
...
</application>
```

微博分享

需要确保应用签名、包名、分享 ID 和在 微博开放平台 中注册的一致，否则将导致分享失败。由此原因导致分享失败时，share 组件的分享回调不会触发分享异常 onException，而会触发分享成功 onComplete。该缺陷属于微博 SDK 缺陷，目前在微博 SDK 官方 Demo 中同样会出现此问题。

相关链接

- 参见 获取代码示例 以获取代码示例以及使用方法和注意事项。
- 相关 API 接口文档：
  - 分享服务接口
  - 分享类型接口
  - 分享内容接口
  - 分享异常接口

2.2 迁移到 10.1.60 基线

本文介绍了如何将社交分享组件从 10.1.60 以前的基线中迁移到 10.1.60 基线。

操作步骤

1. 卸载分享组件。
   如果您之前已经根据 安装指引 安装了老版本的 mPaaS 分享组件，那么您需要在 build.gradle 中删除以下内容进行卸载。

```
provided"com.alipay.android.phone.mobilecommon:share-build:1.3.0.xxxx:api@jar"
bundle"com.alipay.android.phone.mobilecommon:share-build:1.3.0.xxxx@jar"
manifest"com.alipay.android.phone.mobilecommon:share-build:1.3.0.xxxx:AndroidManifest@xml"
```

   升级基线到 10.1.60。如果您已经升级了基线，请跳过此步骤。
   在 Android Studio 的菜单中，点击 **mPaaS** > **基线升级**。选择 10.1.60 基线后，点击 **OK**。

安装 10.1.60 基线下的分享组件。
在 Android Studio 的菜单中，点击 **mPaaS** > **组件管理**，添加 mPaaS 分享组件。

说明：10.1.60 基线下和 10.1.32 基线下分享组件的 API 并没有变化。

2.3 API 说明

2.3.1 分享服务接口

分享服务接口 ShareService：

```
public abstract class ShareService extends ExternalService {

/**
* 静默分享，只能指定一种分享类型，不会显示分享选择界面
* @param content 分享内容
* @param shareType 分享类型
* @param biz biz
*/
public abstract void silentShare(ShareContent content, final int shareType, final String biz);

/**
* 设置分享监听对象
* @param listener 监听对象
*/
public abstract void setShareActionListener(ShareActionListener listener);

/**
* 获取分享监听对象
```

```
* @return 监听对象
*/
public abstract ShareActionListener getShareActionListener();

/**
* 设置app的名字
* @param name app名字
*/
public abstract void setAppName(String name);

/**
* 初始化微信分享
* @param appId 微信appId，在微信渠道中注册获取
* @param appSecret 微信appSecret，在微信渠道中注册获取
*/
public abstract void initWeixin(String appId, String appSecret);

/**
* 初始化微博分享
* @param appId 微博appId，在微博渠道中注册获取
* @param appSecret 微博appSecret，在微博渠道中注册获取
* @param redirectUrl 微博分享重定向链接
*/
public abstract void initWeiBo(String appId, String appSecret, String redirectUrl);

/**
* 初始化QZone分享
* @param appId QZone appId，在QQ渠道中注册获取
*/
public abstract void initQZone(String appId);

/**
* 初始化QQ分享
* @param appId QQ appId，在QQ渠道中注册获取
*/
public abstract void initQQ(String appId);

/**
* 初始化支付宝分享
* @param appId 支付宝 appId，在支付宝渠道中注册获取
*/
public abstract void initAlipayContact(String appId);

/**
* 初始化钉钉分享
* @param appId 钉钉appId，在钉钉渠道中注册获取
*/
public abstract void initDingDing(String appId);

}
```

2.3.2 分享类型接口

ShareType 分享类型接口：

```
public class ShareType {
/**
* 短信
*/
public static final int SHARE_TYPE_SMS = 2;

/**
* 微博
*/
public static final int SHARE_TYPE_WEIBO = 4;

/**
* 微信好友
*/
public static final int SHARE_TYPE_WEIXIN = 8;

/**
* 微信朋友圈
*/
public static final int SHARE_TYPE_WEIXIN_TIMELINE = 16;

/**
* 复制链接
*/
public static final int SHARE_TYPE_LINKCOPY = 32;

/**
* QQ空间动态
*/
public static final int SHARE_TYPE_QZONE = 256;

/**
* QQ好友
*/
public static final int SHARE_TYPE_QQ = 512;

/**
* 联系人
*/
public static final int SHARE_TYPE_CONTACT = 1024;

/**
* 我的生活
*/
public static final int SHARE_TYPE_CONTACT_TIMELINE = 2048;

/**
* 钉钉
*/
public static final int SHARE_TYPE_DINGDING = 4096;

/**
* 圈子
*/
public static final int SHARE_TYPE_GROUP = 8192;
```

```
/**
* 所有
*/
public static final int SHARE_TYPE_ALL = 65535;
}
```

注意事项

在使用 **微信** 分享纯文本的时候，需要在 分享内容接口 的 sharecontent 类中，把 Url、Image、和 ImgUrl 置为 null。

```
shareContent.setUrl(null);
shareContent.setImage(null);
shareContent.setImgUrl(null);
```

在使用 **微博** 分享纯文本的时候，需要在 分享内容接口 的 ShareContent 类中，把 Url 置为空格，Image、ImgUrl 置为 null。

```
shareContent.setUrl("");
shareContent.setImage(null);
shareContent.setImgUrl(null);
```

2.3.3 分享内容接口

ShareContent 分享内容接口：

注意：

- 调用 get 和 set 方法对 ShareContent 的变量进行访问。
- 由于分享没有统一的标准，通过使用 imgUrl 在内部抹平差异。所有分享优先使用 “分享图片 URL（imgUrl）”，其次使用 “分享图片（image）”。

```
public class ShareContent implements Serializable {

/*
* 分享内容
*/
private String content;

/*
* 分享的图片
*/
private byte[] image;

/*
* 分享跳转的 URL
*/
private String url;

/*
```

```
* 分享标题
*/
private String title;

/*
* 分享图片 URL
*/
private String imgUrl;

/*
* 扩展参数: 用于传递生成短链接及短信发送成功等业务参数
*/
private String extData;

/*
* 分享类型："url"为分享链接，"image"为分享图片
*/
private String contentType;

/*
* 分享到联系人：分享预览框中小图的图片 URL
*/
private String iconUrl;

/**
* 本地图片 URL
*/
private String localImageUrl;
}
```

2.3.4 分享异常接口

ShareException 分享异常接口：

```
public class ShareException extends RuntimeException {

/**
* 状态码：用户取消
*/
public static final int USER_CANCEL = 1001;

/**
* 状态码：认证失败
*/
public static final int AUTH_ERROR = 1002;

/**
* 状态码：其他异常
*/
public static final int UNKNOWN_ERROR = 1003;

/**
* 状态码：应用未安装
*/
```

```
public static final int APP_UNINSTALL = 40501;

/**
* 获取状态码
* @return
*/
public int getStatusCode() {
return this.statusCode;
}
}
```

3 接入 iOS

3.1 快速开始

社交分享组件 MPShareKit 提供微博、微信、支付宝、QQ、钉钉、短信等渠道的分享功能，提供给开发者统一的接口，无须处理各 SDK 的接口差异性。

添加 SDK

根据您采用的接入方式，请选择相应的添加方式。更多信息，请参见 编辑模块 。

- 使用 mPaaS Xcode Extension。
  此方式适用于采用了 **基于 mPaaS 框架接入** 或 **基于已有工程且使用 mPaaS 插件接入** 的接入方式。
  - 点击 Xcode 菜单项 **Editor** > **mPaaS** > **编辑工程**，打开编辑工程页面。
  - 选择 **分享**，保存后点击 **开始编辑**，即可完成添加。

- 使用 cocoapods-mPaaS 插件。此方式适用于采用了 **基于已有工程且使用 CocoaPods 接入** 的接入方式。
    - 在 Podfile 文件中，使用 mPaaS_pod"mPaaS_Share" 添加分享组件依赖。
    - 执行 pod install 即可完成接入。

后续步骤

使用 SDK （版本 ≥ 10.1.60 ）

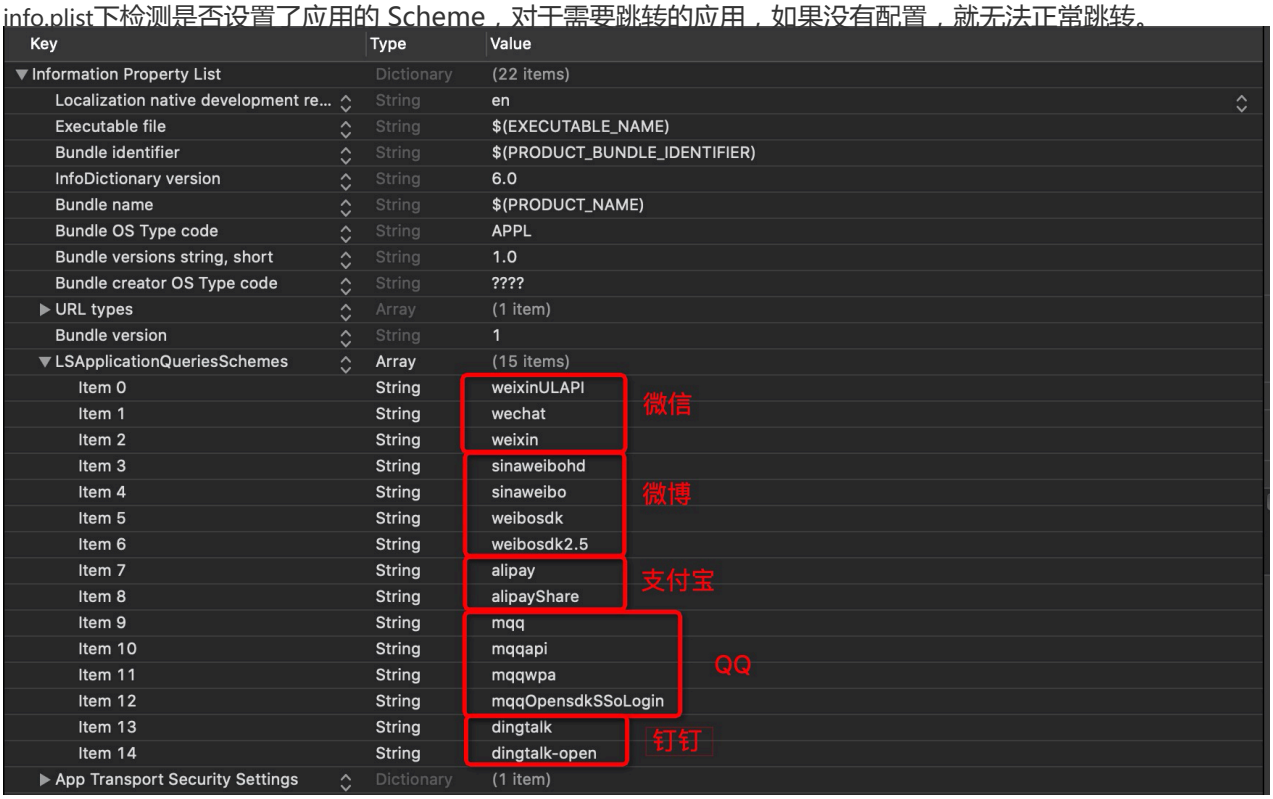
3.2 使用 SDK（版本 ≥ 10.1.60）

在完成添加分享组件的 SDK 后，您还需要对工程进行配置，在初始化后即可开始使用分享组件。本文将结合分享官方 Demo 介绍如何在 10.1.60 及以上版本的基线中使用分享 SDK。

配置工程

配置第三方应用跳转白名单

iOS 9 及以上系统中，要支持应用的分享和授权等操作，需要配置 Scheme 名单。系统会自动到工程的 info.plist下检测是否设置了应用的 Scheme，对于需要跳转的应用，如果没有配置，就无法正常跳转。



配置URL Scheme

为保证可以从各渠道应用正确跳转回当前应用，需在当前工程的 Info.plist 文件中加入 urlScheme（从各分享渠道获取）。

- 微信回调源 App 的 scheme 均为分配的 key
- 微博的 scheme 为 "wb"+ key
- QQ 的 scheme 为 "tencent"+ APPID
- 支付宝的 Identifier 为 alipayShare，scheme 为 'ap' + APPID

初始化设置

使用分享组件时，首先需要在对应第三方平台创建对应的应用信息，然后使用应用的信息进行注册（包括 appId，appSecret，universalLink 等信息），注册使用的方法如下所示。

使用 mPaaS 框架时

在 DTFrameworkInterface 的分类中实现下列方法，并在此方法中以词典的形式返回 key、secret 的值。

注意：mPaaS SDK 版本 ≥ 10.1.60 开始，微信 SDK 已经更新到 1.8.6.1，需要进行 Universal Link 的校验，所以在配置密钥的同时需要配置对应的 Universal Link。

```
- (void)application:(UIApplication *)application beforeDidFinishLaunchingWithOptions:(NSDictionary
*)launchOptions
{
NSDictionary *configDic = @{
@"weixin": @{@"key":@"wxc5c09c98c276ac86", @"secret":@"d56057d8a43031bdc178991f6eb8dcd5",
@"universalLink":@"https://mpaas.demo.com/"},
@"weibo": @{@"key":@"1877934830", @"secret":@"1067b501c42f484262c1803406510af0"},
@"qq": @{@"key":@"1104122330", @"secret":@"WyZkbNmE6d0rDTLf"},
@"alipay": @{@"key":@"2015060900117932"},/*该 key 对应的 bundleID 为"com.alipay.share.demo", 如需用来测试,请
修改为自己申请的 key 或修改 bundleID 为"com.alipay.share.demo"*/
@"dingTalk": @{@"key":@"dingoaa4aipzuf2yifw17s"}};
[APSKClient registerAPPConfig:configDic];
}
```

不使用 mPaaS 框架时

在接入应用的启动方法中注册密钥。

注意： mPaaS SDK 版本 ≥ 10.1.60 开始，微信 SDK 已经更新到 1.8.6.1，需要进行 Universal Link 的校验，所以在配置密钥的同时需要配置对应的 Universal Link。

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
NSDictionary *dic = @{
@"weixin": @{@"key":@"wxc5c09c98c276ac86",
@"secret":@"d56057d8a43031bdc178991f6eb8dcd5",@"universalLink":@"https://mpaas.demo.com/"},
@"weibo": @{@"key":@"1877934830", @"secret":@"1067b501c42f484262c1803406510af0"},
@"qq": @{@"key":@"1104122330", @"secret":@"WyZkbNmE6d0rDTLf"},
@"alipay": @{@"key":@"2015060900117932"},/*该 key 对应的 bundleID 为"com.alipay.share.demo", 如需用来测试,请
修改为自己申请的 key 或修改 bundleID 为"com.alipay.share.demo"*/
@"dingTalk": @{@"key":@"dingoaa4aipzuf2yifw17s"}};
[APSKClient registerAPPConfig:dic];
}
```

基础功能

分享

唤起分享选择面板

在唤起分享面板时可以指定需要显示的渠道。

```
NSArray *channelArr = @[kAPSKChannelQQ, kAPSKChannelLaiwangContacts, kAPSKChannelLaiwangTimeline,
kAPSKChannelWeibo, kAPSKChannelWeixin, kAPSKChannelCopyLink,kAPSKChannelDingTalkSession];

self.launchPad = [[APSKLaunchpad alloc] initWithChannels:channelArr sort:NO];
```

```
self.launchPad.delegate = self;
[self.launchPad showForView:[[UIApplication sharedApplication] keyWindow] animated:YES];
```

完成分享操作

在@protocol APSKLaunchpadDelegate的sharingLaunchpad回调中执行分享操作。

```
- (void)sharingLaunchpad:(APSKLaunchpad *)launchpad didSelectChannel:(NSString *)channelName
{
[self shareUrl:channelName];
[self.launchPad dismissAnimated:YES];
}

- (void)shareUrl:(NSString*)channelName
{
//生成数据，调用对应渠道分享
APSKMessage *message = [[APSKMessage alloc] init];
message.contentType = @"url";//类型分"text","image","url"三种
message.content = [NSURL URLWithString:@"www.sina.com.cn"];
message.icon = [UIImage imageNamed:@"1"];
message.title = @"标题";
message.desc = @"描述";

APSKClient *client = [[APSKClient alloc] init];

[client shareMessage:message toChannel:channelName completionBlock:^(NSError *error, NSDictionary *userInfo) {
//userInfo 为扩展信息
if(!error)
{
//your logistic
}
NSLog(@"error = %@", error);
}];
}
```

从渠道应用跳回的处理

- 当使用 mPaaS 框架时不需要处理，会由框架负责。
- 当不使用 mPaaS 框架时参照下列代码进行处理。

```
- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url sourceApplication:(NSString
*)sourceApplication annotation:(id)annotation
{
//加入分享成功后，从渠道 APP 回到源 APP 的处理
BOOL ret;
ret = [APSKClient handleOpenURL:url];
return ret;
}
```

登录授权

目前支持登录授权的渠道：微信。

获取第三方渠道的授权 code

仅通过第三方授权，获取授权的 auth code（ auth code 用来换取 access token 以及 openId 等信息）。

```
APSKClient *client = [[APSKClient alloc] init];
[client getAuthCodeForChannel:kAPSKChannelWeixin completionHandler:^(NSError *error, NSString *code) {
// 处理返回结果
}];
```

获取授权用户信息

由于获取授权信息需要经过网络请求，请确认是否已经添加了组件 **移动网关** 组件。
分享组件提供两种方式来获取用户信息：

- 客户端直接请求。
- 自定义方式，通过业务自己的服务获取用户数据。

```
- (void)getUserInfo {
APSKClient *client = [[APSKClient alloc] init];
[client requestAuthUserInfoForChannel:kAPSKChannelWeixin delegate:self];
}

/**
使用授权code组装自定义请求
自定义方式，通过业务自己的服务获取用户数据，如果不实现，则直接默认使用客户端发送请求

@param code 授权code
@return 自定义的请求对象
*/
- (NSURLRequest *)customRequestWithAuthCode:(NSString *)code {
// 自定义请求对象
NSURLRequest *request;
// ...

return request;
}

/**
获取authcode之后，将要请求用户信息，可以处理UI操作
如不需要，可以不用实现

@param channelName 当前授权的渠道
*/
- (void)getUserInfoWillRequest:(NSString*)channelName {
// 设置loading等操作
}

/**
获取到用户授权信息结果

@param response 返回数据，包括用户信息和授权code，发生错误返回nil
@param code 授权code
@param error 错误信息
@param channelName 当前授权渠道
*/
```

```
- (void)getUserInfoDidFinishWithResponse:(NSDictionary *)response
authCode:(NSString *)code
error:(NSError*)error
channel:(NSString*)channelName {
// 处理返回结果，如果成功 response 中会包含授权的用户信息
}
```

开放服务

通过第三方开发服务接口，可以调用第三方渠道分享 SDK 中提供的其他开放服务。目前支持的开放服务有 **微信一次性订阅消息** 和 **拉起微信小程序。**

请求开放服务的操作流程

请求开放服务的操作流程如下：

```
// 1. 创建请求对象
APSKOpenServiceRequest *req = [APSKOpenServiceRequest new];
// 2. 设置请求类型
req.requestType = APSKOpenServiceRequestTypeLaunchMini;
// 3. 设置所需参数
MPSKLaunchMiniProgramParam *param = [[MPSKLaunchMiniProgramParam alloc] init];
param.userName = @"xxxxxxxx";
param.path = @"/index.html";
param.miniProgramType = MPSKWXMiniProgramTypeTest;

req.param = param;

// 4. 创建 client 对象
APSKClient *client = [APSKClient new];
// 5. 请求服务
[client requestOpenService:req toChannel:kAPSKChannelWeixin completionBlock:^(NSError *error, NSDictionary
*userInfo) {
// 6. 回调
if (error) {
NSLog(@"%@", error);
}
}];
```

从渠道应用跳回的处理

从渠道应用跳回后，需要根据客户端是否采用 mPaaS 框架进行不同的处理。

- 如果客户端使用了 mPaaS 框架，在跳回后框架会负责处理。
- 如果客户端没有使用 mPaaS 框架，请您参照下列代码进行跳回后的处理。

```
- (BOOL)application:(UIApplication *)application openURL:(NSURL *)url sourceApplication:(NSString
*)sourceApplication annotation:(id)annotation
{
// 从渠道 APP 回到源 APP 的处理
BOOL ret;
ret = [APSKClient handleOpenURL:url];
```

```
return ret;
}
```


蚂蚁集团
ANT GROUP