即时通信 IM

推送功能

产品文档

腾讯云

文档目录

推送功能
推送功能介绍

最近更新时间：2024-06-13 10:39:26

概述

即时通信 IM 为您提供了两种集成方式：推送插件（推荐）和 自集成推送。
两种集成方式均支持小米、华为、荣耀、OPPO、vivo、魅族、APNs、一加、realme、iQOO、 FCM 等厂商通道，但在接入集成、推送类型、数据统计、链路追踪等方面均有不同，详见下方表格：

对比项		推送插件	自集成推送
价格	刊例价	299 USD/月	免费
支持平台	Android & iOS	✓	✓
	uni-app	✓	×
	Flutter	✓	×
	React Native	✓	×
接入集成	厂商配置	一键式集成	逐一集成7个 Android 厂商和1个 iOS 厂商，共8个 SDK
	SDK 部署	一键式配置	逐一进行8个厂商的配置
	推送注册、token 上报	✓	自行上报
	接入测试工具	✓	✓
	接入周期	1小时	1周
推送类型	普通消息推送	✓	✓
	全员/标签推送-落地推送	✓	-
	全员/标签推送-不落地推送	✓	-
数据统计	普通推送记录查询	✓	-

	全员/标签推送记录查询	✓	-
	普通推送数据统计（实发率、触达率、点击率等）	✓	-
	全员/标签推送数据统计（实发率、触达率、点击率等）	✓	-
链路追踪	推送下发通道查询	✓	-
	推送设备情况查询	✓	-
	推送全链路状态查询（包含 IM 服务器 > 厂商服务器 > 终端设备 > 用户点击的整个链路）	✓	-
离线推送可触达范围		30天内有活跃的用户	7天内有活跃的用户

联系我们

如果您在使用中遇到问题，可通过查阅 常见问题 解决，也可以 点击进入交流群 直接咨询。

插件集成推送（荐）
功能概述

最近更新时间：2024-06-13 10:39:26

概述

推送插件（TIMPush）为您提供稳定、及时、多样化的推送服务。相比自集成推送，推送插件只需进行简单配置，即可一键式集成接入多个厂商的推送服务。推送插件支持普通消息推送和全员/标签推送，提供完整的推送生命周期查询、数据统计、问题排查服务。
如果您有聊天、音视频通话、信令等场景，需要离线场景也能及时触达，可以关注**普通消息推送功能**。
如果您有营销广告、通知、新闻资讯等需要推送给所有用户或指定群体，可以关注**全员/标签推送功能**。
离线推送厂商支持小米、华为、荣耀、OPPO、vivo、魅族、APNs，包含各厂商子品牌如一加、realme、iQOO等，境外支持 Google FCM。
推送管理控制台为您提供全链路排查工具、推送记录和各类型指标统计数据，方便您查看推送触达率、点击率和转化率等各类指标。

功能介绍

3分钟快速集成

不再需要逐个厂商分别配置推送信息，只需在 IM 控制台下载引入 json 配置文件，即可一键式完成所有手机厂商的推送信息配置。
支持按需集成一个或者多个对应厂商的推送渠道包，轻松应对合规要求。
不需要自行处理推送注册、token 上报、前后台状态上报等，推送插件自闭环。
不再需要自行添加打点和处理上报逻辑，推送插件自行上报和汇总，还支持链路排查和指标统计等。
插件封装界面跳转、图标自定义等方法，直接使用即可。

普通消息推送

在普通 IM 消息收发场景，应用离线下消息也可及时抵达设备并支持定制跳转界面。

全员/标签推送

全员/标签推送功能旨在帮助您向全员或者标签用户，进行营销、广告和通知等类型的推送，保证在合适的时机用合适的方式给合适的用户发送合适的消息。

全员/标签推送支持落地和不落地两种推送方式：

落地推送

支持漫游，推送消息也会进入 IM 消息系统，会触发对应会话、消息和未读等模块更新，用户在线时候可以收到推送的消息，用户不在线时候下次登录后可自动拉取到推送的消息。

不落地推送

推送消息产生系统通知抵达设备，消息不会保存在 IM 系统。

自定义跳转界面

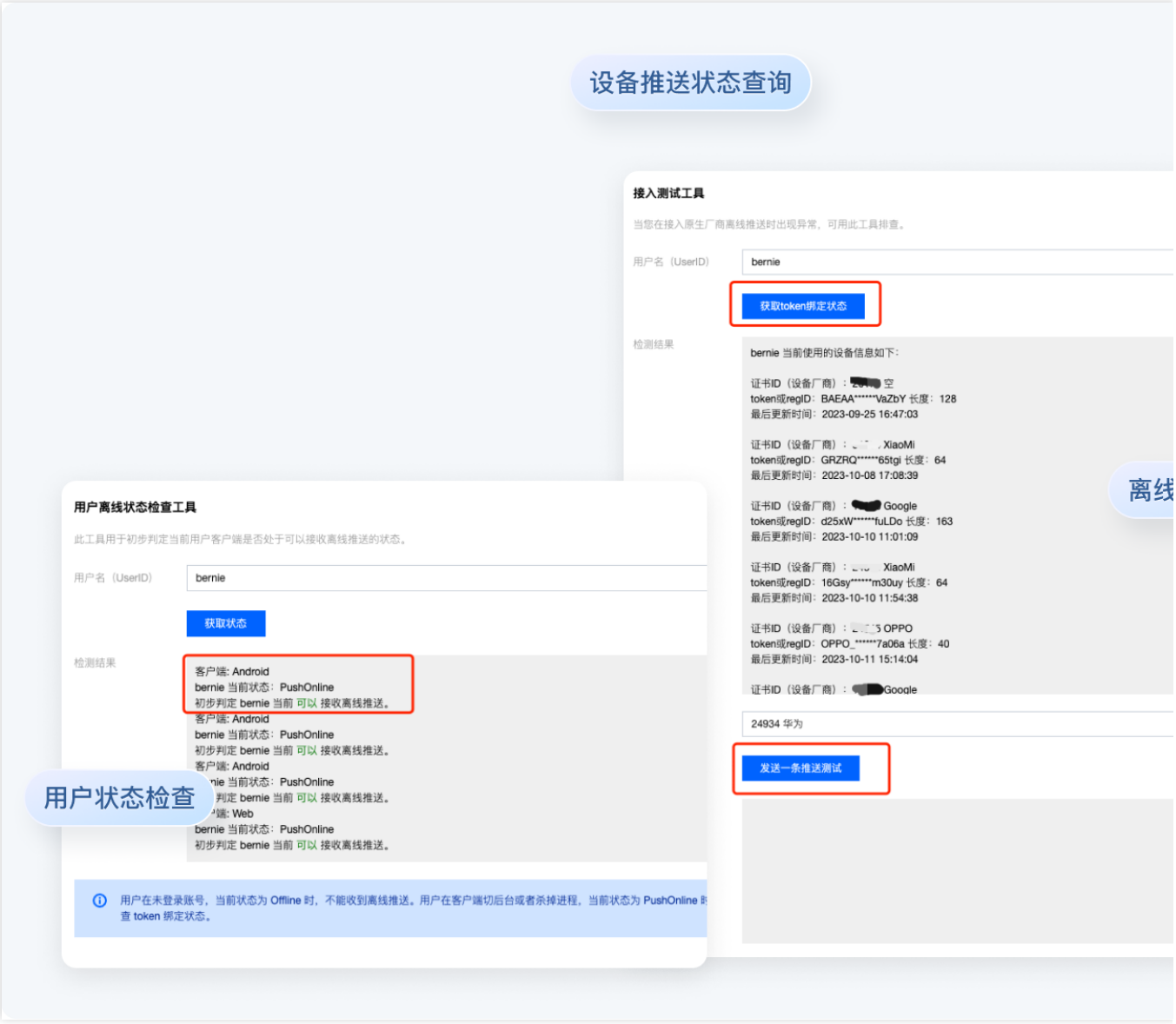
收到推送后，用户单击了通知栏，支持自定义跳转界面。

推送自定义样式

支持小图标、右侧图标、长文本、大图片、角标和铃音自定义样式。

设备推送状态查询

集成推送插件完成后，可以使用控制台接入测试功能自查各个厂商是否配置正常，达到可推送状态。



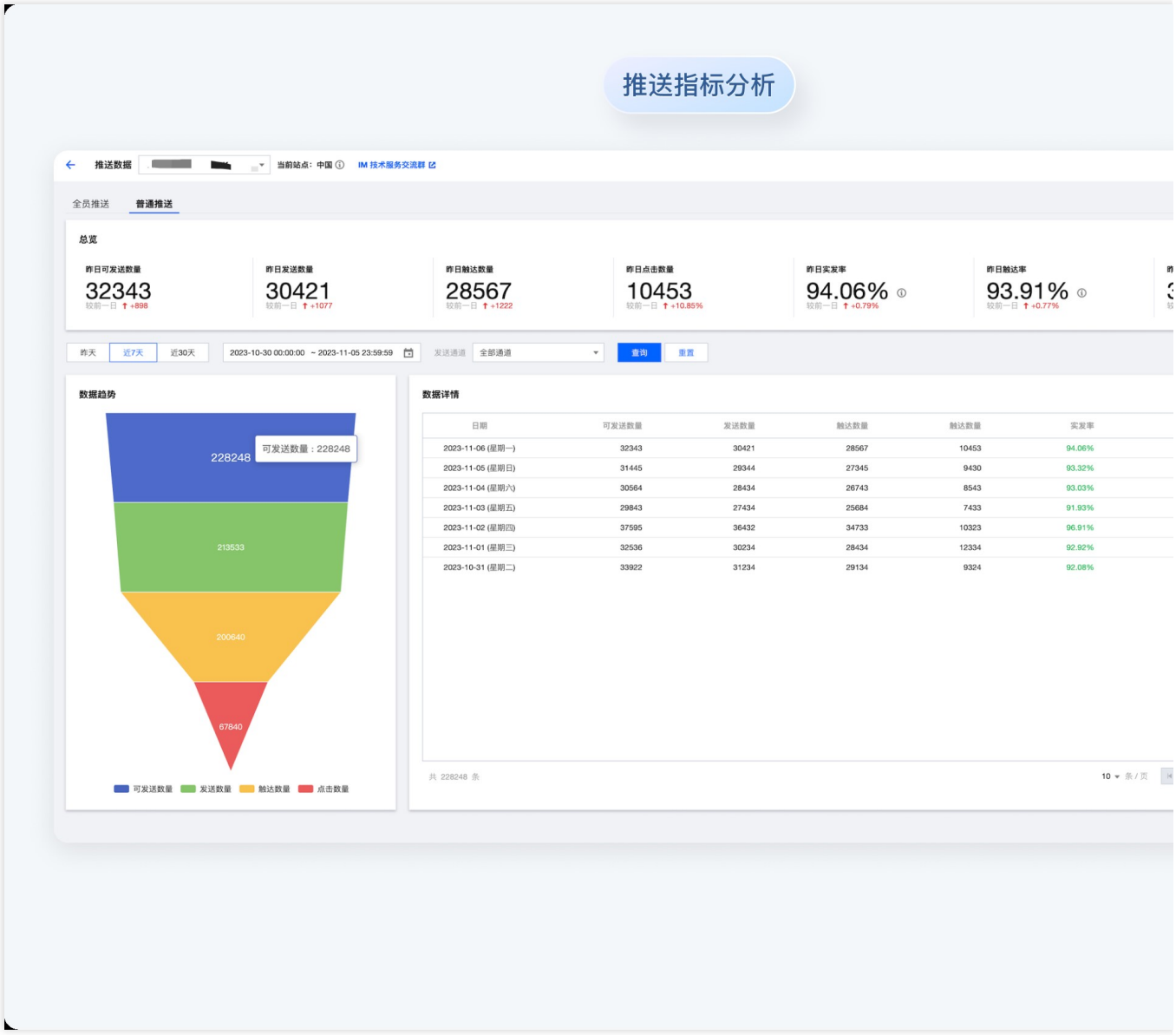
全链路排查工具

提供自助式排查工具，支持查看整个推送链路详情，支持分析推送失败和点击失败原因，提升转化。

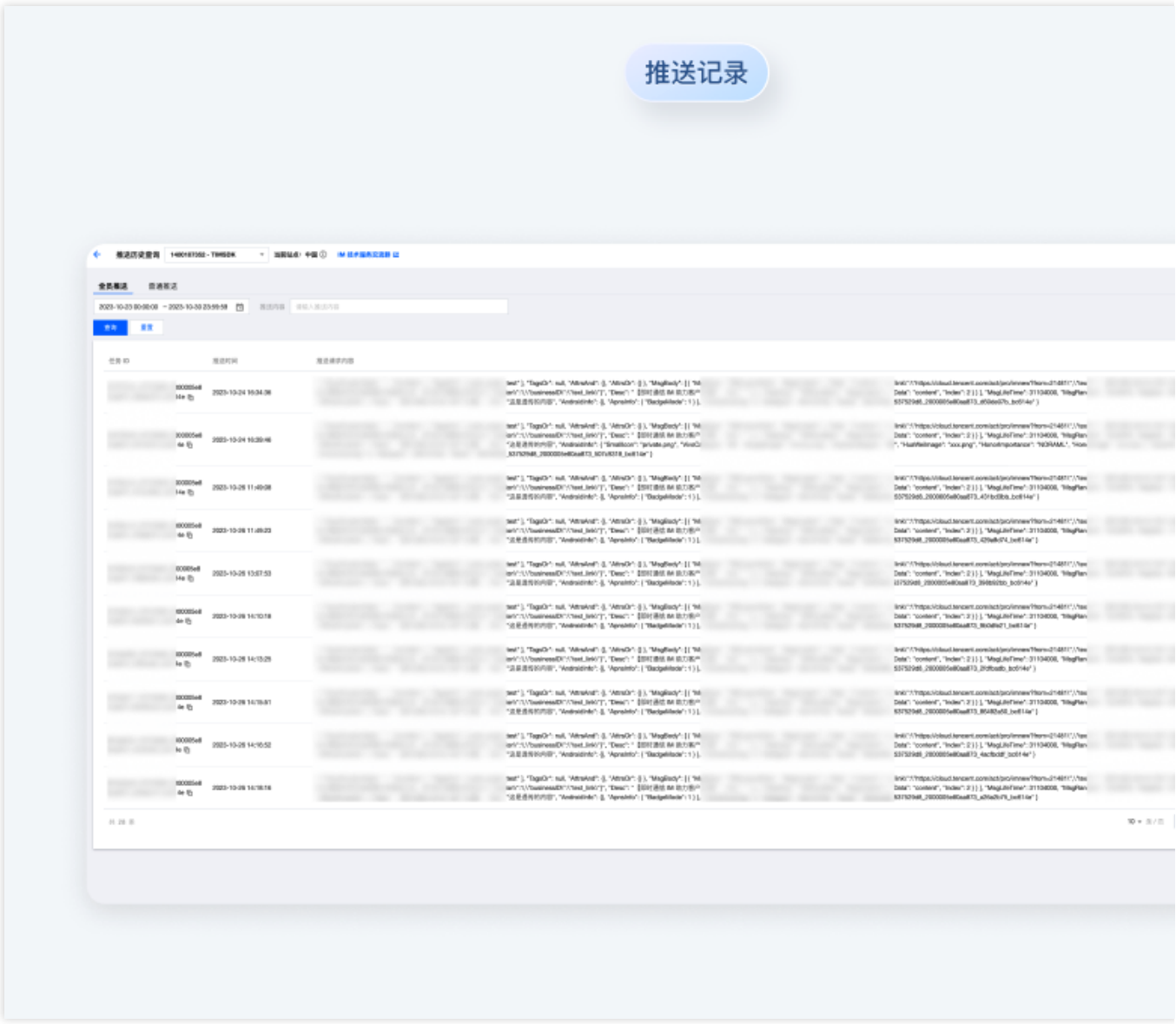
统计数据

记录查询用户所有推送数据，整理和分析用户每日推送各类型指标数据，生成近日**发送 > 触达 > 点击**漏斗转化图表，支持按照厂商通道分类查看。

推送指标分析


推送指标分析
推送数据
当前站点：中国
IM 技术服务交流群
全员推送
普通推送
总览
昨日可发送数量
32343
较前一日 +898
昨日发送数量
30421
较前一日 +1077
昨日触达数量
28567
较前一日 +1222
昨日点击数量
10453
较前一日 +10.85%
昨日实发率
94.06%
较前一日 +0.79%
昨日触达率
93.91%
较前一日 +0.77%
昨天
近7天
近30天
2023-10-30 00:00:00 ~ 2023-11-05 23:59:59
发送通道
全部通道
查询
重置
数据趋势
228248
可发送数量：228248
213533
200640
67840
可发送数量
发送数量
触达数量
点击数量
数据详情
日期
可发送数量
发送数量
触达数量
触达数量
实发率
2023-11-06 (星期一) 32343 30421 28567 10453 94.06%
2023-11-05 (星期日) 31445 29344 27345 9430 93.32%
2023-11-04 (星期六) 30564 28434 26743 8543 93.03%
2023-11-03 (星期五) 29843 27434 25684 7433 91.93%
2023-11-02 (星期四) 37595 36432 34733 10323 96.91%
2023-11-01 (星期三) 32536 30234 28434 12334 92.92%
2023-10-31 (星期二) 33922 31234 29134 9324 92.08%
共 228248 条
10 条 / 页

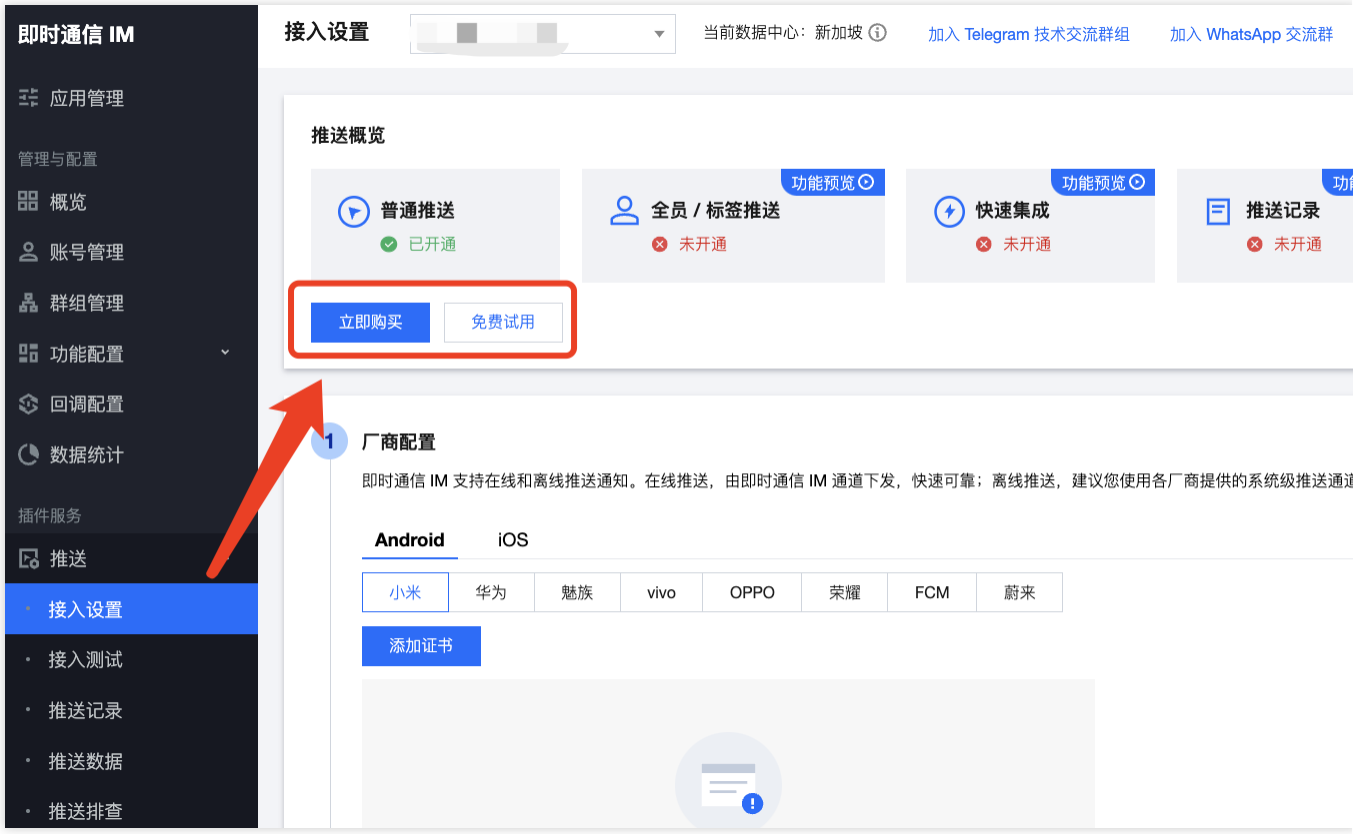
推送记录



开通使用

步骤1：开通推送插件

进入 IM 控制台 > 推送，单击 **立即购买** 或 **免费试用** 。（每个应用可免费试用一次，有效期7天）



注意：

推送插件**试用或购买到期后，将自动停止提供推送服务（包括普通消息离线推送、全员/标签推送等服务）**。为避免影响您业务正常使用，请提前购买/续费。

步骤2：集成和使用推送插件

厂商配置 Android & iOS & Flutter & uniapp & react-native。

快速接入 Android & iOS & Flutter & uniapp & react-native。

步骤3：发送全员/标签推送消息

全员/标签推送消息发送，详情请参见 发起全员/标签推送。

步骤4：推送多样化实现

角标

铃音

小图标

自定义点击跳转

消息分类

步骤5：推送管理和分析

数据统计

排查工具

联系我们

如果您在使用中遇到问题，可通过查阅常见问题解决，也可以 点击进入交流群 直接咨询。

厂商配置

Android

最近更新时间：2024-06-13 10:39:26

操作步骤

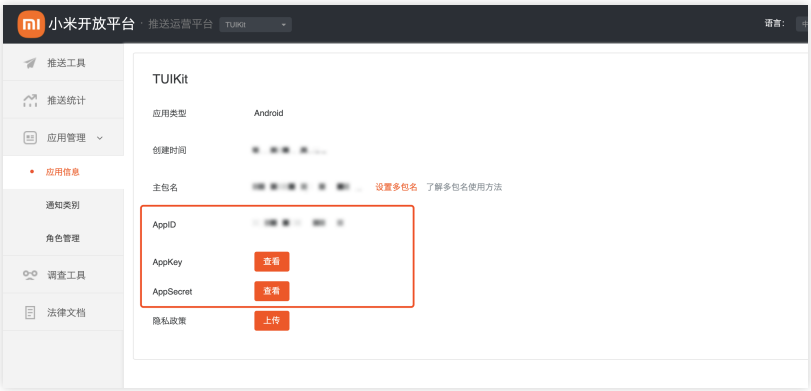
步骤1：注册应用到厂商推送平台

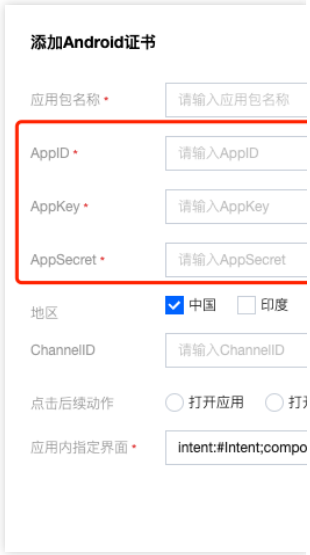
离线推送需要将您自己的应用注册到各个厂商的推送平台，得到 AppID 和 AppKey 等参数，来实现离线推送功能。
目前国内支持的手机厂商有：小米、华为、荣耀、OPPO、VIVO、魅族，境外支持 Google FCM。

步骤2：IM 控制台配置

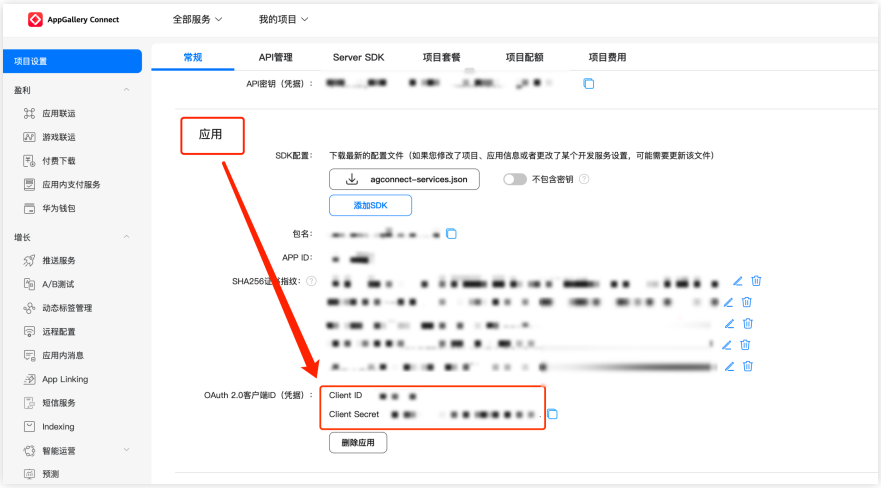
登录腾讯云 即时通信 IM 控制台 ，在**推送管理** > **接入设置**功能栏添加各个厂商推送证书，并将您在步骤一中获取的各厂商的 AppId、AppKey、AppSecret 等参数配置给添加的推送证书。
关于**点击后续动作**选项的说明：
打开应用：单击通知栏后拉起应用，默认启动应用的 Launcher 界面；
打开网页：单击通知栏会跳转到配置的网页链接；
打开应用内指定界面：单击通知栏会根据配置自定义跳转界面，详见 自定义点击跳转。
小米
华为
OPPO
vivo
魅族
荣耀
Google FCM

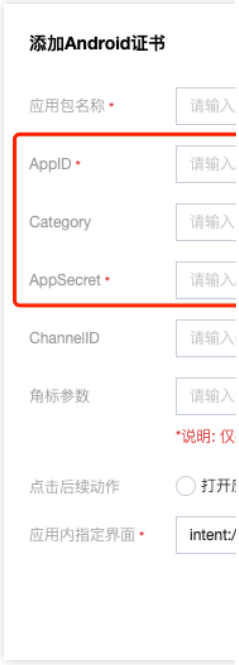
厂商推送平台	IM 控制台配置





厂商推送平台	IM 控制台配置

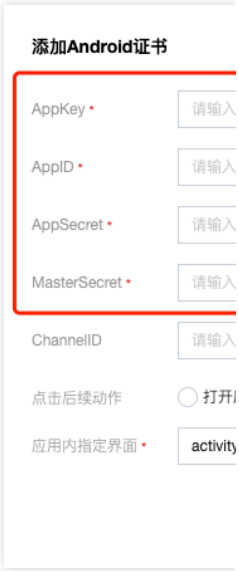




说明：

Client ID 对应 AppID，Cli

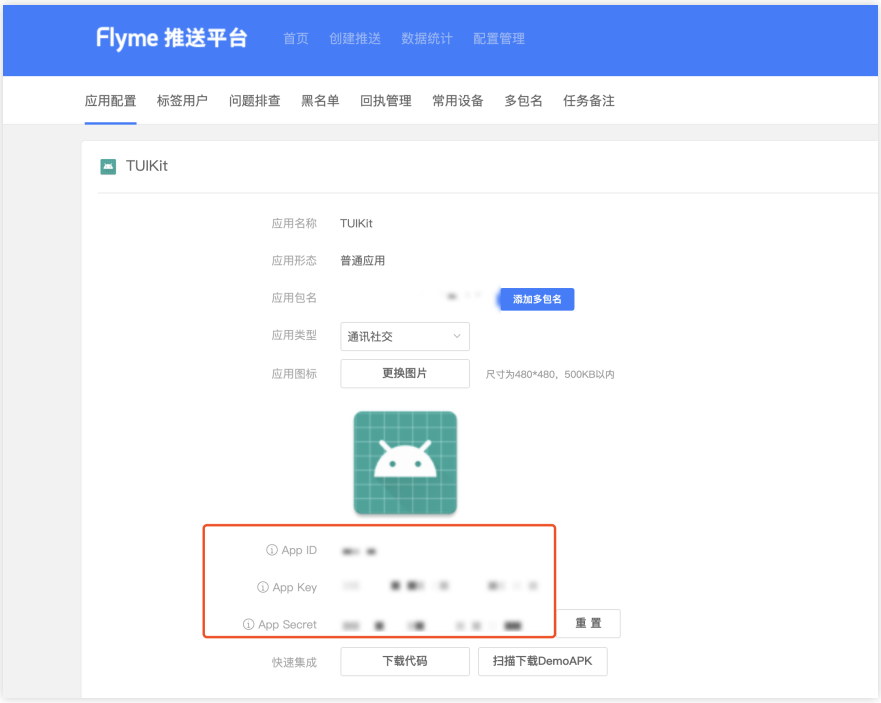
厂商推送平台	IM 控制台配置

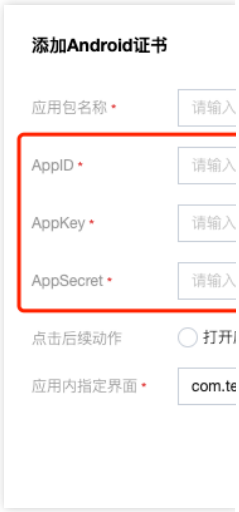


厂商推送平台	IM 控制台配置

回执配置请参考：消息触达统计配置-vivo

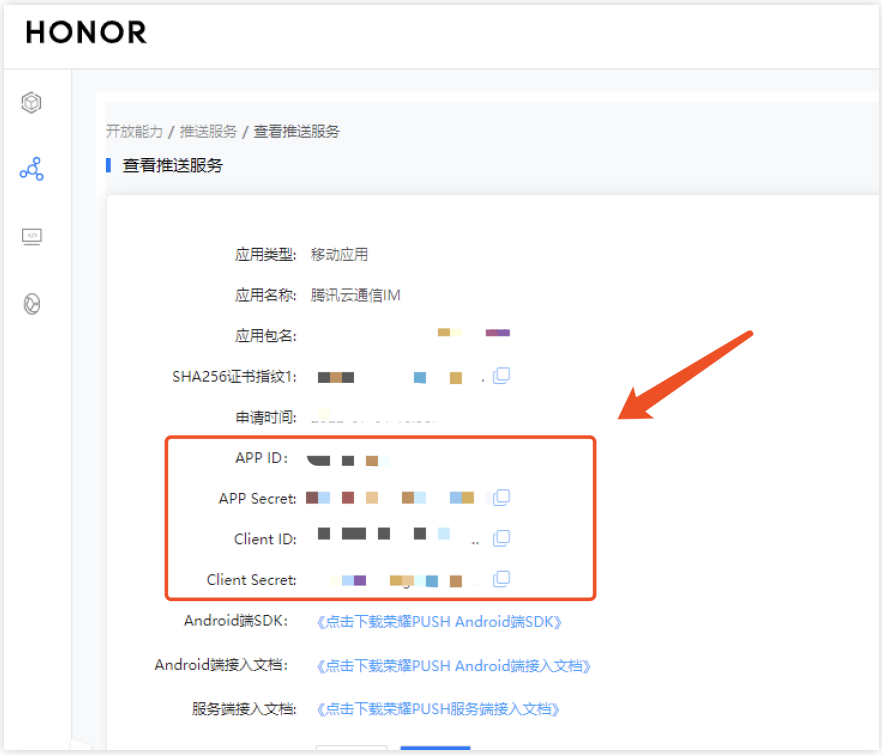
厂商推送平台	IM 控制台配置

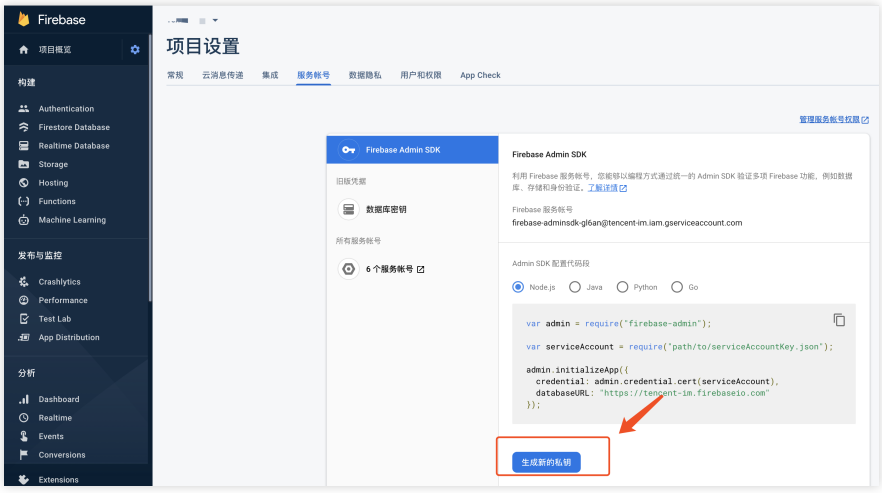




回执配置请参考：消息触达统计配置-魅族

厂商推送平台	IM 控制台配置
	



厂商推送平台	IM 控制台配置
	

注意：

关于**点击后续动作**支持上报统计功能：

1. 如果选择打开应用和打开网页，购买插件后会默认支持上报统计。
2. 如果选择打开应用内指定界面：
新增证书情况，请直接使用自动填写的默认值即可支持点击上报统计。
如果之前有证书且已配置，继续使用旧证书需要修改为默认值，才可以支持上报统计，或者重新生成新的证书。

关于 FCM 数据消息

FCM 提供两种推送方式是通知消息和数据消息。
通知消息，样式简单不区分设备，成功集成即可进行离线推送；
数据消息，样式定制丰富，特定设备有效，支持触达和点击上报，需要集成后在设备上做好测试开放上线。
控制台默认选择为通知消息，两种模式切换可在 IM 控制台操作：

注意：
FCM 数据消息能力仅支持 TIMPush 7.8 及以上版本的 pixel 手机，其他厂商设备需自测支持情况；

iOS

最近更新时间：2024-06-13 10:39:26

集成 TIMPush 组件之前，需要先向 Apple 申请 APNs 推送证书，然后上传推送证书到 IM 控制台 。之后按照快速接入步骤接入即可。

Apple 厂商配置目前有两种主流的证书，p12 证书和 p8 证书。两种证书各有优劣，您可按需要选择其中的一种。

证书类型：

p12 证书：p12 证书是一个包含公钥和私钥的二进制文件，用于基于证书的身份验证。它将公钥证书和私钥捆绑在一个文件中，扩展名为 .p12 或 .pfx。

p8 证书：p8 证书是一个 Auth Key（授权密钥），用于基于令牌的身份验证。它是一个包含私钥的文本文件，扩展名为 .p8。

有效期和管理：

p12 证书：p12 证书通常有一年的有效期，过期后需要重新生成和部署。每个应用程序都需要单独的 P12 证书来处理推送通知。

p8 证书：p8 证书没有到期日期，因此您无需担心证书过期。此外，使用 P8 证书可以简化证书管理，因为您可以使用一个 p8 证书为多个应用程序提供推送通知服务。

安全性：

p12 证书：p12 证书使用基于证书的身份验证，需要在服务器上存储私钥。这可能会增加安全风险，因为私钥可能会被未经授权的用户访问。

p8 证书：p8 证书使用基于令牌的身份验证，这意味着您的服务器会周期性地生成一个 JSON Web Token（JWT）来与 APNs 建立连接。这种方法更安全，因为它不需要在服务器上存储私钥。

灵动岛:

p12 证书：不支持。

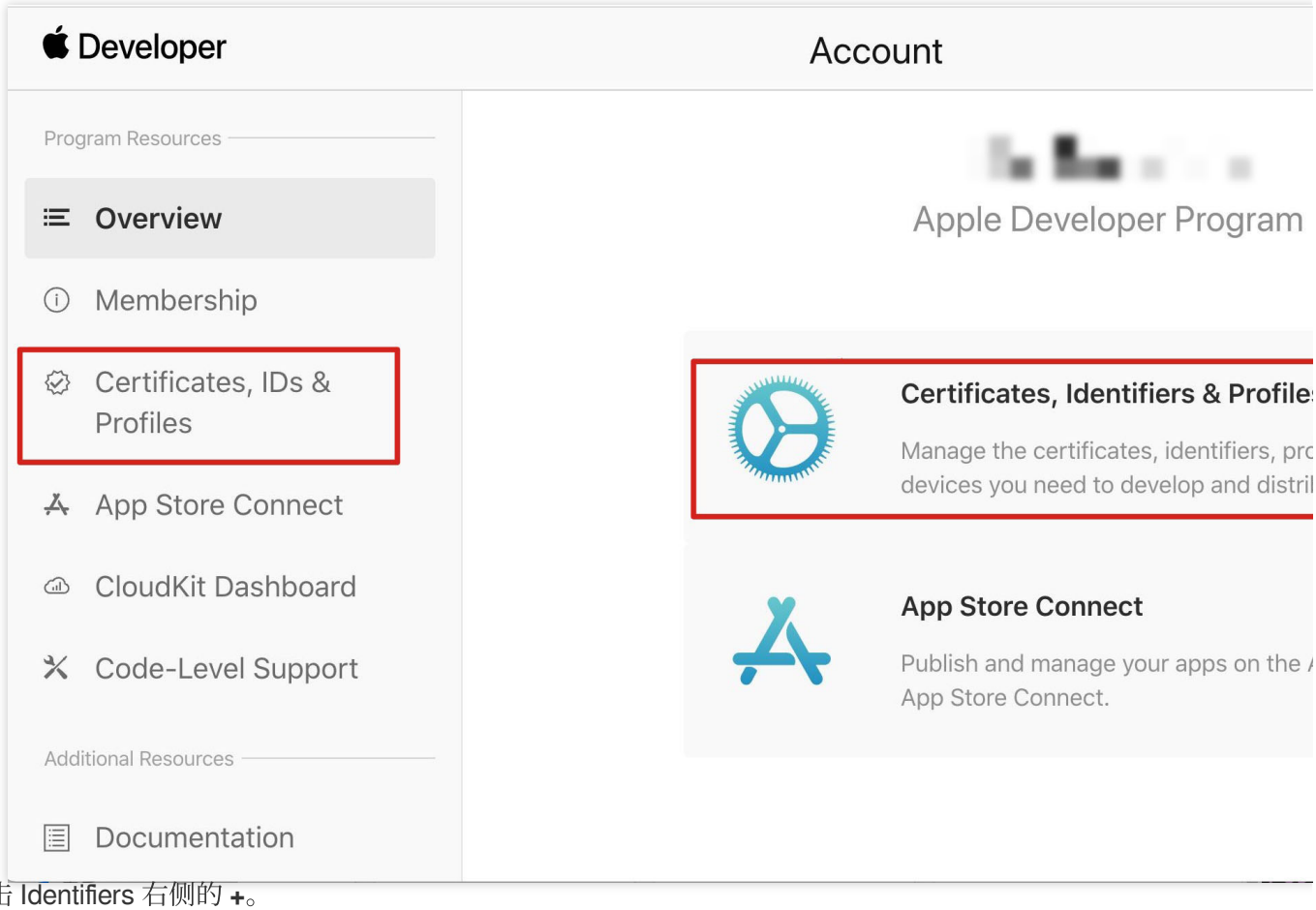
p8 证书：支持灵动岛推送。

一、使用 p12 证书 （传统推送证书）

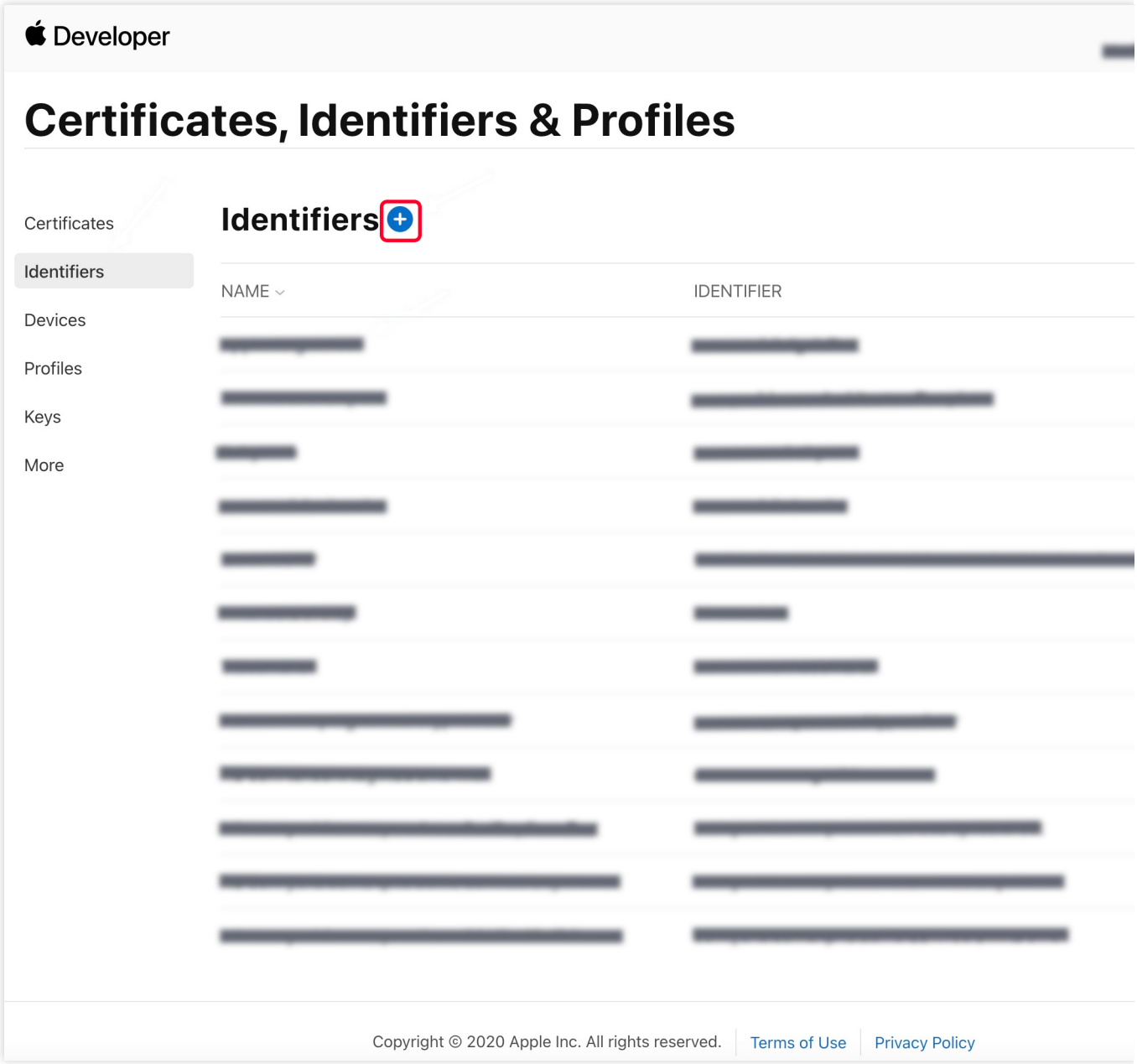
步骤1：申请 APNs 证书

开启 App 远程推送

1. 登录 苹果开发者中心 网站，单击 **Certificates,Identifiers & Profiles** 或者侧栏的 **Certificates,IDs & Profiles**，进入 **Certificates, IDS & Profiles** 页面。



2. 单击 Identifiers 右侧的 +。

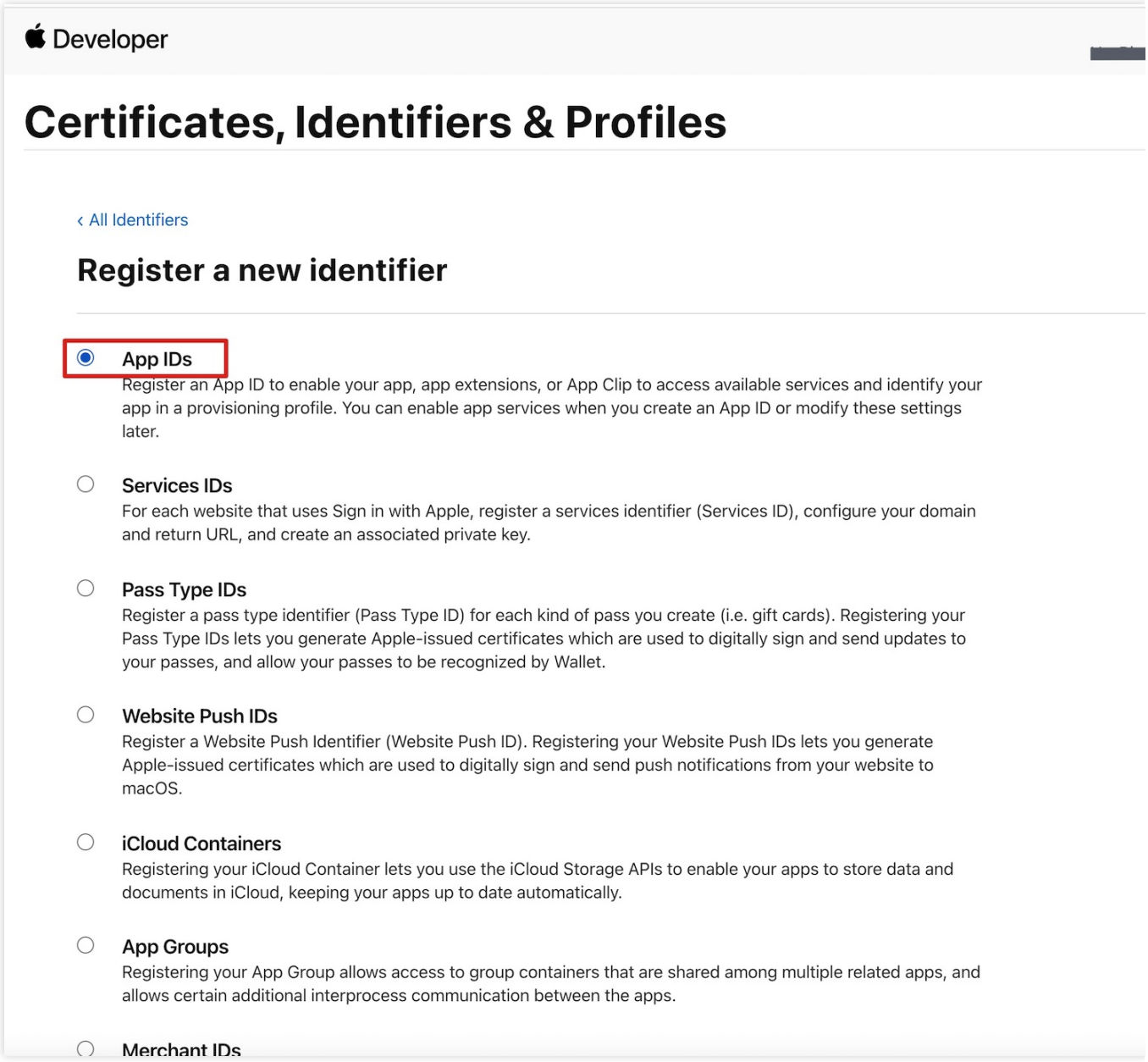


3. 您可以参见如下步骤新建一个 AppID，或者在您原有的 AppID 上增加 `Push Notification` 的 `Service` 。

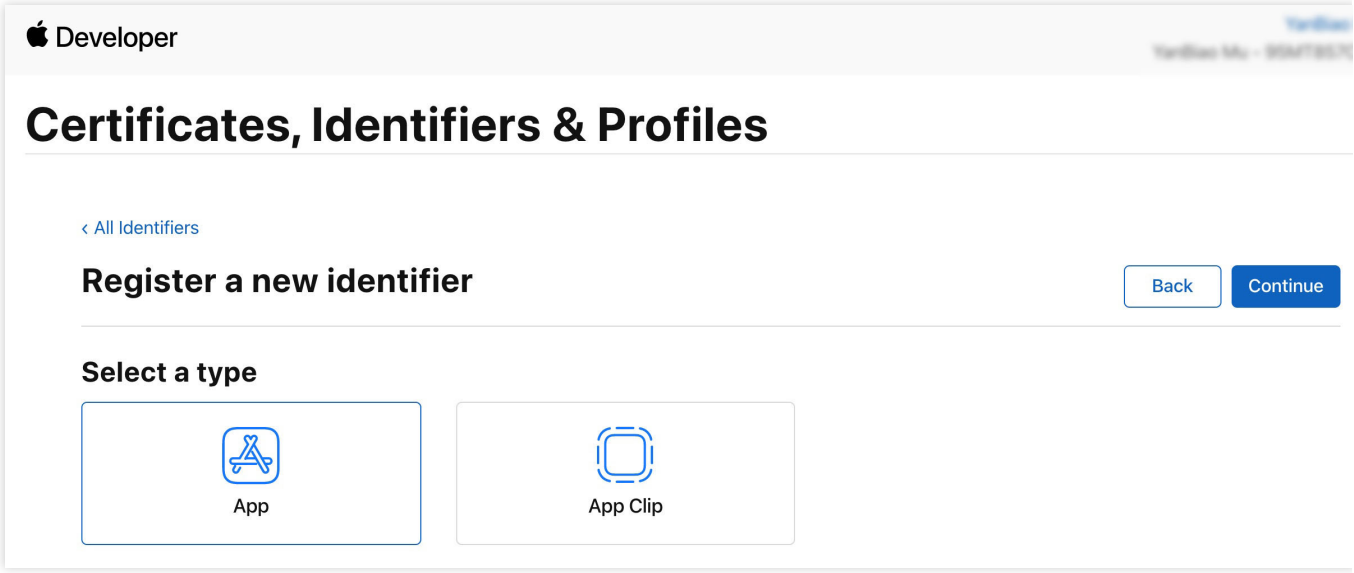
说明：

您 App 的 `Bundle ID` 不能使用通配符 `*` ，否则将无法使用远程推送服务。

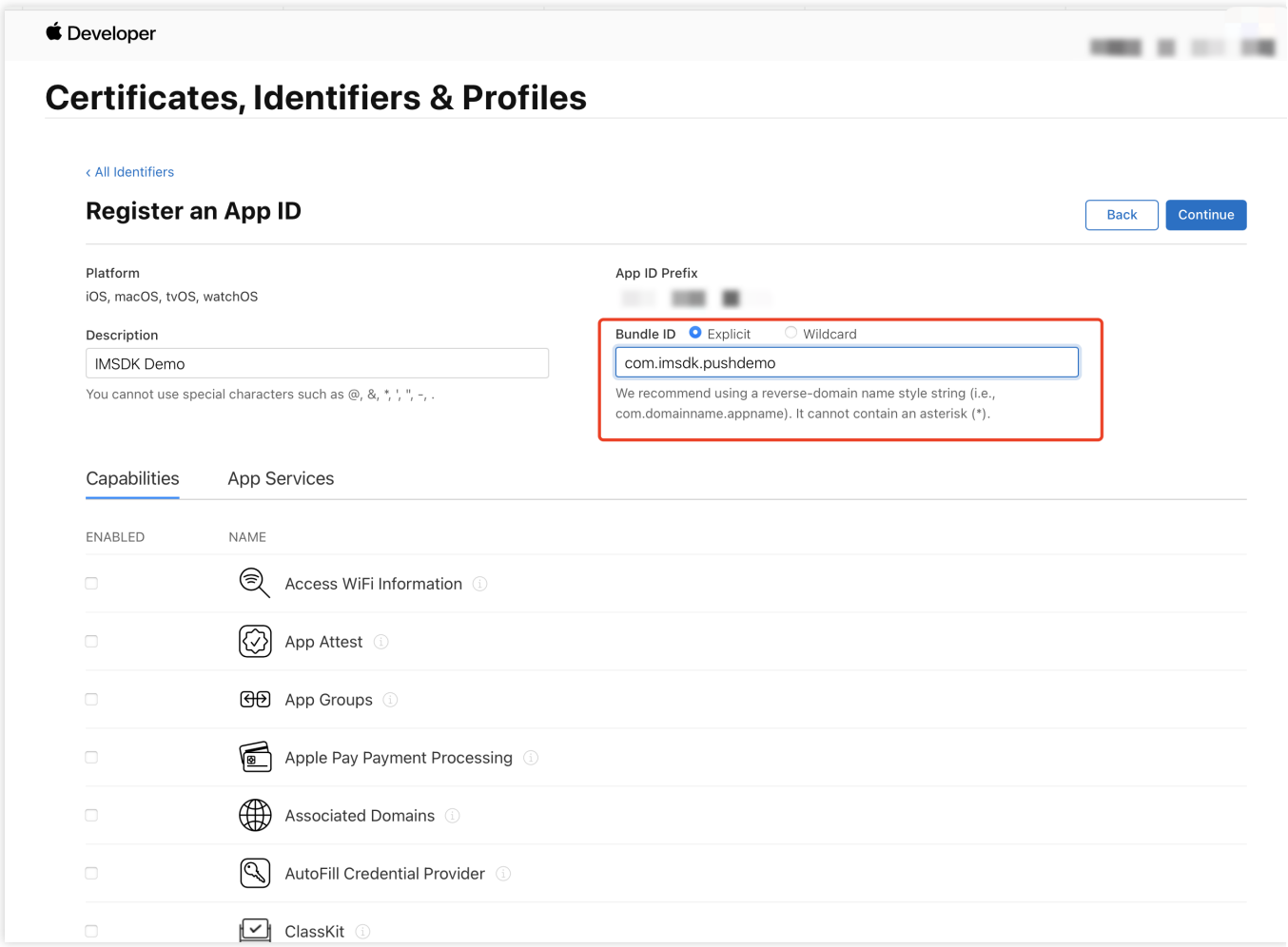
4. 勾选 **App IDs**，单击 **Continue** 进行下一步。



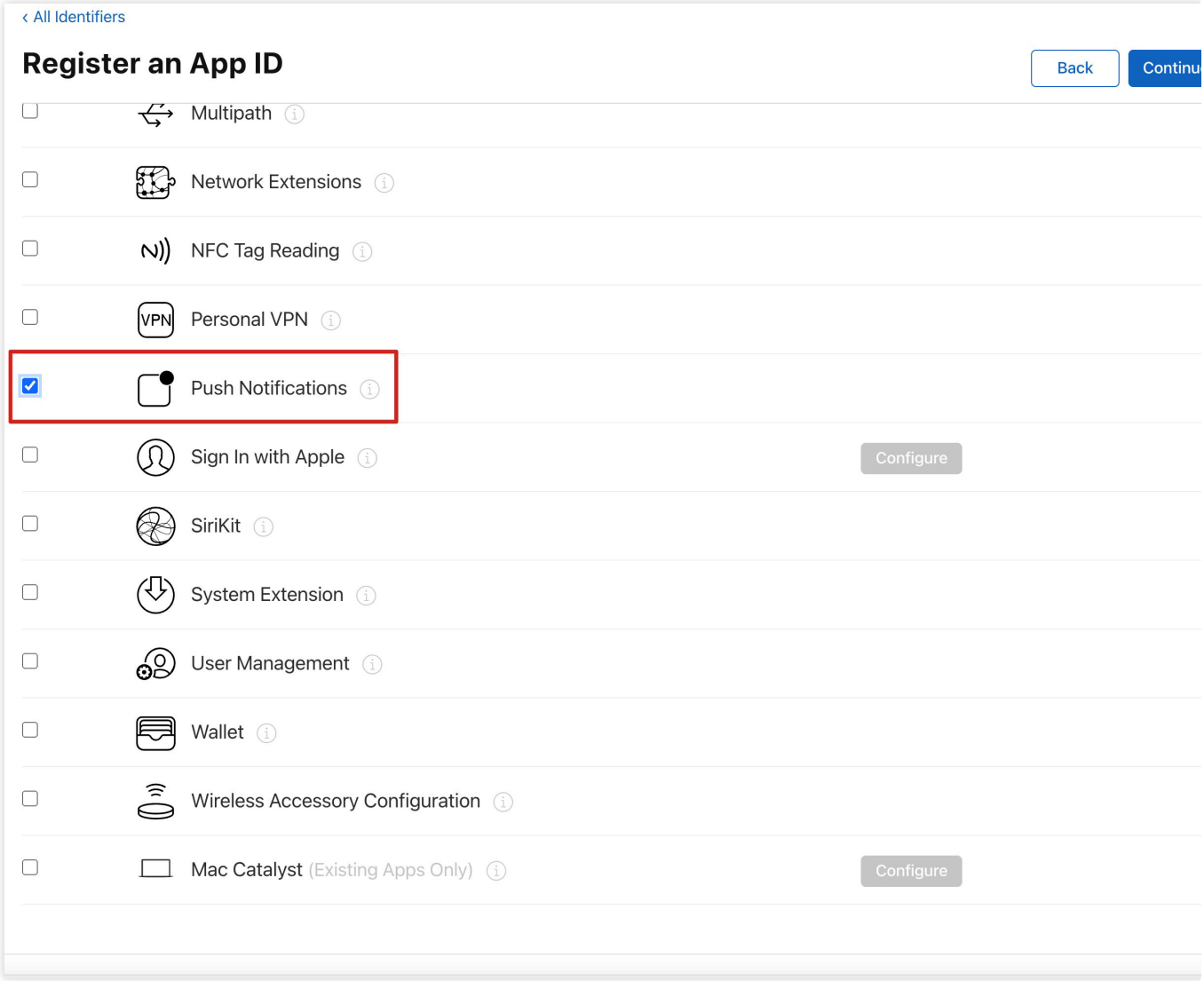
5. 选择 **App**，单击 **Continue** 进行下一步。



6. 配置 `Bundle ID` 等其他信息，单击 **Continue** 进行下一步。

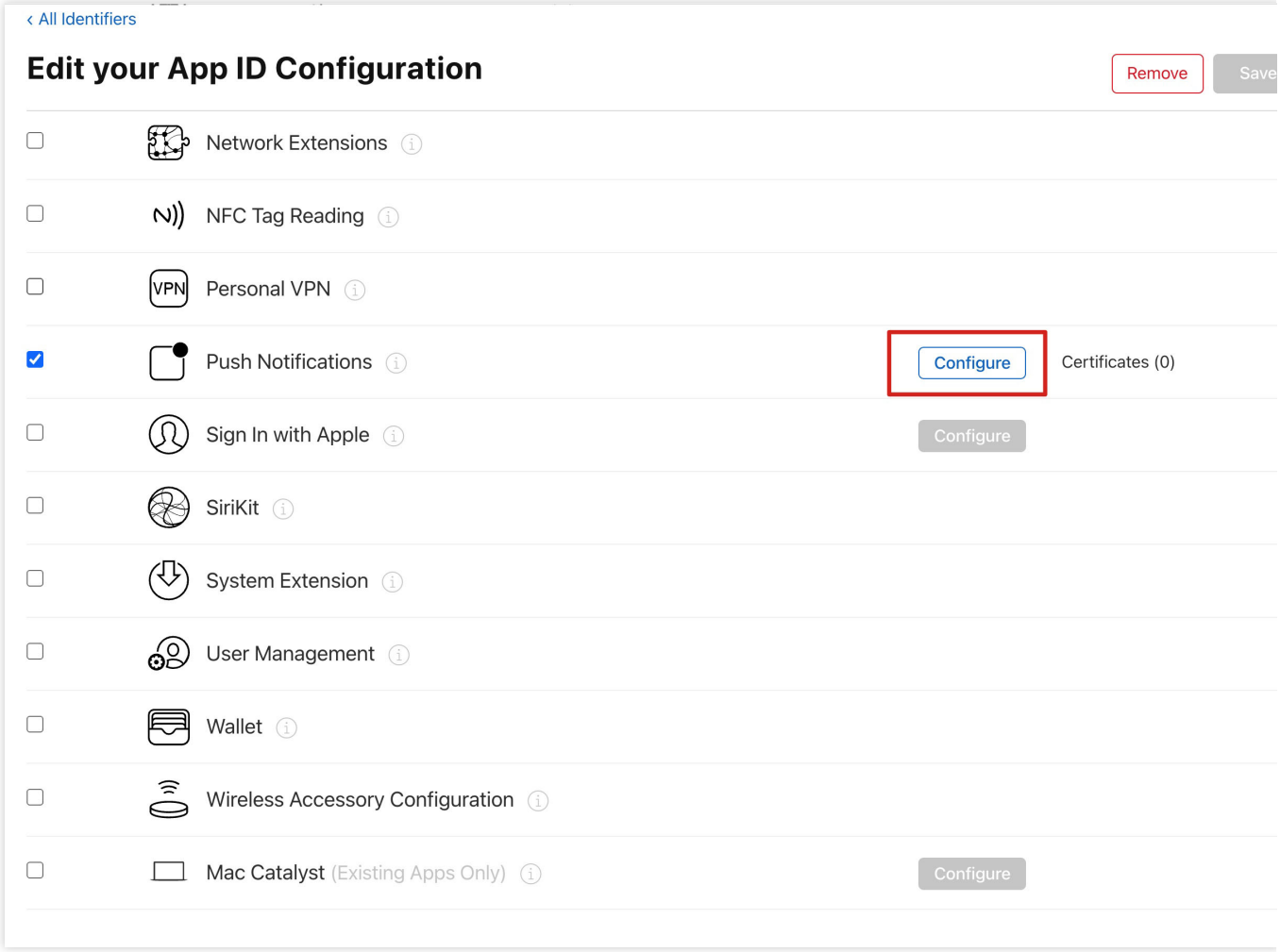


7. 勾选 **Push Notifications**，开启远程推送服务。

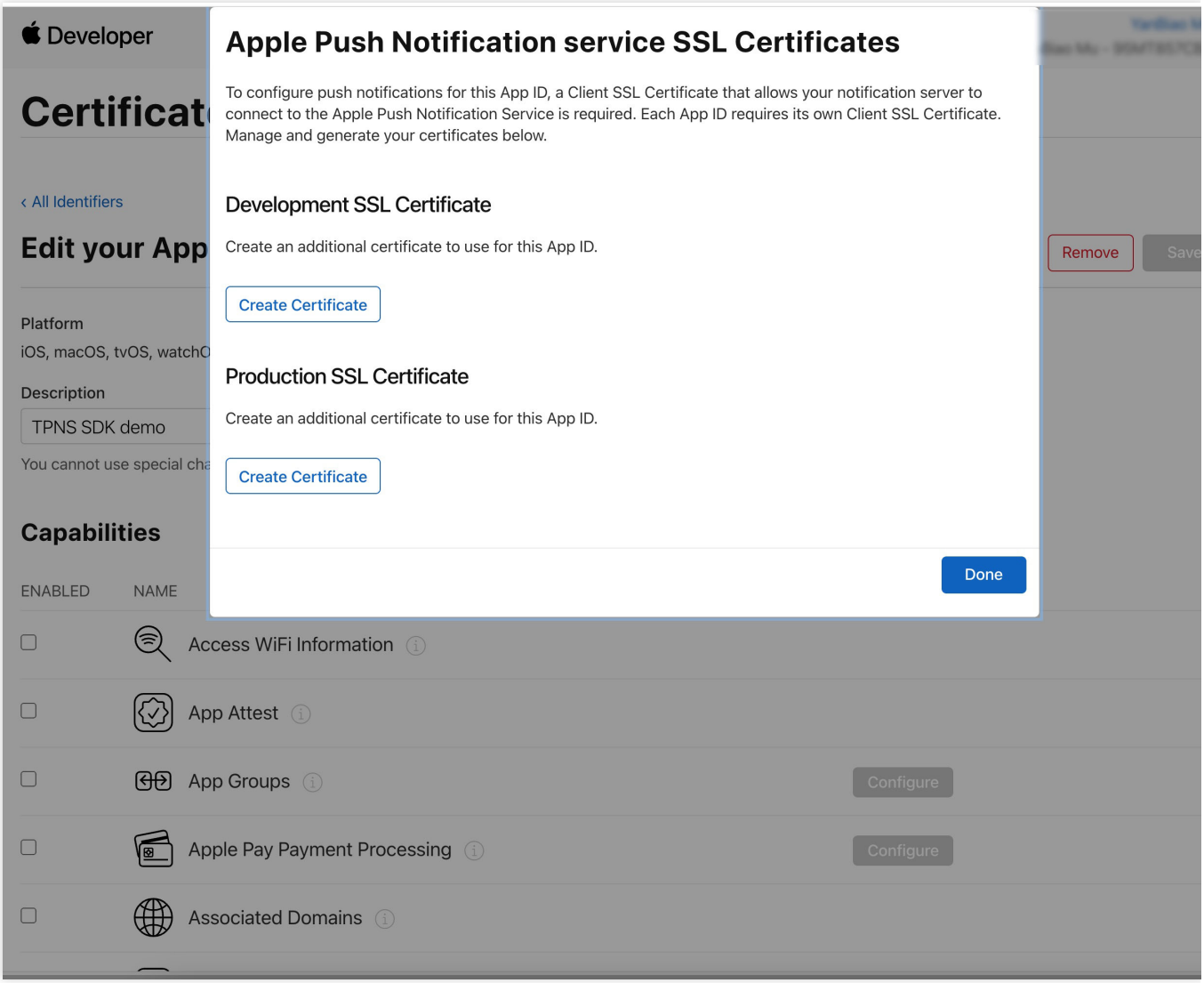


生成证书

1. 选中您的 AppID，选择 **Configure**。



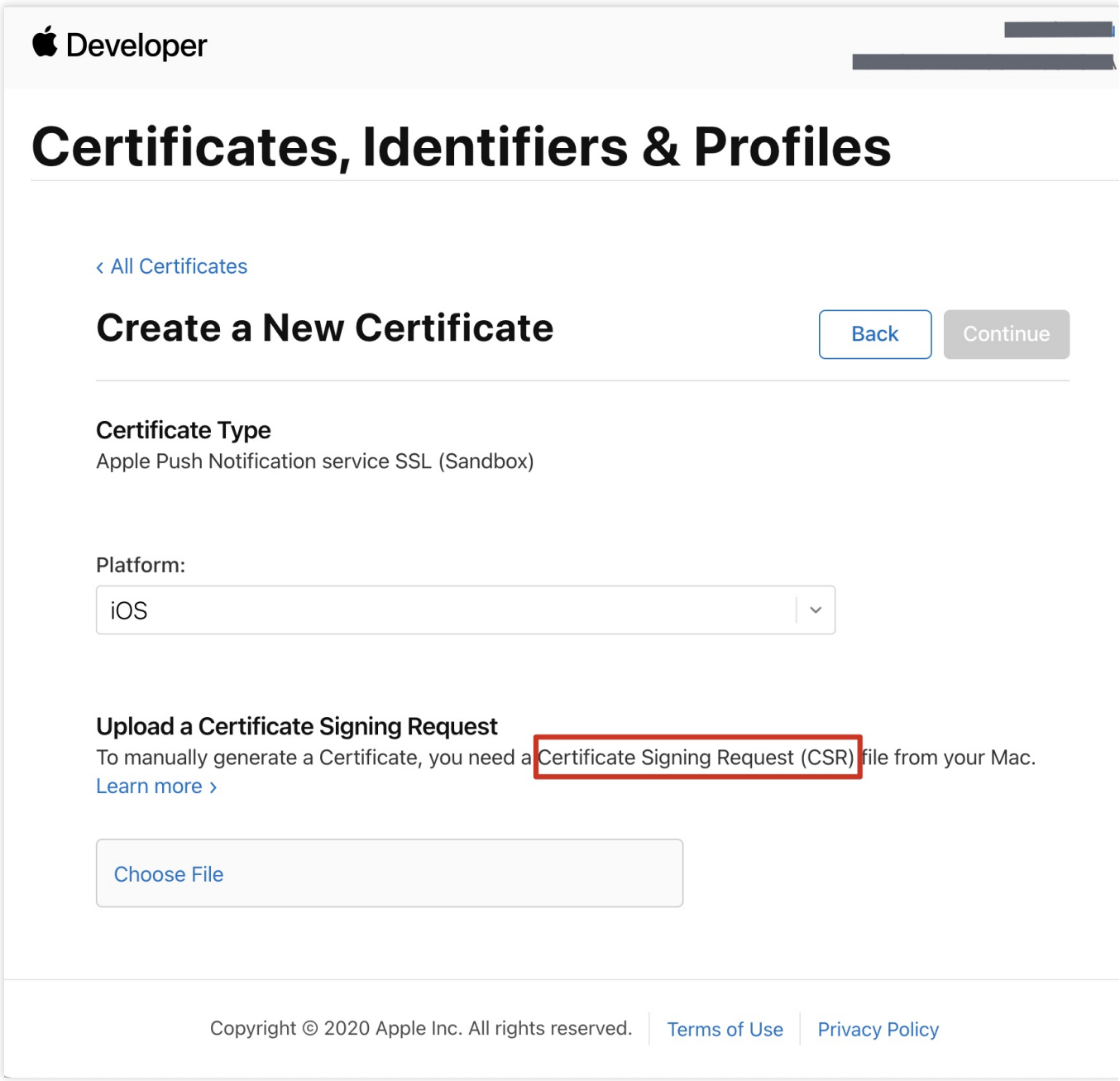
2. 可以看到在 **Apple Push Notification service SSL Certificates** 窗口中有两个 `SSL Certificate`，分别用于开发环境（Development）和生产环境（Production）的远程推送证书，如下图所示：



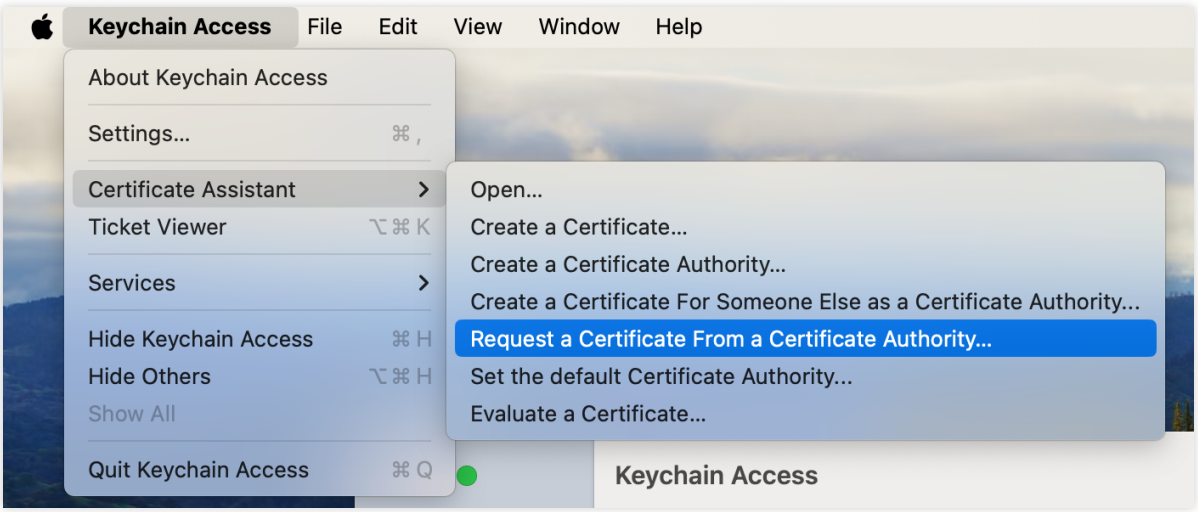
3.

我

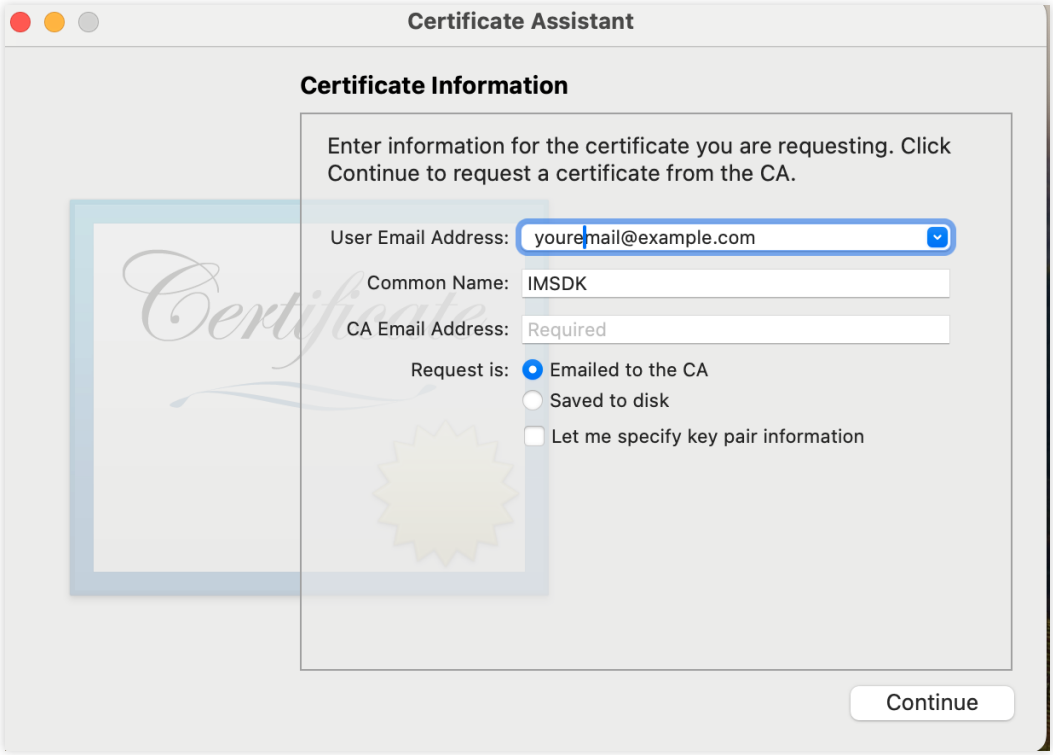
们先选择开发环境（Development）的 **Create Certificate**，系统将提示我们需要一个 Certificate Signing Request（CSR）。



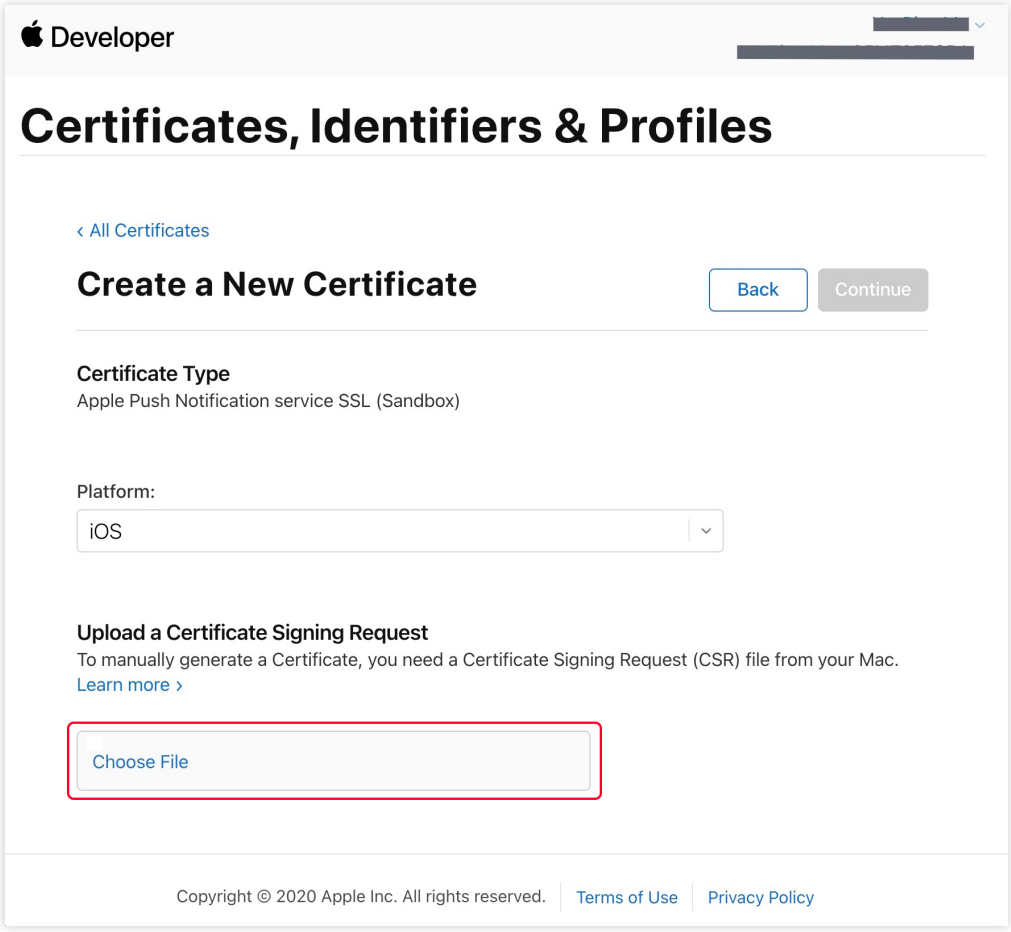
4. 在 Mac 上打开**钥匙串访问工具（Keychain Access）**，在菜单中选择**钥匙串访问 > 证书助理 > 从证书颁发机构请求证书**（`Keychain Access - Certificate Assistant - Request a Certificate From a Certificate Authority`）。



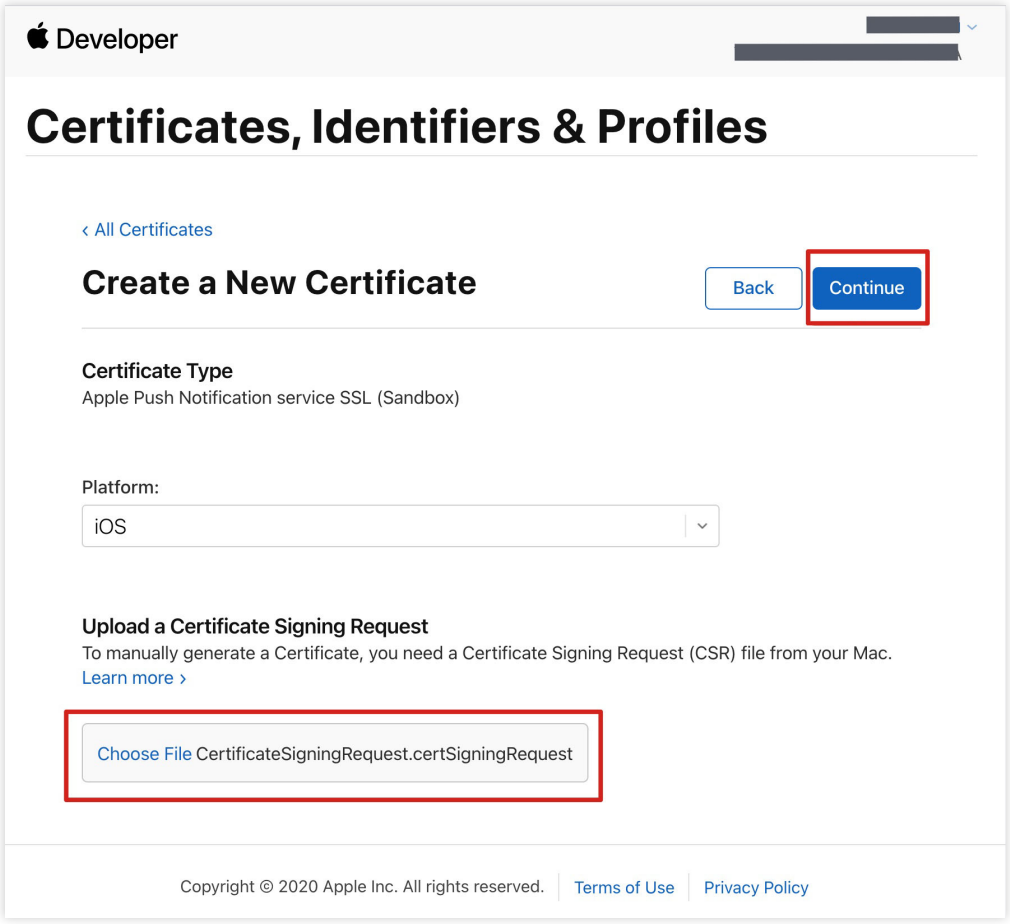
5. 输入用户电子邮件地址（您的邮箱）、常用名称（您的名称或公司名），选择**存储到磁盘**，单击**继续**，系统将生成一个 `*.certSigningRequest` 文件。



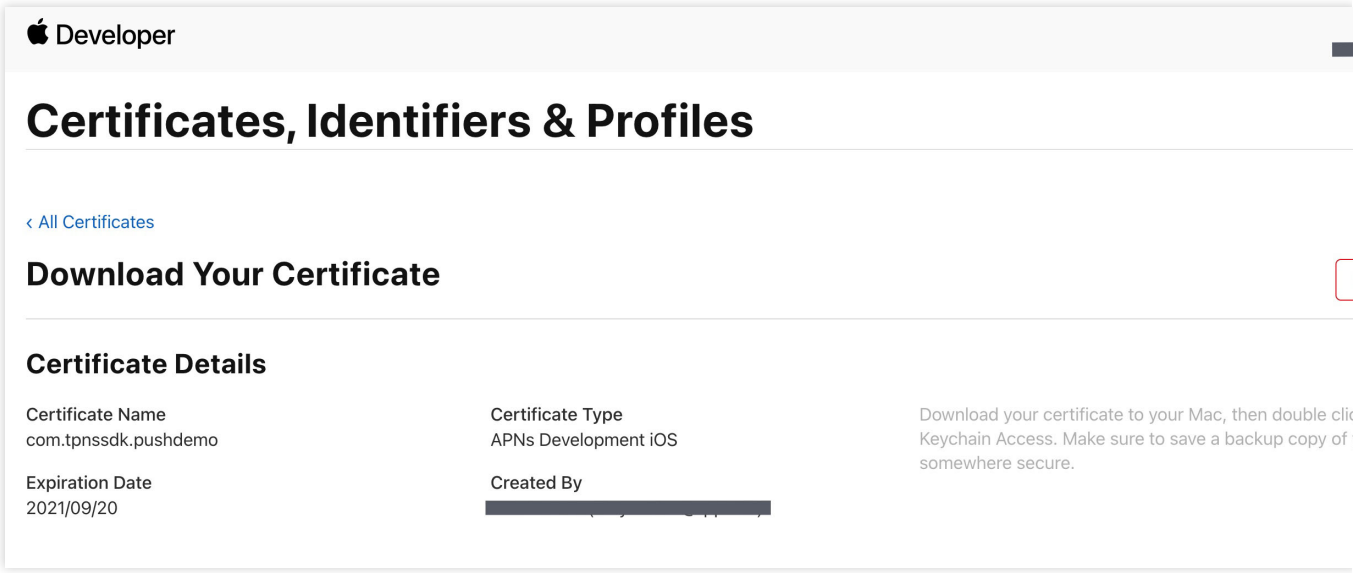
6. 返回上述 步骤3 中 Apple Developer 网站刚才的页面，单击 **Choose File** 上传生成的 `*.certSigningRequest` 文件。



7. 单击 **Continue**，即可生成推送证书。



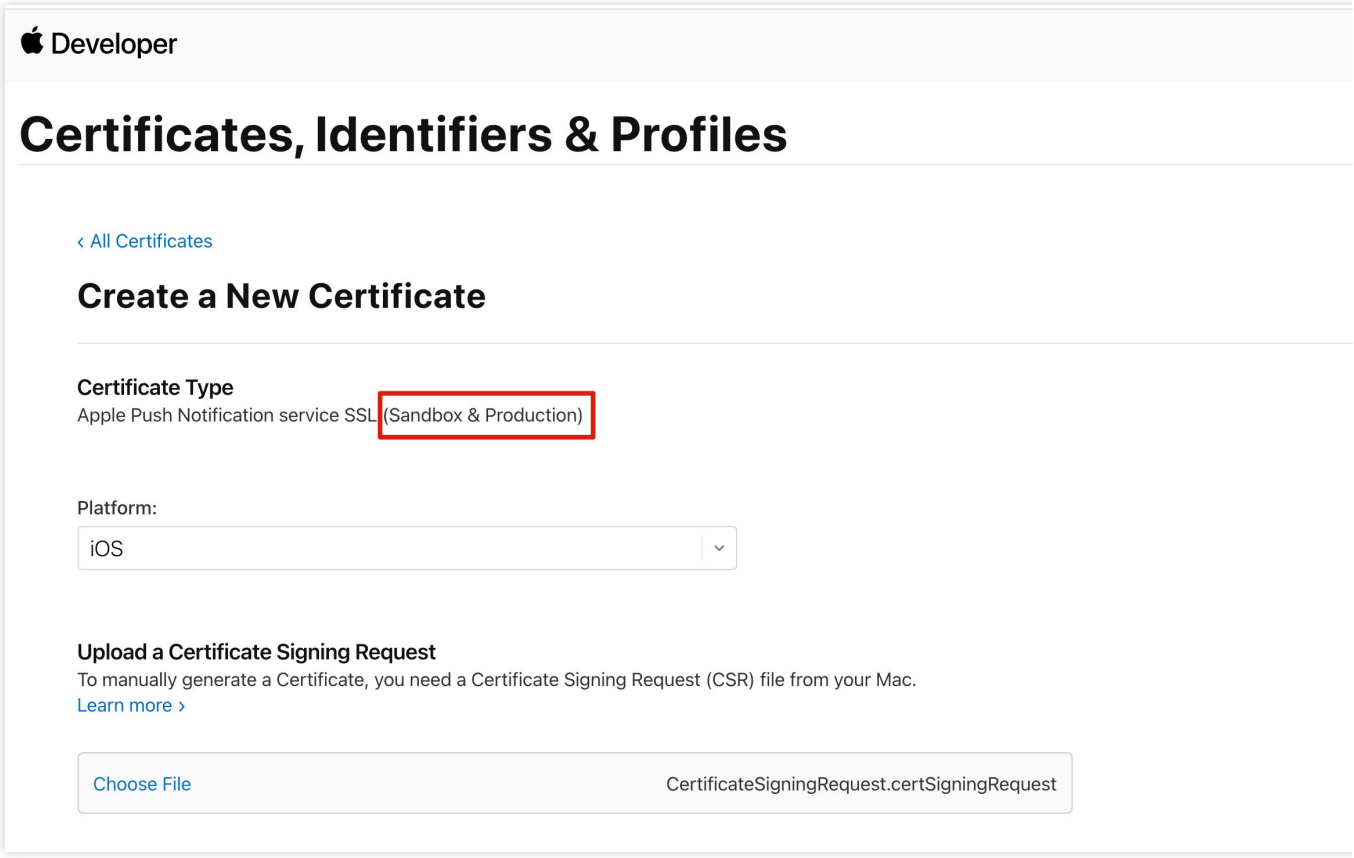
8. 单击 **Download** 下载开发环境的 `Development SSL Certificate` 到本地。

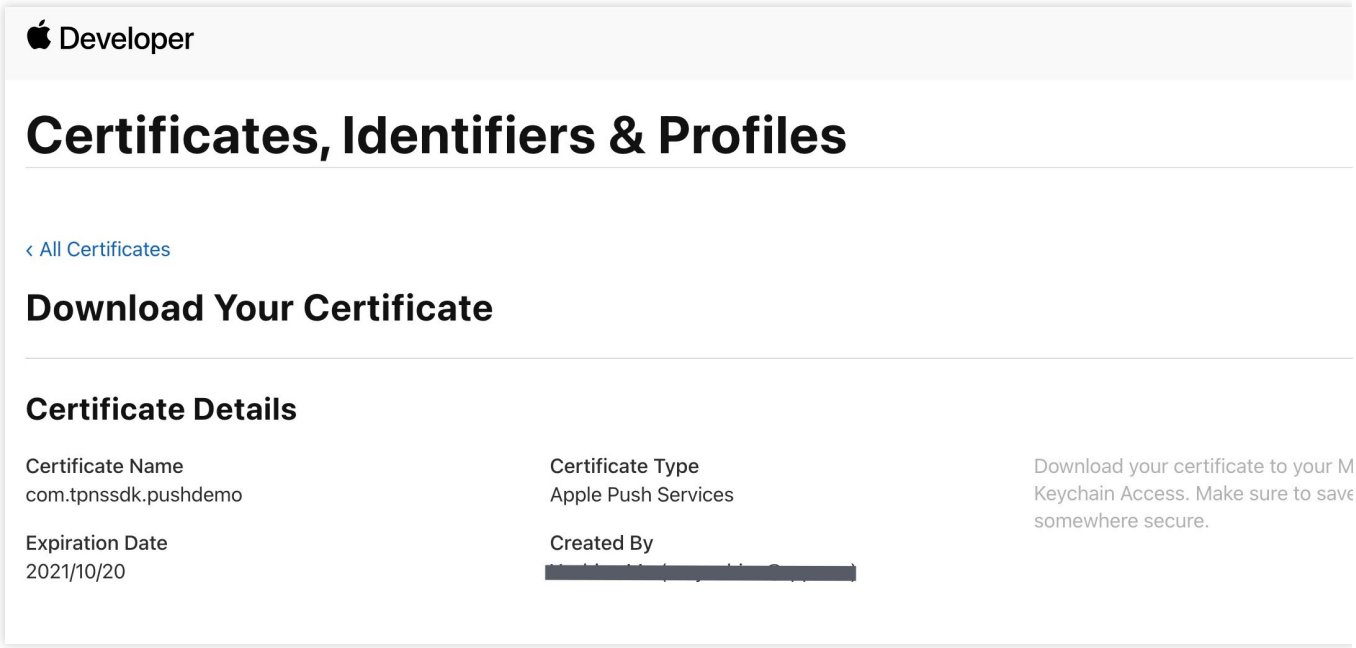


9. 再次按照上述步骤1 - 8，将生产环境的 `Production SSL Certificate` 下载到本地。

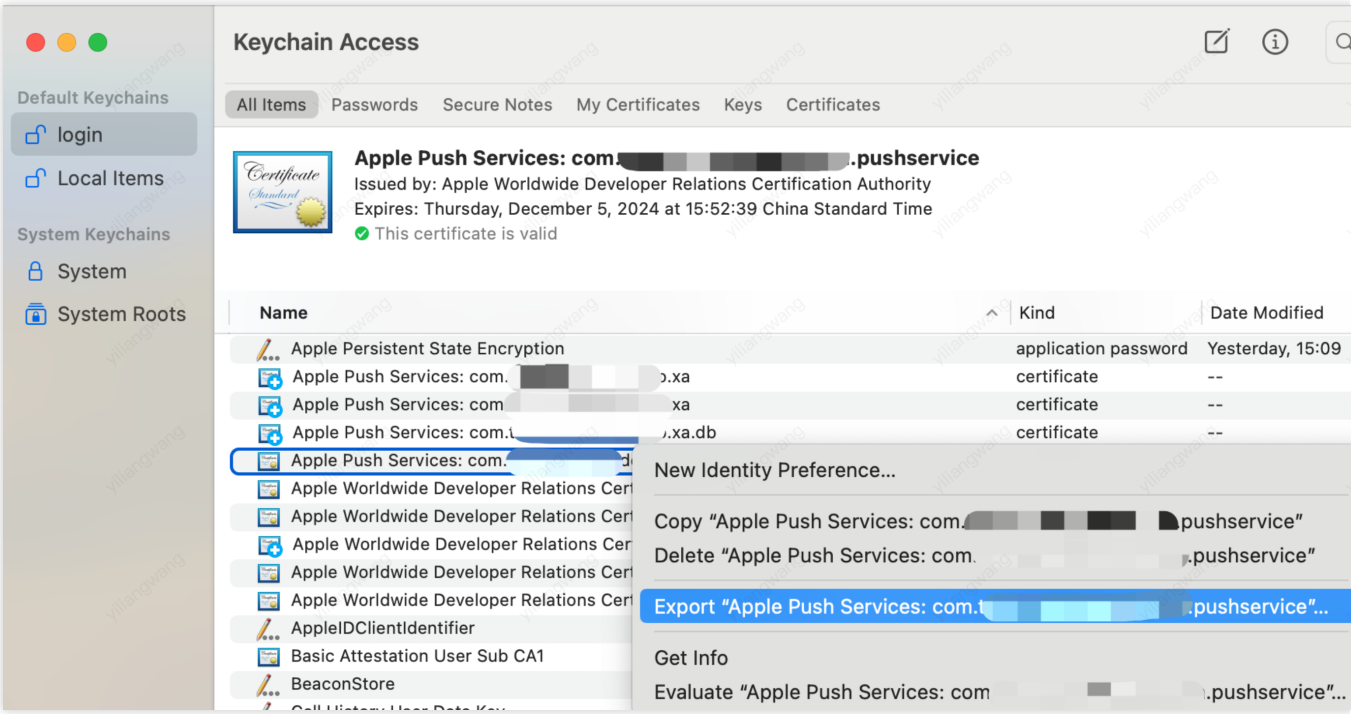
说明

生产环境的证书实际是开发（Sandbox）+生产（Production）的合并证书，可以同时作为开发环境和生产环境的证书使用。





10. 双击打开下载的开发环境和生产环境的 `SSL Certificate`，系统会将其导入钥匙串中。
11. 打开钥匙串应用，在**登录 > 我的证书**，右键分别导出刚创建的开发环境（ `Apple Development IOS Push Service` ）和生产环境（ `Apple Push Services` ）的 `p12` 文件。

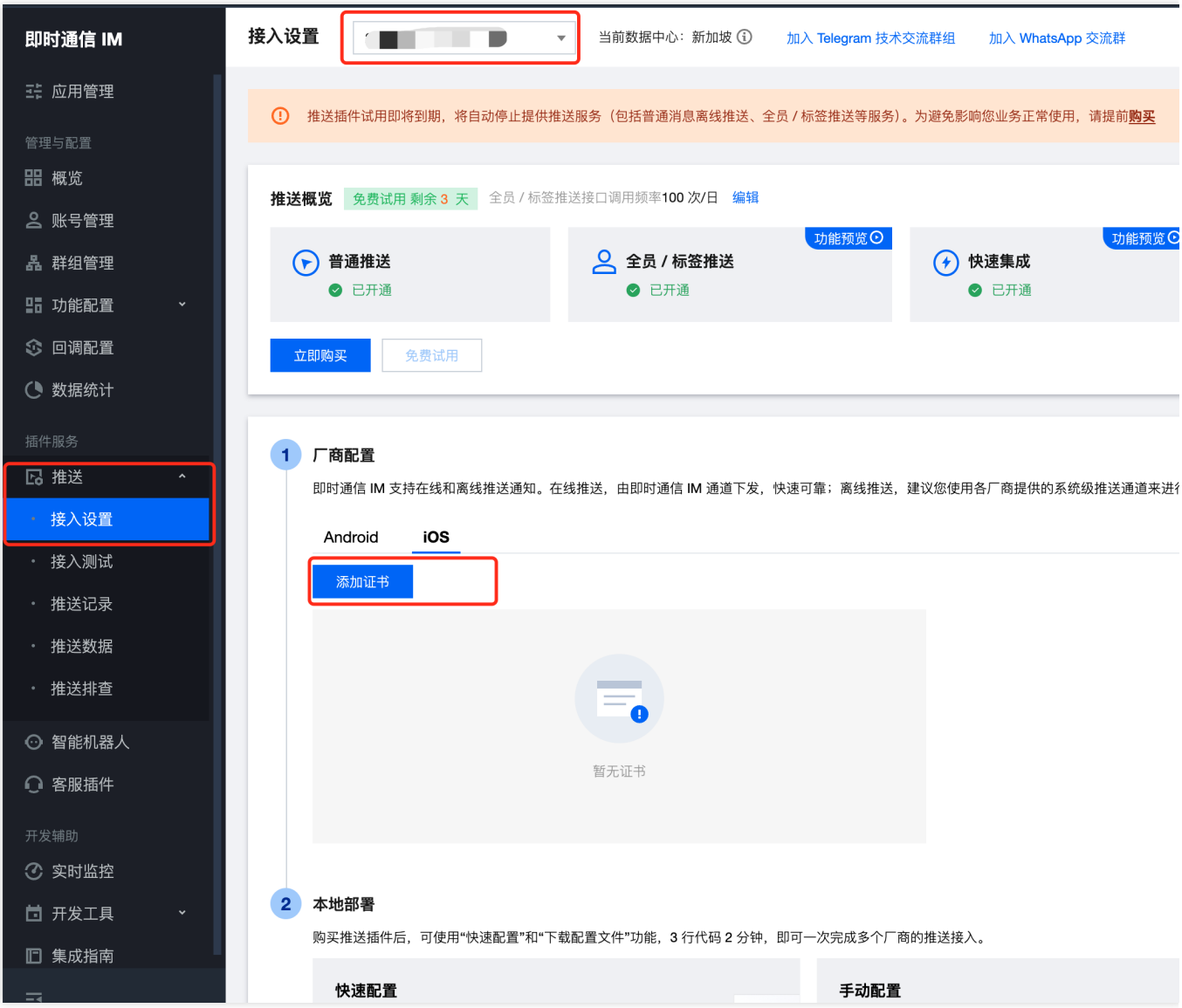


注意

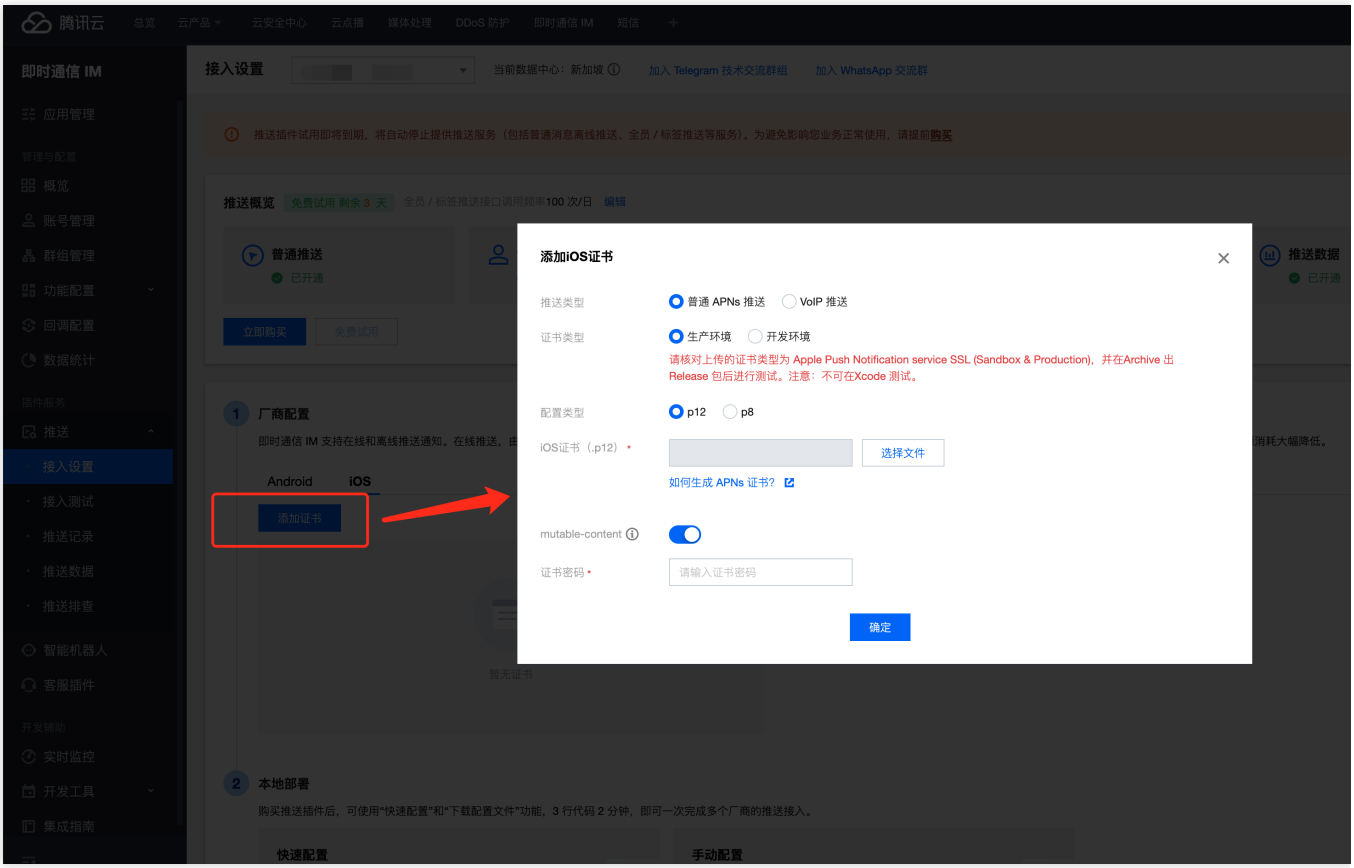
保存 `.p12` 文件时，请务必要为其设置密码。

步骤2：上传证书到控制台

1. 登录 即时通信 IM 控制台。
2. 单击插件服务-推送-接入设置，进入接入设置页面。



3. 单击厂商配置中的 **iOS** 底部的**添加证书**。
4. 选择证书类型，上传 iOS 证书（p.12），设置证书密码，单击**确定**。



注意：

上传证书名最好使用全英文（尤其不能使用括号等特殊字符）。

上传证书需要设置密码，无密码收不到推送。

发布 App Store 的证书需要设置为生产环境，否则无法收到推送。

上传的 p12 证书必须是自己申请的真实有效的证书。

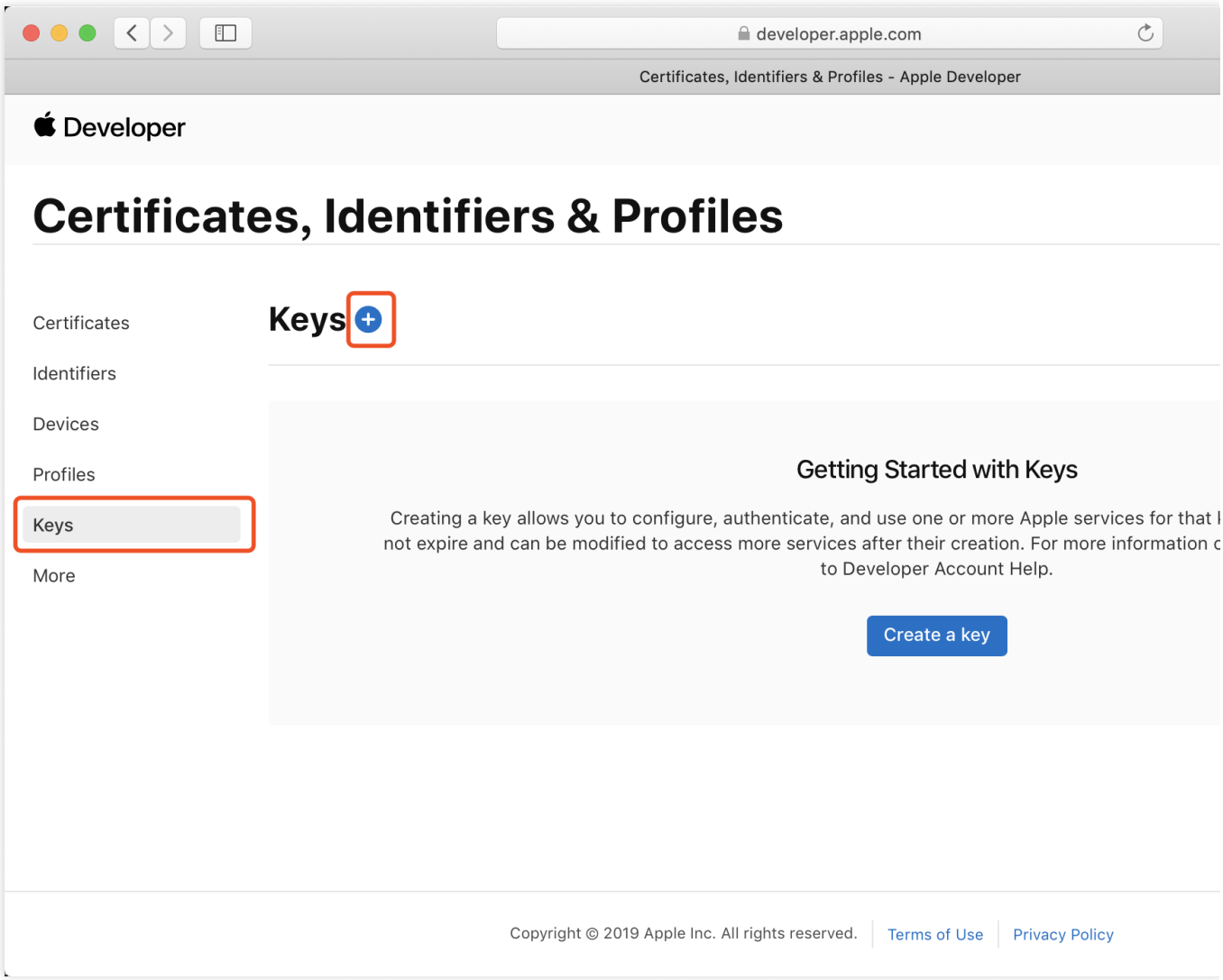
5. 待推送证书信息生成后，记录证书的 ID。

二、使用 p8 证书 （支持灵动岛推送）

p8 证书：p8 证书没有到期日期，因此您无需担心证书过期。此外，使用 p8 证书可以简化证书管理，因为您可以使用一个p8 证书为多个应用程序提供推送通知服务。另外，p8 证书支持灵动岛推送。

步骤1：申请 APNs 证书

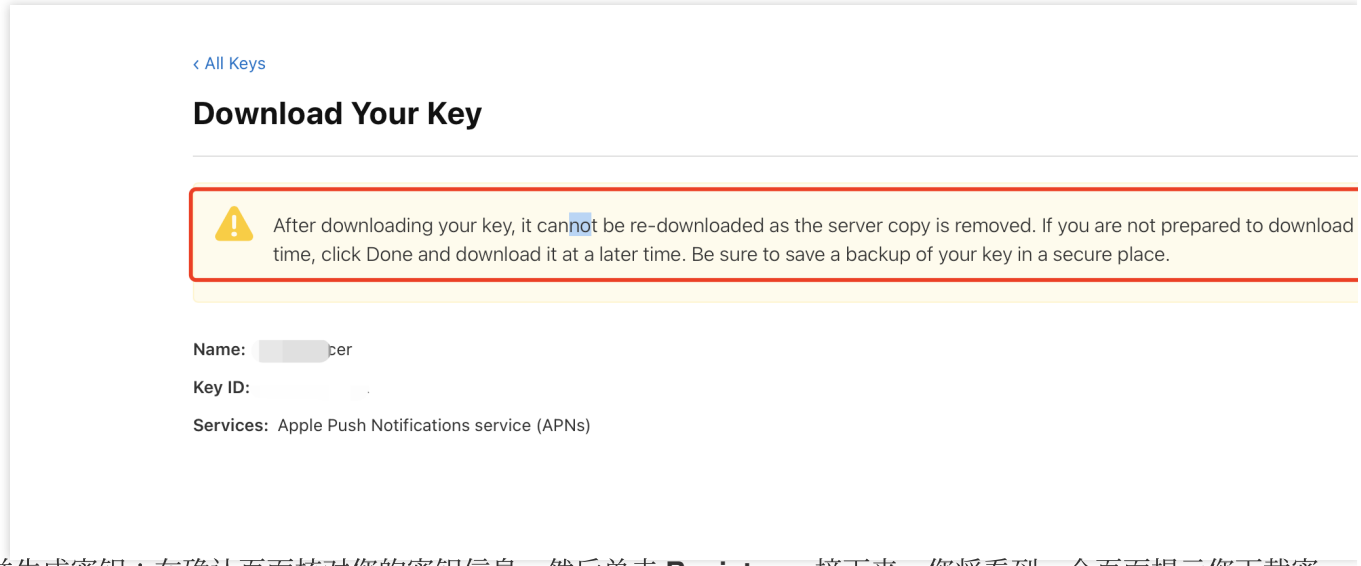
要创建 p8 证书文件，首先需要登录 苹果开发者中心。



1. 进入Certificates，Identifiers & Profiles：在页面右上角单击 **Account**，然后在下拉菜单中选择 **Certificates, Identifiers & Profiles** 。

2. 创建一个新的 App ID：在左侧菜单中单击 **Identifiers** ，然后单击右侧的 **+** 创建一个新的 App ID。填写相应的信息并单击 **Continue** 。

3. 创建一个新的密钥：在左侧菜单中单击 **Keys**，然后单击右侧的 **+** 创建一个新的密钥。输入密钥的名称，然后勾选 **Apple Push Notifications service (APNs)**，单击 **Continue** 。



确认并生成密钥：在确认页面核对您的密钥信息，然后单击 **Register** 。接下来，您将看到一个页面提示您下载密钥。单击 **Download**，将生成的 .p8 文件保存到您的计算机上。

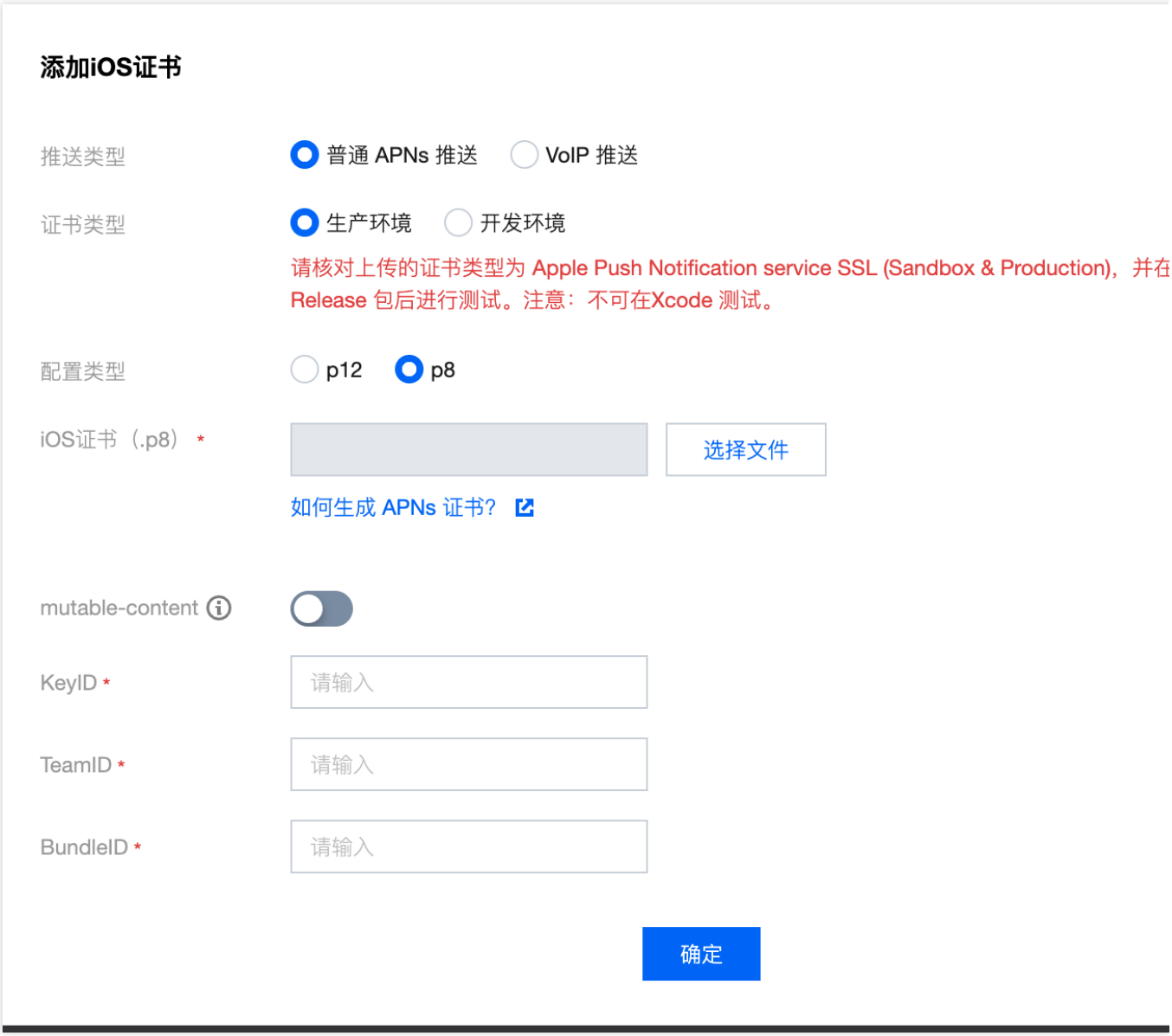
注意：

p8 证书只可以下载一次，请妥善保存。

请妥善保管下载的 p8 文件，因为您将无法再次下载该文件。您可以使用此P8证书配置您的iOS应用程序以接收推送通知。

步骤2：上传 p8 证书到IM控制台

1. 登录 即时通信 IM 控制台。
2. 单击插件服务-推送-接入设置，进入接入设置页面。
3. 单击厂商配置中的 **iOS** 底部的**添加证书**。
4. 选择 .p8 证书



说明：

Key ID：这是您的 APNs Auth Key 的唯一标识符。当您在 Apple Developer Center 创建一个新的 APNs Auth Key 时，系统会为您生成一个 Key ID。您可以在 "Certificates, Identifiers & Profiles" 部分的 "Keys" 中找到它。

Team ID：这是您的开发者账户的唯一标识符。您可以在 Apple Developer Center 的账户详情页面找到它。点击右上角的 "Membership"，在 "Membership Details" 部分可以找到您的 Team ID。

Bundle ID：这是您的应用程序的唯一标识符，也称为应用程序 ID。您可以在 Apple Developer Center 的 "Certificates, Identifiers & Profiles" 部分找到它。选择 "Identifiers"，然后在您的应用程序列表中找到对应的 Bundle ID。

uni-app

最近更新时间：2024-06-13 10:39:26

TencentCloud-TIMPush 是腾讯云即时通信 IM Push 插件。目前推送支持小米、华为、荣耀、OPPO、vivo、魅族、APNs、一加、realme、iQOO 和 苹果等厂商通道。

Android

iOS

注册应用到厂商推送平台

离线推送需要将您自己的应用注册到各个厂商的推送平台，得到 AppID 和 AppKey 等参数，来实现离线推送功能。

目前国内支持的手机厂商有：小米、华为、荣耀、OPPO、VIVO、魅族，境外支持 Google FCM。

小米

华为

OPPO

vivo

魅族

荣耀

Google FCM

步骤1：注册小米开发者账号

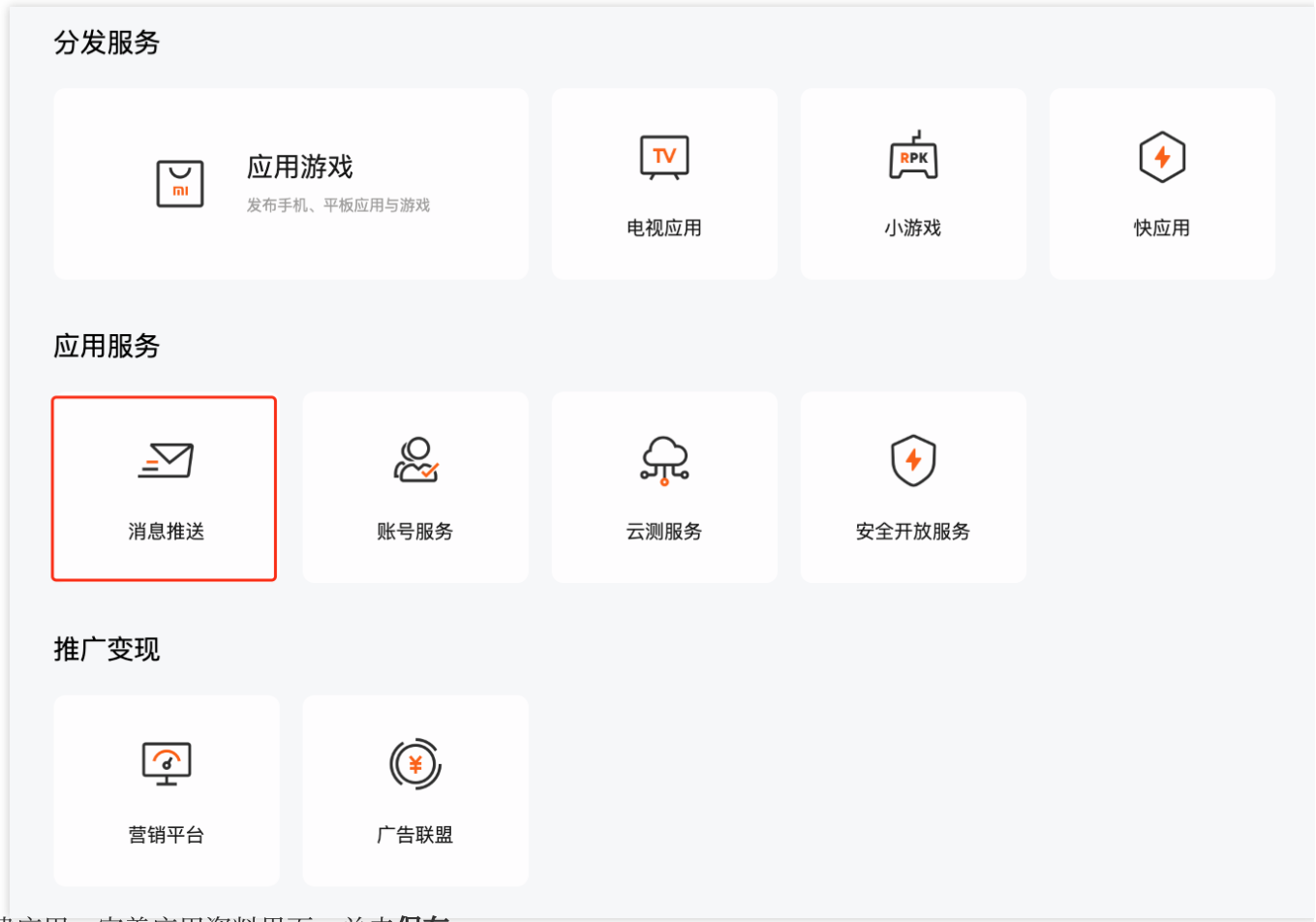
进入 小米开放平台，注册小米开发者账号，详情请参见 企业开发者账号注册流程。

步骤2：创建应用

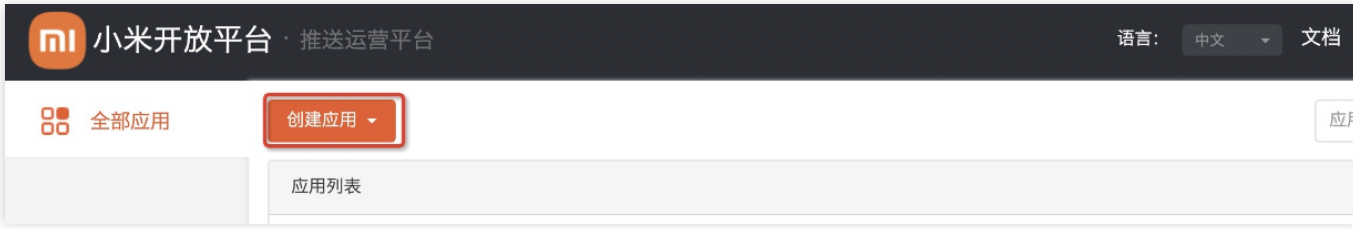
1. 在管理控制台单击**消息推送**。

注意：

若使用个人账号登录，会显示“抱歉，您当前没有推送/审核权限”。



2. 创建应用，完善应用资料界面，单击**保存**。



注意：

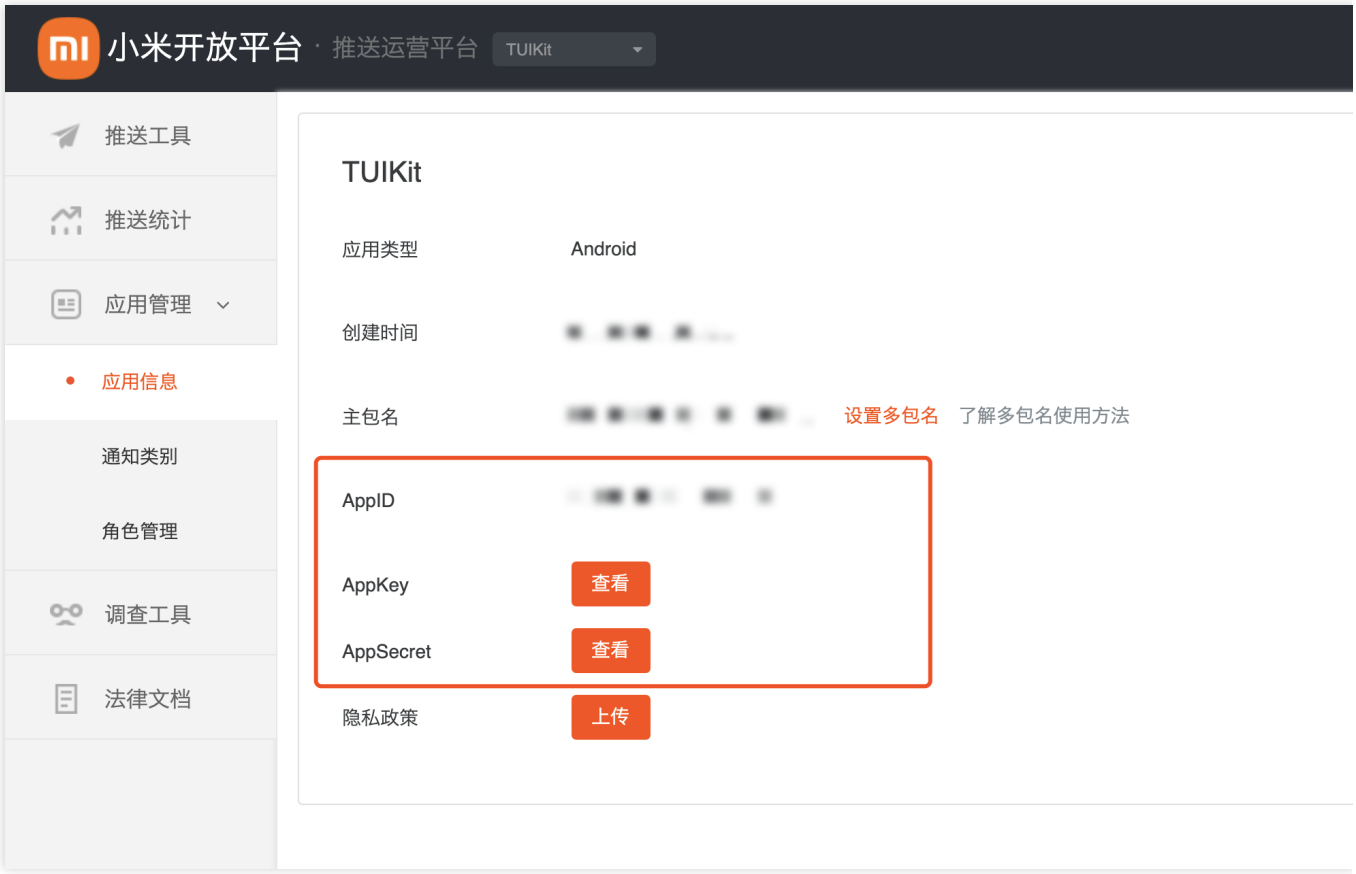
应用包名与插件应用包名保持一致。

步骤3：启用推送

进入推送运营平台的**应用列表**页面，在对应的应用名称单击**启用推送**，确定启用。

步骤4：查看获取应用信息

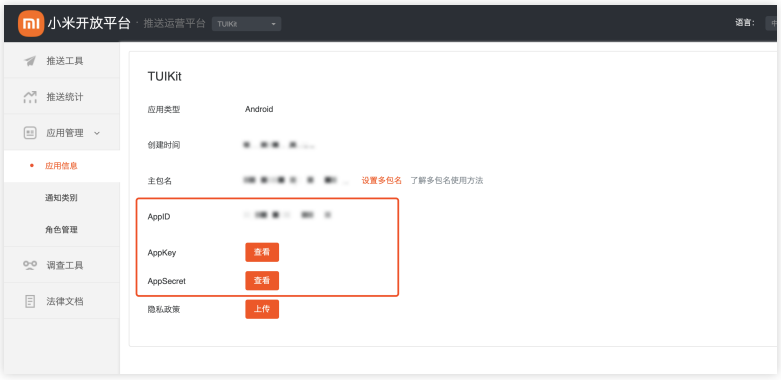
进入推送运营平台的**应用信息**页面，查看**应用信息**。



步骤5：配置推送证书

登录腾讯云 即时通信 IM 控制台，在**推送管理 > 接入设置**功能栏添加各个厂商推送证书，并将您获取的厂商的 AppId、AppKey、AppSecret 等参数配置给添加的推送证书。

厂商推送平台	IM 控制台配置
	注意：



应用内指定界面链接，不可以修改
监听，不可以直接配置应用内页面

步骤1：注册华为开发者账号

进入 华为开发者联盟，注册华为开发者账号，详情请参见 注册账号。

步骤2：创建应用

1. 在管理中心单击**我的项目**，添加一个新的项目。

2. 在**项目设置**栏单击**推送服务** > **立即开通**。

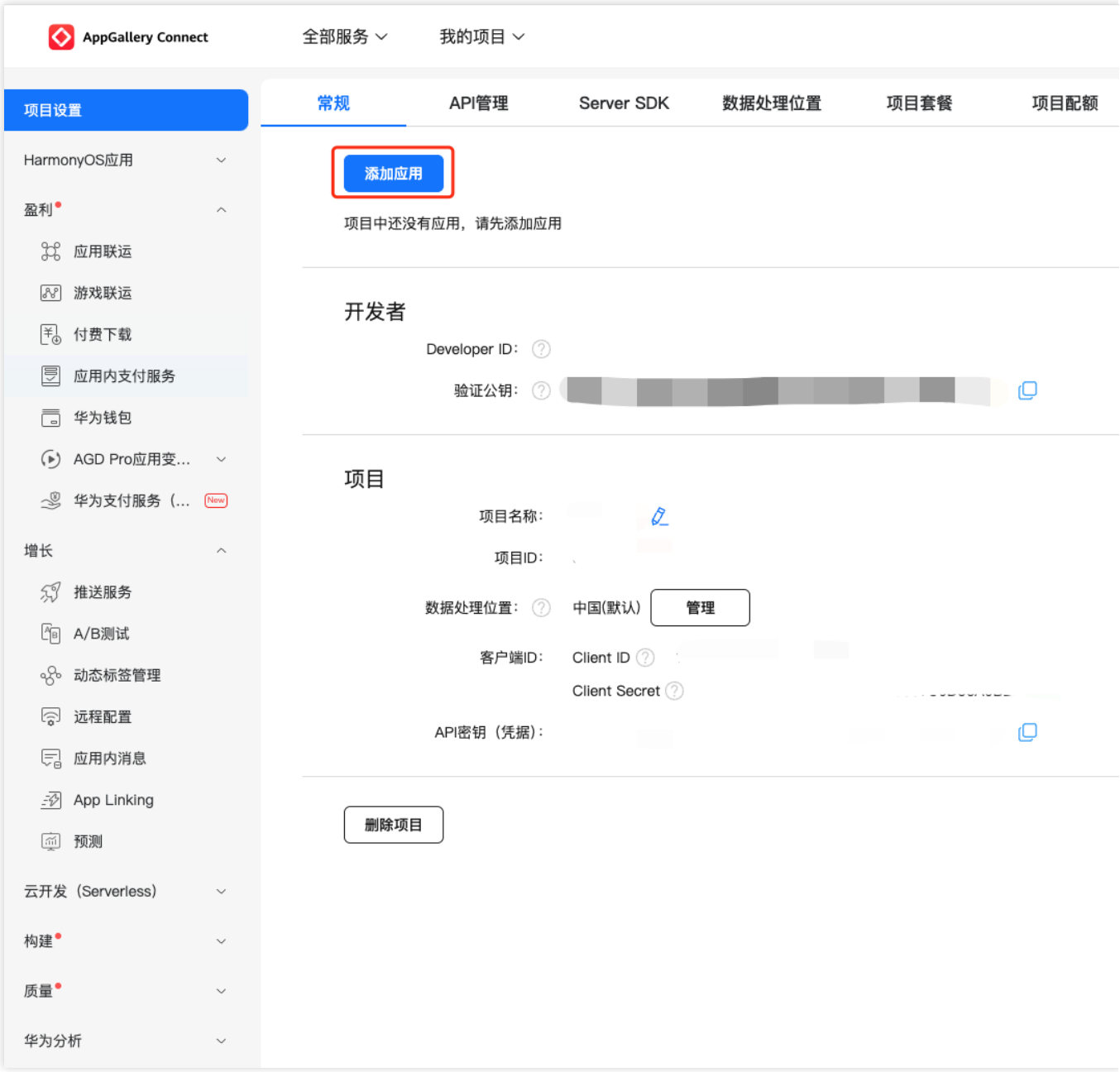
3.单击**项目设置 > API 管理**，开启推送服务的权限。

步骤3. 添加应用

单击**项目设置 > 常规**，添加应用。

注意：

应用包名与插件应用包名保持一致。



步骤4：获取应用信息

单击**项目设置 > 常规**，获取应用信息。

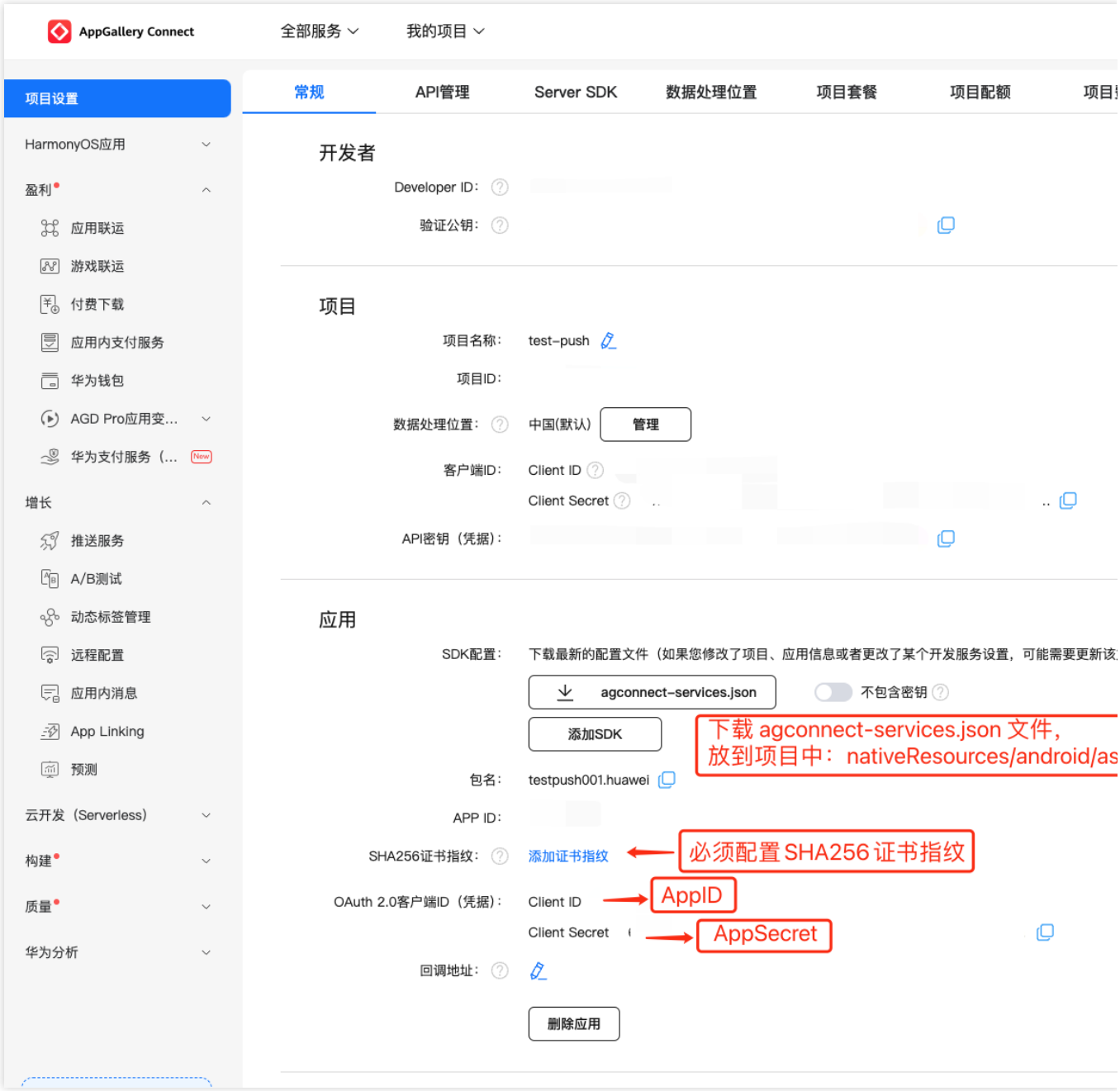
说明：

常规页面包含项目和应用的 Client ID 和 Client Secret，两者对应的参数不一致，请下拉至页面底部，获取应用的 Client ID 和 Client Secret。

必须添加打包的 SHA256证书指纹，SHA256 证书指纹需与自己的打包证书一致。

下载 agconnect-services.json 文件，放到项目中：nativeResources/android/assets/ 路径下。

修改了项目、应用信息、开发服务设置，都需要重新下载配置 agconnect-services.json 文件。



步骤5：添加推送证书

登录腾讯云 即时通信 IM 控制台，单击**推送管理** > **接入设置**添加各个厂商推送证书，并将您获取的厂商的 AppId、AppKey、AppSecret 等参数配置给添加的推送证书。

厂商推送平台	IM 控制台配置
	注意： Client ID 对应 AppID，Client **应用内指定界面**链接，不可以 监听，不可以直接配置应用内

注意：

通知栏推送：应用需在 OPPO 软件商店上架；

通知栏推送测试权限：每天仅可推送1000条消息，限测试使用。应用上架后需重新申请“通知栏推送”权限，以获得正常消息推送数量；

平台将会在1个工作日内返回审核结果，开发者可以在申请页面查看审核结果，其他问题可咨询开放平台客服。

步骤1：注册 OPPO 开发者账号

进入OPPO开放平台，注册 OPPO 开发者账号，详情参见 OPPO 企业开发者账号注册。

步骤2：创建应用

进入 OPPO 开放平台，单击**产品 > 应用分发> OPPO 软件商店 > 发布应用**进入管理中心，创建应用。

步骤3：开通 PUSH 服务

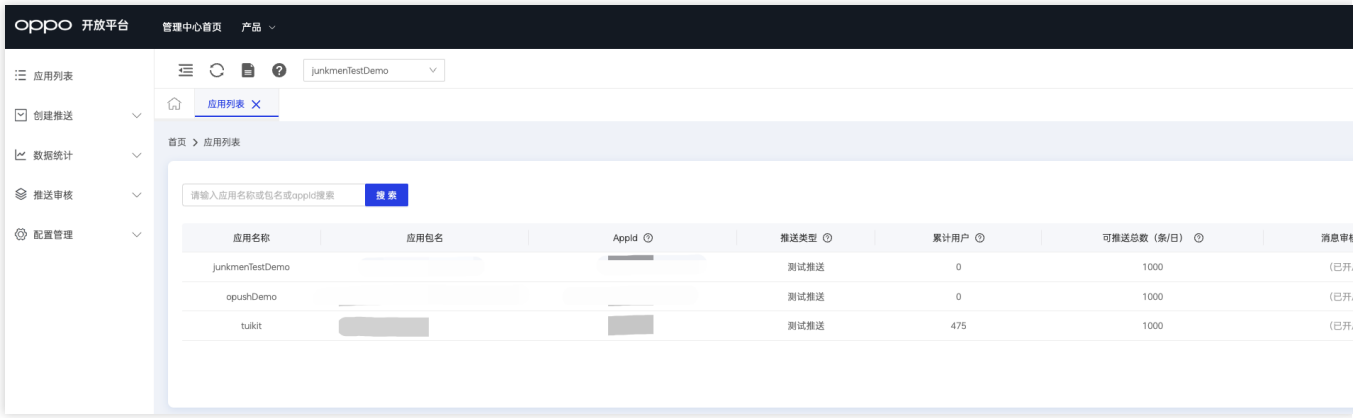
1. 进入 OPPO 开放平台，单击**产品** > **移动服务** > **推送服务**进入推送主页，单击**申请接**入开通推送服务。

2. 单击进入**管理中心** > **应用列表** > **申请推送服务**界面，为未开启服务的应用申请推送权限。

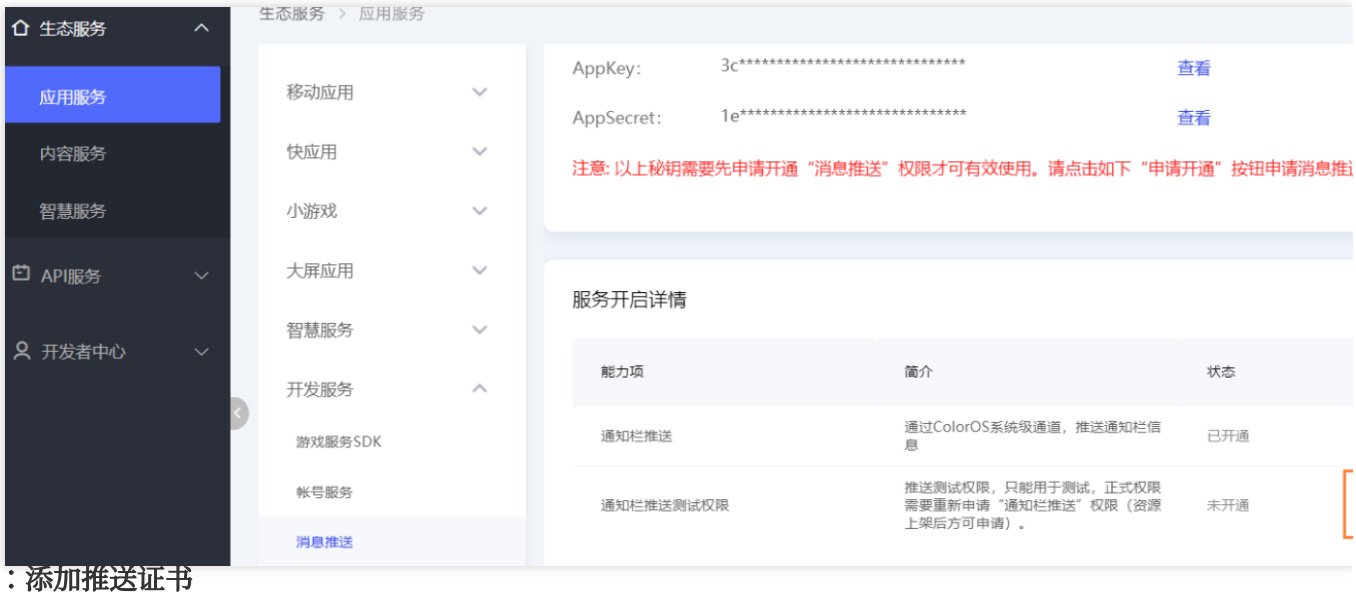
说明：

已开启服务：已申请 PUSH 权限并通过的应用。

未开启服务：可申请 PUSH 权限的应用。



3. 单击**申请开通**。在未开启服务中单击需要申请 PUSH 权限的应用，进入 PUSH 服务并点击**申请开通**。



步骤4：添加推送证书

登录腾讯云 即时通信 IM 控制台，在**推送管理 > 接入设置**功能栏添加各个厂商推送证书，并将您获取的厂商的 AppId、AppKey、AppSecret 等参数配置给添加的推送证书。

厂商推送平台	IM 控制台配置
	注意： 应用内指定界面链接，不可以 该配置用于派发单击后离线推 跳转。 

注意：

若应用没有上架应用市场，推送权限受限，不可在 vivo 官网的 Web 界面和 API 后台发送正式消息，可在 API 后台向设置的测试设备发送测试消息进行测试。

步骤1：注册 vivo 开发者账号

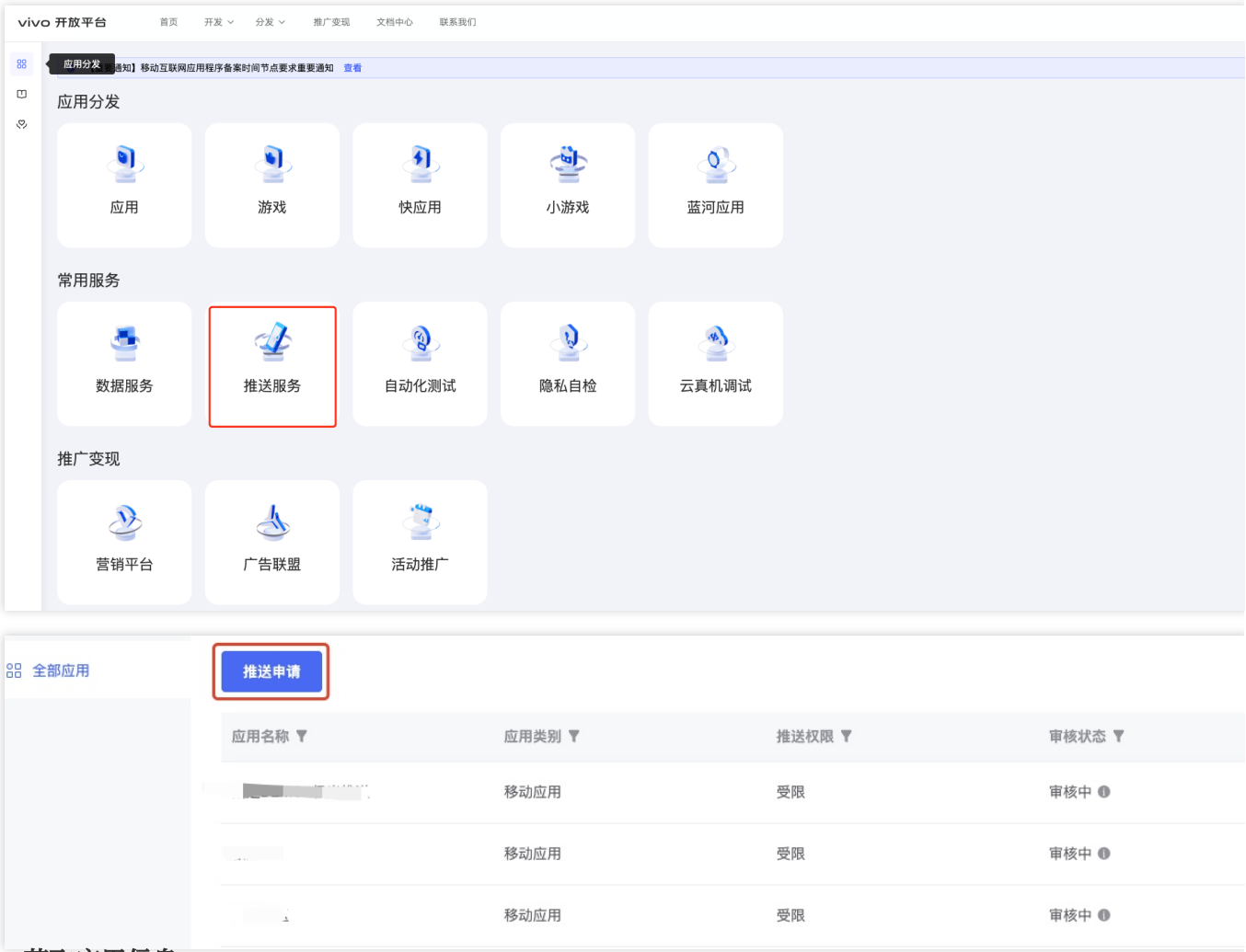
进入 vivo开放平台，注册 vivo 开发者账号，详情参见 vivo 企业开发者账号注册。

步骤2：新建应用

进入**vivo 开放平台，**单击**分发 > 应用分发 > 应用商店 > 上传应用**来新建您的应用。

步骤3：开通推送

进入管理中心单击**推送服务 > 推送申请**为新建的应用申请开通推送。



步骤4：获取应用信息

进入推送运营平台，单击**应用管理 > 应用信息**，获取应用信息。

步骤5：添加推送证书

登录腾讯云 即时通信 IM 控制台，单击**推送管理** > **接入设置**添加各个厂商推送证书，并将您获取的厂商的 AppId、AppKey、AppSecret 等参数配置给添加的推送证书。

厂商推送平台	IM 控制台配置
	**注意：** **应用内指定界面**链接，不可以 监听，不可以直接配置应用内 

回执配置请参见：消息触达统计配置 > vivo

步骤1：注册魅族开发者账号

注册魅族开发者账号，详情参见 开发者注册。

步骤2：创建应用

1. 单击**控制台** > **Flyme 推送**。

2. 填写应用信息后，创建应用。

注意：

应用包名与插件应用包名保持一致。

步骤3：获取应用信息

在应用列表中单击**打开应用**。进入配置管理页面，获取应用信息。

步骤4：添加推送证书

登录腾讯云 即时通信 IM 控制台，单击**推送管理 > 接入设置**功能栏添加各个厂商推送证书，并将您获取的厂商的 AppId、AppKey、AppSecret 等参数配置给添加的推送证书。

厂商推送平台	IM 控制台配置
	**注意：** **应用内指定界面**链接，不可以 监听，不可以直接配置应用内 

回执配置请参见：消息触达统计配置 > 魅族

步骤1. 注册荣耀开发者账号

注册荣耀开发者账号，详情参见 开发者注册。

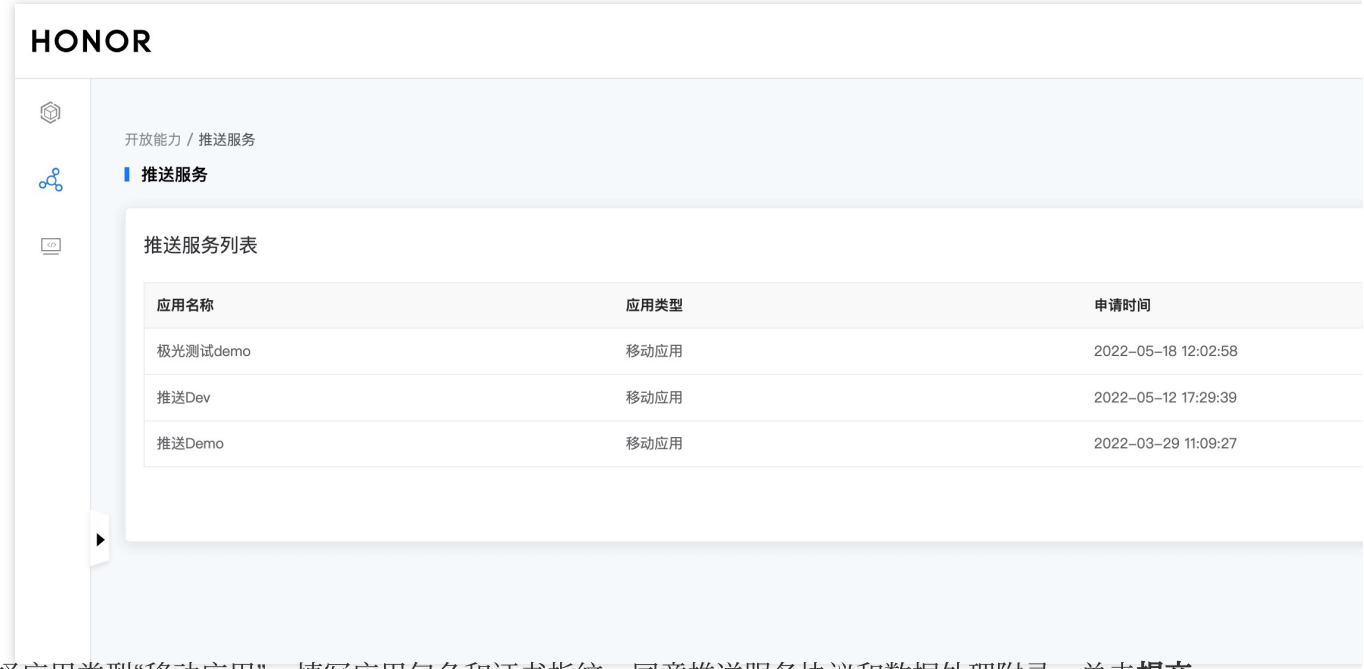
步骤2: 进入管理中心页面。

步骤3: 进入推送服务列表

单击**推送服务**进入推送服务列表页面。

步骤4: 创建应用

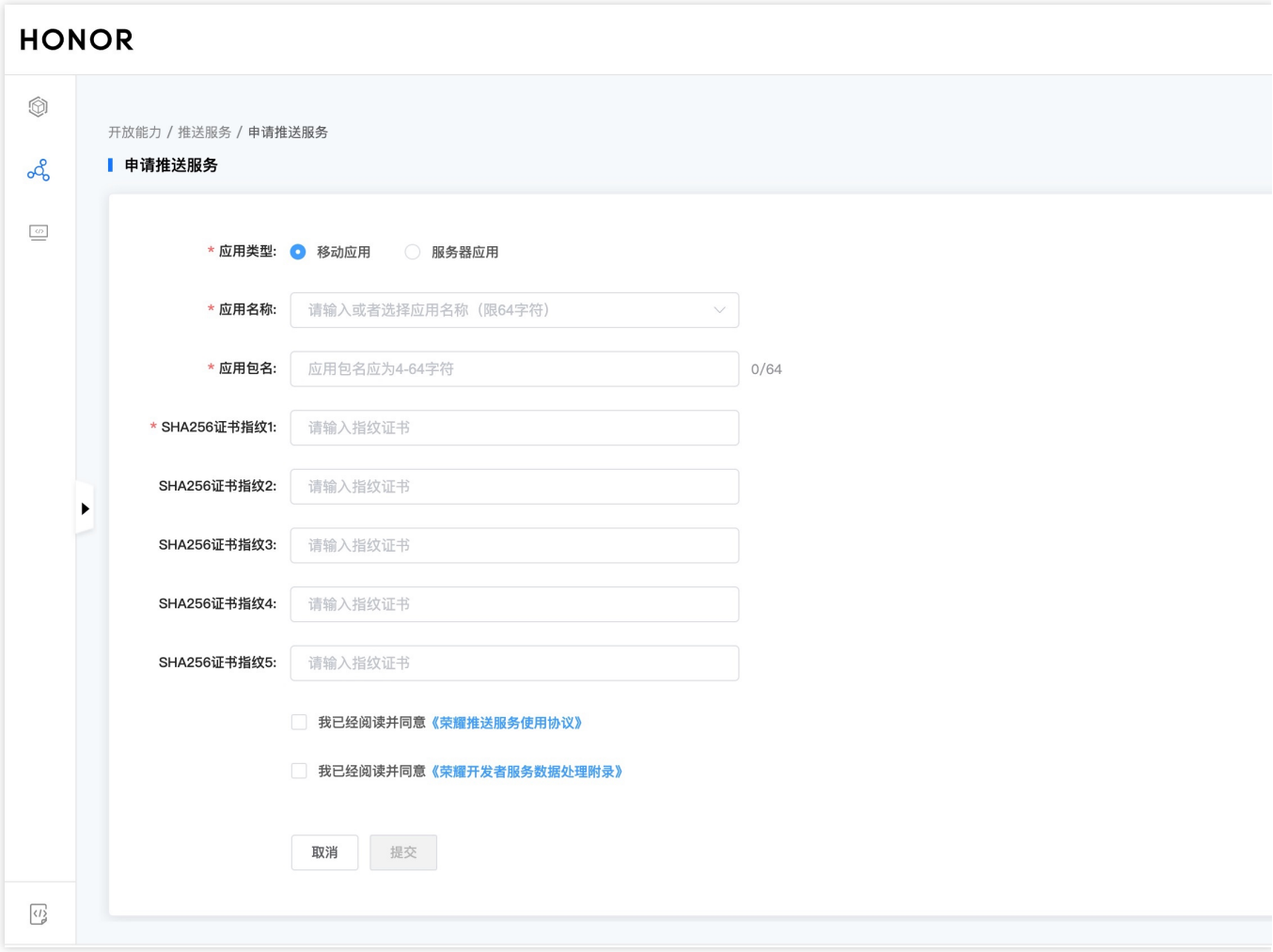
1. 单击**申请推送服务**进入应用申请页面。



2. 选择应用类型“移动应用”，填写应用包名和证书指纹、同意推送服务协议和数据处理附录，单击**提交**。

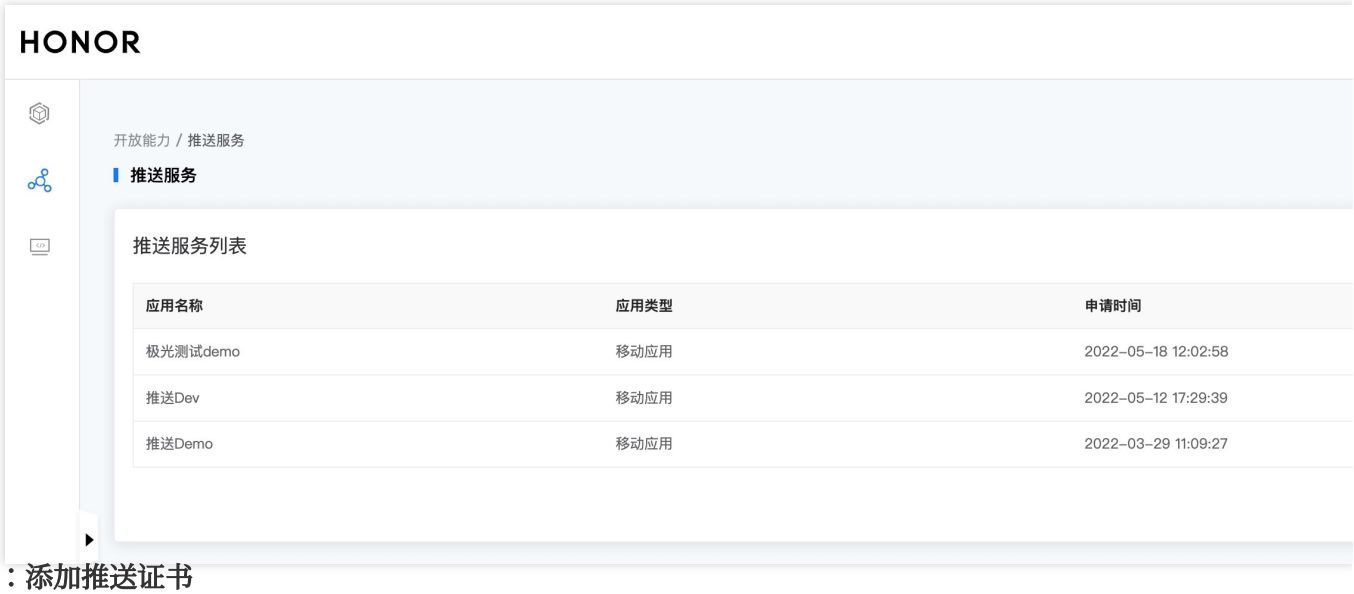
注意：

需要添加打包的 SHA256 证书指纹，SHA256 证书指纹需与自己的打包证书一致。



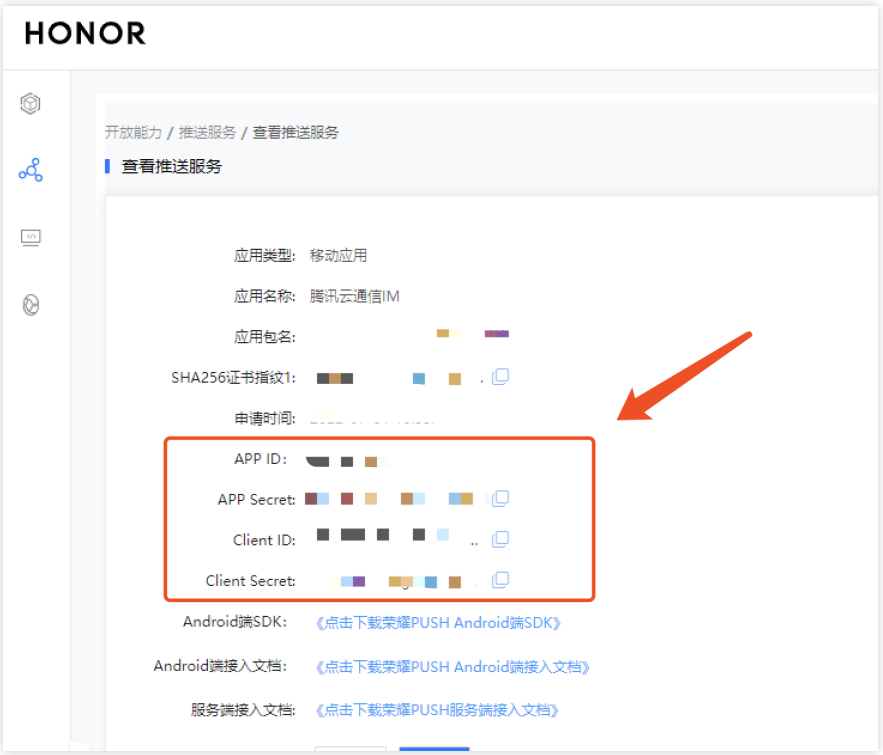
步骤5: 获取应用信息

在**推送服务**列表中，单击**查看**，获取应用信息。



步骤6：添加推送证书

登录腾讯云 即时通信 IM 控制台，单击**推送管理 > 接入设置**，添加各个厂商推送证书，并将您获取的厂商的 AppID、AppKey、AppSecret 等参数配置给添加的推送证书。

厂商推送平台	IM 控制台配置
	注意： **应用内指定界面**链接，不 监听，不可以直接配置应 

Google FCM 正在开发中，敬请期待~~

集成 TIMPush 组件之前，需要先向 Apple 申请 APNs 推送证书，然后上传推送证书到 IM 控制台 。之后按照快速接入步骤接入即可。

Apple 厂商配置目前有两种主流的证书，p12 证书和 p8 证书。两种证书各有优劣，您可按需要选择其中的一种。

	证书类型	有效期和管理	安全性	灵动岛
p12 证书	p12 证书是一个包含公钥和私钥的二进制文件，用于基于证书的身份验证。它将公钥证书和私钥捆绑在一个文件中，扩展名为 .p12 或 .pfx。	p12 证书通常有一年的有效期，过期后需要重新生成和部署。每个应用程序都需要单独的 P12 证书来处理推送通知。	证书：p12 证书使用基于证书的身份验证，需要在服务器上存储私钥。这可能会增加安全风险，因为私钥可能会被未经授权的用户访问。	不支持
p8 证书	p8 证书是一个 Auth Key（授权密钥），用	p8 证书没有到期日期，因此您无需担心证书过	p8 证书使用基于令牌的身份验证，这意味着您的服	支持灵动岛推送

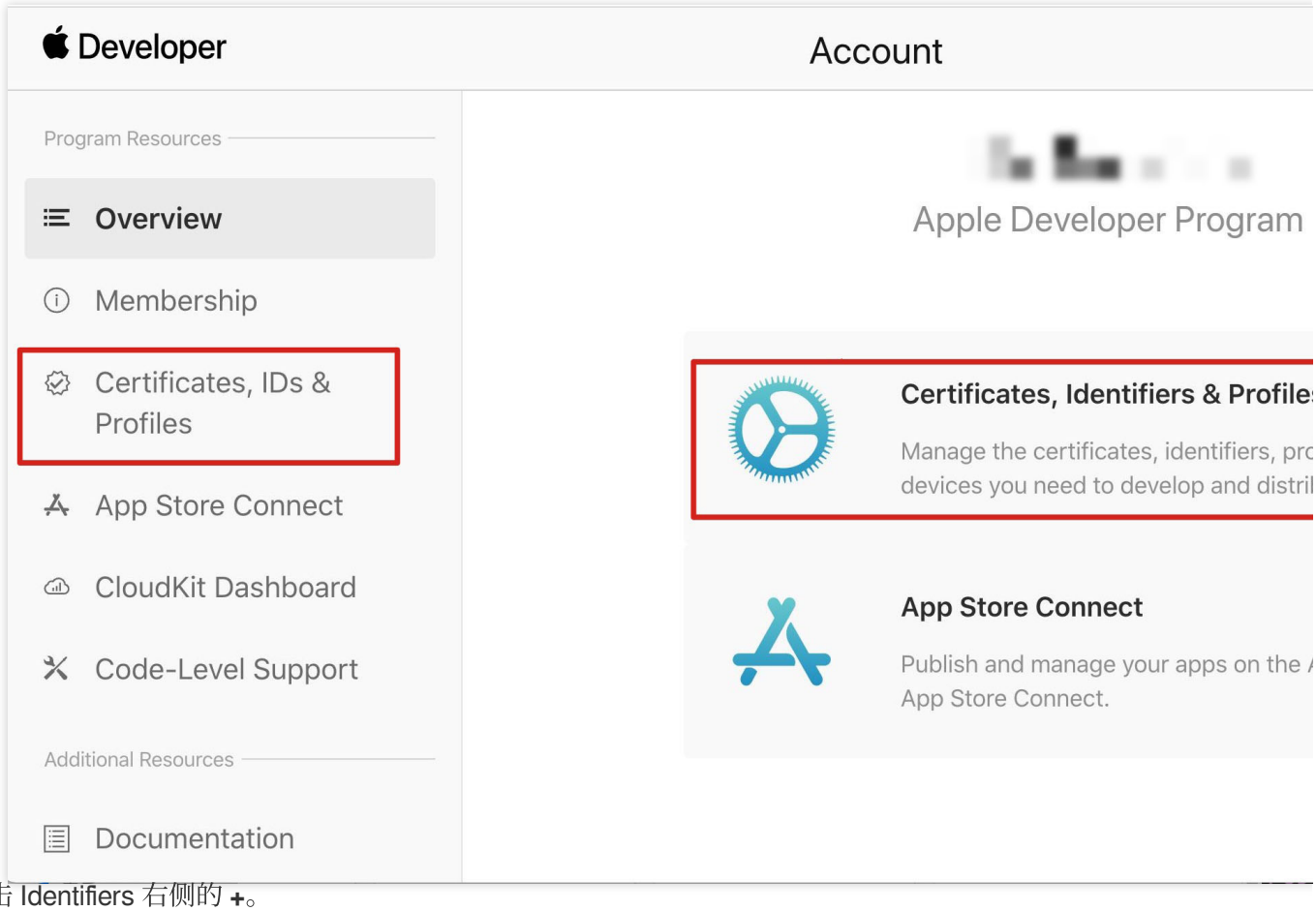
	于基于令牌的身份验证。它是一个包含私钥的文本文件，扩展名为 .p8。	期。此外，使用 P8 证书可以简化证书管理，因为您可以使用一个 p8 证书为多个应用程序提供推送通知服务。	务器会周期性地生成一个 JSON Web Token（JWT）来与 APNs 建立连接。这种方法更安全，因为它不需要在服务器上存储私钥。	

一、使用 p12 证书 （传统推送证书）

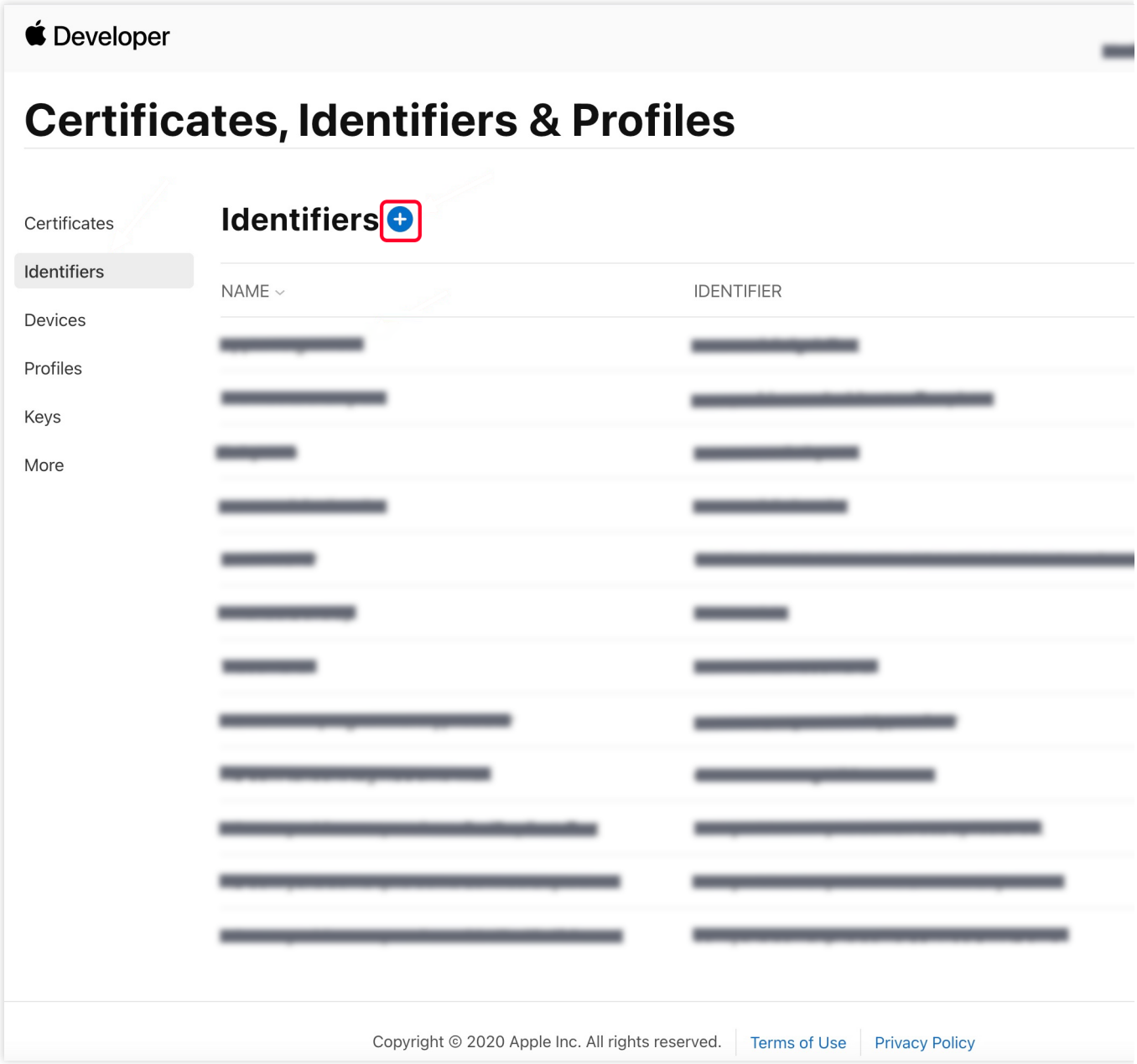
步骤1：申请 APNs 证书

开启 App 远程推送

1. 登录 苹果开发者中心 网站，单击 **Certificates,Identifiers & Profiles** 或者侧栏的 **Certificates,IDs & Profiles**，进入 **Certificates, IDS & Profiles** 页面。



2. 单击 Identifiers 右侧的 +。

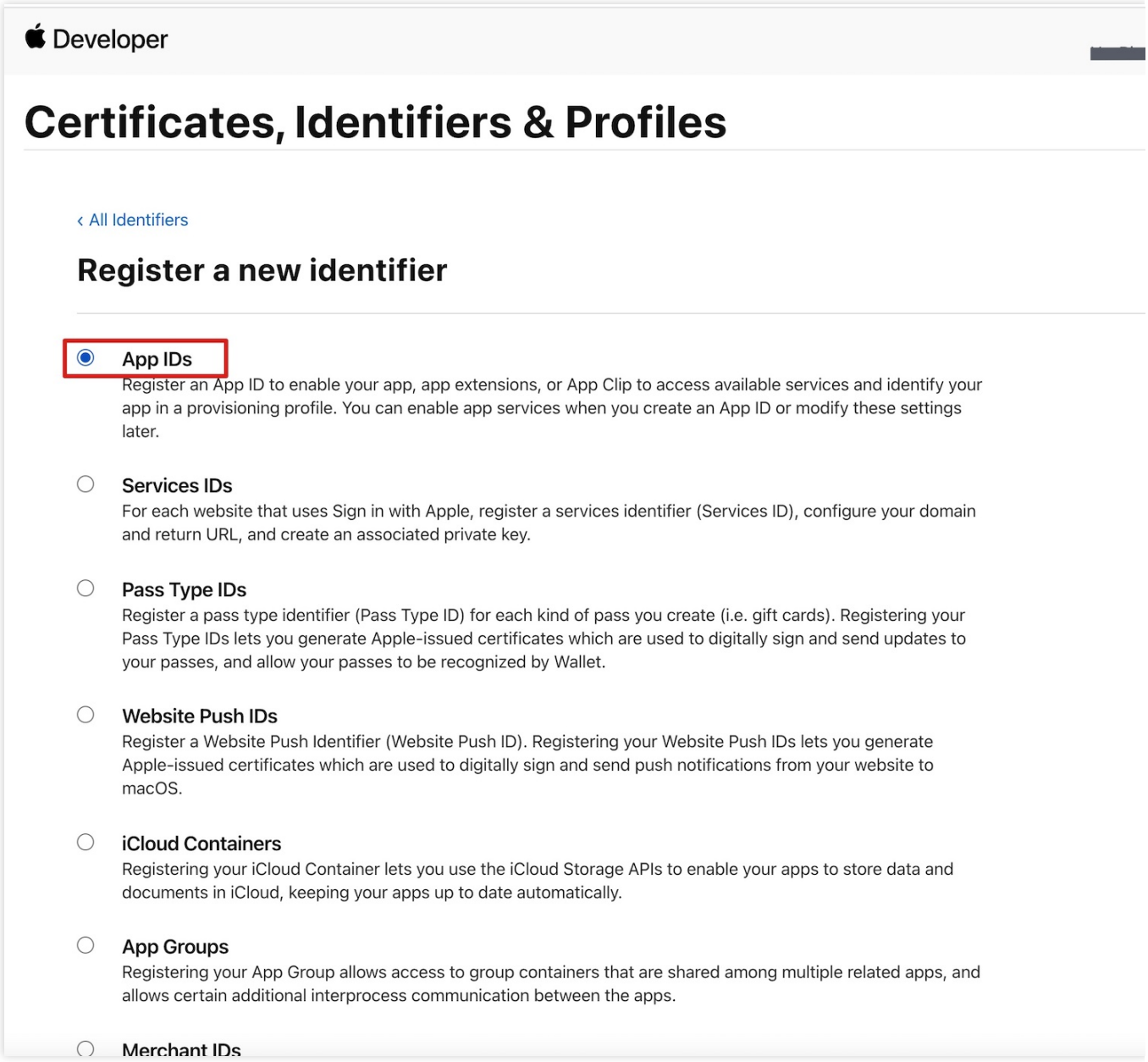


3. 您可以参见如下步骤新建一个 AppID，或者在您原有的 AppID 上增加 `Push Notification` 的 `Service` 。

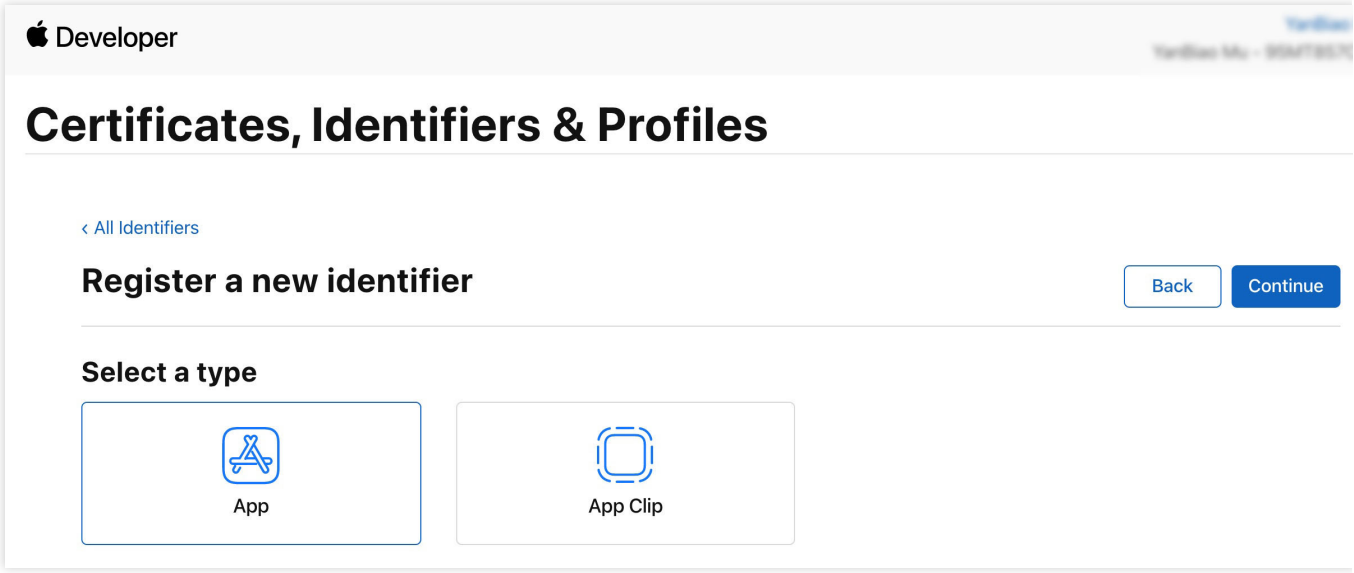
说明：

您 App 的 `Bundle ID` 不能使用通配符 `*` ，否则将无法使用远程推送服务。

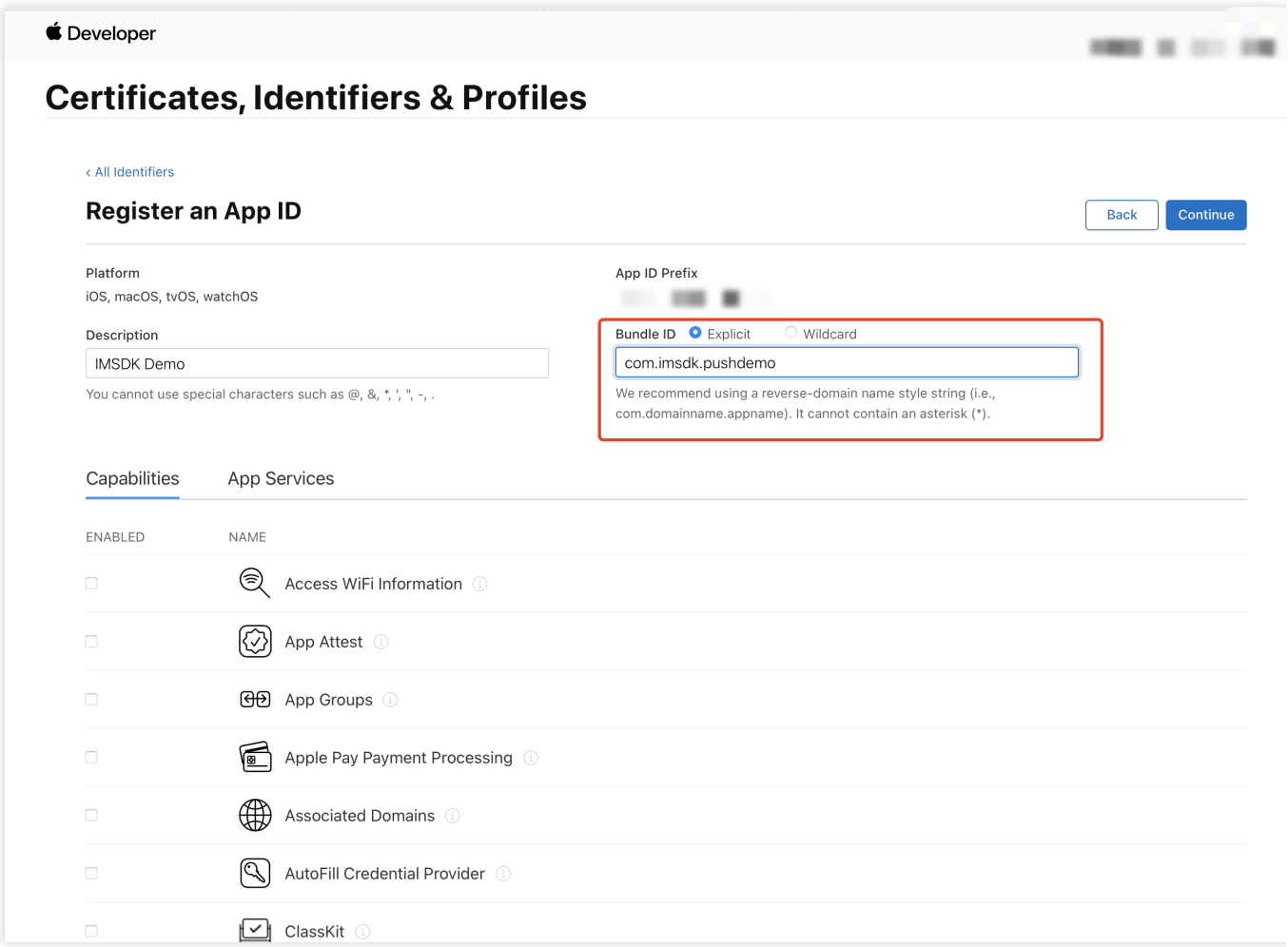
4. 勾选 **App IDs**，单击 **Continue** 进行下一步。



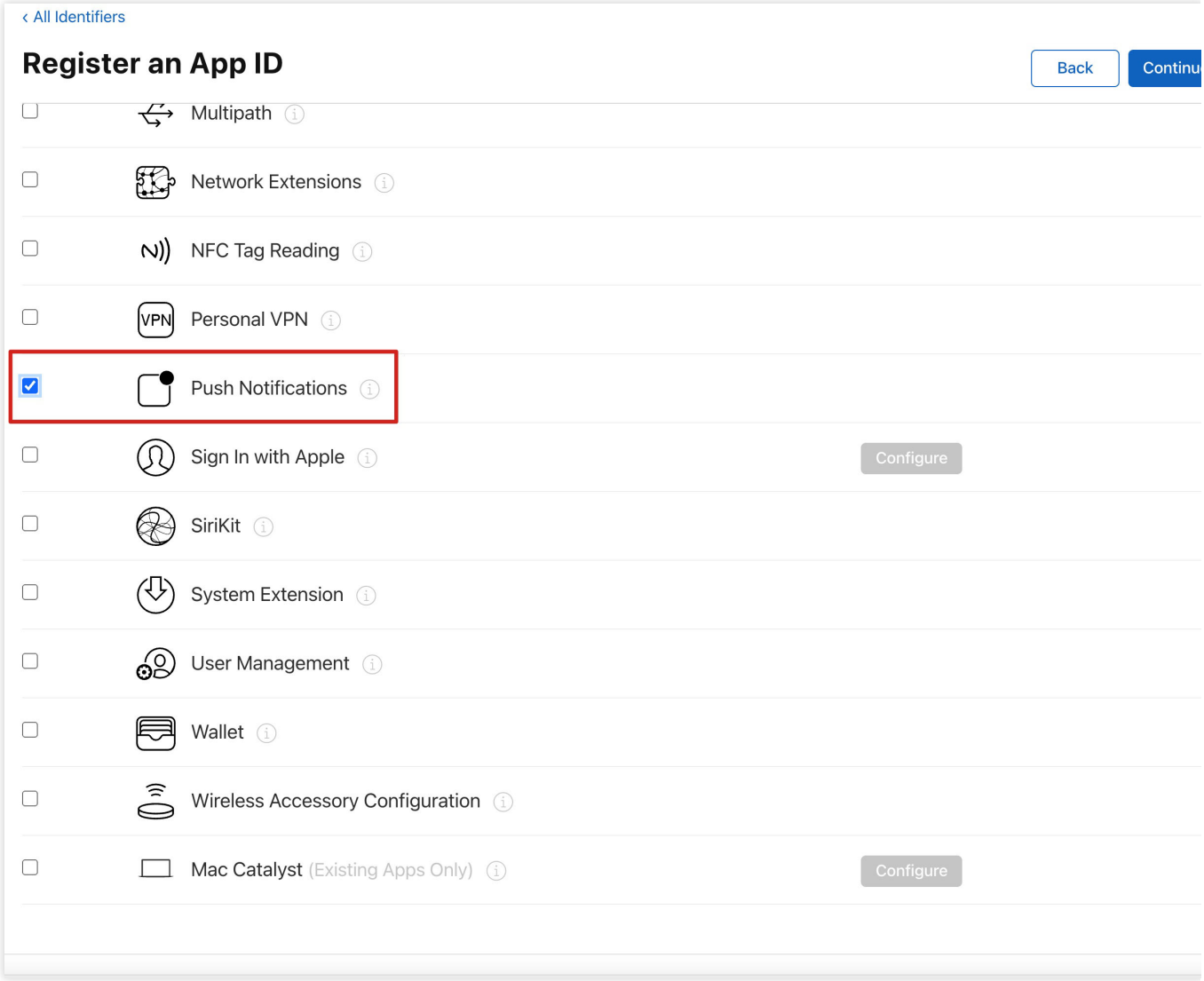
5. 选择 **App**，单击 **Continue** 进行下一步。



6. 配置 `Bundle ID` 等其他信息，单击 **Continue** 进行下一步。

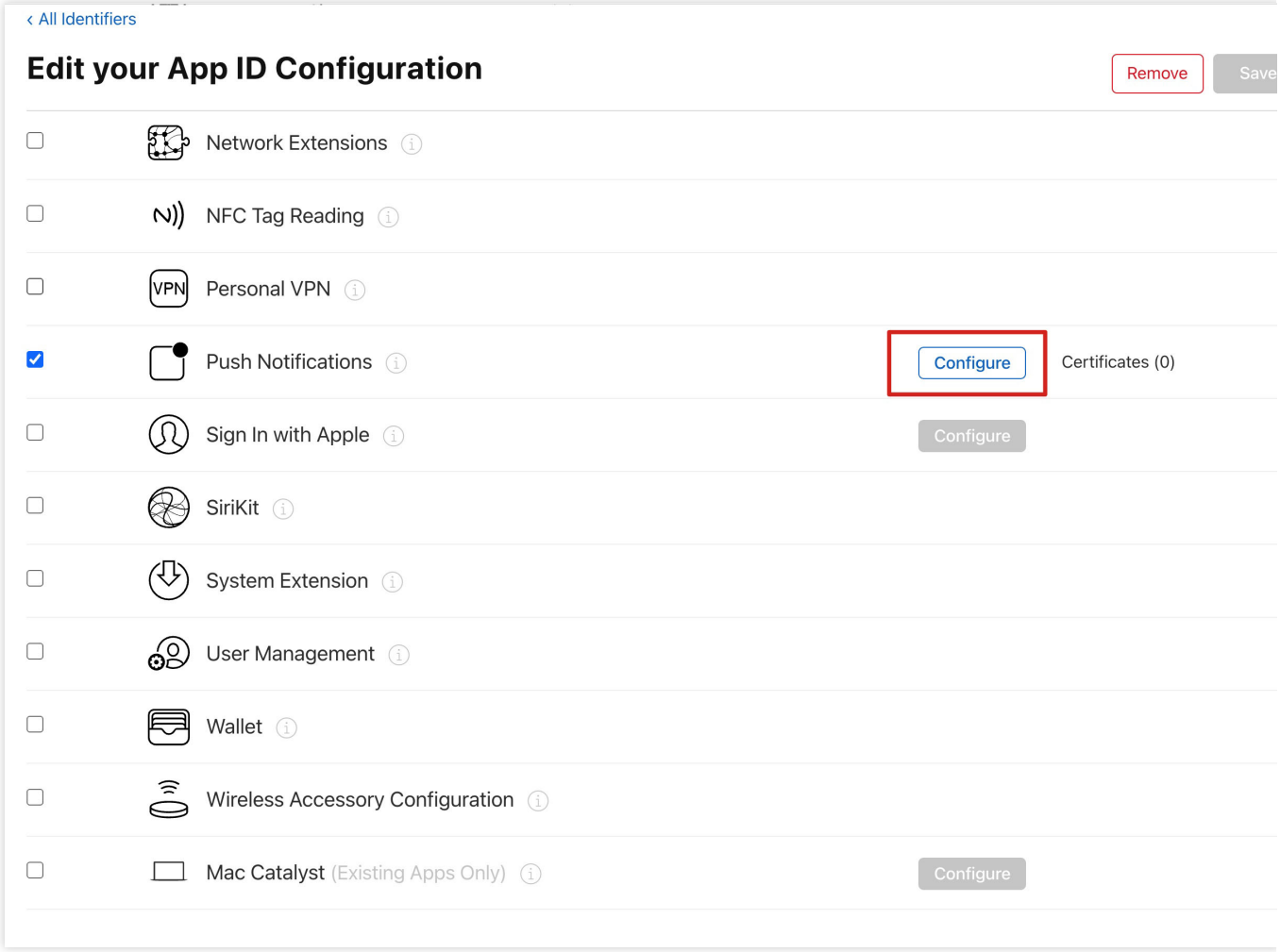


7. 勾选 **Push Notifications**，开启远程推送服务。

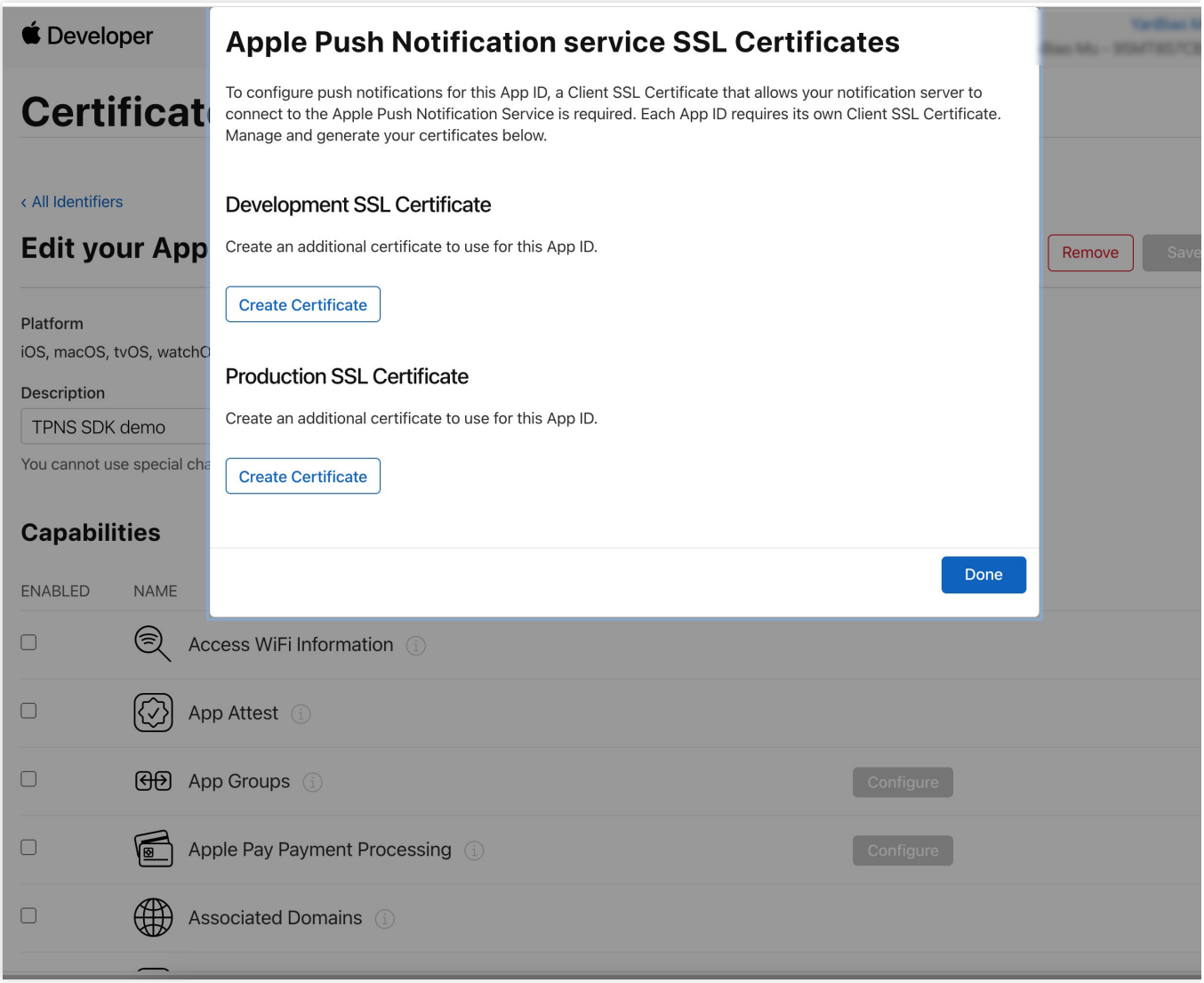


生成证书

1. 选中您的 AppID，选择 **Configure**。



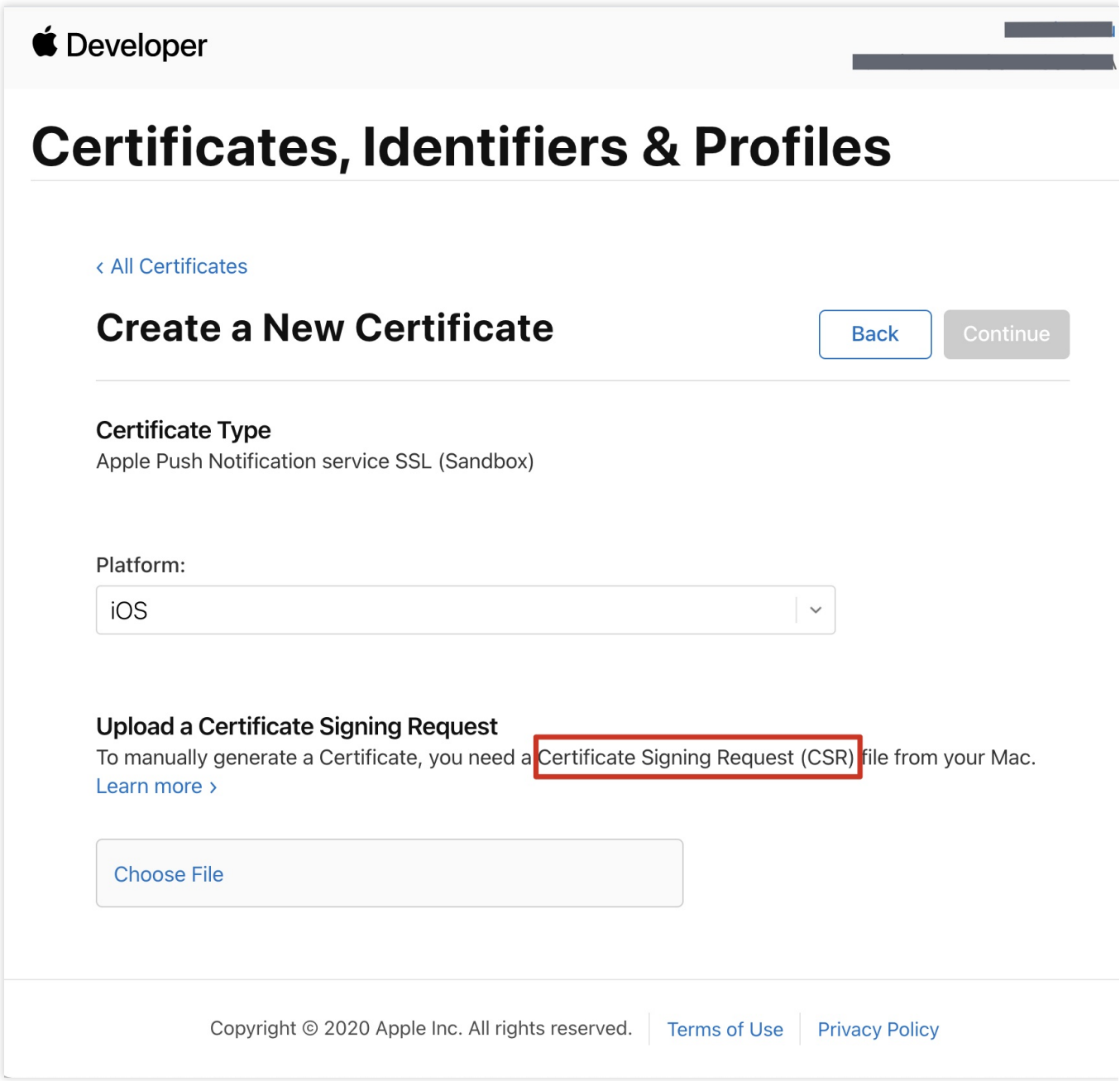
2. 可以看到在 **Apple Push Notification service SSL Certificates** 窗口中有两个 `SSL Certificate`，分别用于开发环境（Development）和生产环境（Production）的远程推送证书，如下图所示：



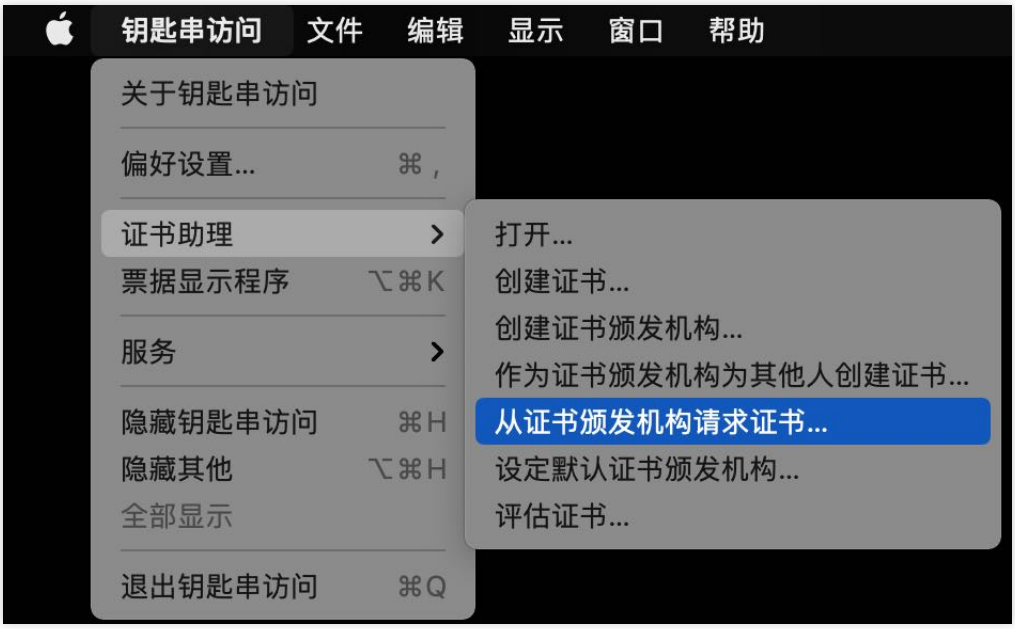
3.

我

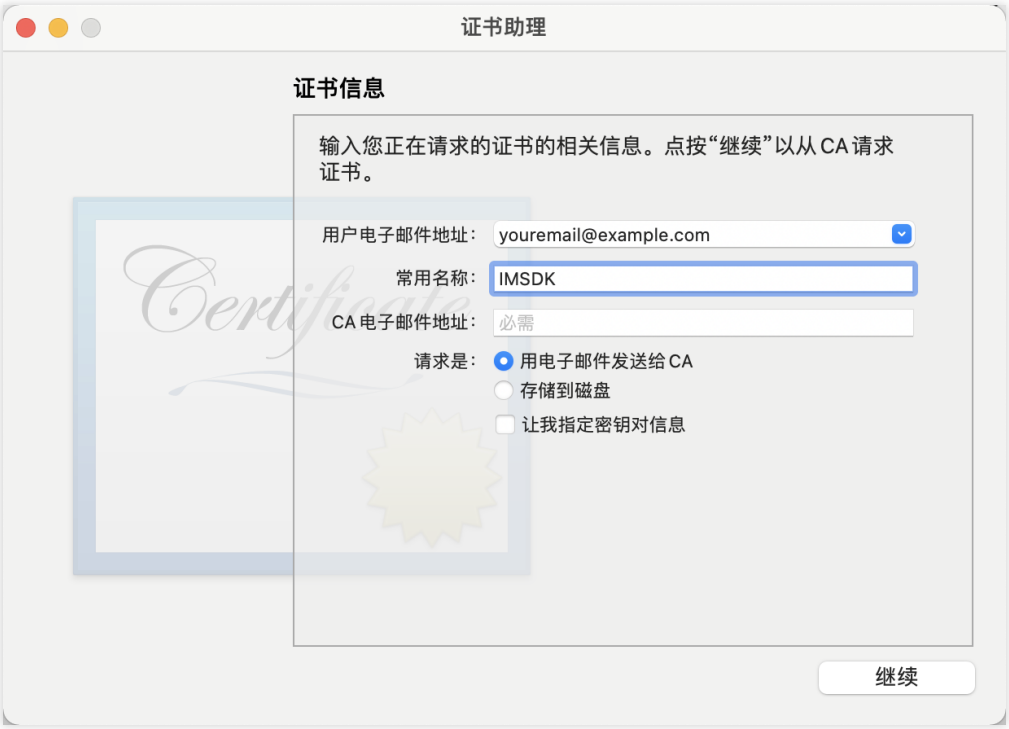
们先选择开发环境（Development）的 **Create Certificate**，系统将提示我们需要一个 Certificate Signing Request（CSR）。



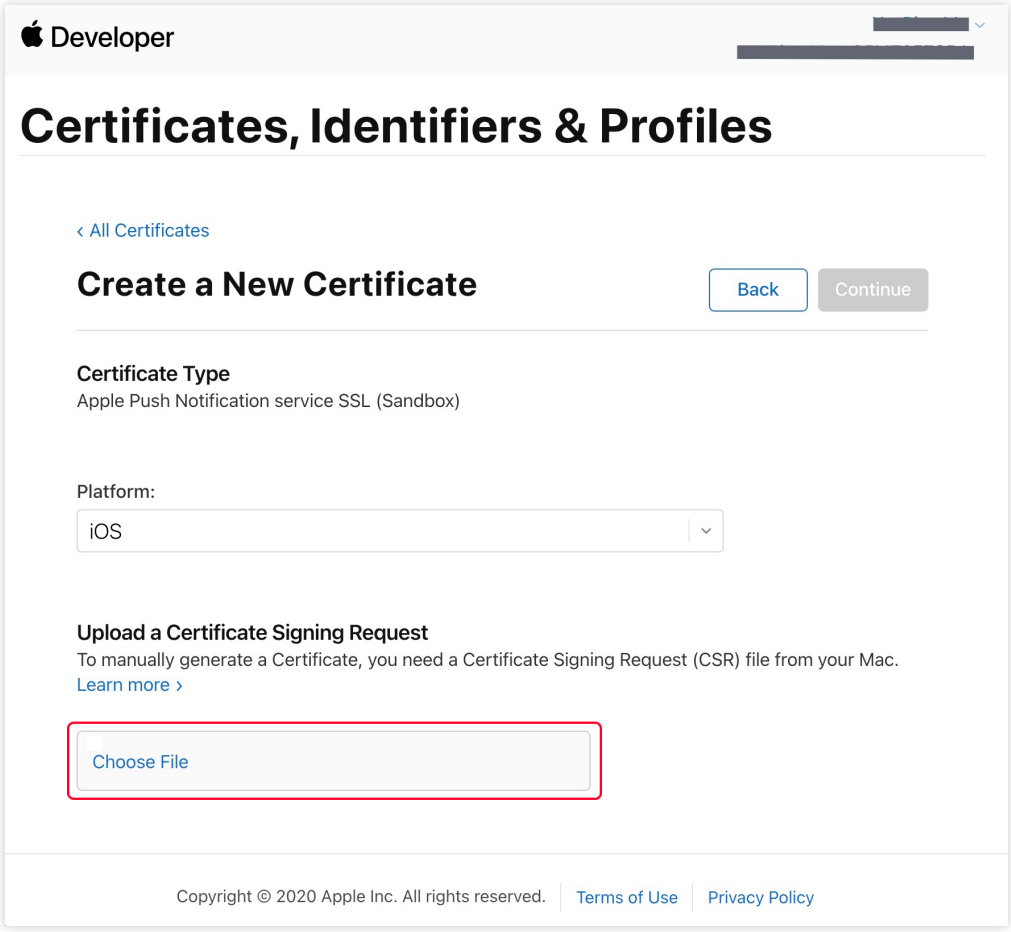
4. 在 Mac 上打开**钥匙串访问工具（Keychain Access）**，在菜单中选择**钥匙串访问 > 证书助理 > 从证书颁发机构请求证书**（`Keychain Access - Certificate Assistant - Request a Certificate From a Certificate Authority`）。



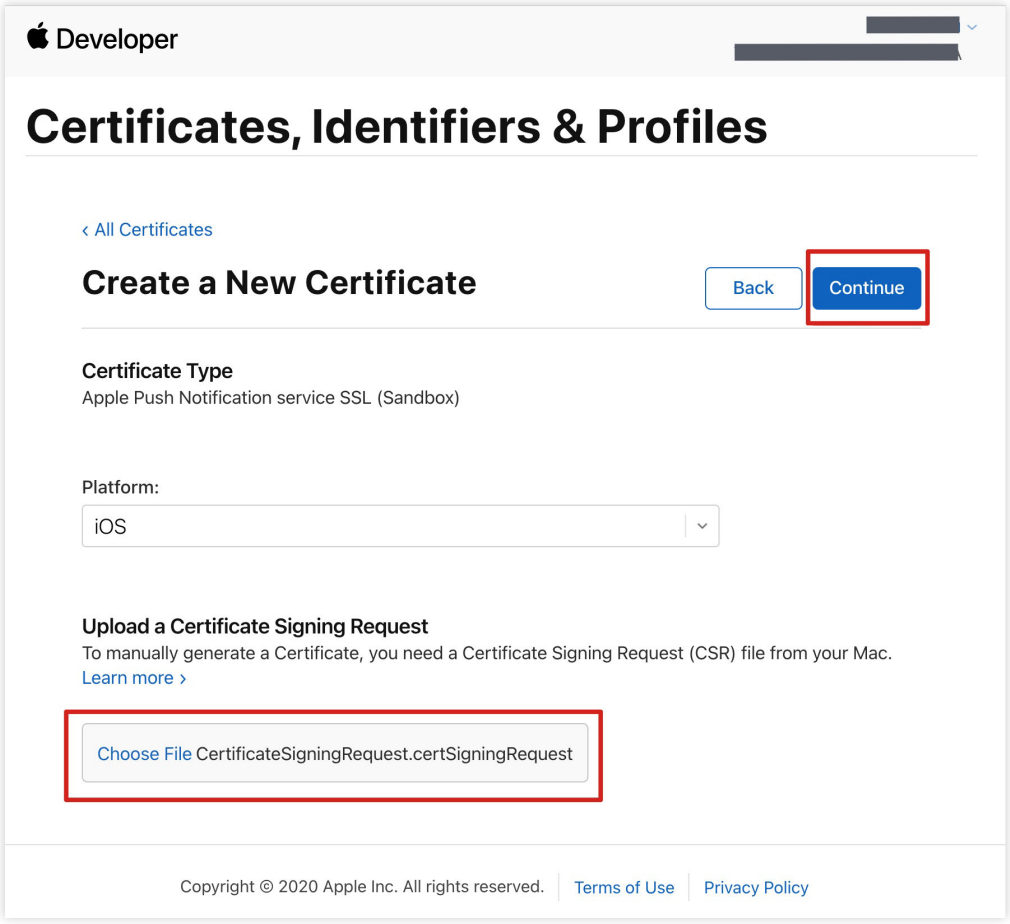
5. 输入用户电子邮件地址（您的邮箱）、常用名称（您的名称或公司名），选择**存储到磁盘**，单击**继续**，系统将生成一个 `*.certSigningRequest` 文件。



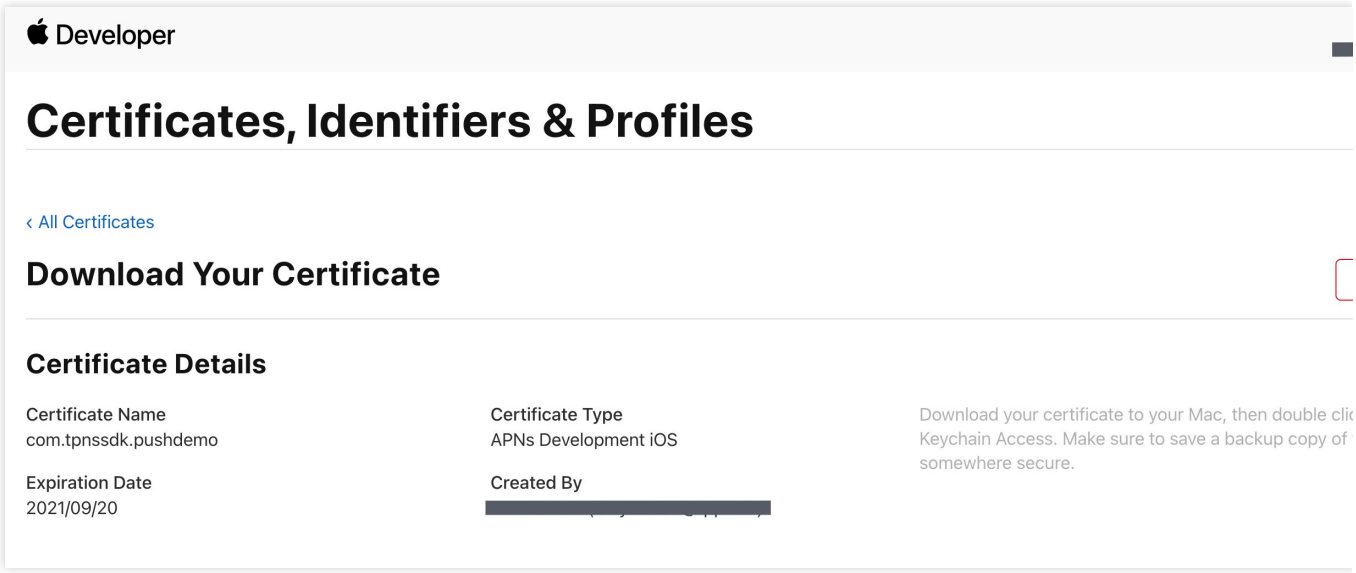
6. 返回上述 步骤3 中 Apple Developer 网站刚才的页面，单击 **Choose File** 上传生成的 `*.certSigningRequest` 文件。



7. 单击 **Continue**，即可生成推送证书。



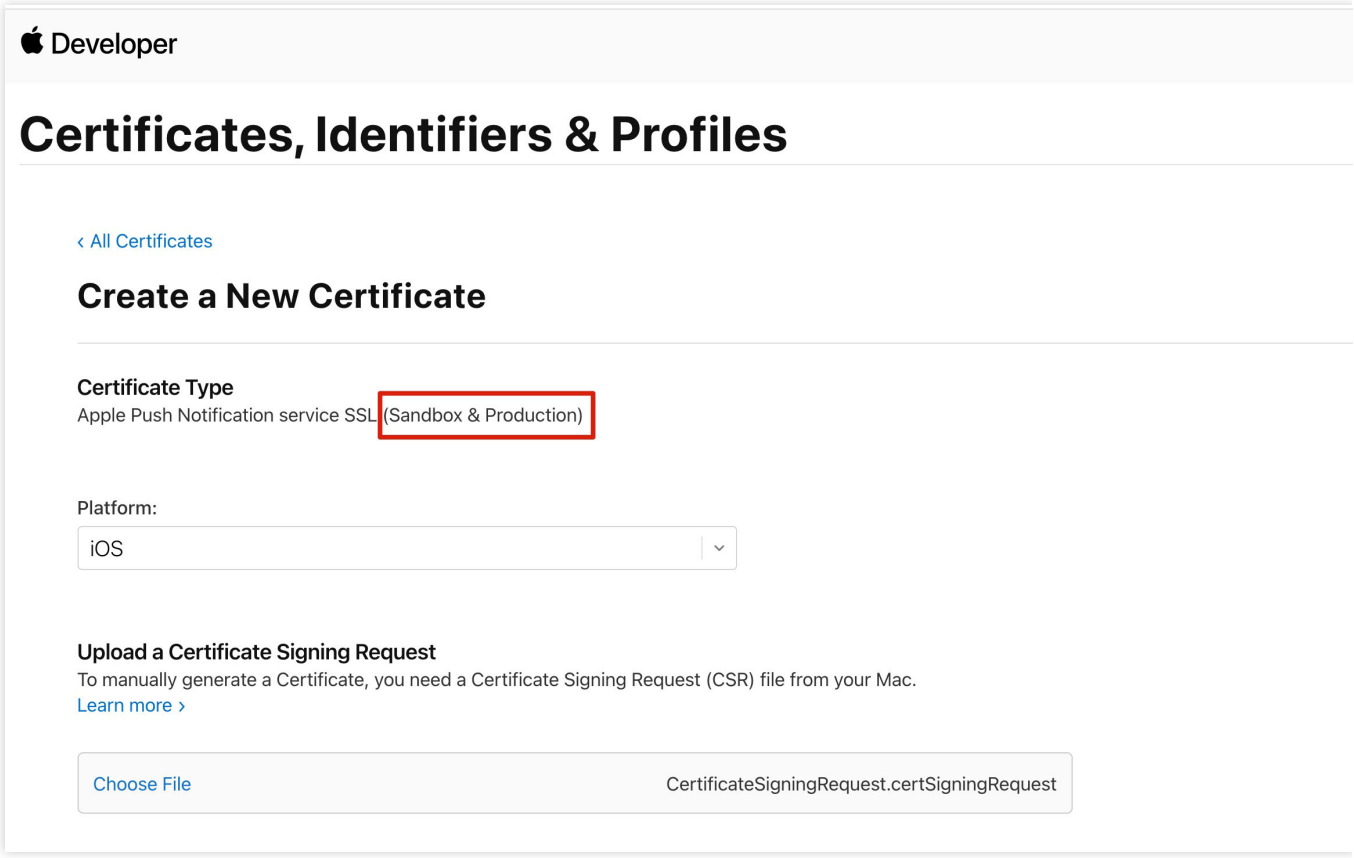
8. 单击 **Download** 下载开发环境的 `Development SSL Certificate` 到本地。

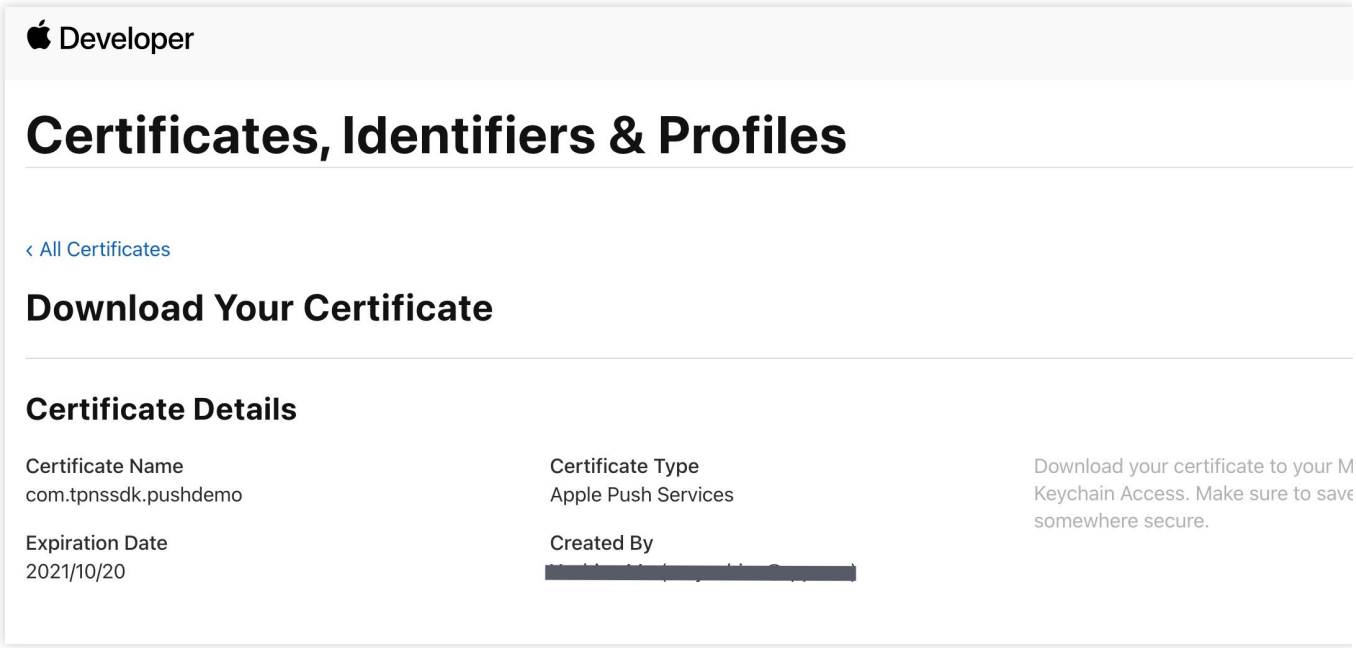


9. 再次按照上述步骤1 - 8，将生产环境的 `Production SSL Certificate` 下载到本地。

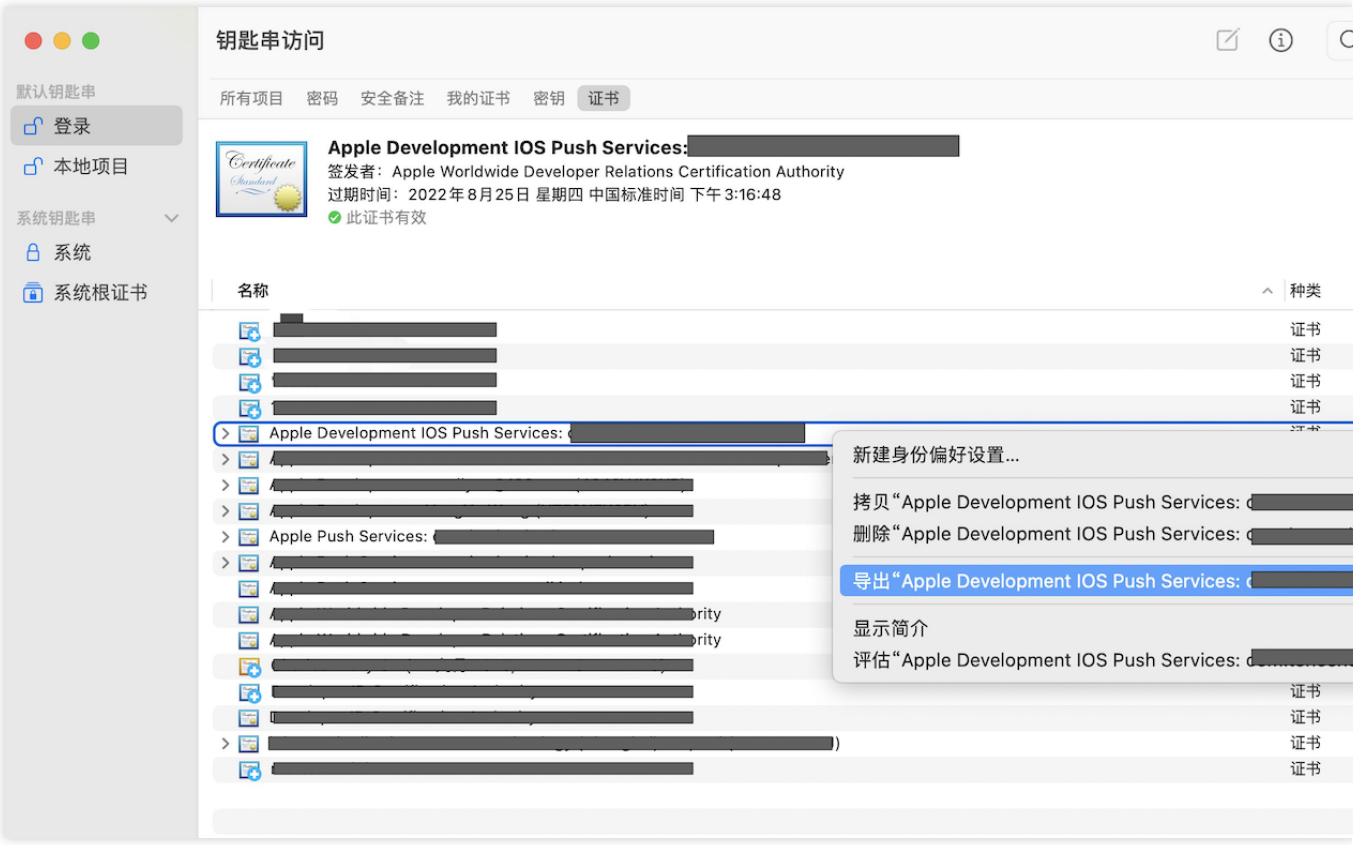
说明

生产环境的证书实际是开发（Sandbox）+生产（Production）的合并证书，可以同时作为开发环境和生产环境的证书使用。





10. 双击打开下载的开发环境和生产环境的 `SSL Certificate`，系统会将其导入钥匙串中。
11. 打开钥匙串应用，在**登录 > 我的证书**，右键分别导出刚创建的开发环境（ `Apple Development IOS Push Service` ）和生产环境（ `Apple Push Services` ）的 `p12` 文件。

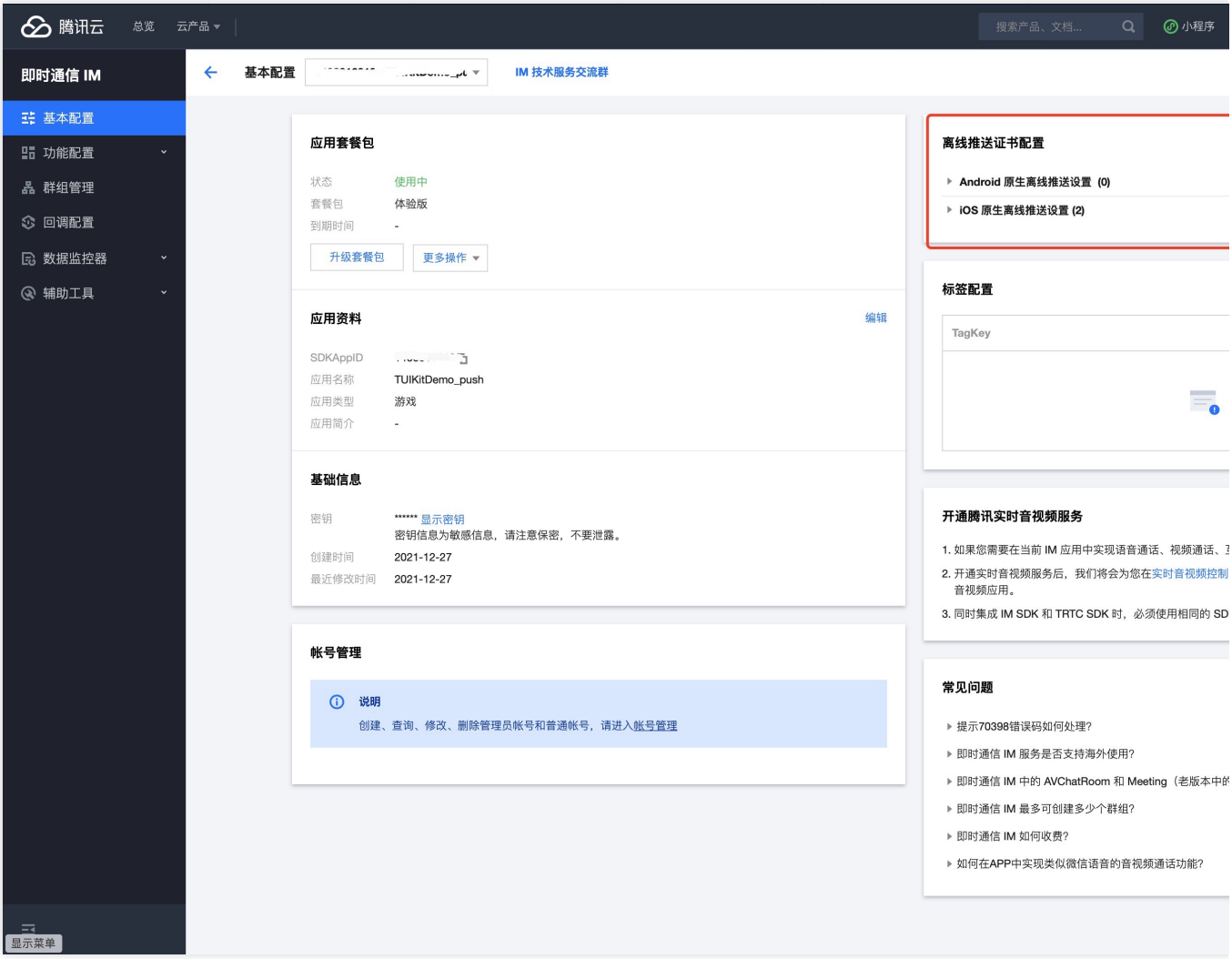


注意

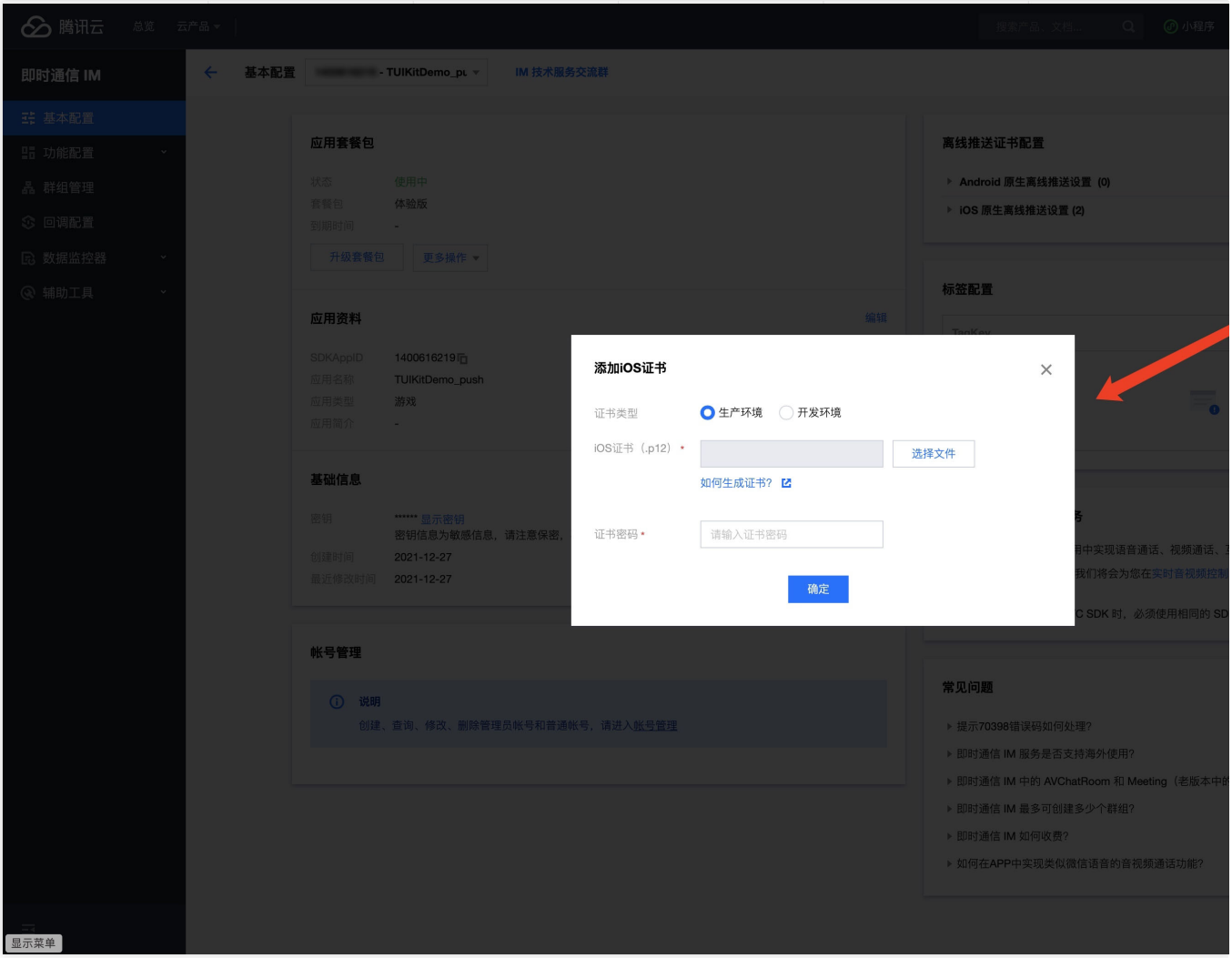
保存 `.p12` 文件时，请务必为其设置密码。

步骤2：上传证书到控制台

1. 登录 即时通信 IM 控制台。
2. 单击目标应用卡片，进入应用的基础配置页面。



3. 单击 **iOS 原生离线推送设置**右侧的**添加证书**。
4. 选择证书类型，上传 iOS 证书（p.12），设置证书密码，单击**确认**。



注意：

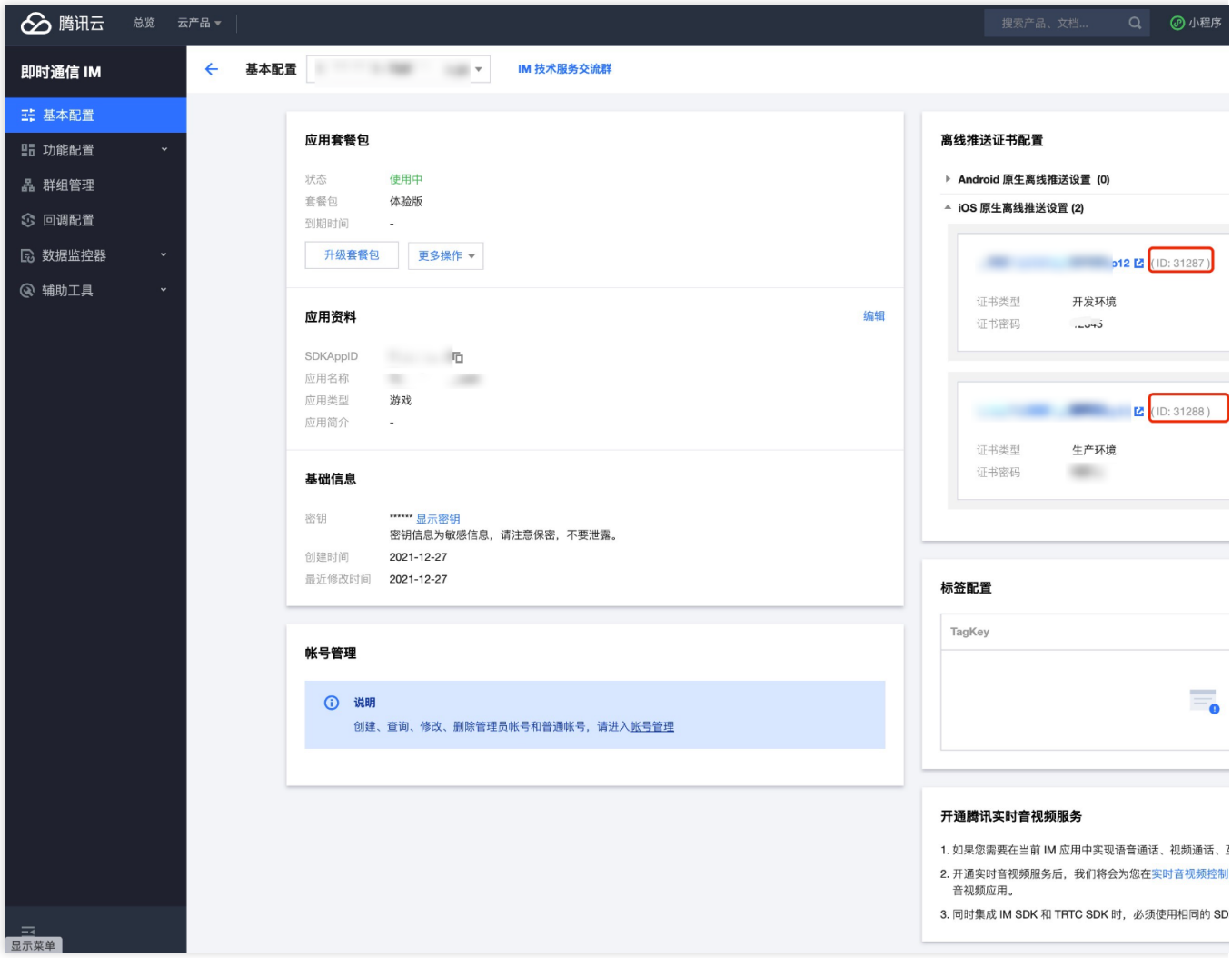
上传证书名最好使用全英文（尤其不能使用括号等特殊字符）。

上传证书需要设置密码，无密码收不到推送。

发布 App Store 的证书需要设置为生产环境，否则无法收到推送。

上传的 p12 证书必须是自己申请的真实有效的证书。

5. 待推送证书信息生成后，记录证书的 ID。

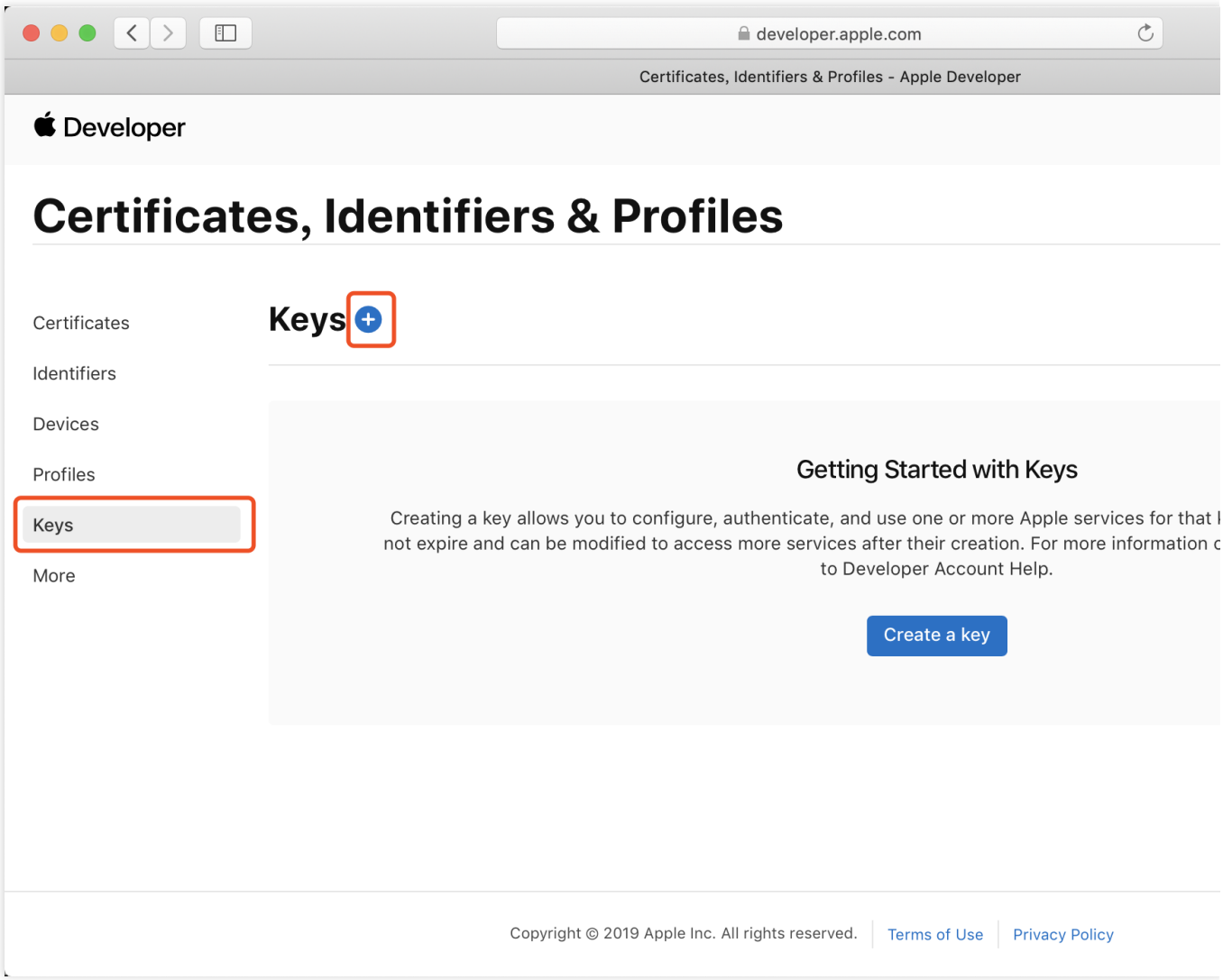


二、使用 p8 证书 （支持灵动岛推送）

p8 证书：p8 证书没有到期日期，因此您无需担心证书过期。此外，使用 p8 证书可以简化证书管理，因为您可以使用一个p8 证书为多个应用程序提供推送通知服务。另外，p8 证书支持灵动岛推送。

步骤1：申请 APNs 证书

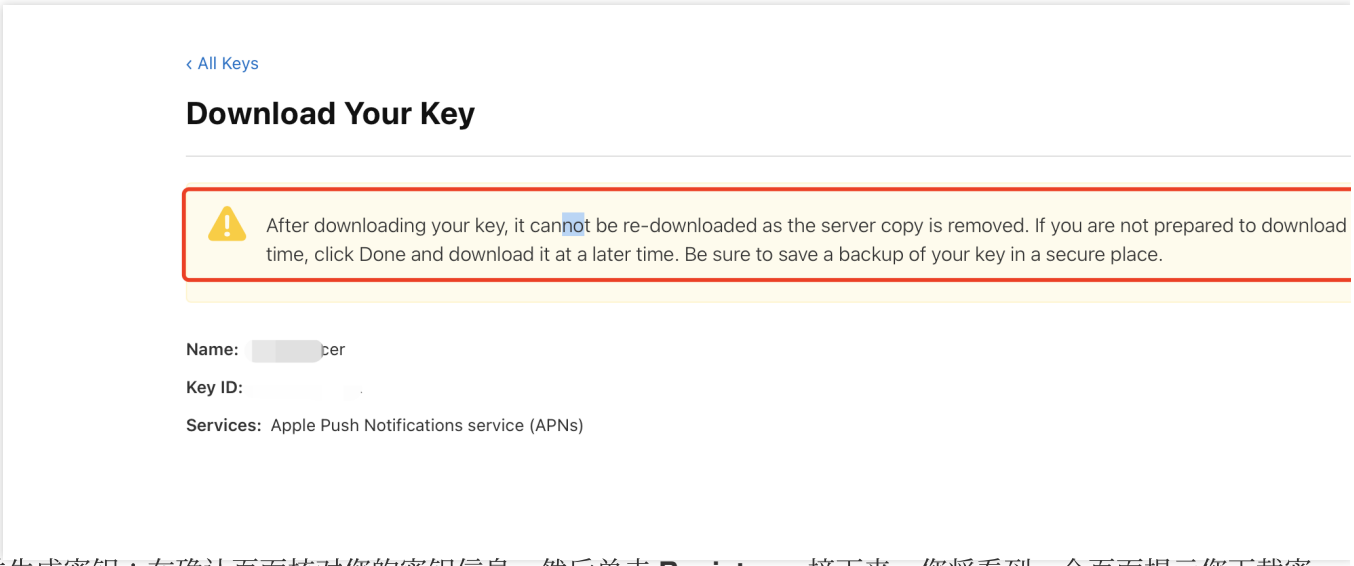
要创建 p8 证书文件，首先需要登录 苹果开发者中心。



1. 进入Certificates, Identifiers & Profiles：在页面右上角单击 **Account**，然后在下拉菜单中选择 **Certificates, Identifiers & Profiles** 。

2. 创建一个新的 App ID：在左侧菜单中单击 **Identifiers** ，然后单击右侧的 **+** 创建一个新的 App ID。填写相应的信息并单击 **Continue** 。

3. 创建一个新的密钥：在左侧菜单中单击 **Keys**，然后单击右侧的 **+** 创建一个新的密钥。输入密钥的名称，然后勾选 **Apple Push Notifications service (APNs)**，单击 **Continue** 。



确认并生成密钥：在确认页面核对您的密钥信息，然后单击 **Register** 。接下来，您将看到一个页面提示您下载密钥。单击 **Download**，将生成的 .p8 文件保存到您的计算机上。

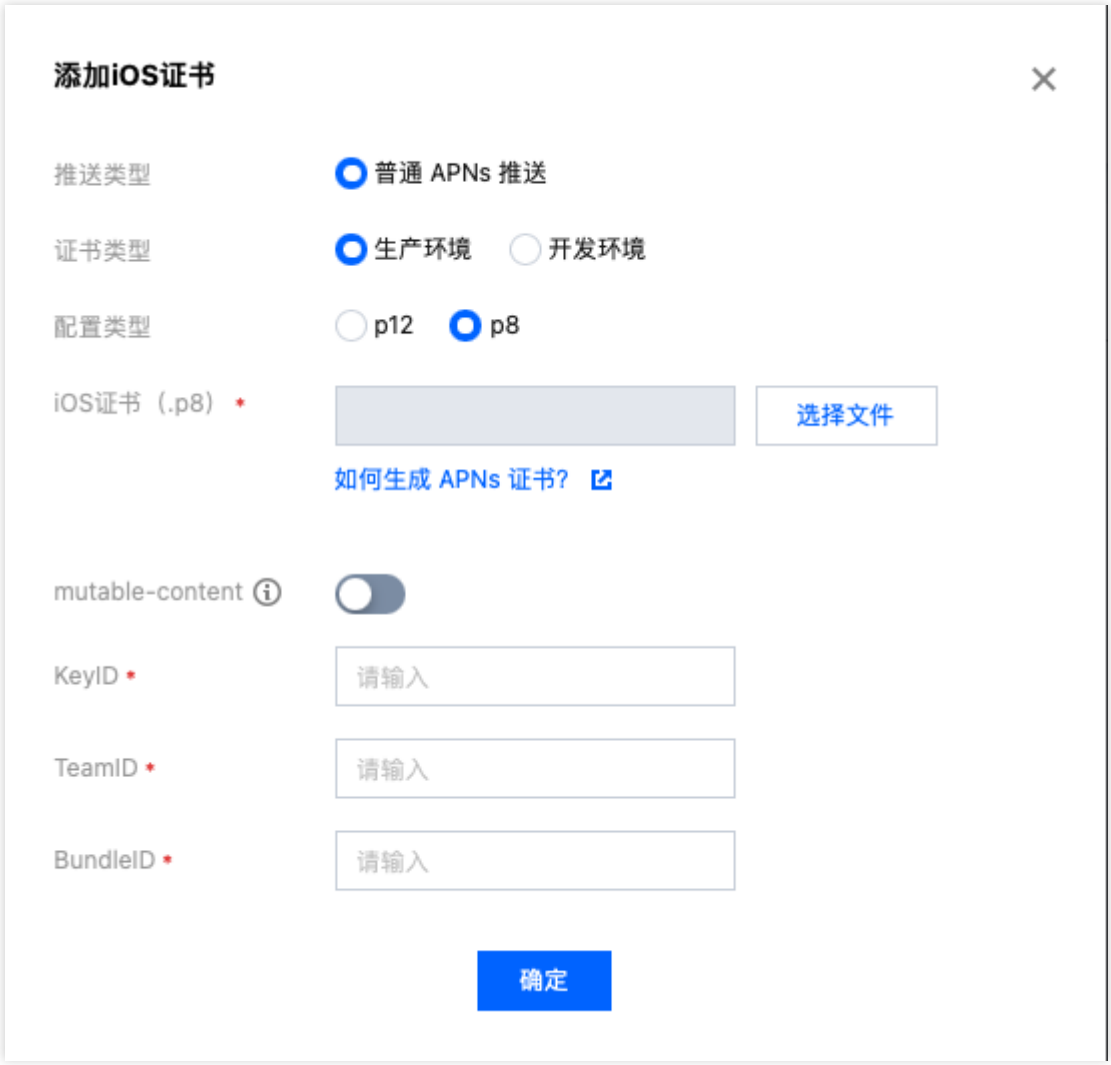
注意：

p8 证书只可以下载一次，请妥善保存。

请妥善保管下载的 p8 文件，因为您将无法再次下载该文件。您可以使用此P8证书配置您的iOS应用程序以接收推送通知。

步骤2：上传 p8 证书到IM控制台

1. 登录 即时通信 IM 控制台。
2. 单击目标应用卡片，进入应用的基础配置页面。
3. 单击 **iOS 原生离线推送设置**右侧的**添加证书**。
4. 选择 .p8 证书



说明：

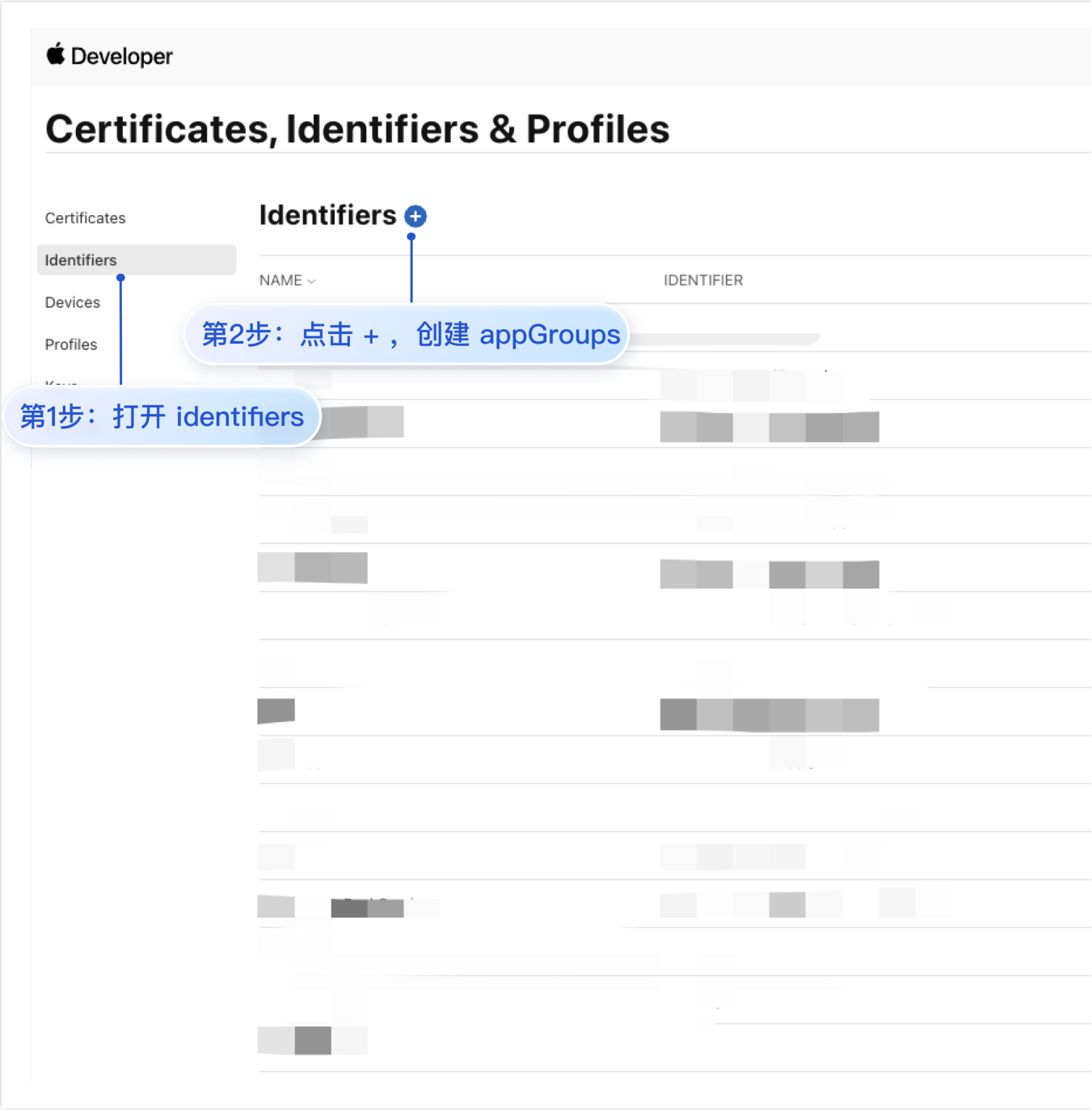
Key ID：这是您的 APNs Auth Key 的唯一标识符。当您在 Apple Developer Center 创建一个新的 APNs Auth Key 时，系统会为您生成一个 Key ID。您可以在 "Certificates, Identifiers & Profiles" 部分的 "Keys" 中找到它。

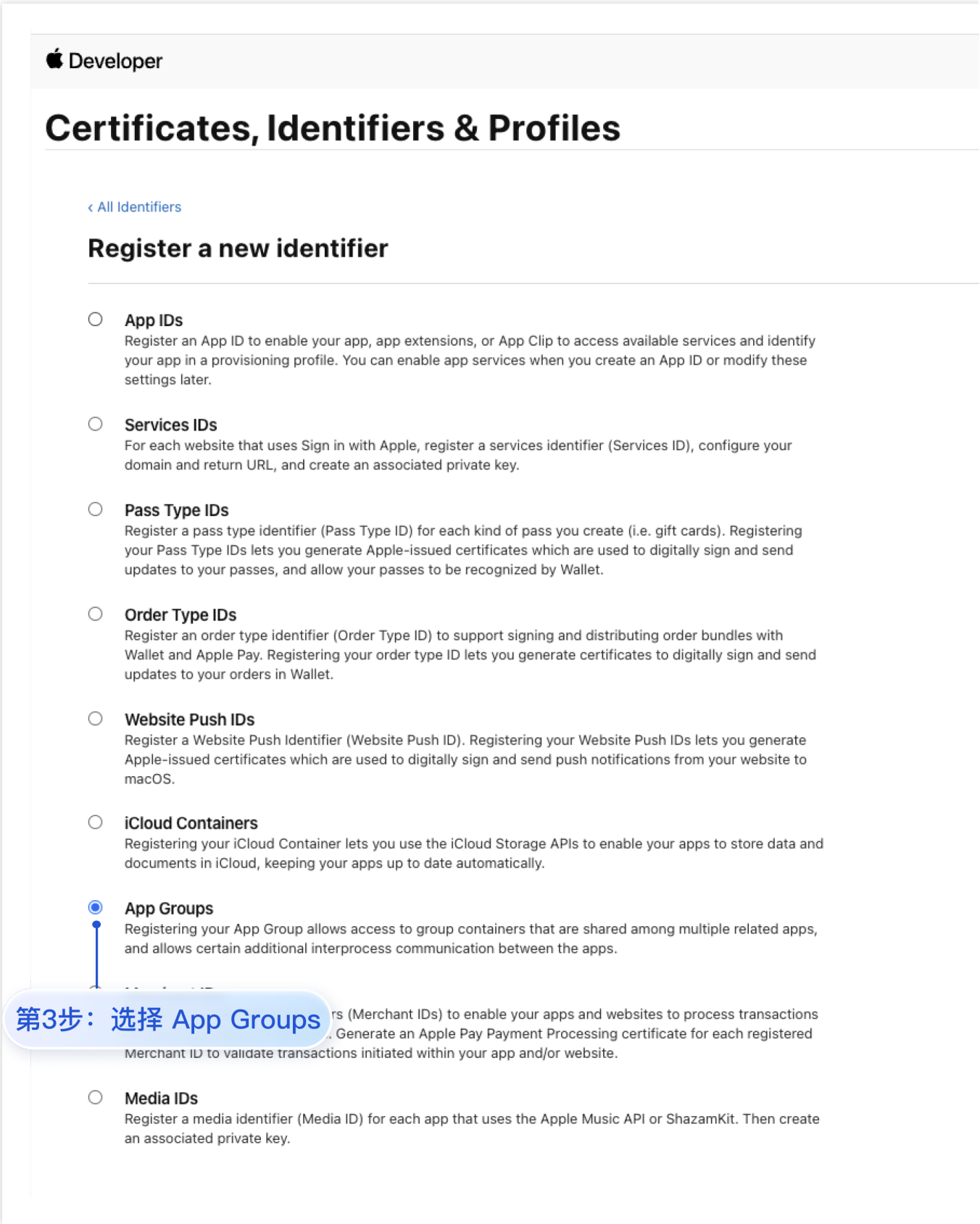
Team ID：这是您的开发者账户的唯一标识符。您可以在 Apple Developer Center 的账户详情页面找到它。点击右上角的 "Membership"，在 "Membership Details" 部分可以找到您的 Team ID。

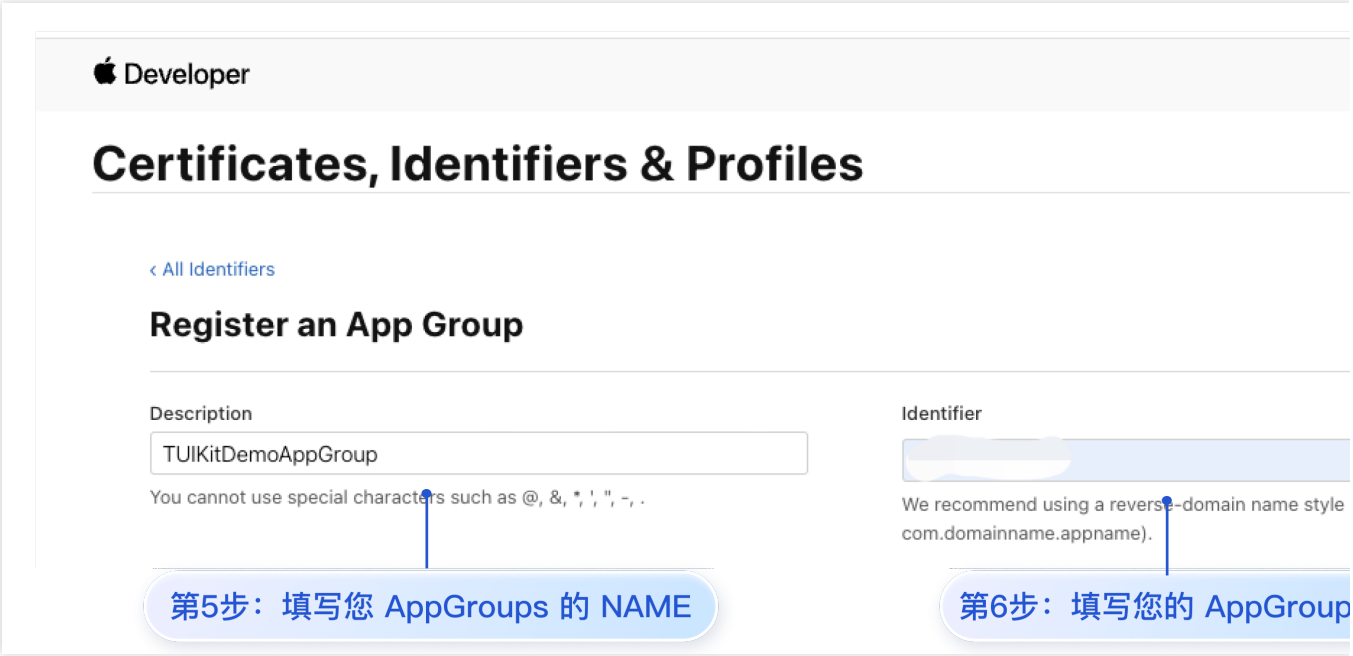
Bundle ID：这是您的应用程序的唯一标识符，也称为应用程序 ID。您可以在 Apple Developer Center 的 "Certificates, Identifiers & Profiles" 部分找到它。选择 "Identifiers"，然后在您的应用程序列表中找到对应的 Bundle ID。

三. 生成 TIMPushAppGroupID

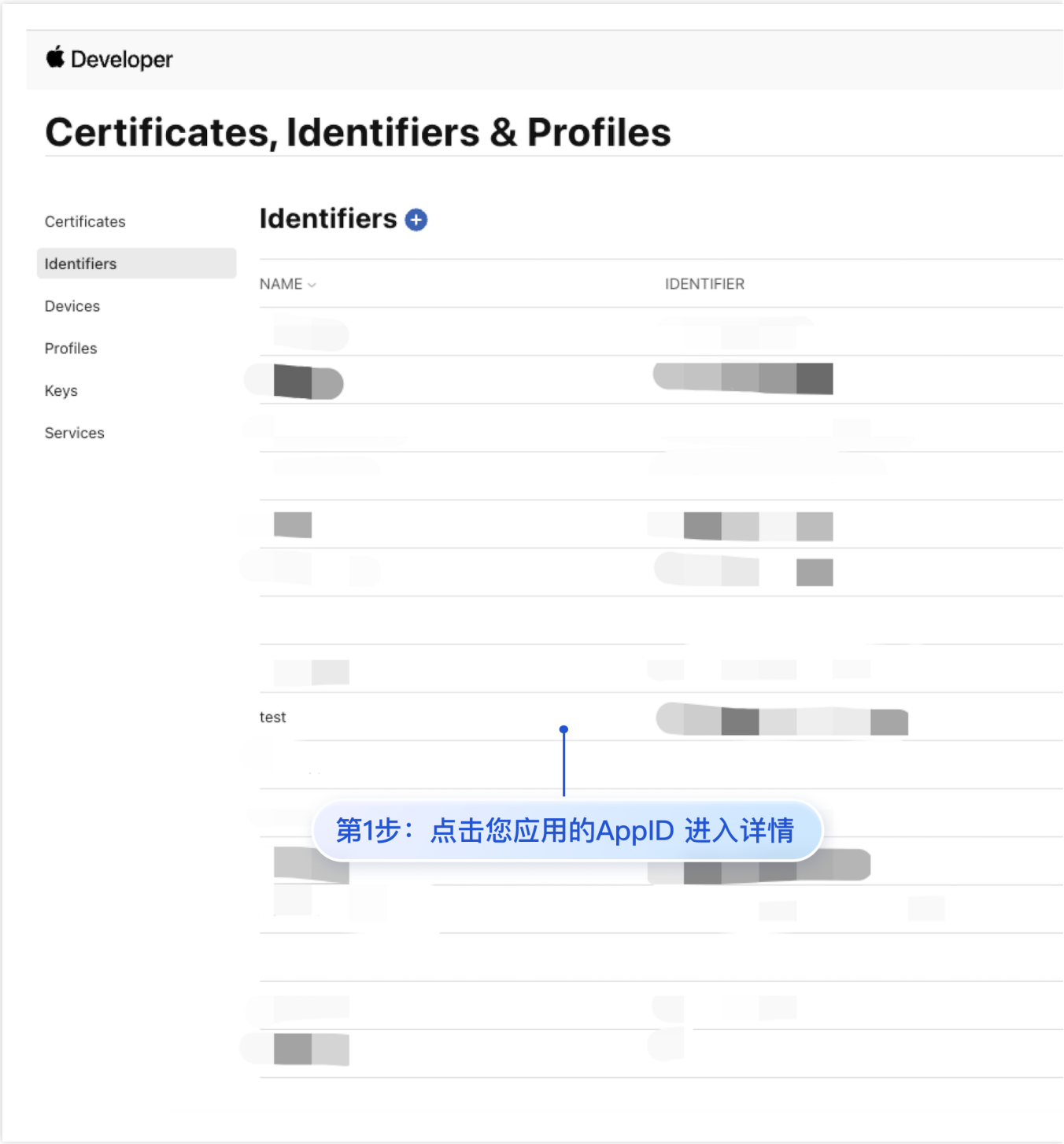
步骤1：登录苹果开发者中心网站，进入【identifiers】 ->【App Groups】创建 AppGroups。

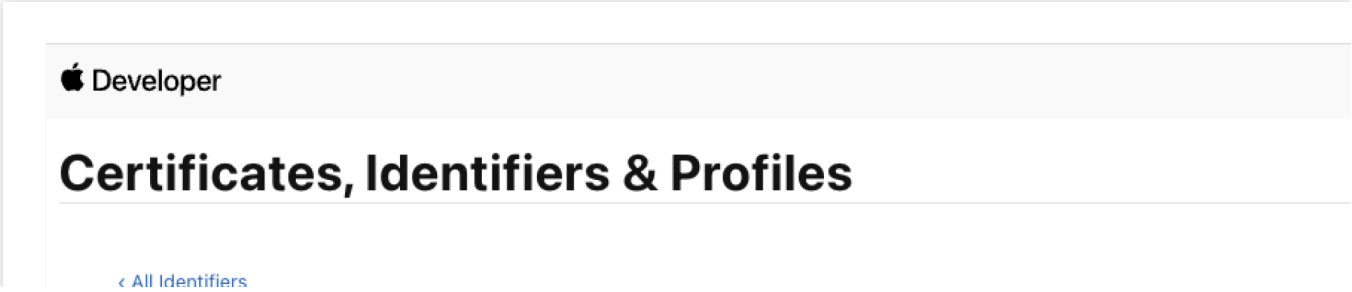

Developer
Certificates, Identifiers & Profiles
Certificates
Identifiers
Devices
Profiles
Identifiers
NAME
IDENTIFIER
第2步：点击 + ，创建 appGroups
第1步：打开 identifiers

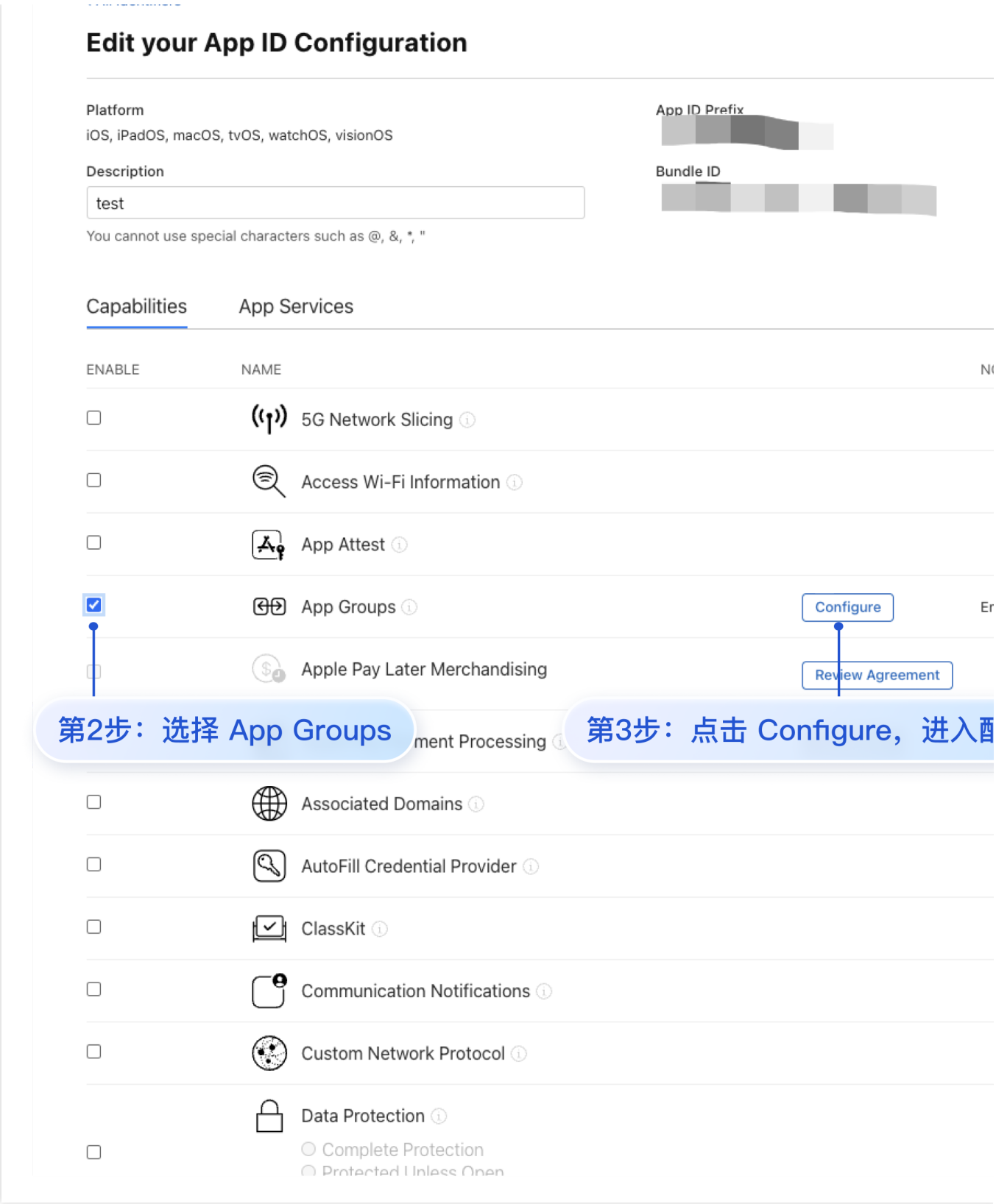

Developer
Certificates, Identifiers & Profiles
‹ All Identifiers
Register a new identifier
App IDs
Register an App ID to enable your app, app extensions, or App Clip to access available services and identify your app in a provisioning profile. You can enable app services when you create an App ID or modify these settings later.
Services IDs
For each website that uses Sign in with Apple, register a services identifier (Services ID), configure your domain and return URL, and create an associated private key.
Pass Type IDs
Register a pass type identifier (Pass Type ID) for each kind of pass you create (i.e. gift cards). Registering your Pass Type IDs lets you generate Apple-issued certificates which are used to digitally sign and send updates to your passes, and allow your passes to be recognized by Wallet.
Order Type IDs
Register an order type identifier (Order Type ID) to support signing and distributing order bundles with Wallet and Apple Pay. Registering your order type ID lets you generate certificates to digitally sign and send updates to your orders in Wallet.
Website Push IDs
Register a Website Push Identifier (Website Push ID). Registering your Website Push IDs lets you generate Apple-issued certificates which are used to digitally sign and send push notifications from your website to macOS.
iCloud Containers
Registering your iCloud Container lets you use the iCloud Storage APIs to enable your apps to store data and documents in iCloud, keeping your apps up to date automatically.
App Groups
Registering your App Group allows access to group containers that are shared among multiple related apps, and allows certain additional interprocess communication between the apps.
第3步：选择 App Groups
rs (Merchant IDs) to enable your apps and websites to process transactions
. Generate an Apple Pay Payment Processing certificate for each registered
Merchant ID to validate transactions initiated within your app and/or website.
Media IDs
Register a media identifier (Media ID) for each app that uses the Apple Music API or ShazamKit. Then create an associated private key.

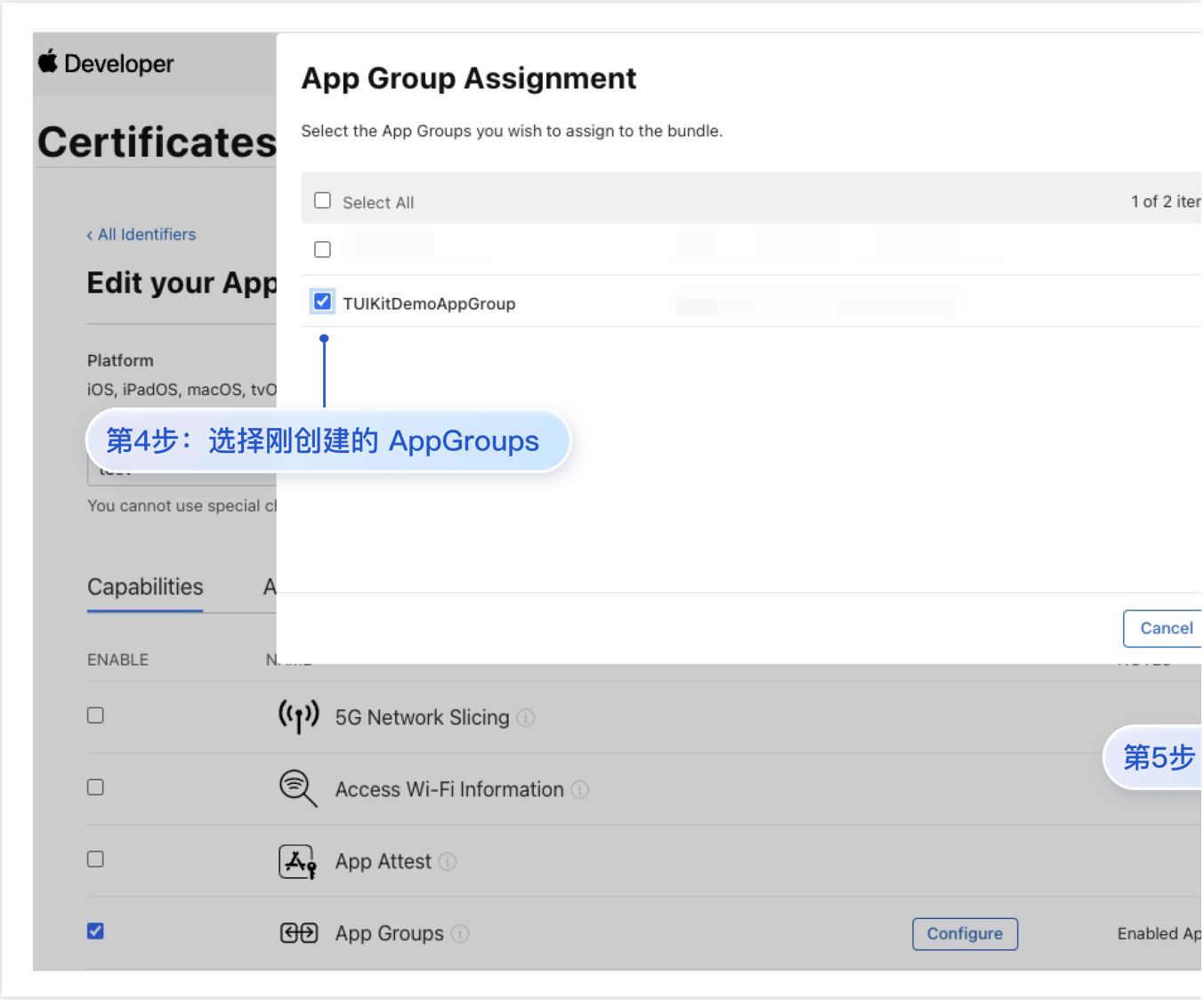


步骤2：绑定使用 TencentCloud-TIMPush 应用的 appID 到 AppGroups。

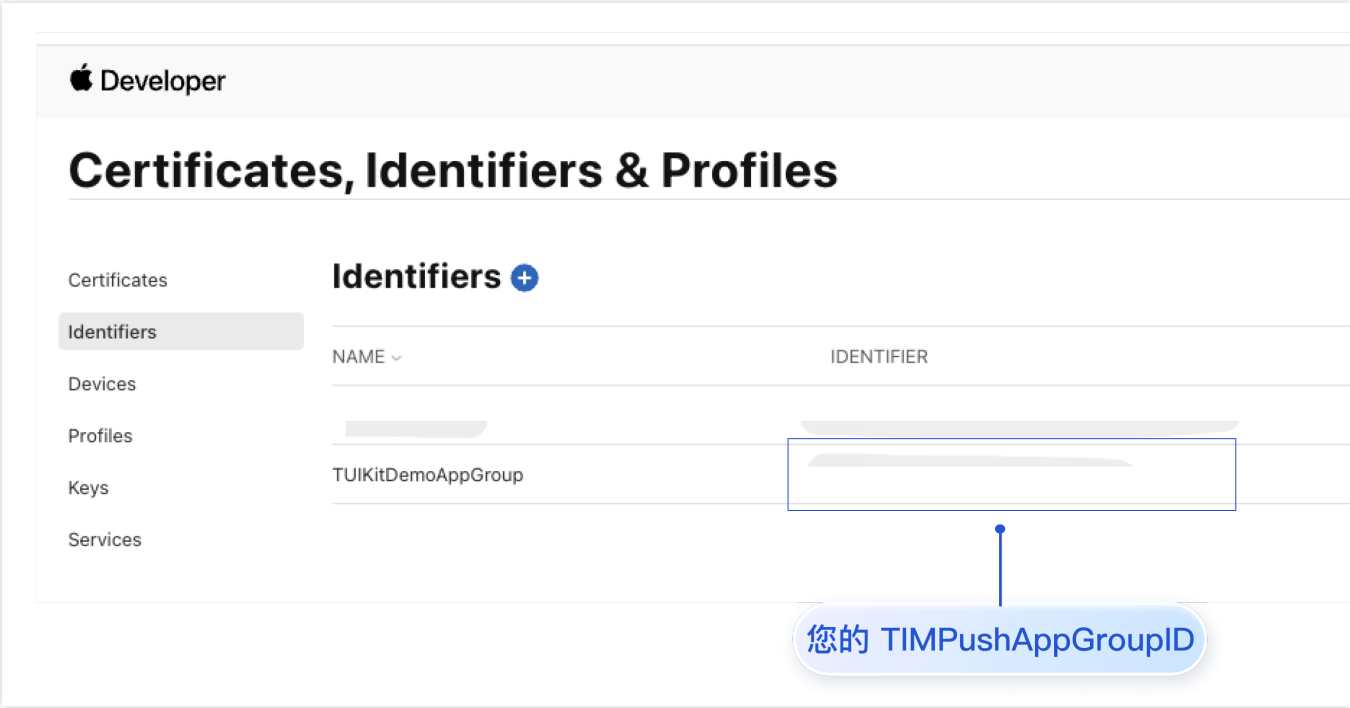

Developer
Certificates, Identifiers & Profiles
Certificates
Identifiers
Devices
Profiles
Keys
Services
Identifiers
NAME
IDENTIFIER
test
第1步：点击您应用的AppID 进入详情


Developer
Certificates, Identifiers & Profiles
All Identifiers


Edit your App ID Configuration
Platform
iOS, iPadOS, macOS, tvOS, watchOS, visionOS
App ID Prefix
Description
test
Bundle ID
You cannot use special characters such as @, &, *, "
Capabilities
App Services
ENABLE
NAME
5G Network Slicing
Access Wi-Fi Information
App Attest
App Groups
Configure
Apple Pay Later Merchandising
Review Agreement
第2步：选择 App Groups
第3步：点击 Configure，进入配
Associated Domains
AutoFill Credential Provider
ClassKit
Communication Notifications
Custom Network Protocol
Data Protection
Complete Protection



步骤3. 获取您的 TIMPushAppGroupID。


Developer
Certificates, Identifiers & Profiles
Certificates
Identifiers
Devices
Profiles
Keys
Services
Identifiers
NAME
IDENTIFIER
TUIKitDemoAppGroup
您的 TIMPushAppGroupID

Flutter

最近更新时间：2024-06-13 10:39:26

目前消息推送插件, 在 Flutter 使用中, 仅支持推送至 Android (含各厂商通道) 和 iOS 设备.

iOS

Android

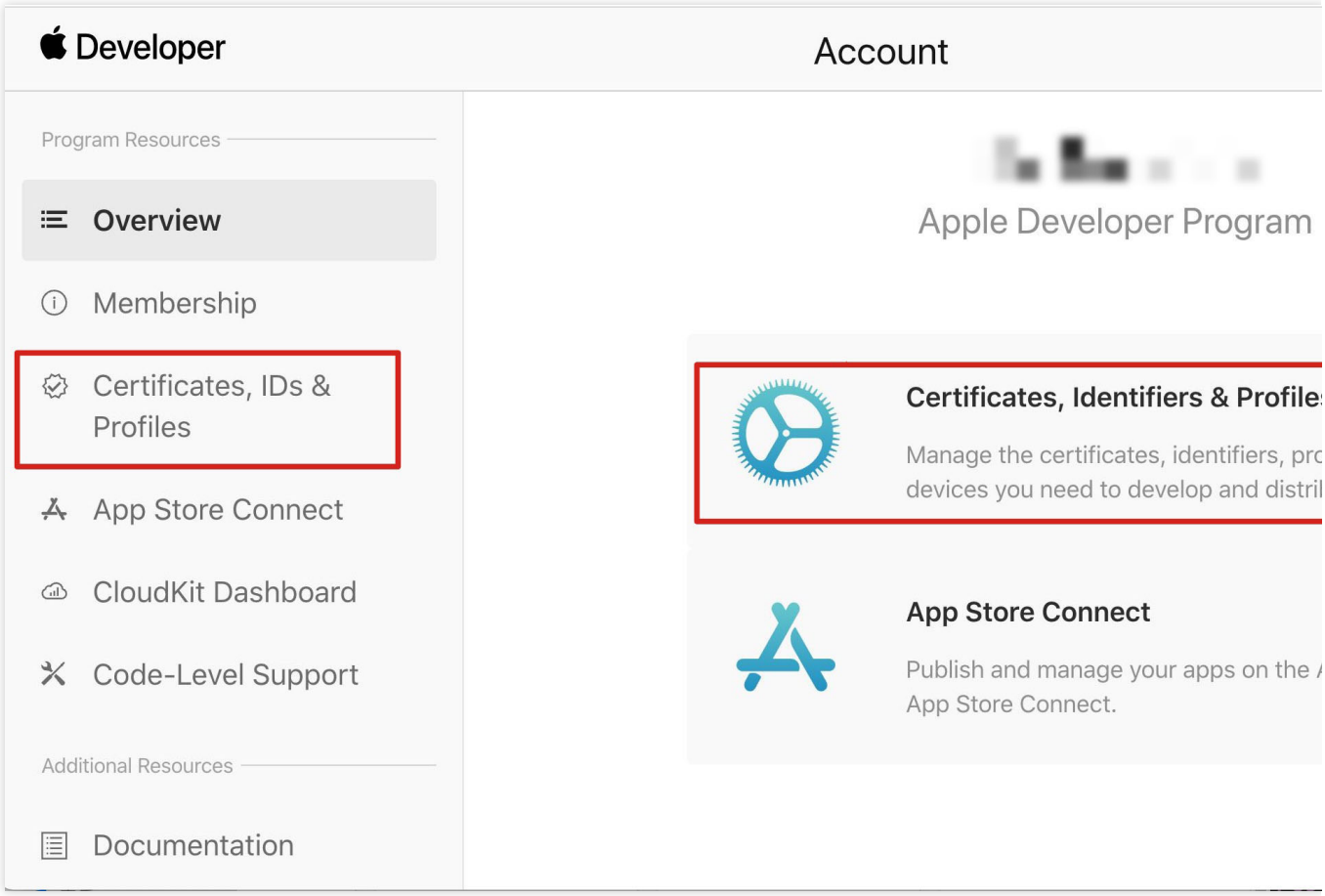
集成 消息推送 插件之前，需要先向 Apple 申请 APNs 推送证书，然后上传推送证书到 IM 控制台 。之后按照快速接入步骤接入即可。

操作步骤

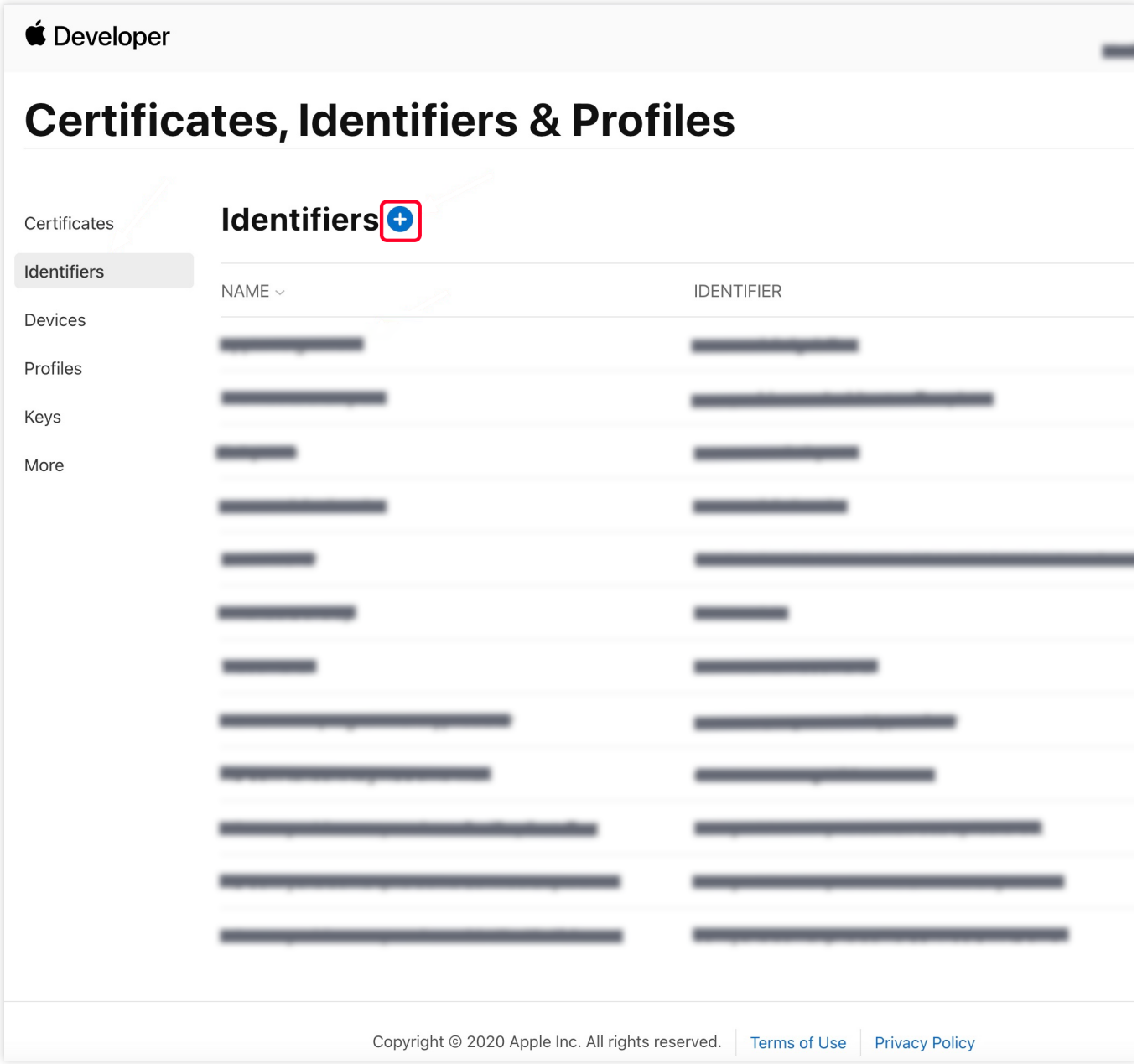
步骤1：申请 APNs 证书

开启 App 远程推送

1. 登录 苹果开发者中心 网站，单击 **Certificates,Identifiers & Profiles** 或者侧栏的 **Certificates,IDs & Profiles**，进入 **Certificates, IDS & Profiles** 页面。



2. 单击 Identifiers 右侧的 +。

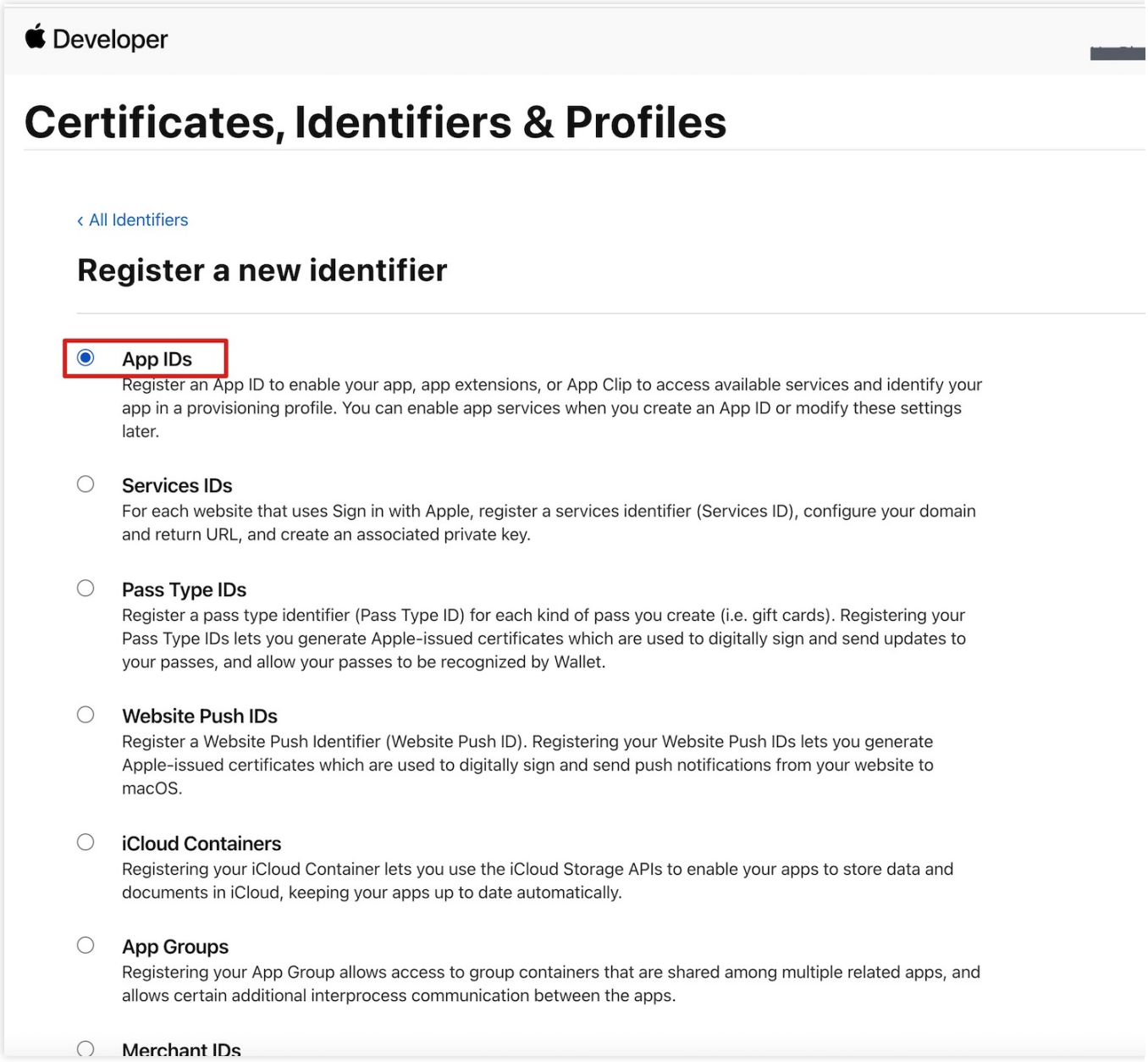


3. 您可以参见如下步骤新建一个 AppID，或者在您原有的 AppID 上增加 `Push Notification` 的 `Service` 。

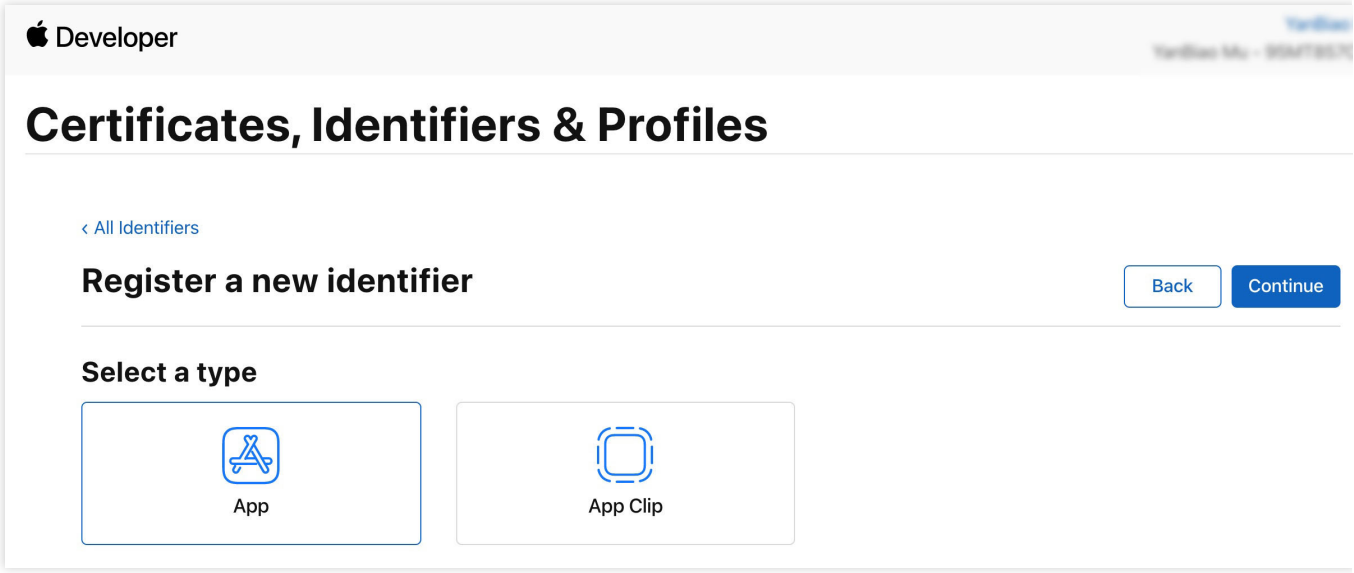
说明

您 App 的 `Bundle ID` 不能使用通配符 `*` ，否则将无法使用远程推送服务。

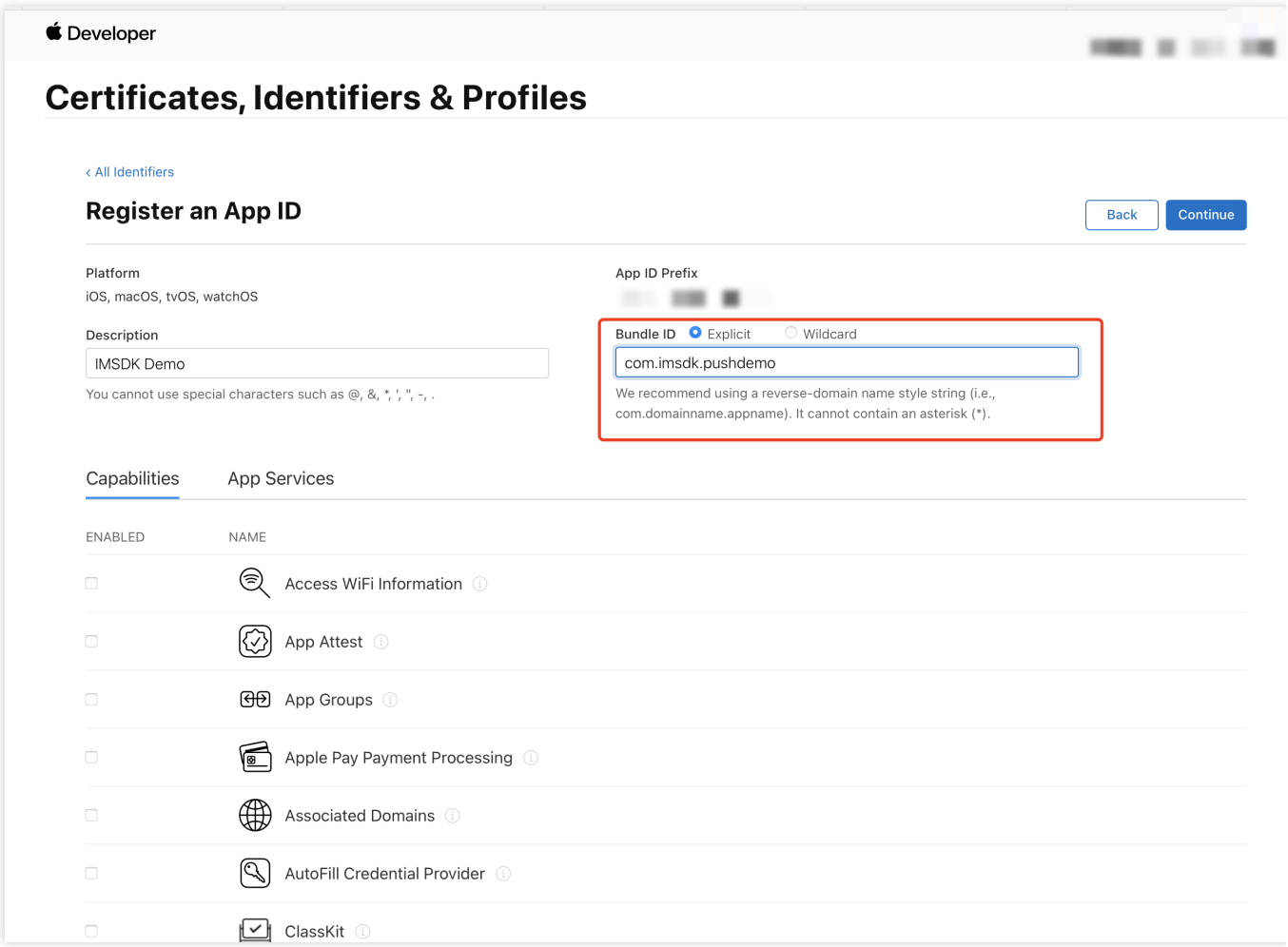
4. 勾选 **App IDs**，单击 **Continue** 进行下一步。



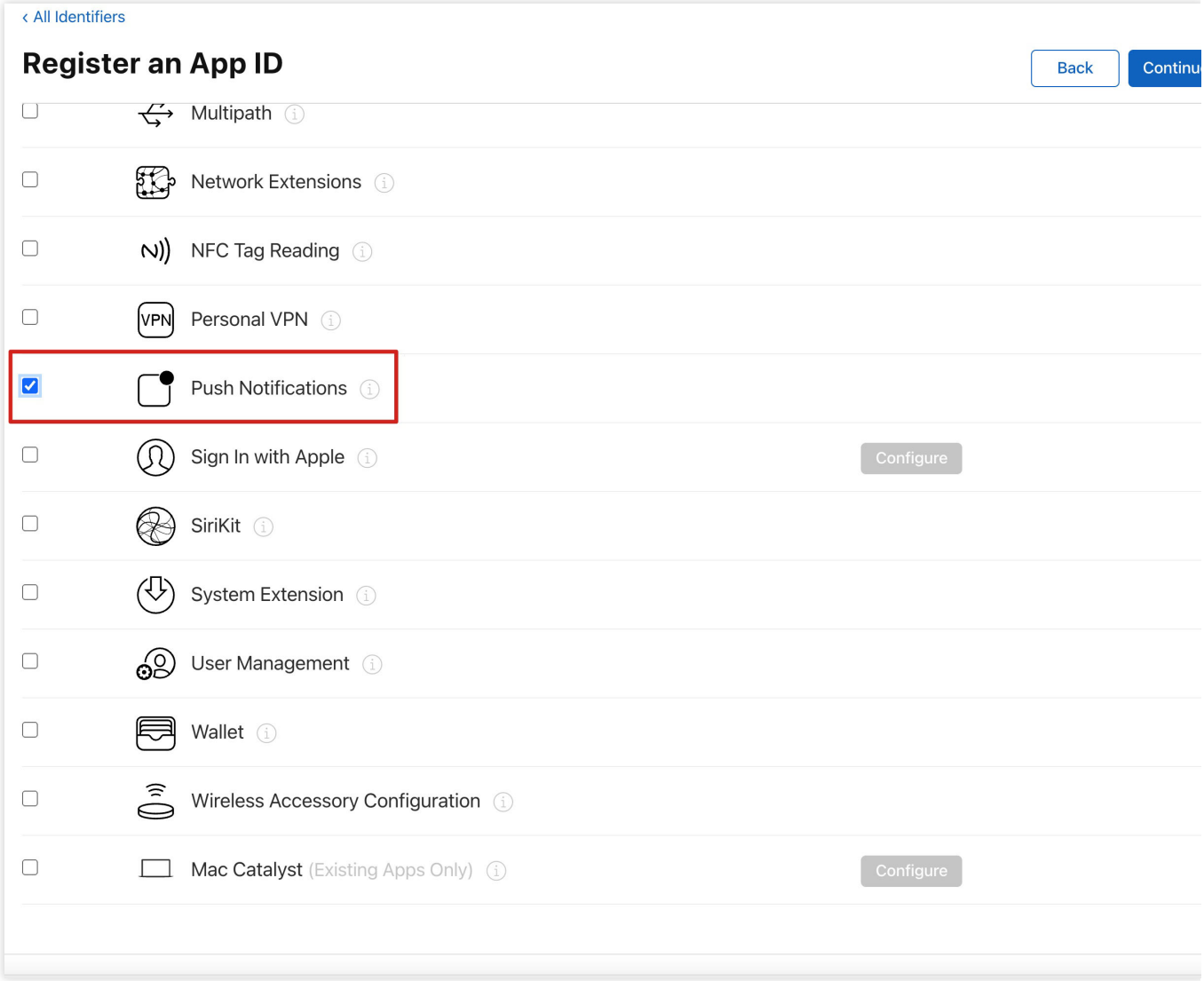
5. 选择 **App**，单击 **Continue** 进行下一步。



6. 配置 `Bundle ID` 等其他信息，单击 **Continue** 进行下一步。

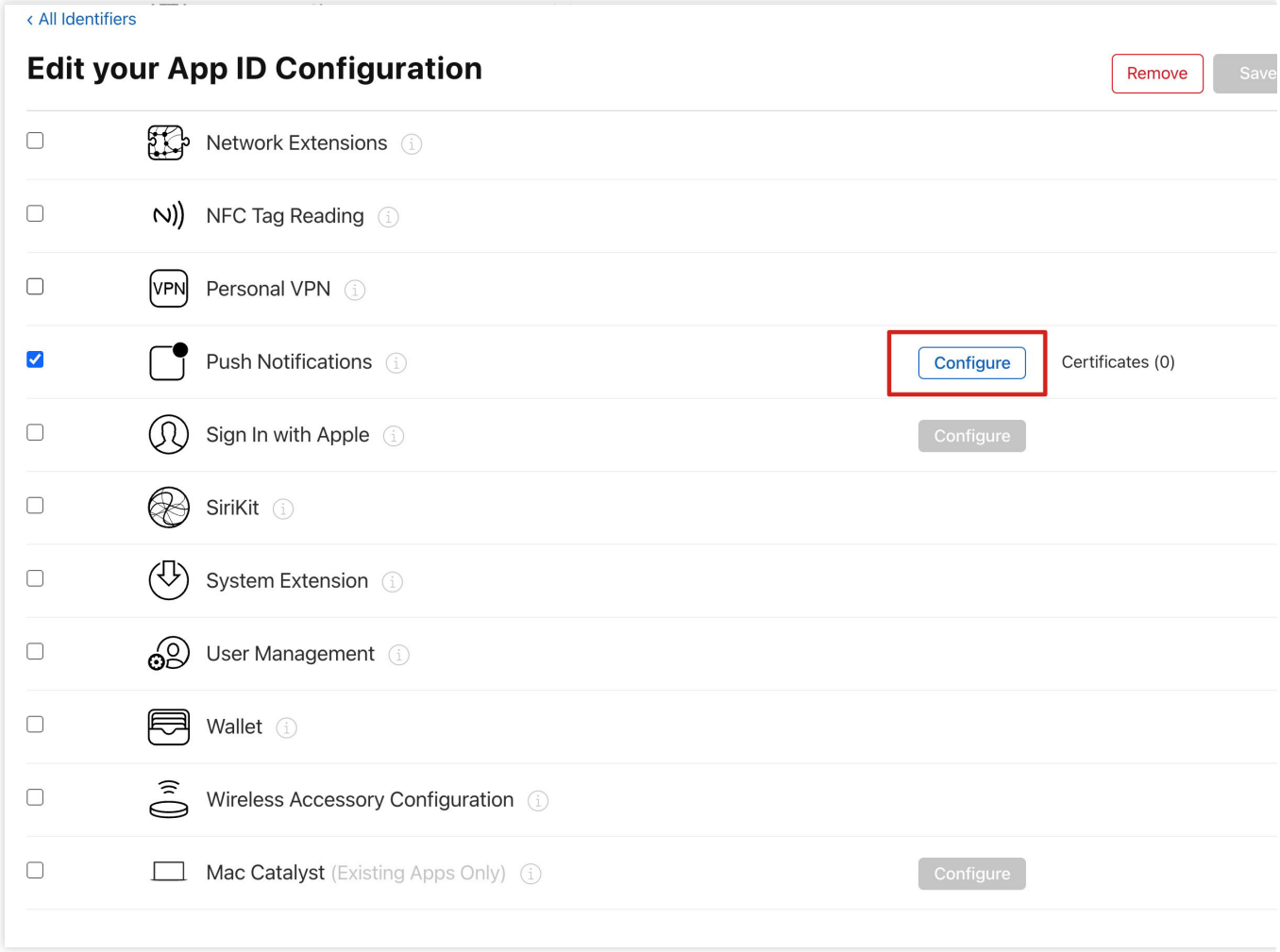


7. 勾选 **Push Notifications**，开启远程推送服务。

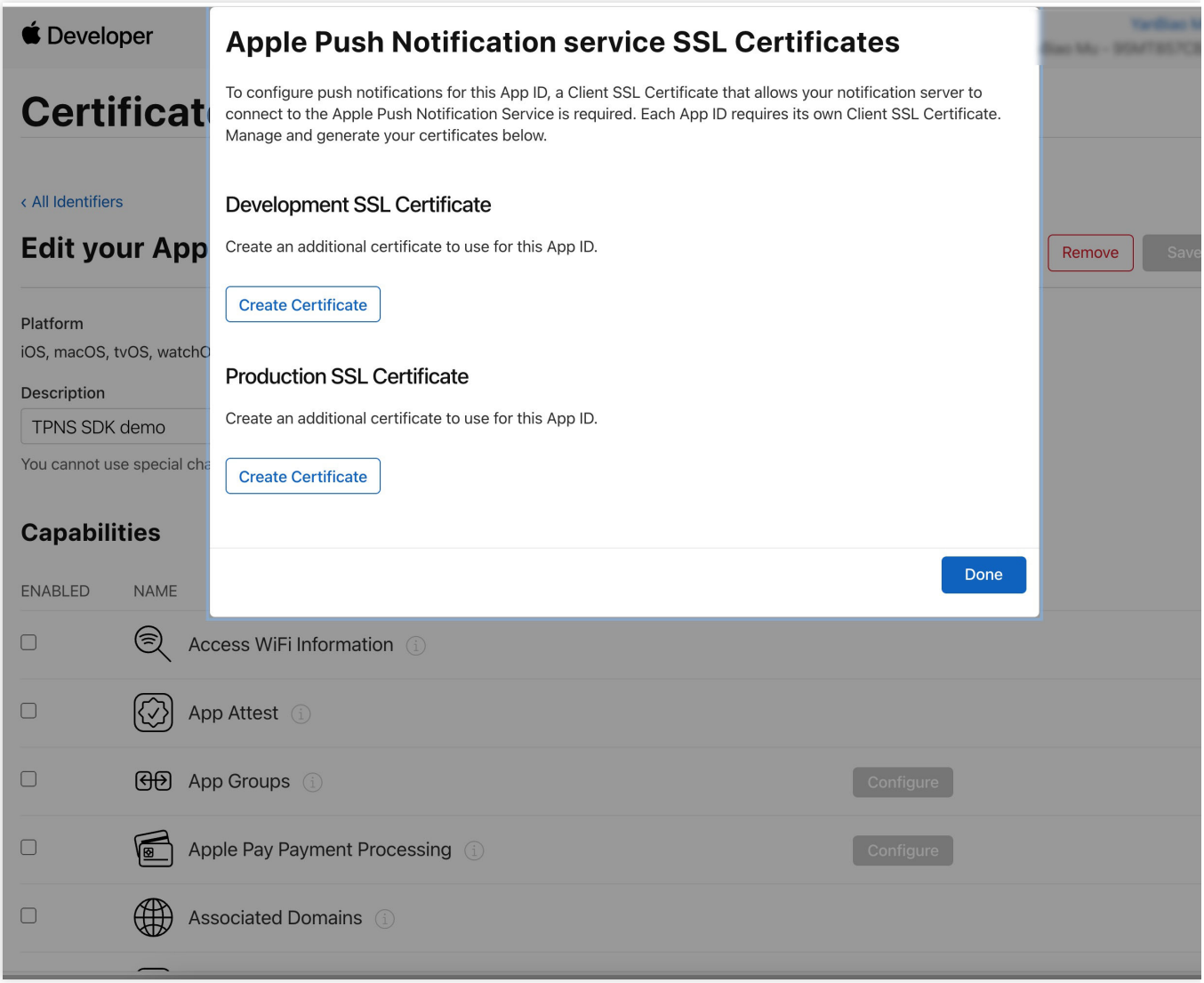


生成证书

1. 选中您的 AppID，选择 **Configure**。



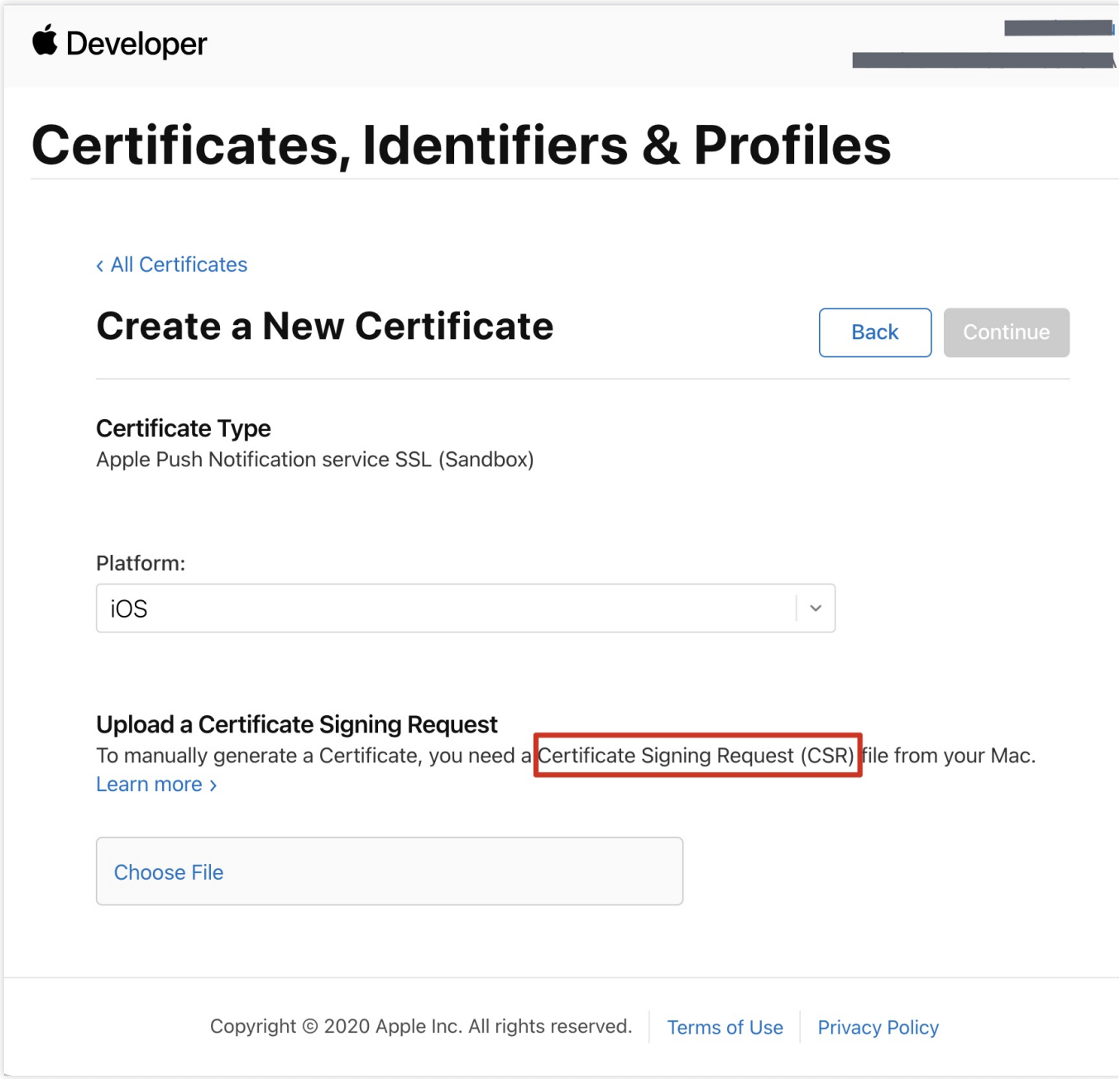
2. 可以看到在 **Apple Push Notification service SSL Certificates** 窗口中有两个 `SSL Certificate`，分别用于开发环境（Development）和生产环境（Production）的远程推送证书，如下图所示：



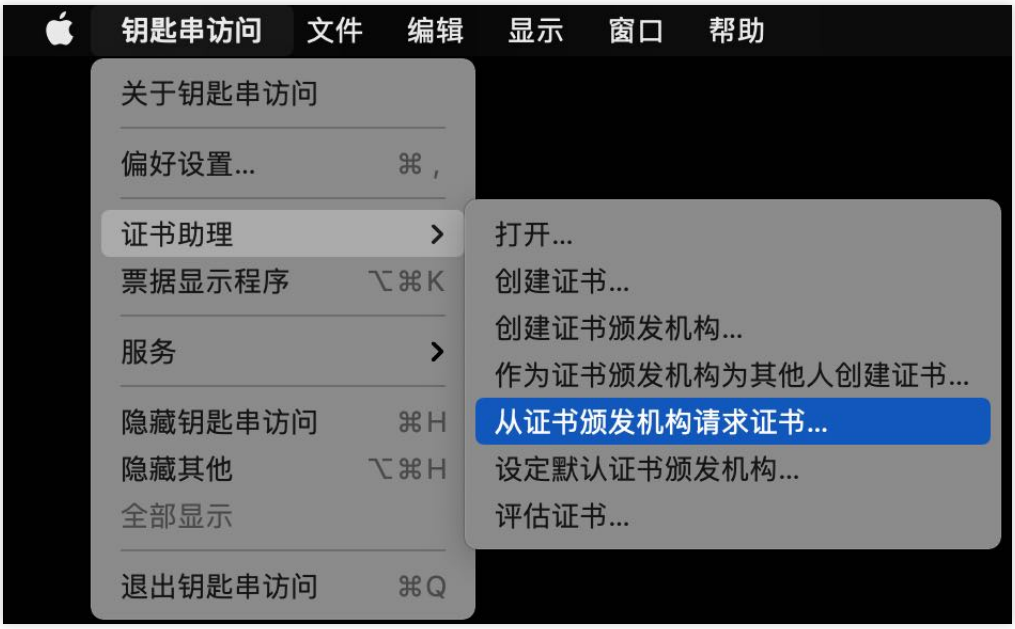
3.

我

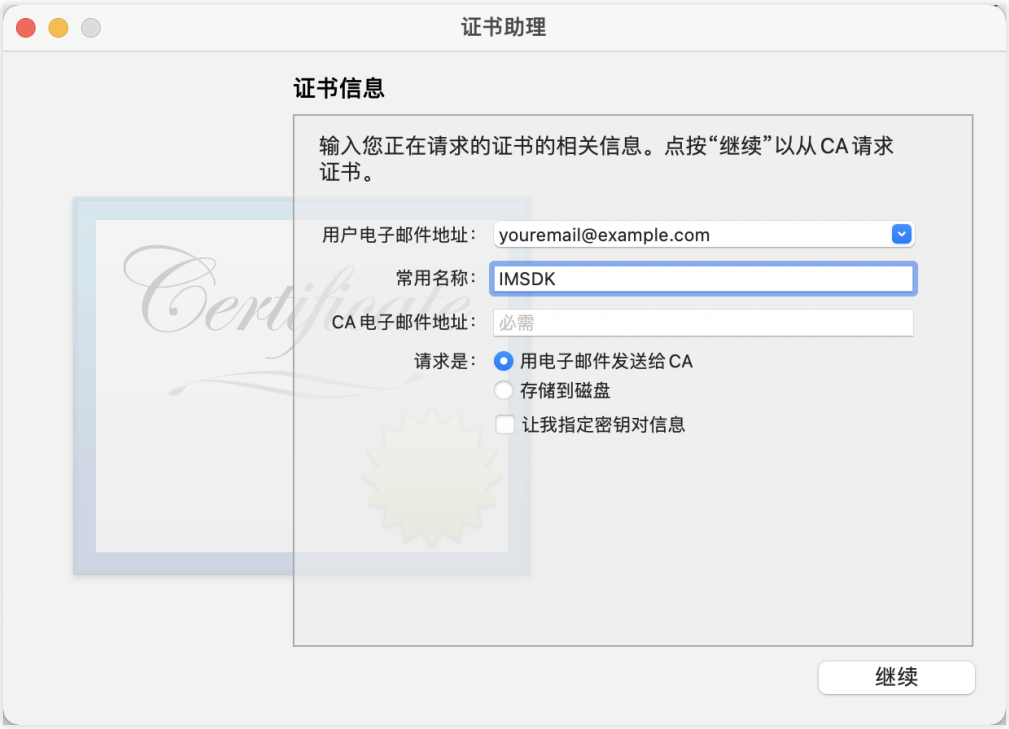
们先选择开发环境（Development）的 **Create Certificate**，系统将提示我们需要一个 Certificate Signing Request（CSR）。



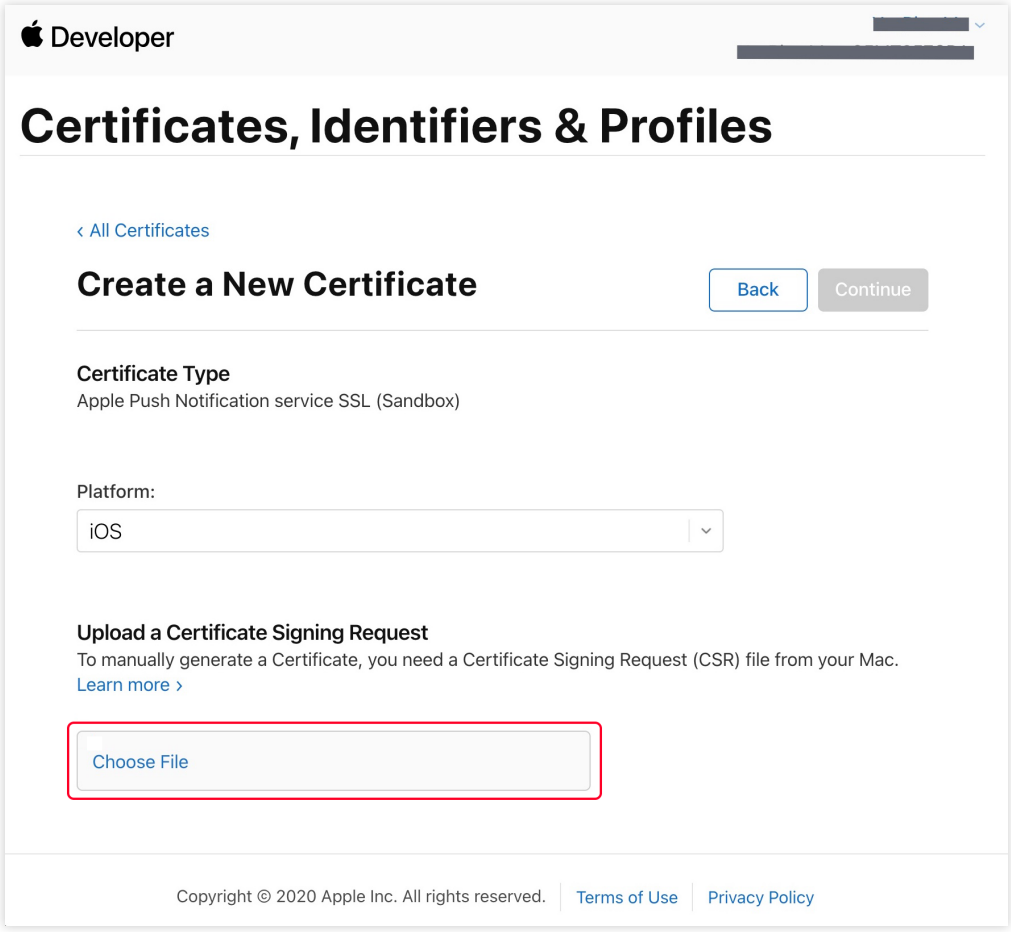
4. 在 Mac 上打开**钥匙串访问工具（Keychain Access）**，在菜单中选择**钥匙串访问 > 证书助理 > 从证书颁发机构请求证书**（`Keychain Access - Certificate Assistant - Request a Certificate From a Certificate Authority`）。



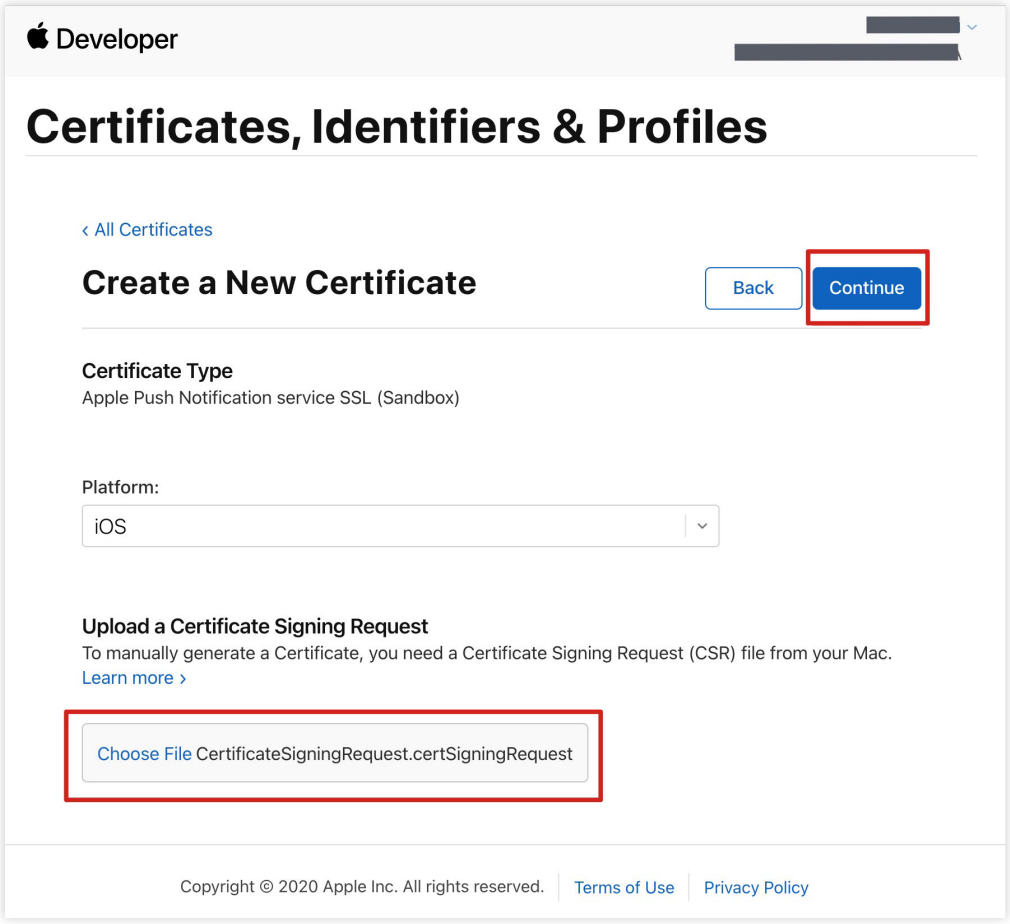
5. 输入用户电子邮件地址（您的邮箱）、常用名称（您的名称或公司名），选择**存储到磁盘**，单击**继续**，系统将生成一个 `*.certSigningRequest` 文件。



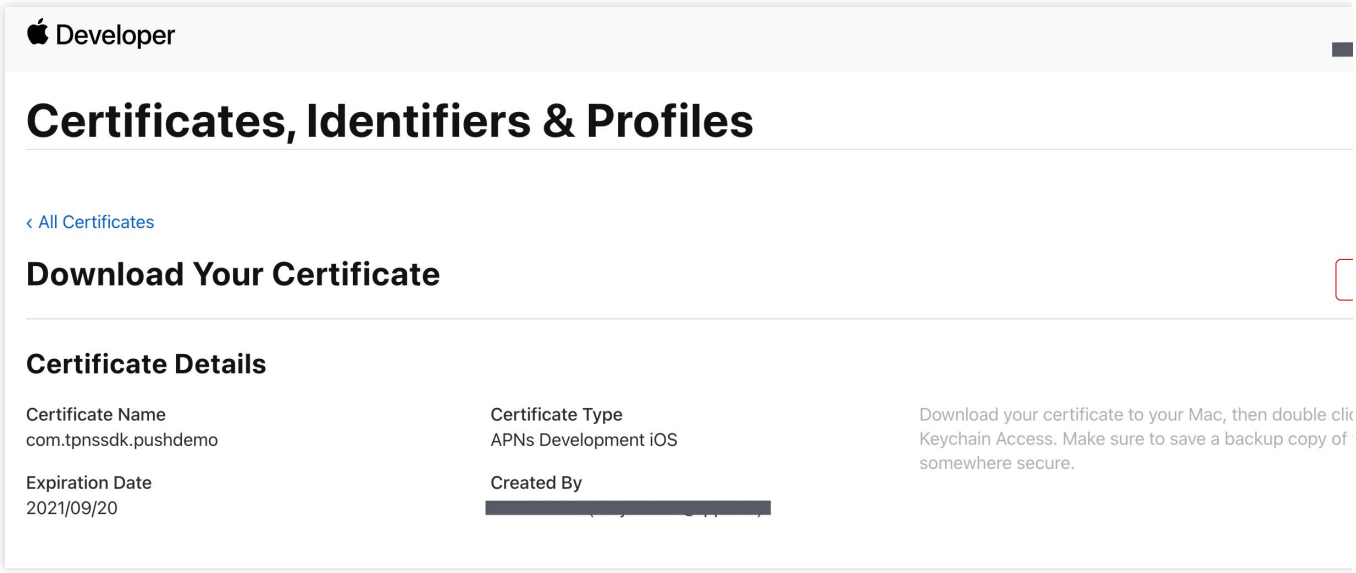
6. 返回上述 步骤3 中 Apple Developer 网站刚才的页面，单击 **Choose File** 上传生成的 `*.certSigningRequest` 文件。



7. 单击 **Continue**，即可生成推送证书。



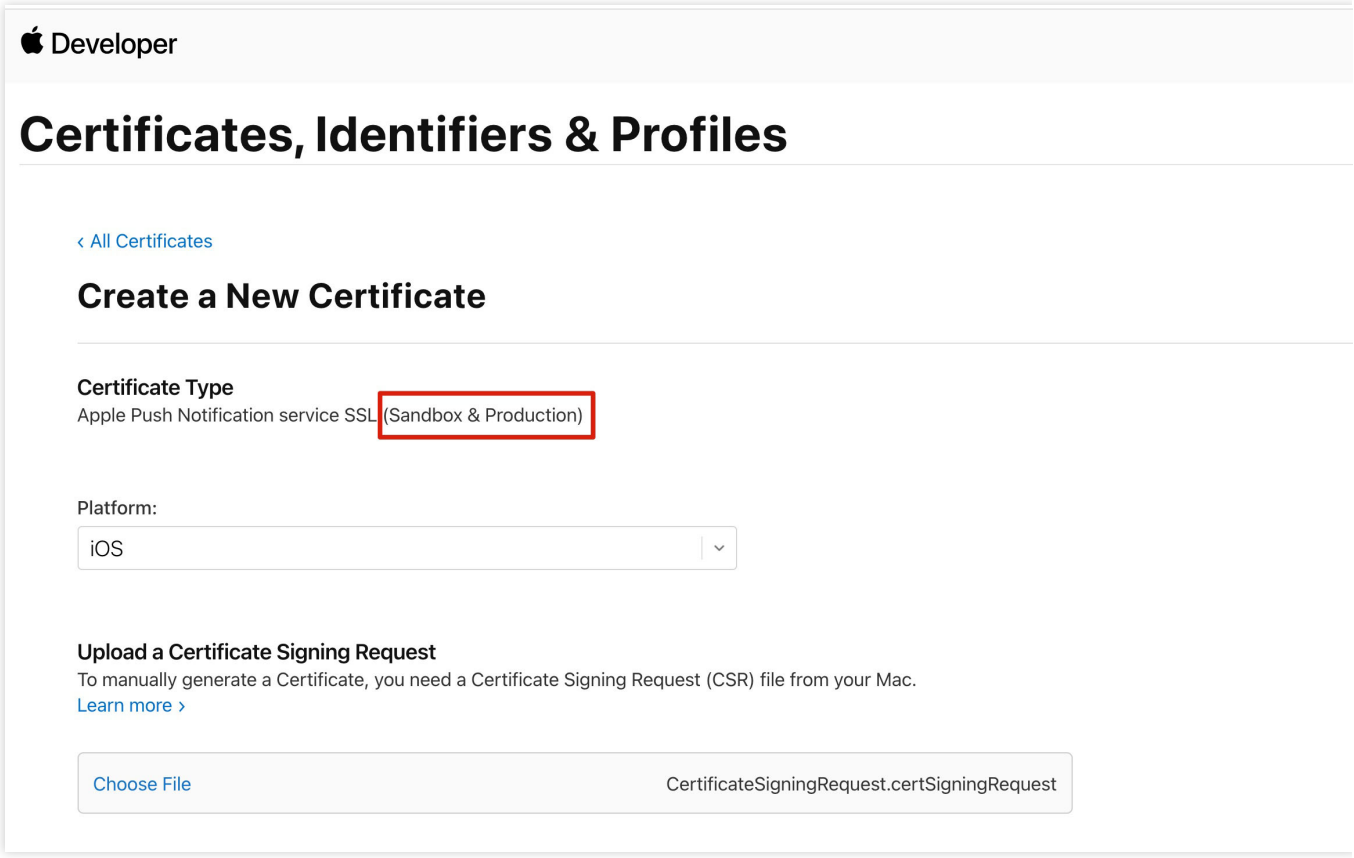
8. 单击 **Download** 下载开发环境的 `Development SSL Certificate` 到本地。

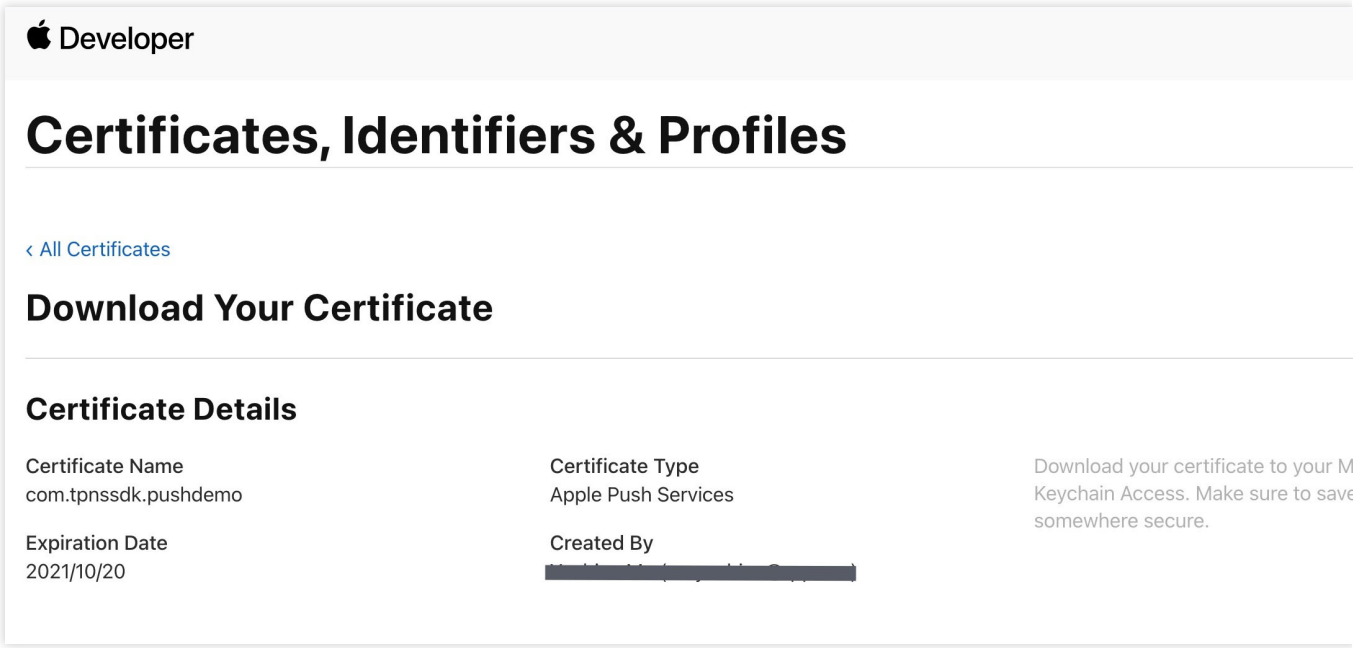


9. 再次按照上述步骤1 - 8，将生产环境的 `Production SSL Certificate` 下载到本地。

说明

生产环境的证书实际是开发（Sandbox）+生产（Production）的合并证书，可以同时作为开发环境和生产环境的证书使用。





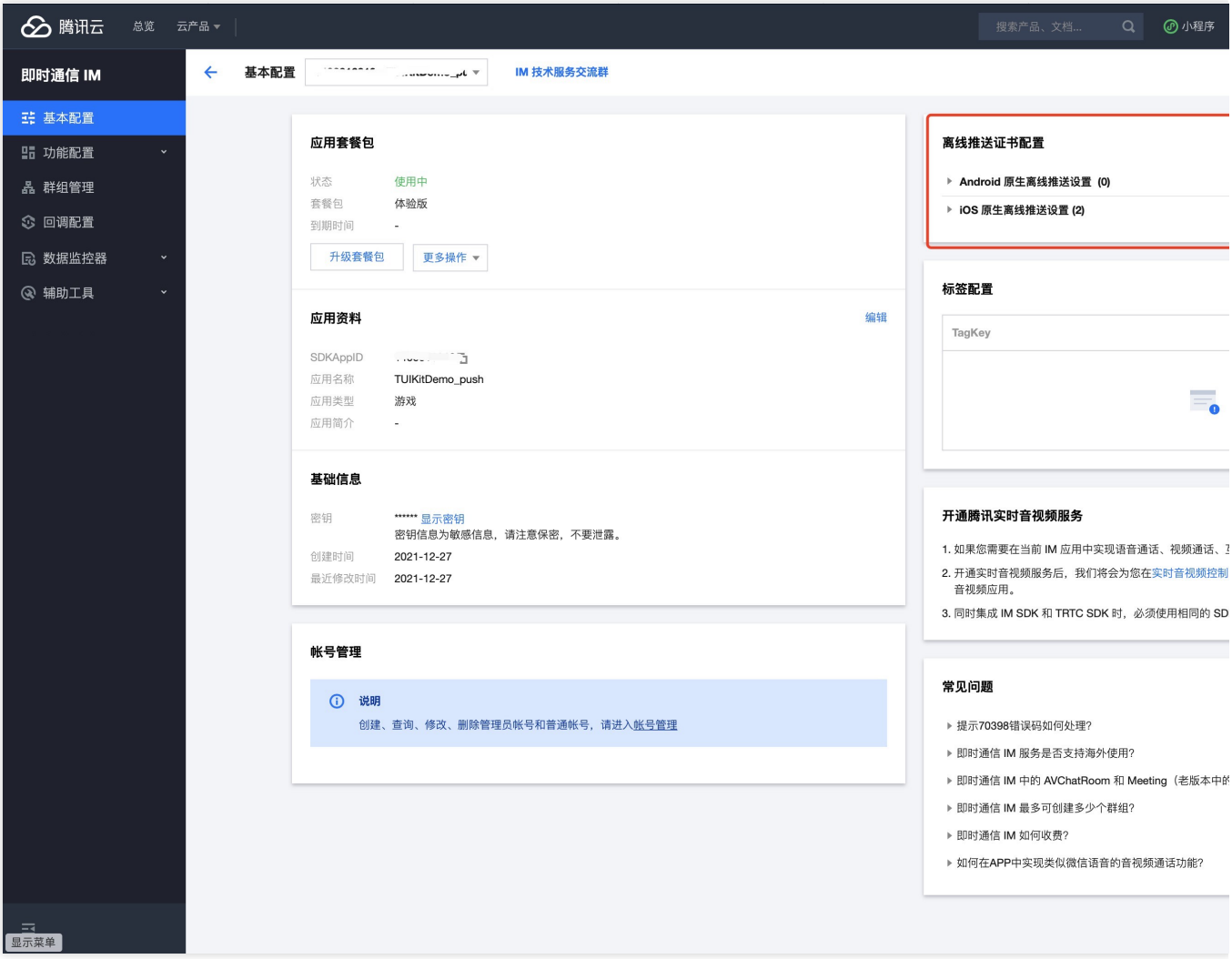
10. 双击打开下载的开发环境和生产环境的 `SSL Certificate` ，系统会将其导入钥匙串中。
11. 打开钥匙串应用，在**登录 > 我的证书**，右键分别导出刚创建的开发环境（ `Apple Development IOS Push Service` ）和生产环境（ `Apple Push Services` ）的 `P12` 文件。

注意

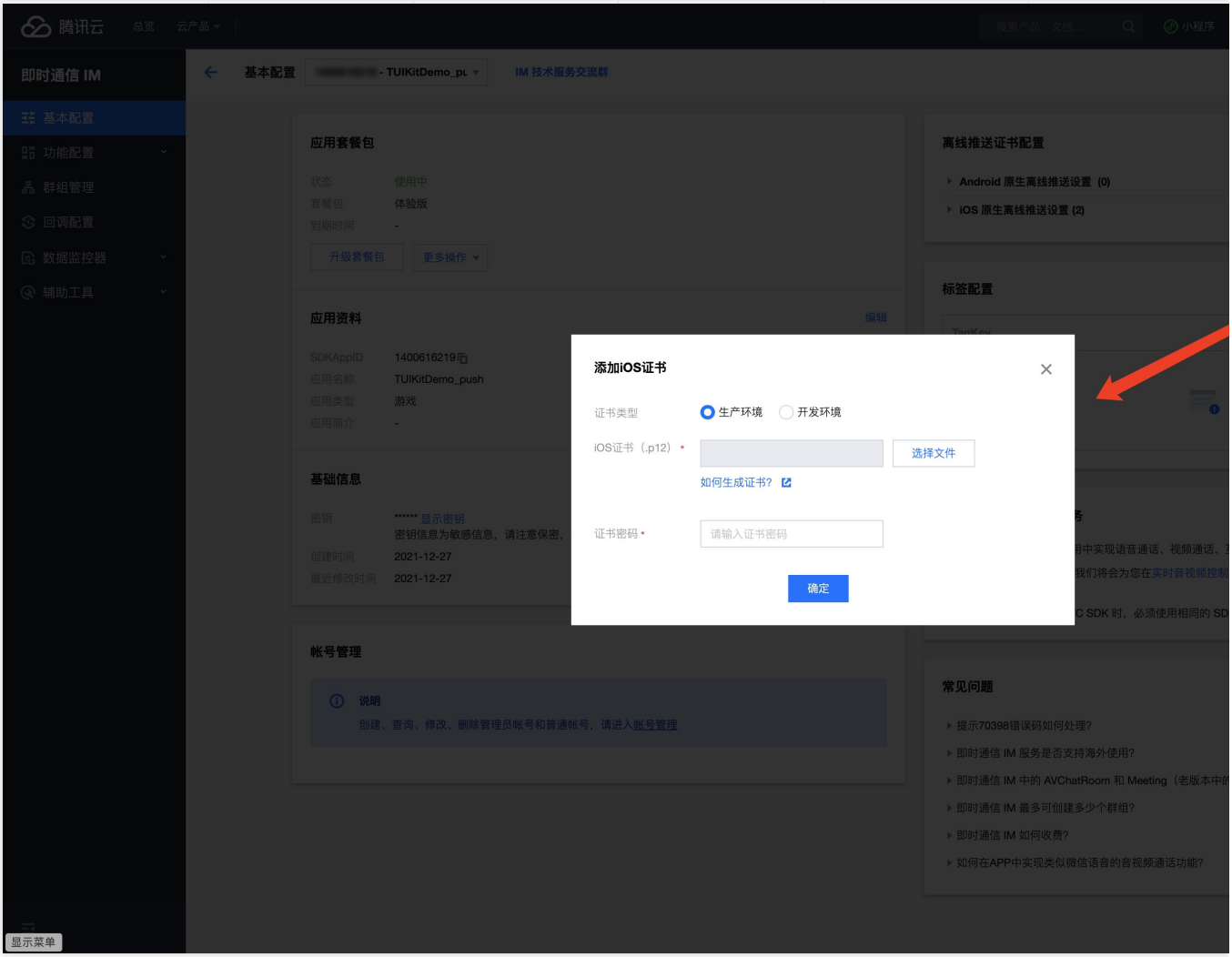
保存 `P12` 文件时，请务必要为其设置密码。

步骤2：上传证书到控制台

1. 登录 即时通信 IM 控制台。
2. 单击目标应用卡片，进入应用的基础配置页面。



3. 单击 **iOS 原生离线推送设置**右侧的**添加证书**。
4. 选择证书类型，上传 iOS 证书（p.12），设置证书密码，单击**确认**。



注意

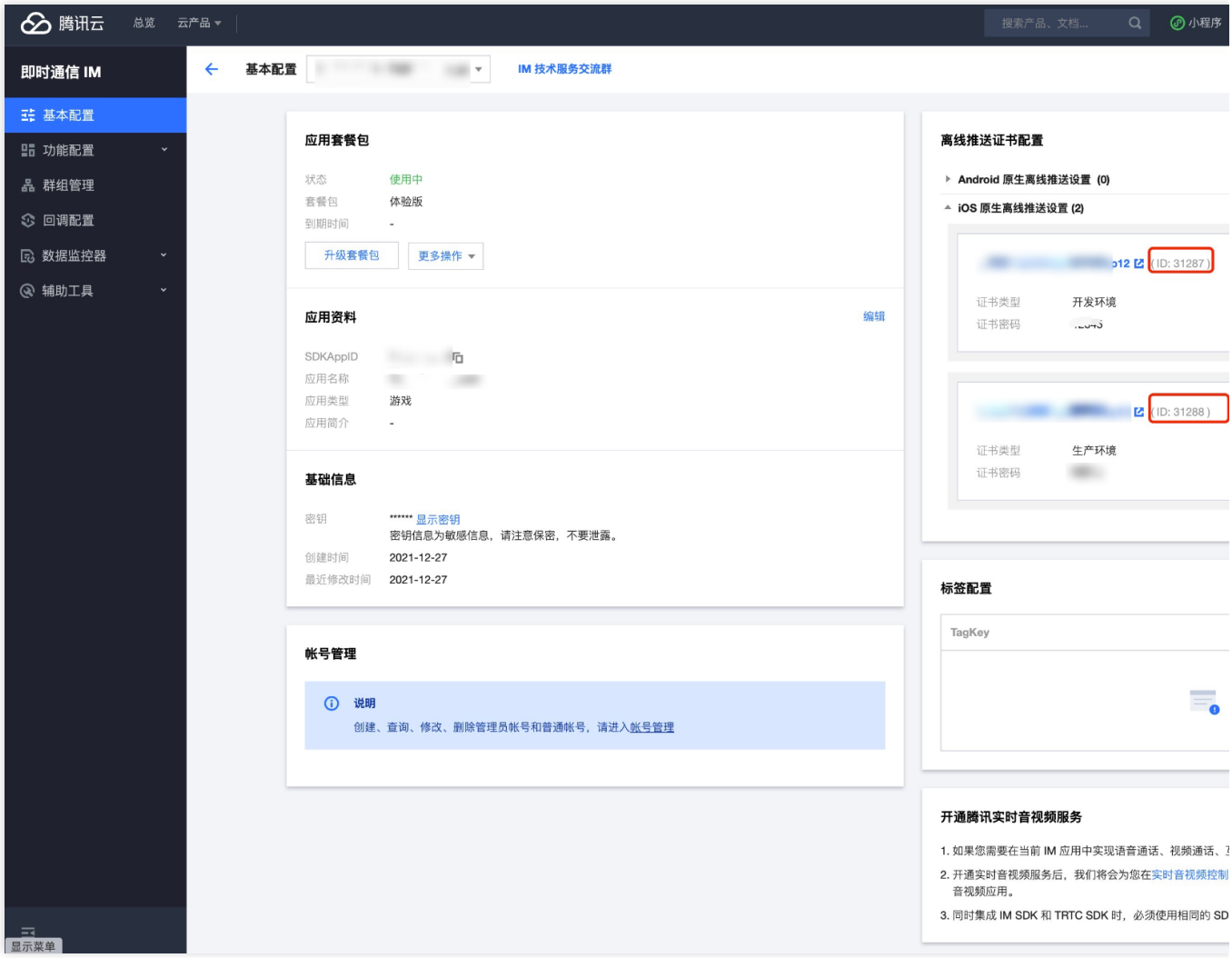
上传证书名最好使用全英文（尤其不能使用括号等特殊字符）。

上传证书需要设置密码，无密码收不到推送。

发布 App Store 的证书需要设置为生产环境，否则无法收到推送。

上传的 p12 证书必须是自己申请的真实有效的证书。

5. 待推送证书信息生成后，记录证书的 ID。



操作步骤

步骤1：注册应用到厂商推送平台

离线推送需要将您自己的应用注册到各个厂商的推送平台，得到 AppID 和 AppKey 等参数，来实现离线推送功能。目前国内支持的手机厂商有：小米、华为、荣耀、OPPO、VIVO、魅族，境外支持 Google FCM。

步骤2：IM 控制台配置

登录腾讯云 即时通信 IM 控制台 ，在**推送管理 > 接入设置**功能栏添加各个厂商推送证书，并将您在步骤一中获取的各厂商的 AppId、AppKey、AppSecret 等参数配置给添加的推送证书。

注意：

关于**点击后续动作**选项, 如需使用本插件提供的点击跳转能力, 请保持默认值不变, 即通常是 `打开应用内指定页面` 并带有默认配置.

如需使用上报统计功能, 也请保持此项默认值不变,

小米

华为

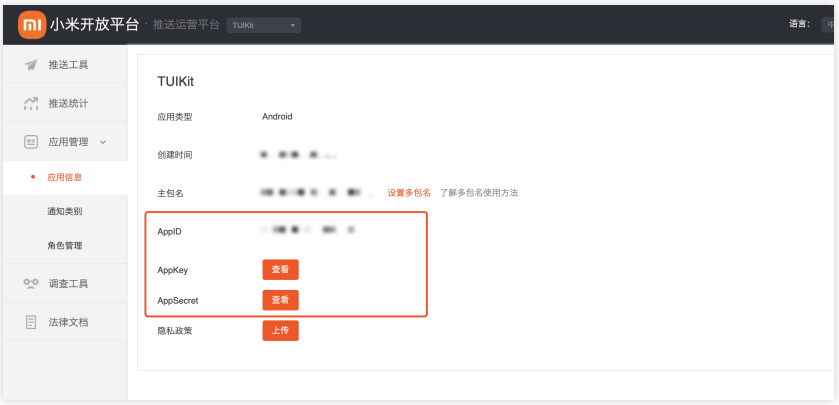
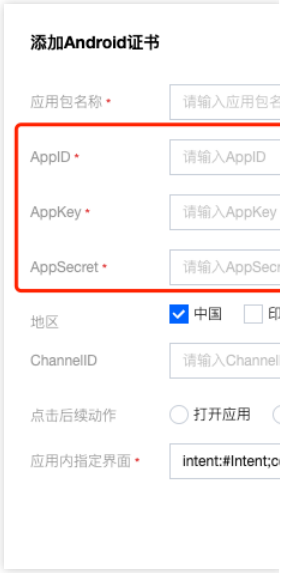
OPPO

vivo

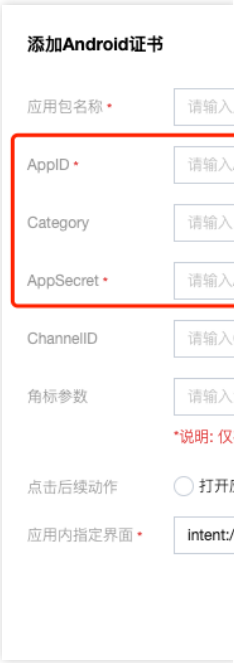
魅族

荣耀

Google FCM

厂商推送平台	IM 控制台配置
	

厂商推送平台	IM 控制台配置

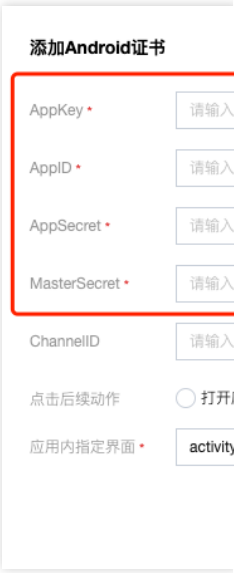


说明：

Client ID 对应 AppID，Cli

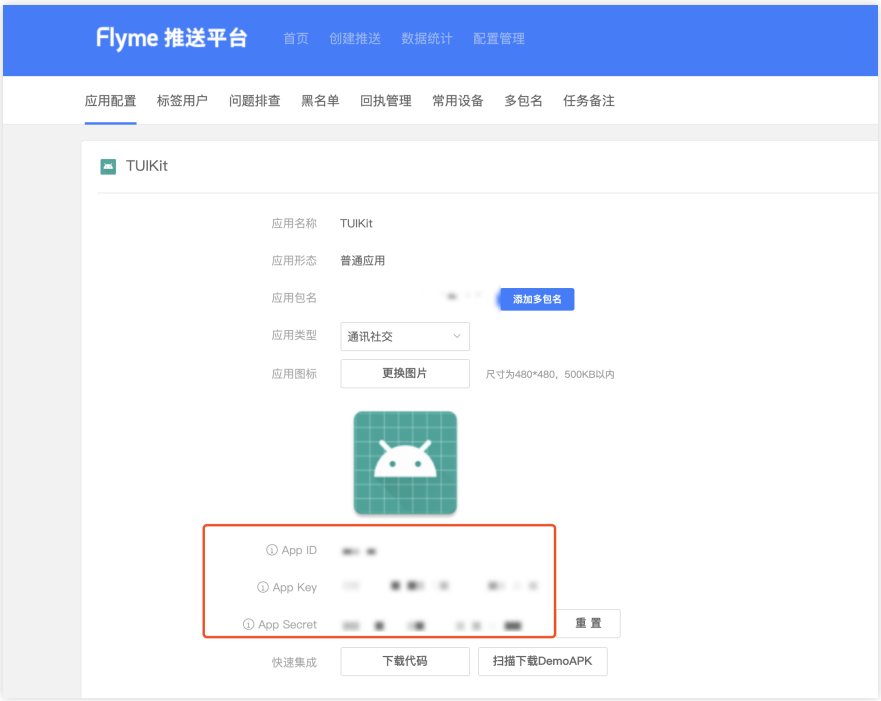
厂商推送平台

IM 控制台配置

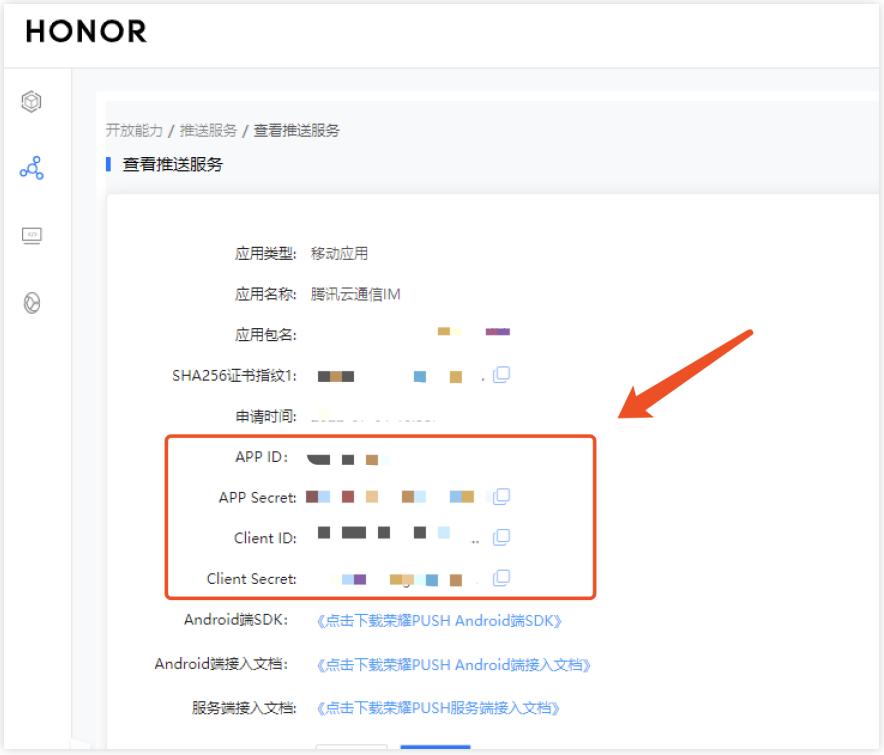


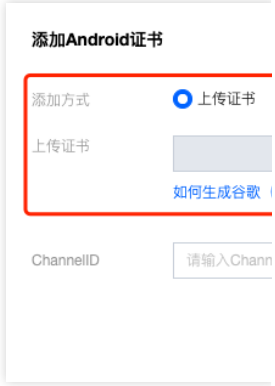
厂商推送平台	IM 控制台配置
	

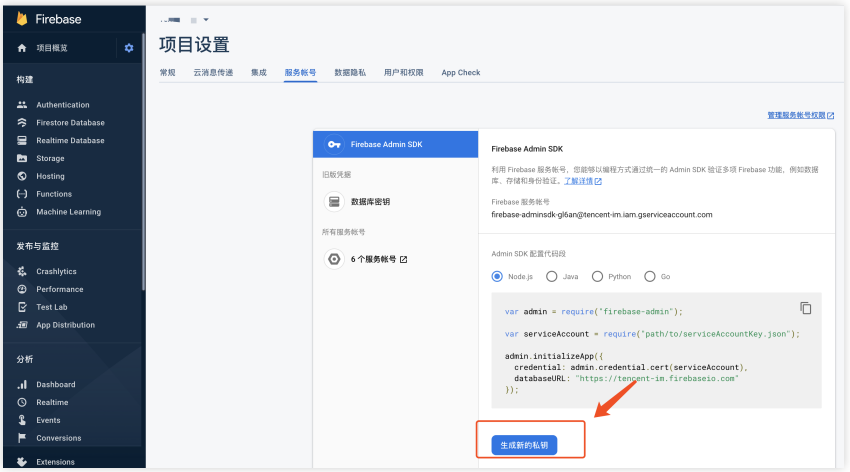
回执配置请参考：消息触达统计配置-vivo

厂商推送平台	IM 控制台配置
	

回执配置请参考：消息触达统计配置 > 魅族

厂商推送平台	IM 控制台配置
	

厂商推送平台	IM 控制台配置
	


Firebase
项目概览
构建
Authentication
Firestore Database
Realtime Database
Storage
Hosting
Functions
Machine Learning
发布与监控
Crashlytics
Performance
Test Lab
App Distribution
分析
Dashboard
Realtime
Events
Conversions
Extensions
项目设置
常规
云消息传递
集成
服务帐号
数据隐私
用户和权限
App Check
管理服务帐号权限
Firebase Admin SDK
旧版凭据
数据库密钥
所有服务帐号
6 个服务帐号
Firebase Admin SDK
利用 Firebase 服务帐号，您能够以编程方式通过统一的 Admin SDK 验证多项 Firebase 功能，例如数据库、存储和身份验证。了解详情
Firebase 服务帐号
firebase-adminsdk-gl6an@tencent-im.iam.gserviceaccount.com
Admin SDK 配置代码段
Node.js
Java
Python
Go
var admin = require("firebase-admin");
var serviceAccount = require("path/to/serviceAccountKey.json");
admin.initializeApp({
credential: admin.credential.cert(serviceAccount),
databaseURL: "https://tencent-im.firebaseio.com"
});
生成新的私钥

React-Native

最近更新时间：2024-06-24 15:45:53

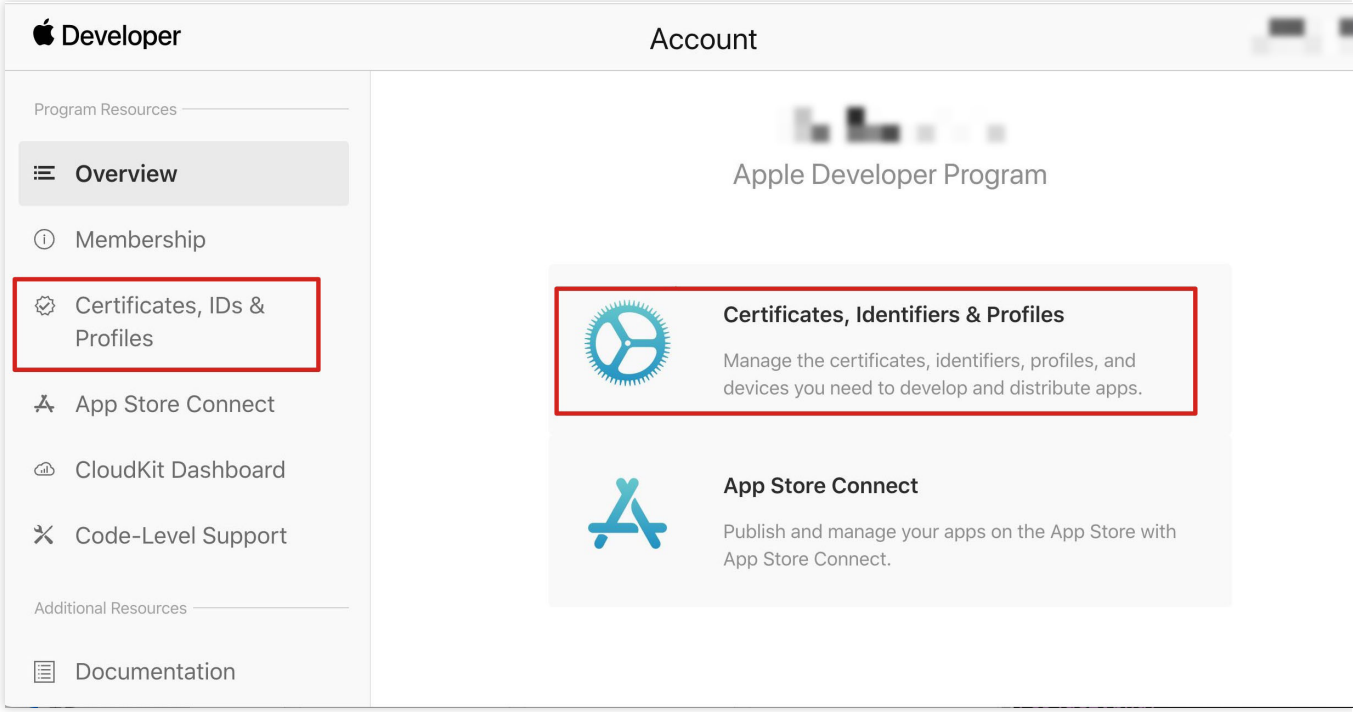
目前消息推送插件, 在 React Native 使用中, 仅支持推送至 Android (含各厂商通道) 和 iOS 设备.
iOS
Android
集成 消息推送 插件之前，需要先向 Apple 申请 APNs 推送证书，然后上传推送证书到 IM 控制台 。之后按照快速接入步骤接入即可。

操作步骤

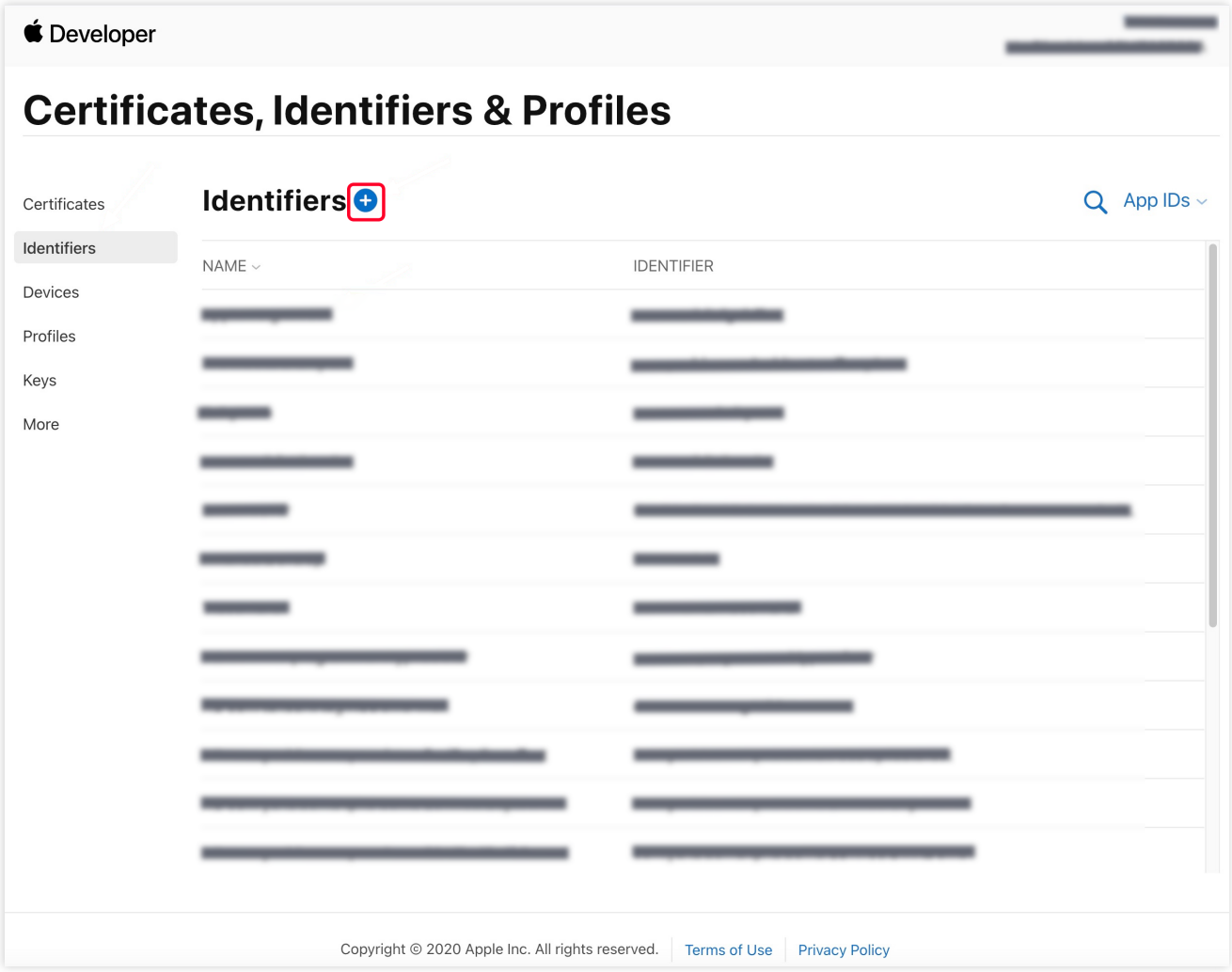
步骤1：申请 APNs 证书

开启 App 远程推送

1. 登录 苹果开发者中心 网站，单击 **Certificates,Identifiers & Profiles** 或者侧栏的 **Certificates,IDs & Profiles**，进入 **Certificates, IDS & Profiles** 页面。



2. 单击 Identifiers 右侧的 +。

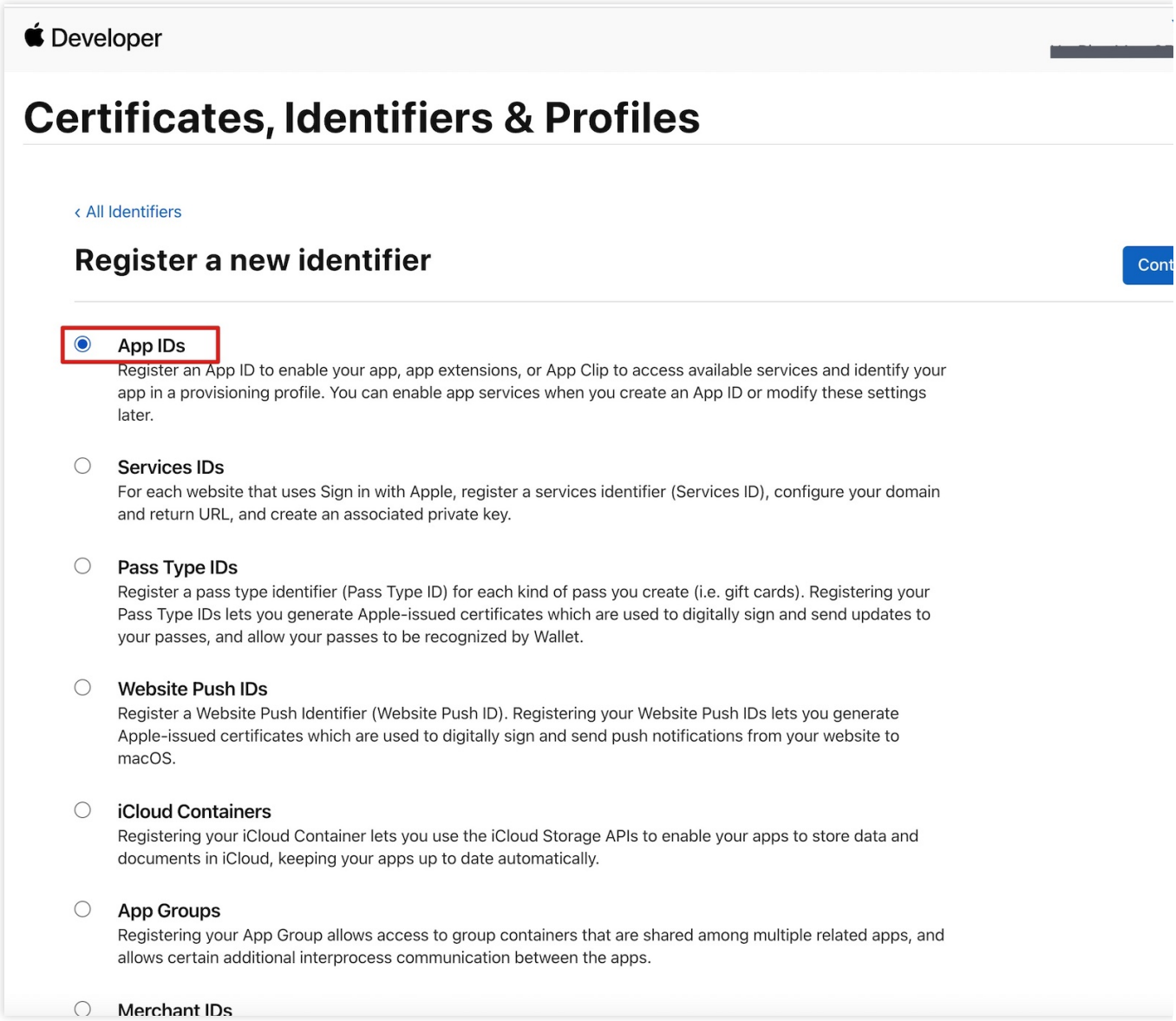


3. 您可以参见如下步骤新建一个 AppID，或者在您原有的 AppID 上增加 `Push Notification` 的 `Service`。

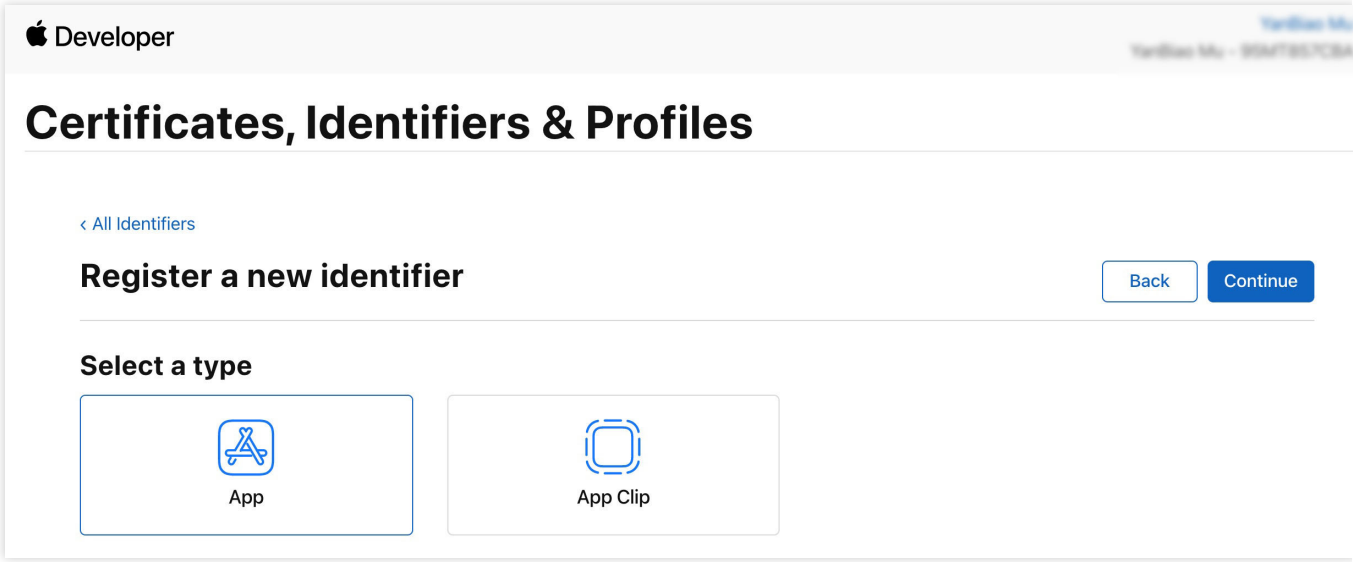
说明

您 App 的 `Bundle ID` 不能使用通配符 `*`，否则将无法使用远程推送服务。

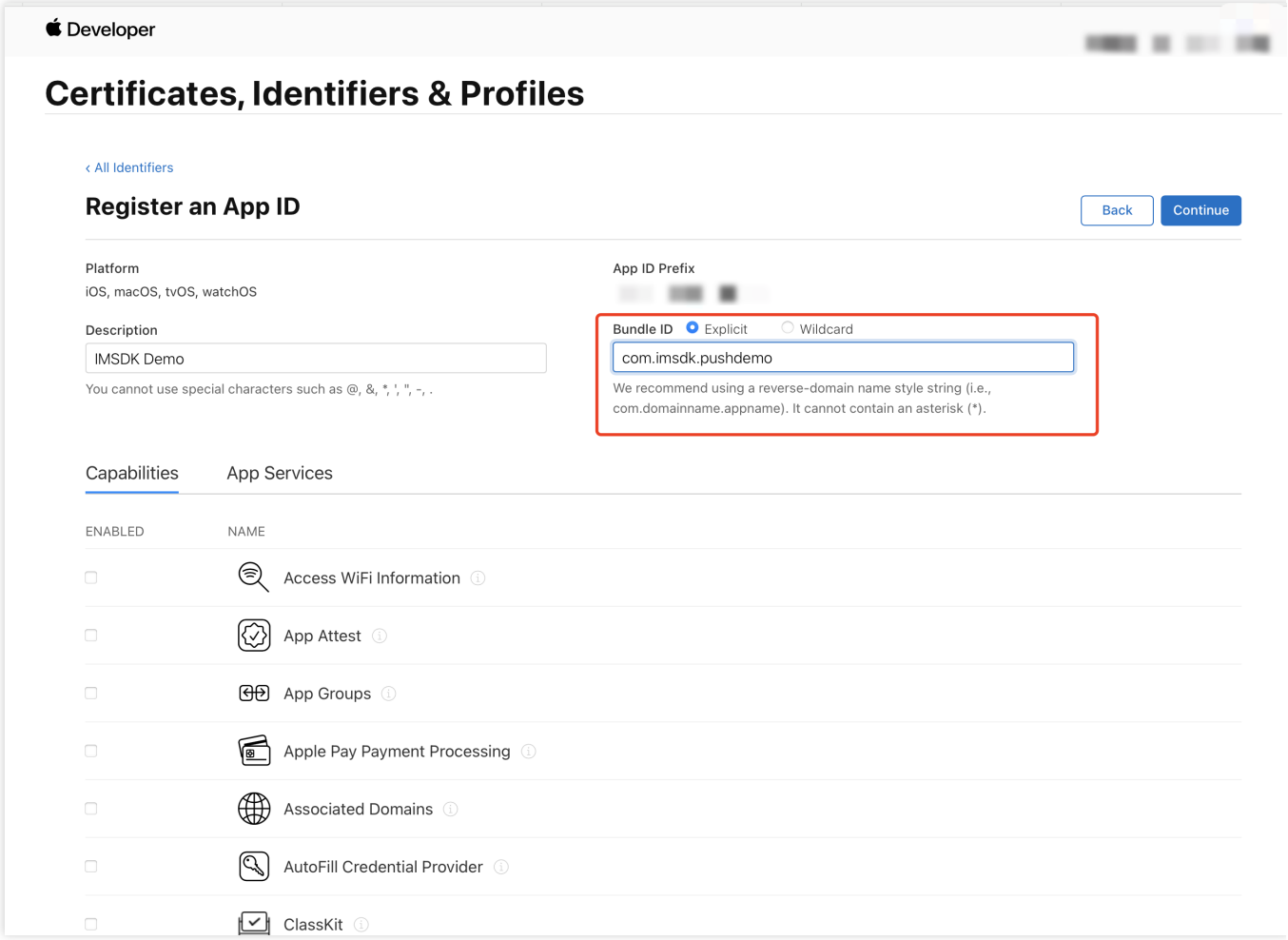
4. 勾选 **App IDs**，单击 **Continue** 进行下一步。



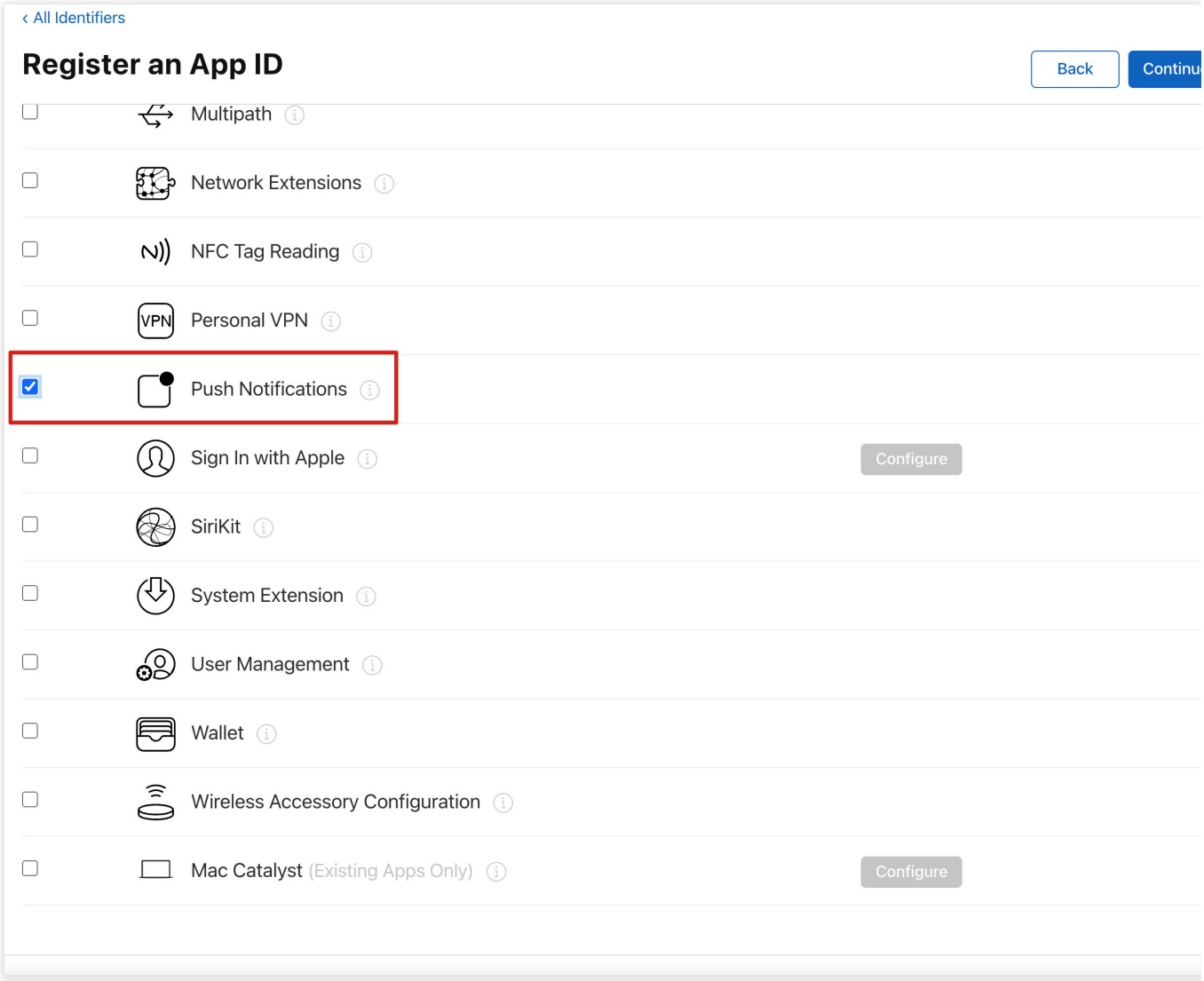
5. 选择 **App**，单击 **Continue** 进行下一步。



6. 配置 `Bundle ID` 等其他信息，单击 **Continue** 进行下一步。

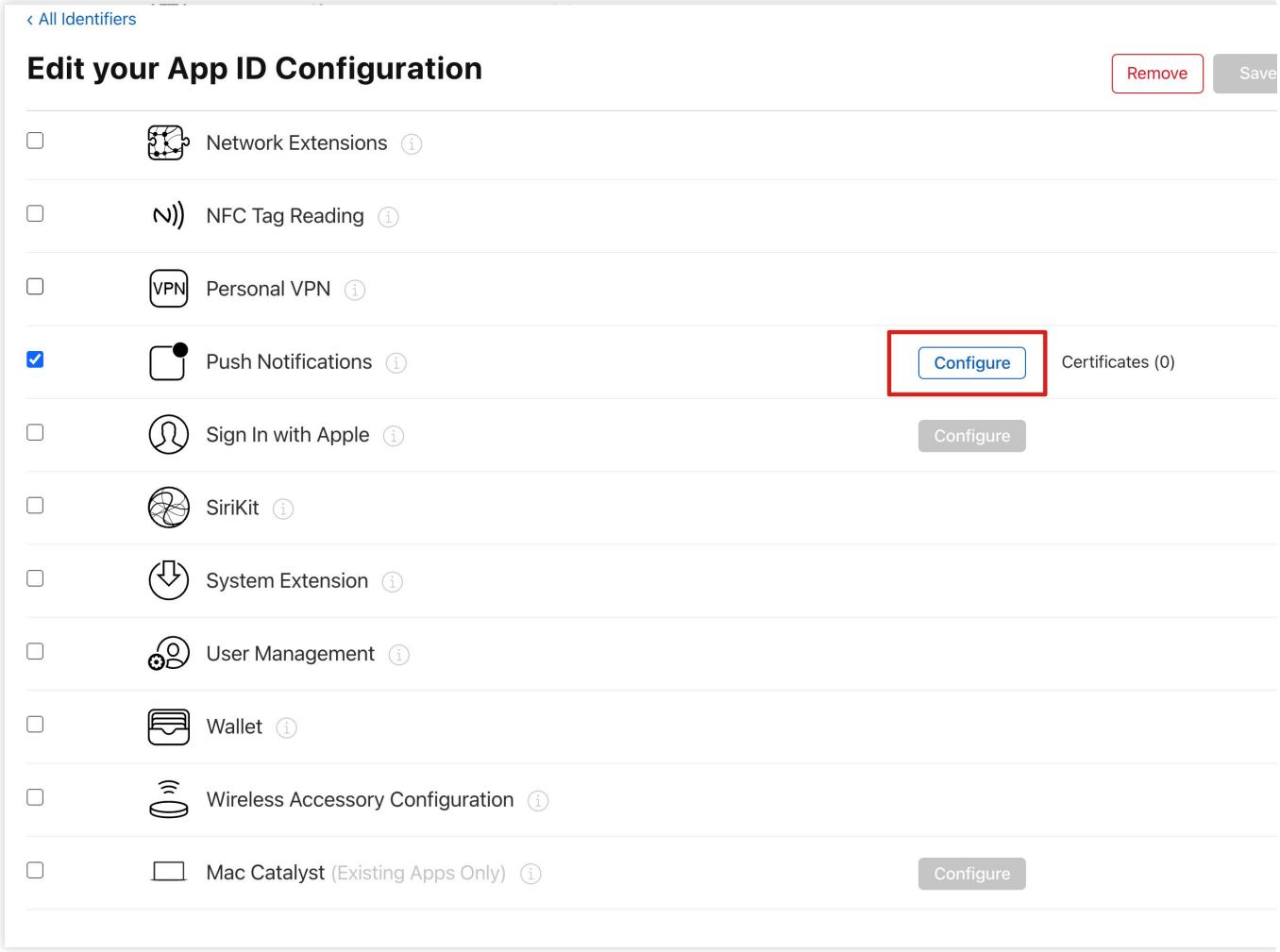


7. 勾选 **Push Notifications**，开启远程推送服务。

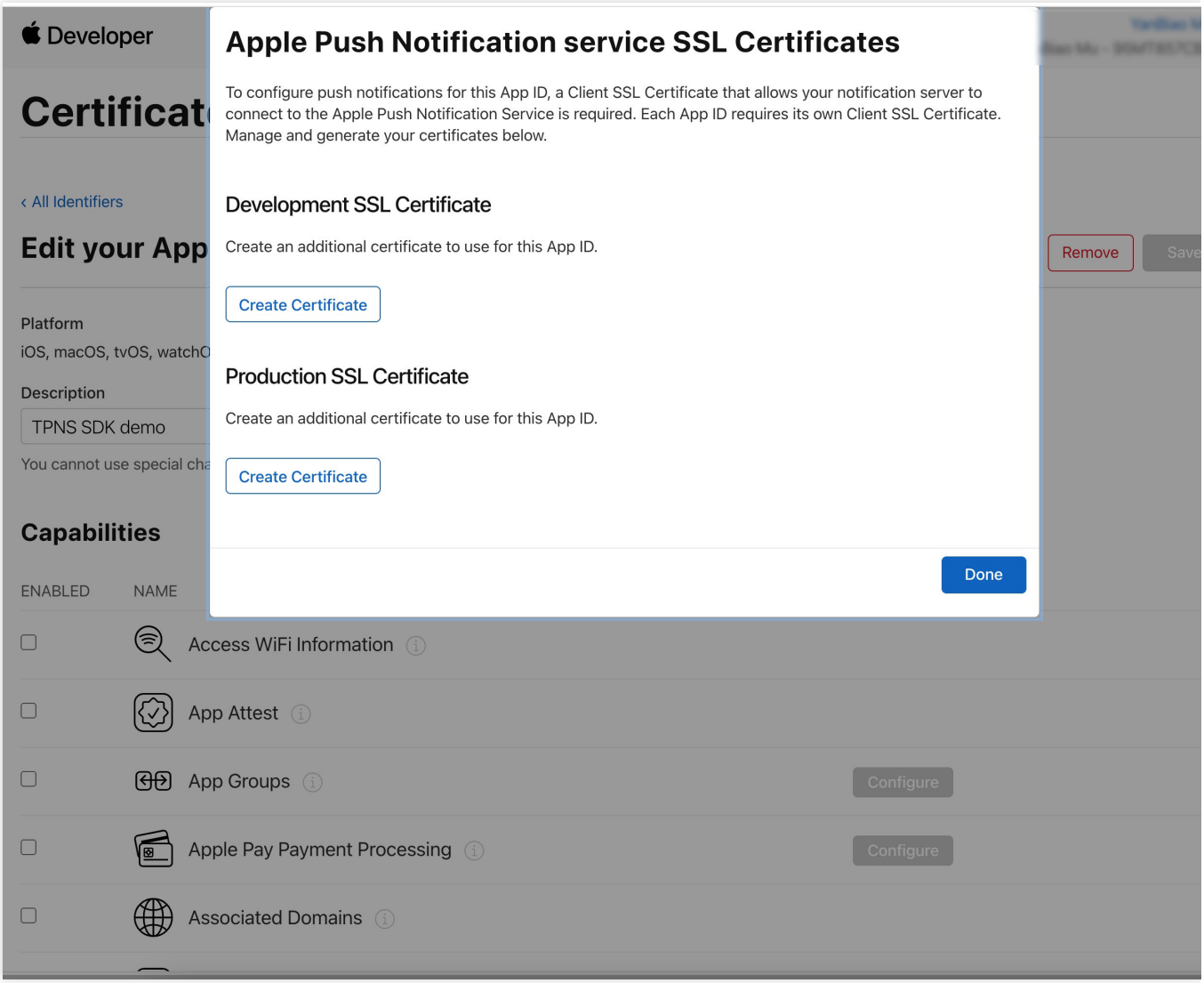


生成证书

1. 选中您的 AppID，选择 **Configure**。



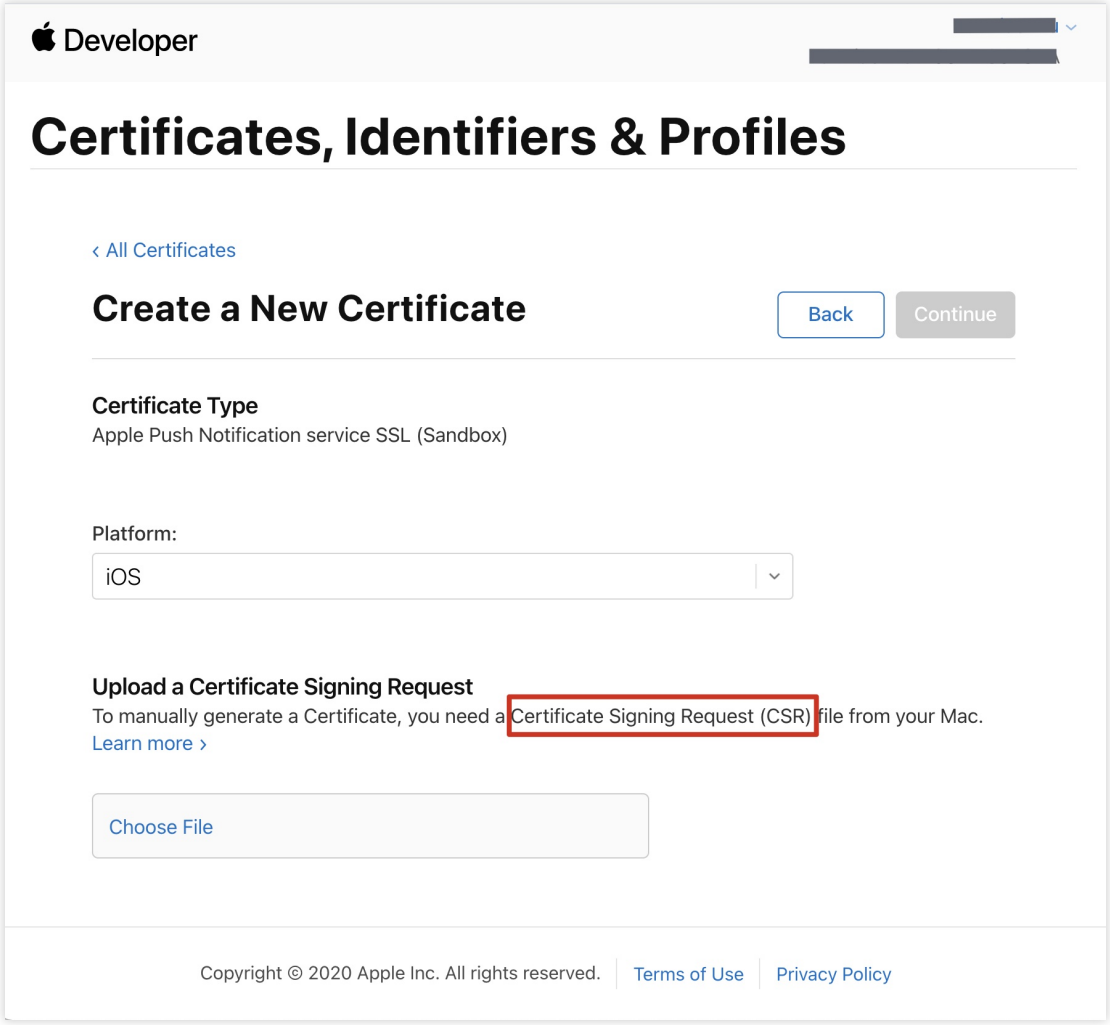
2. 可以看到在 **Apple Push Notification service SSL Certificates** 窗口中有两个 `SSL Certificate`，分别用于开发环境（Development）和生产环境（Production）的远程推送证书，如下图所示：



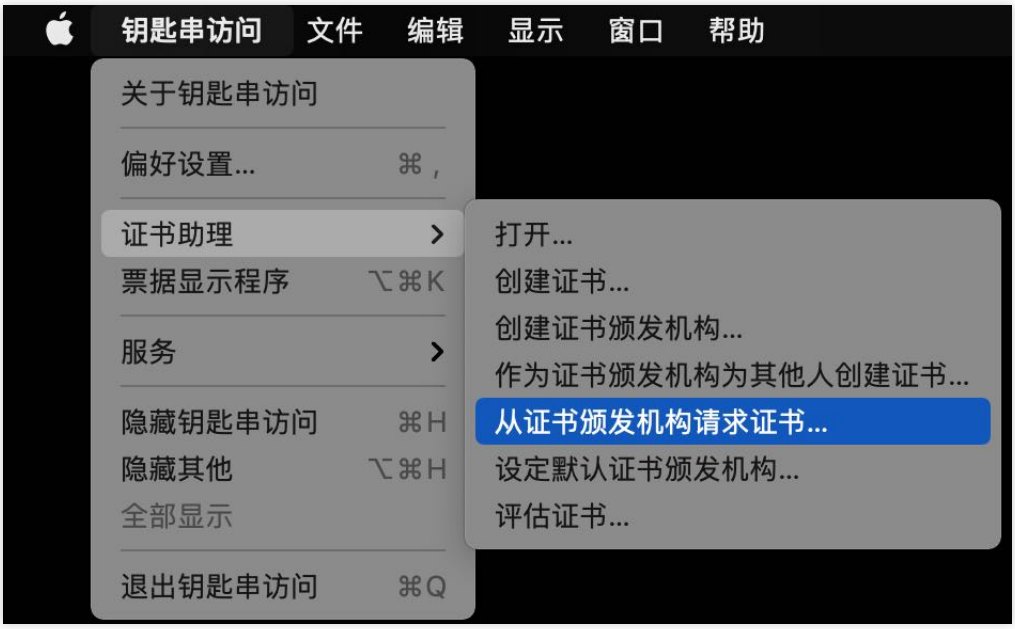
3.

我

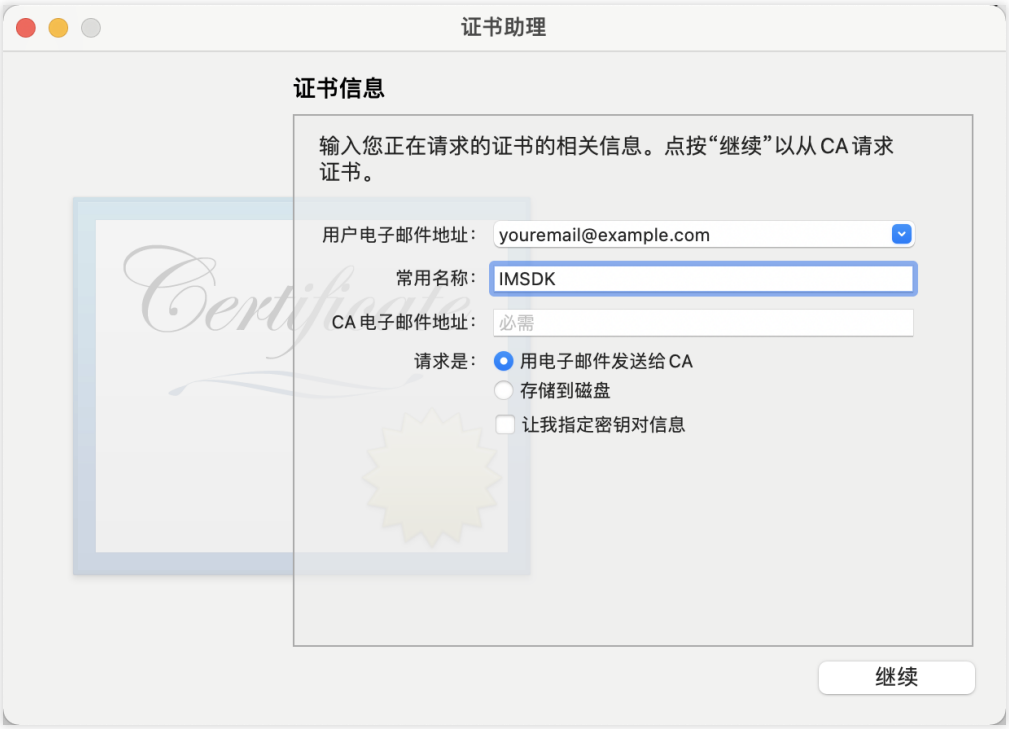
们先选择开发环境（Development）的 **Create Certificate**，系统将提示我们需要一个 Certificate Signing Request（CSR）。



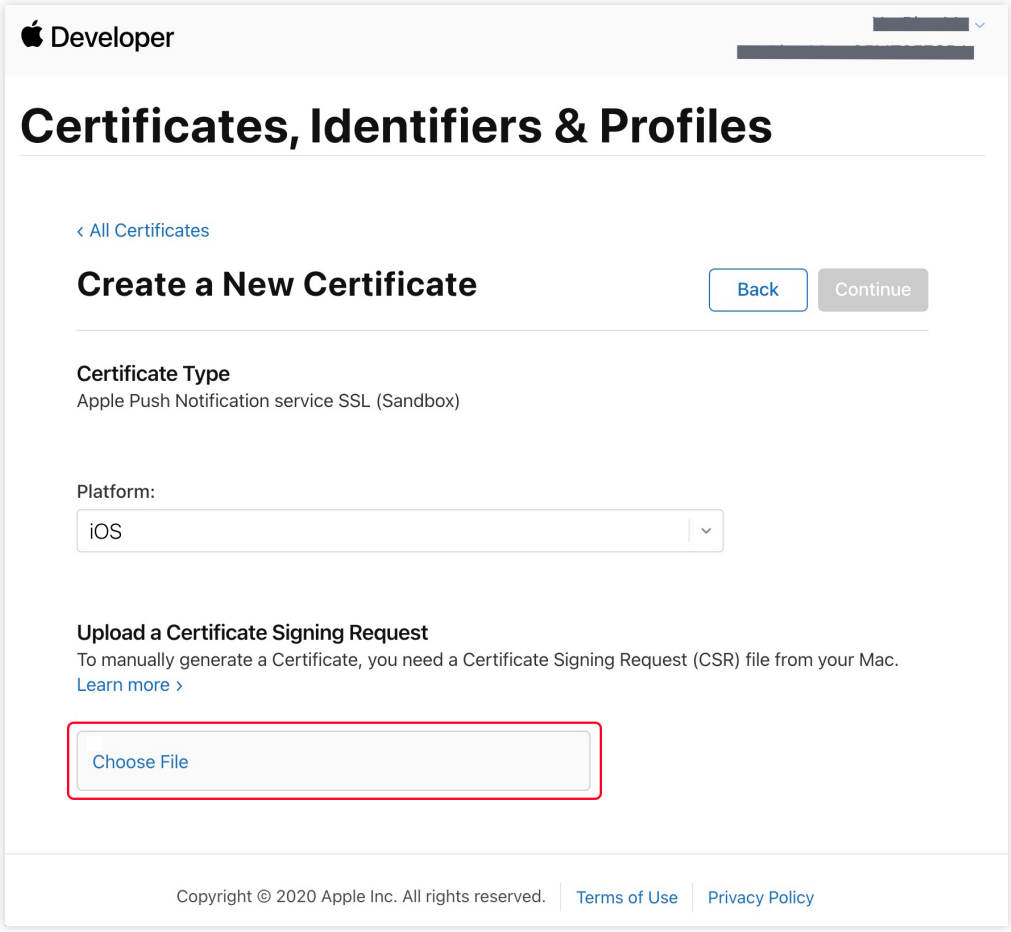
4. 在 Mac 上打开**钥匙串访问工具（Keychain Access）**，在菜单中选择**钥匙串访问 > 证书助理 > 从证书颁发机构请求证书**（`Keychain Access – Certificate Assistant – Request a Certificate From a Certificate Authority`）。



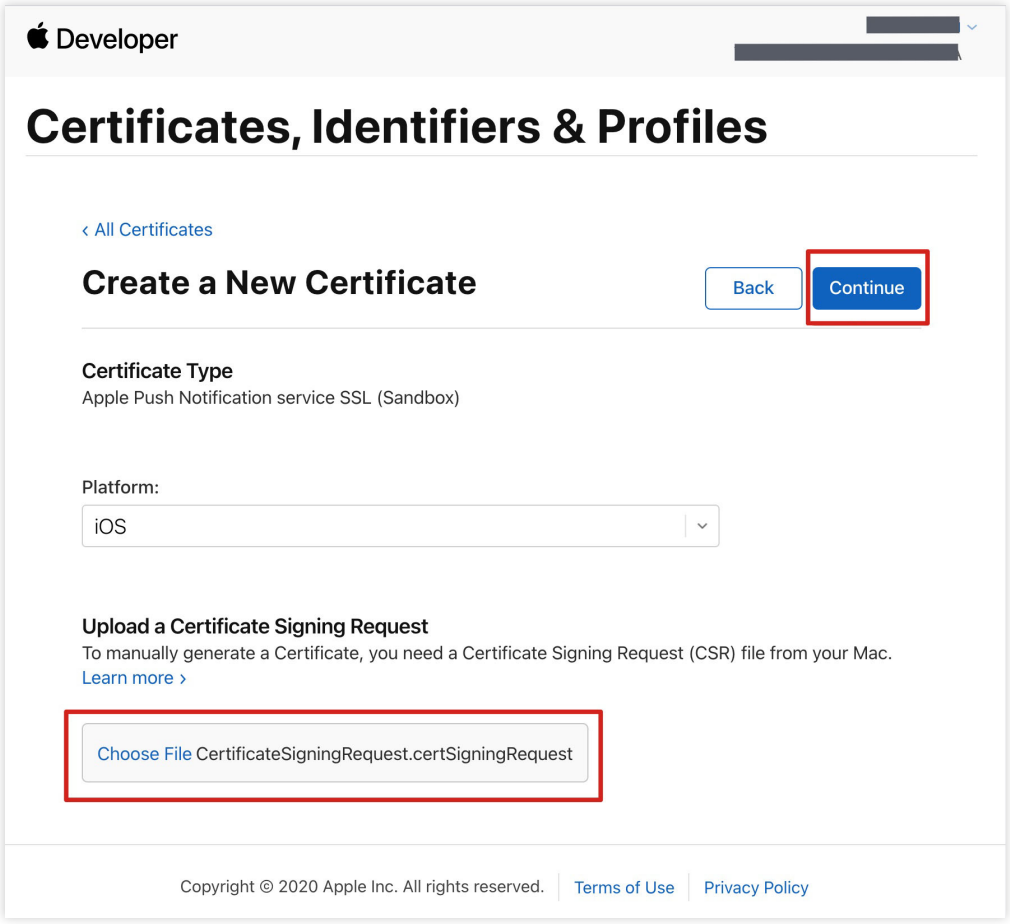
5. 输入用户电子邮件地址（您的邮箱）、常用名称（您的名称或公司名），选择**存储到磁盘**，单击**继续**，系统将生成一个 `*.certSigningRequest` 文件。



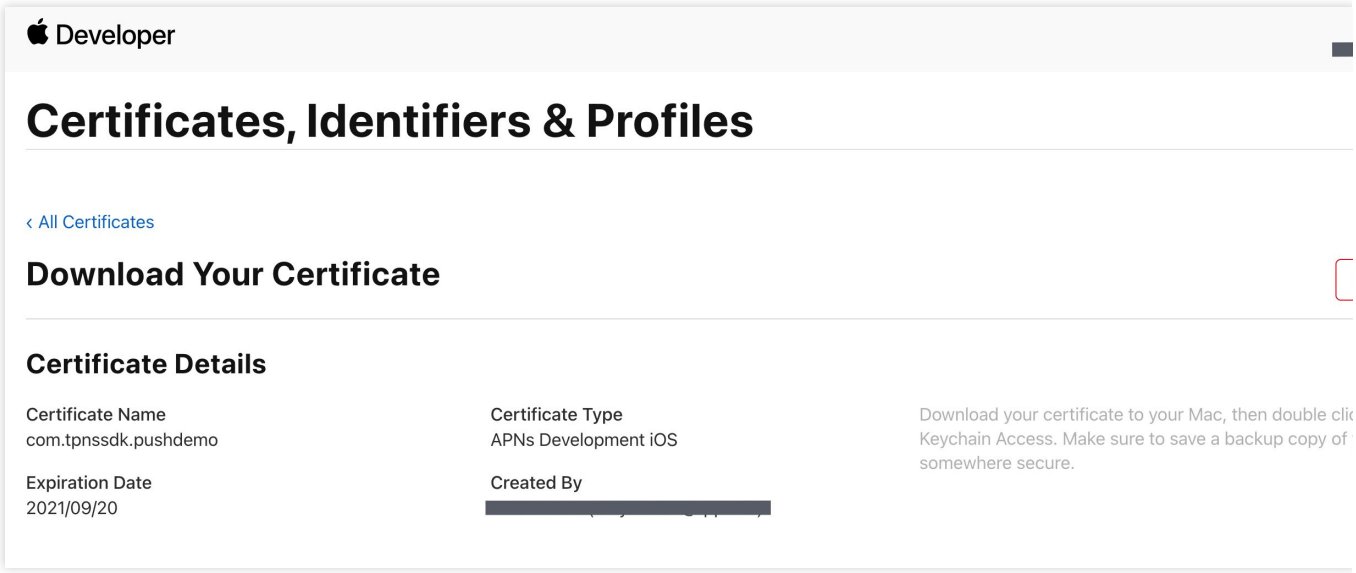
6. 返回上述 步骤3 中 Apple Developer 网站刚才的页面，单击 **Choose File** 上传生成的 `*.certSigningRequest` 文件。



7. 单击 **Continue**，即可生成推送证书。



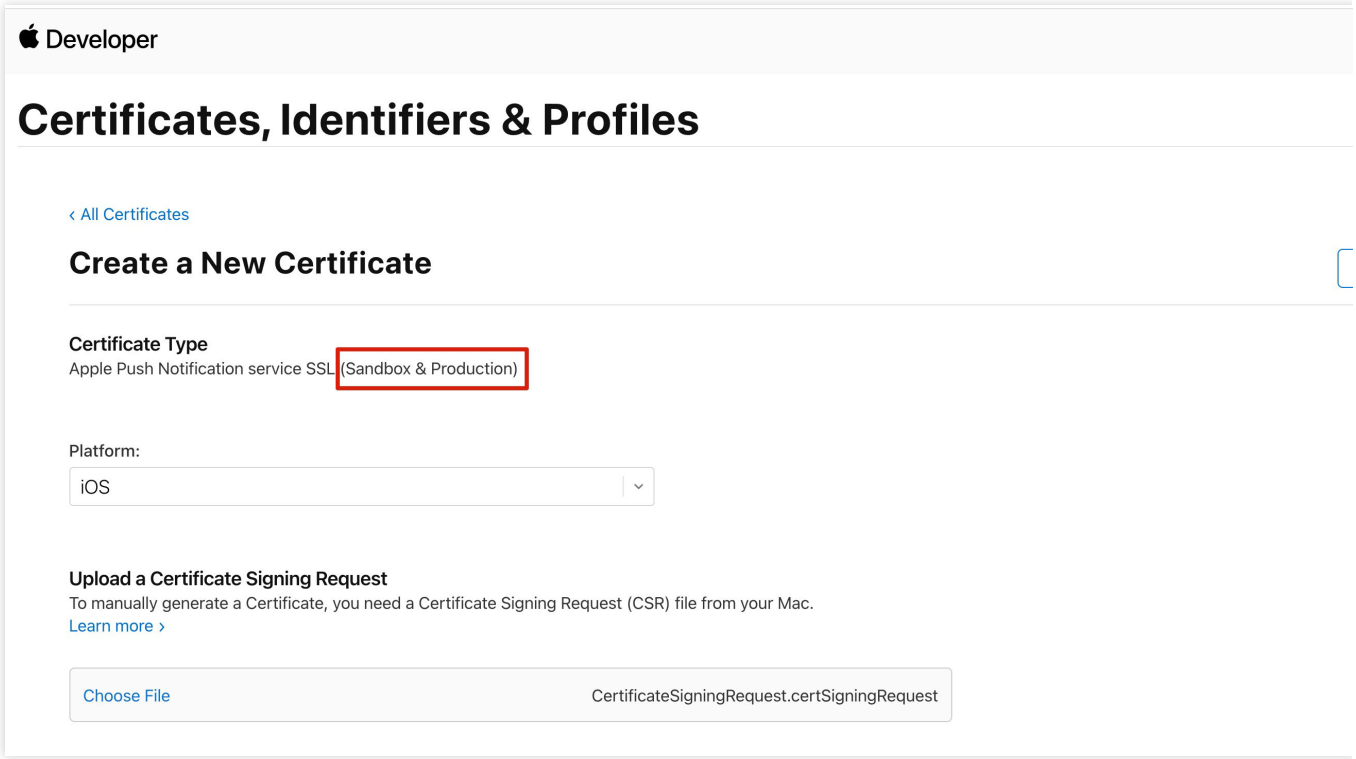
8. 单击 **Download** 下载开发环境的 `Development SSL Certificate` 到本地。

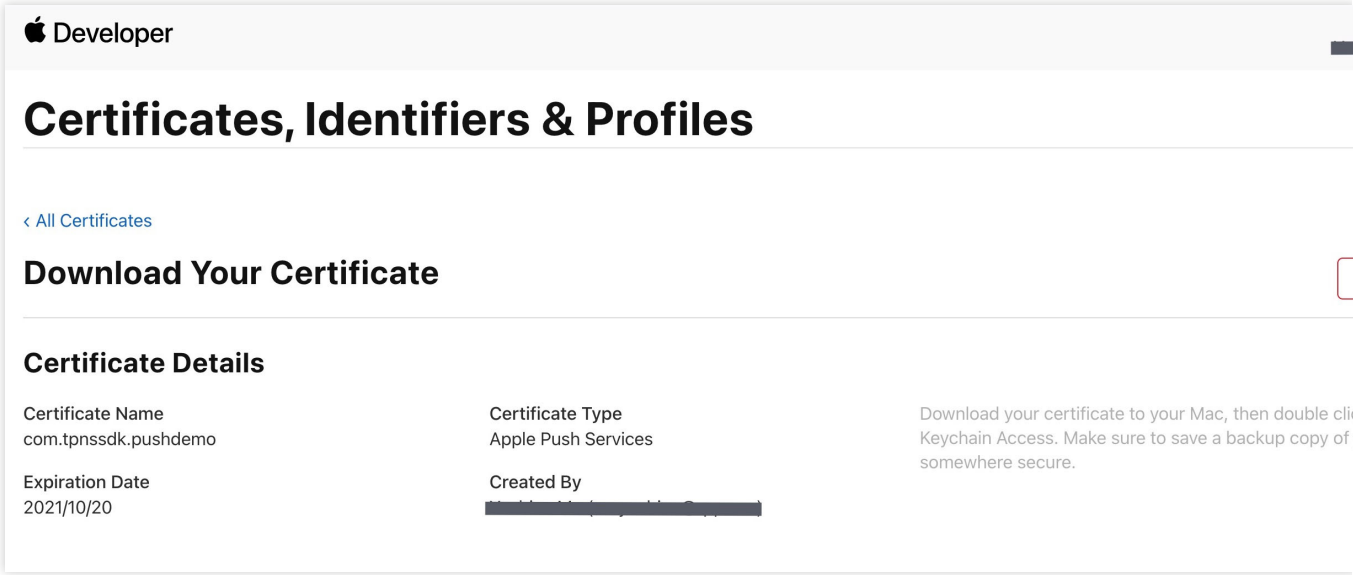


9. 再次按照上述步骤1 - 8，将生产环境的 `Production SSL Certificate` 下载到本地。

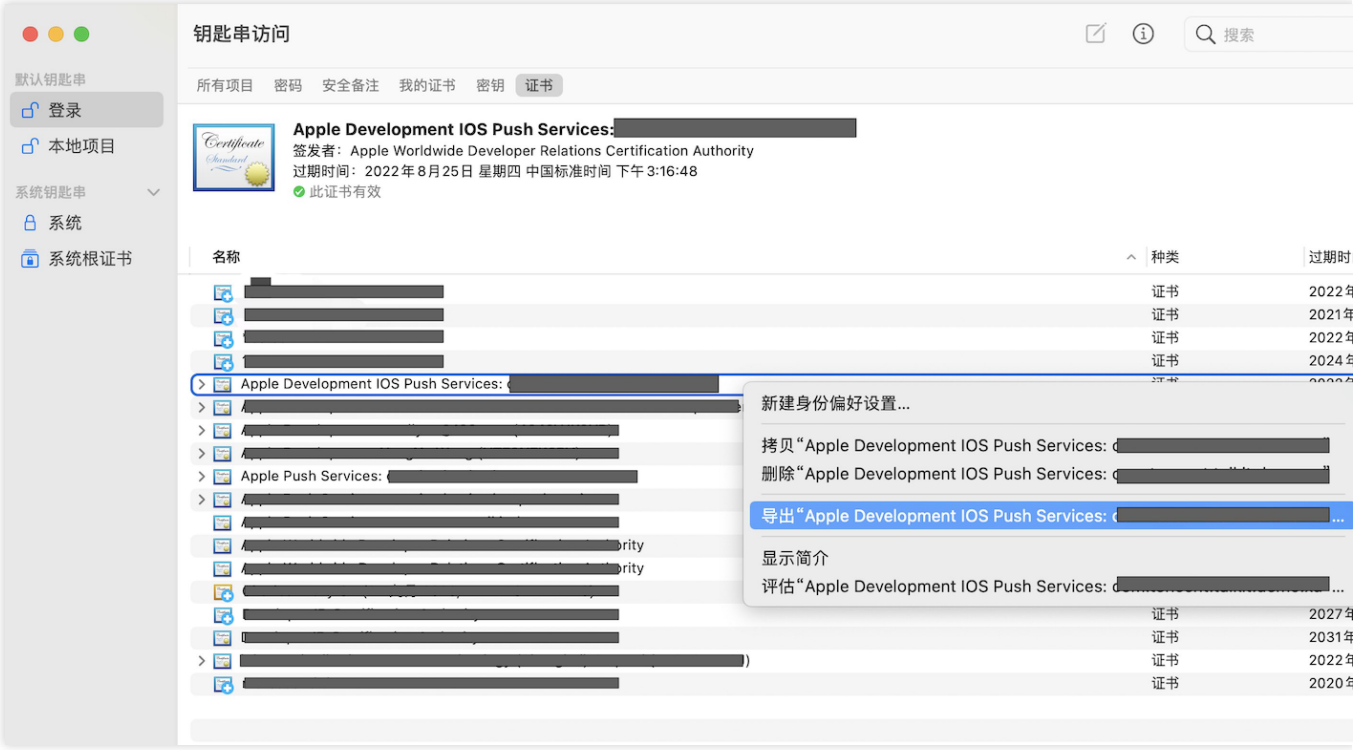
说明

生产环境的证书实际是开发（Sandbox）+生产（Production）的合并证书，可以同时作为开发环境和生产环境的证书使用。





10. 双击打开下载的开发环境和生产环境的 `SSL Certificate` ，系统会将其导入钥匙串中。
11. 打开钥匙串应用，在**登录 > 我的证书**，右键分别导出刚创建的开发环境（ `Apple Development IOS Push Service` ）和生产环境（ `Apple Push Services` ）的 `P12` 文件。

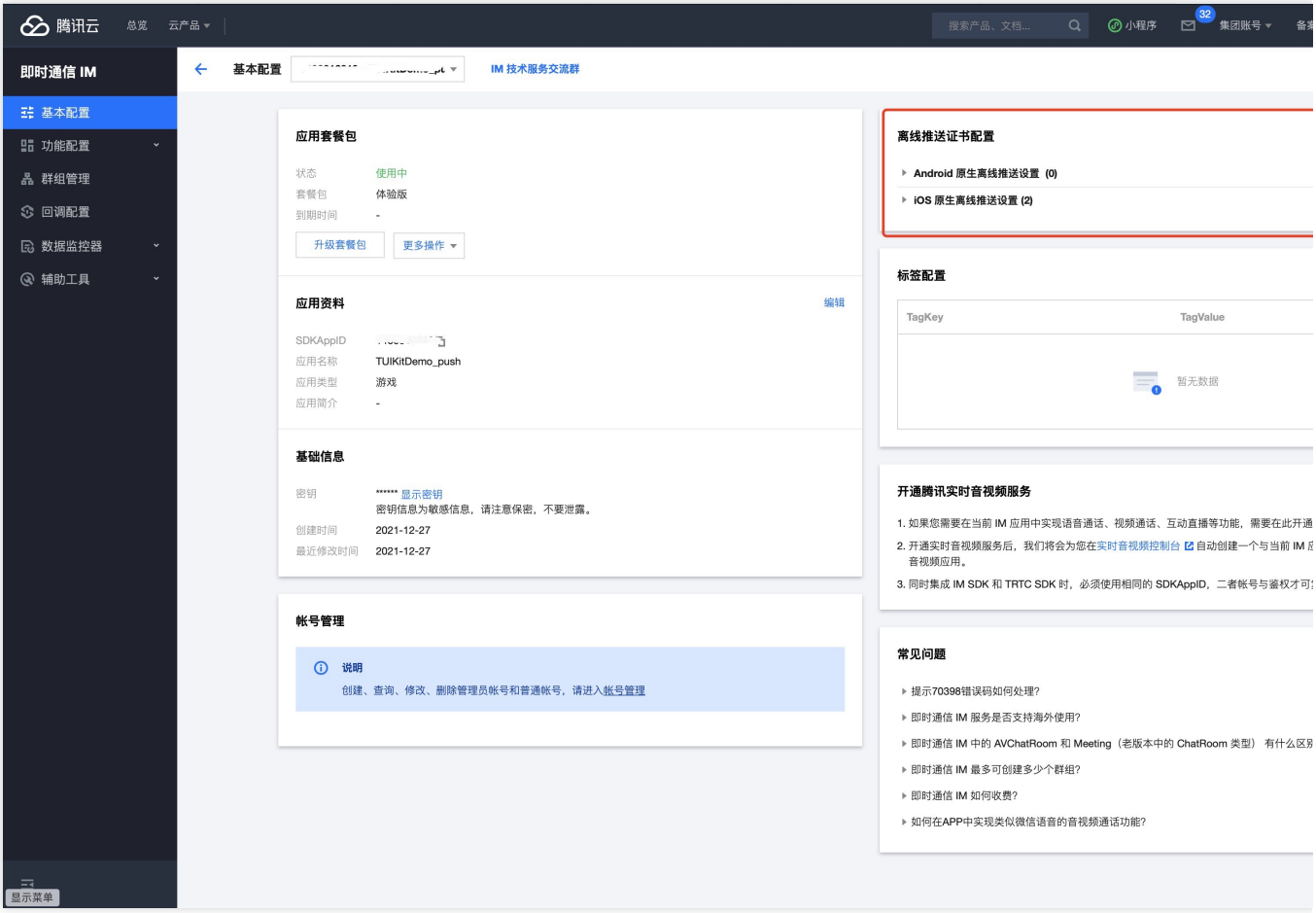


注意

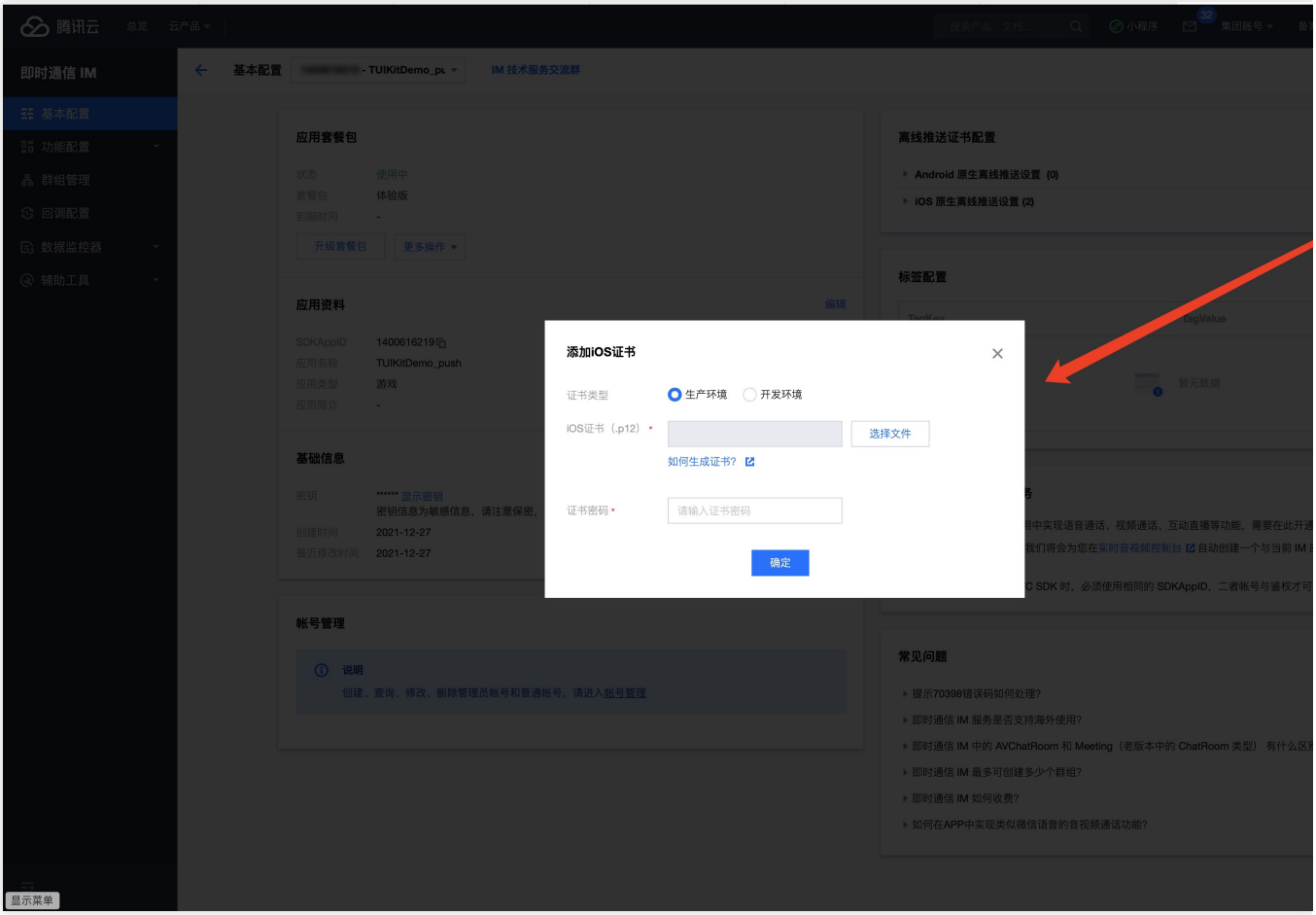
保存 `P12` 文件时，请务必要为其设置密码。

步骤2：上传证书到控制台

1. 登录 即时通信 IM 控制台。
2. 单击目标应用卡片，进入应用的基础配置页面。



3. 单击 **iOS 原生离线推送设置**右侧的**添加证书**。
4. 选择证书类型，上传 iOS 证书（p.12），设置证书密码，单击**确认**。



注意

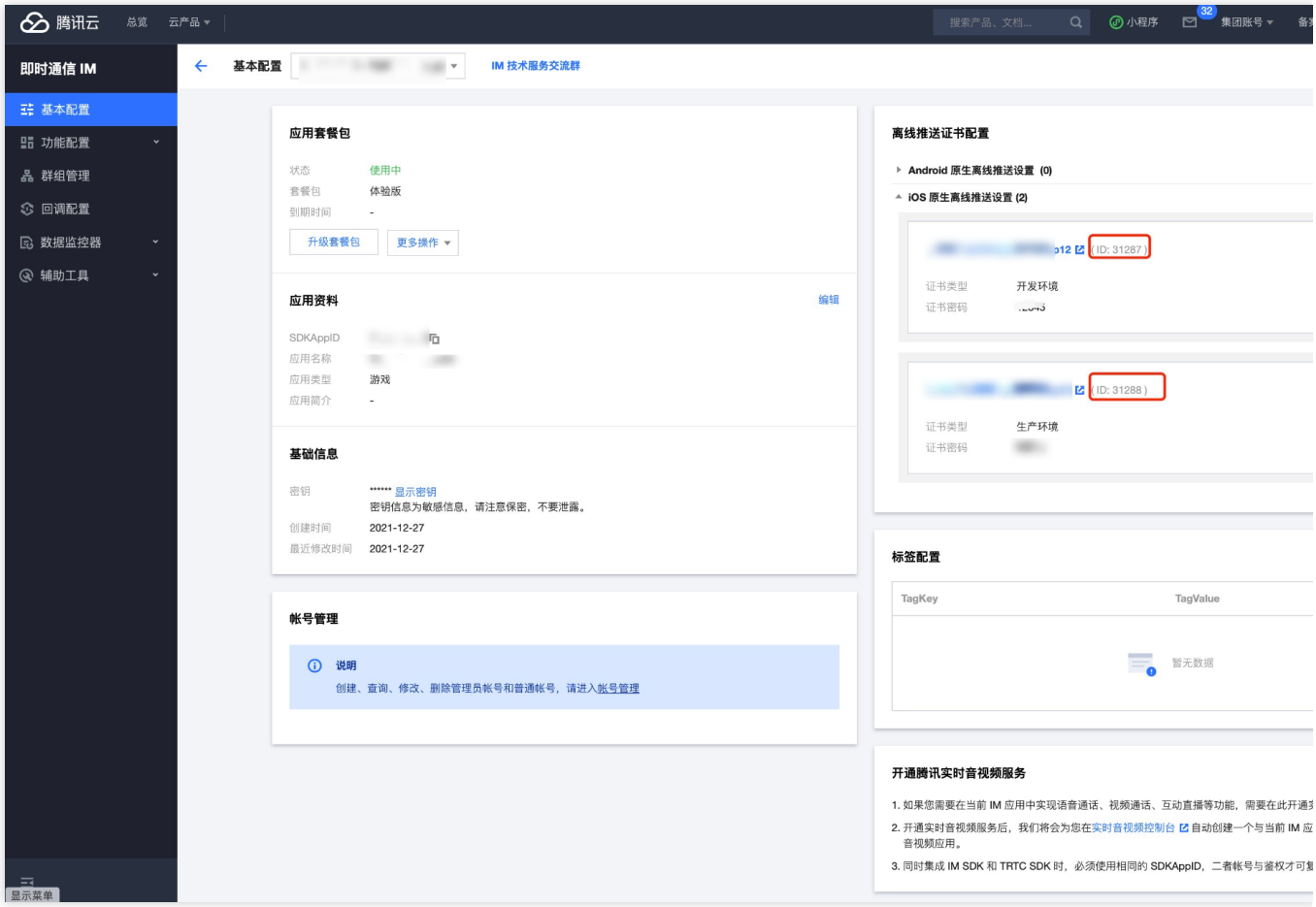
上传证书名最好使用全英文（尤其不能使用括号等特殊字符）。

上传证书需要设置密码，无密码收不到推送。

发布 App Store 的证书需要设置为生产环境，否则无法收到推送。

上传的 p12 证书必须是自己申请的真实有效的证书。

5. 待推送证书信息生成后，记录证书的 ID。



操作步骤

步骤1：注册应用到厂商推送平台

离线推送需要将您自己的应用注册到各个厂商的推送平台，得到 AppID 和 AppKey 等参数，来实现离线推送功能。目前国内支持的手机厂商有：小米、华为、荣耀、OPPO、VIVO、魅族，境外支持 Google FCM。

步骤2：IM 控制台配置

登录腾讯云 即时通信 IM 控制台 ，在**推送管理 > 接入设置**功能栏添加各个厂商推送证书，并将您在步骤一中获取的各厂商的 AppId、AppKey、AppSecret 等参数配置给添加的推送证书。

注意：

关于**点击后续动作**选项, 如需使用本插件提供的点击跳转能力, 请保持默认值不变, 即通常是 `打开应用内指定页面` 并带有默认配置.

如需使用上报统计功能, 也请保持此项默认值不变,

小米

华为

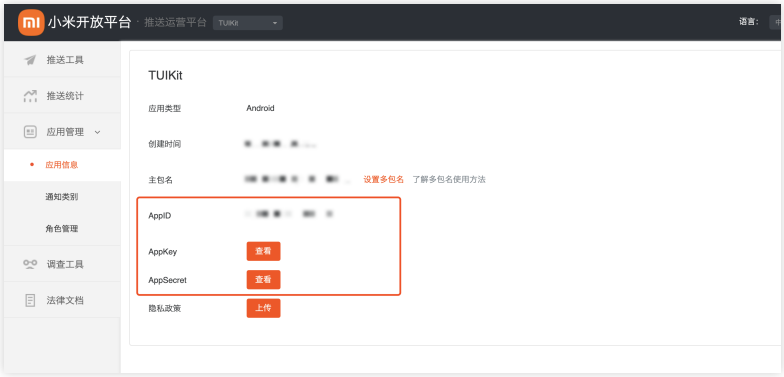
OPPO

vivo

魅族

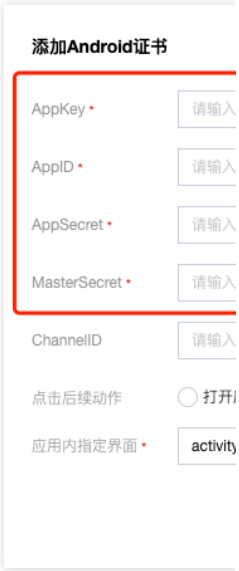
荣耀

Google FCM

厂商推送平台	IM 控制台配置
	

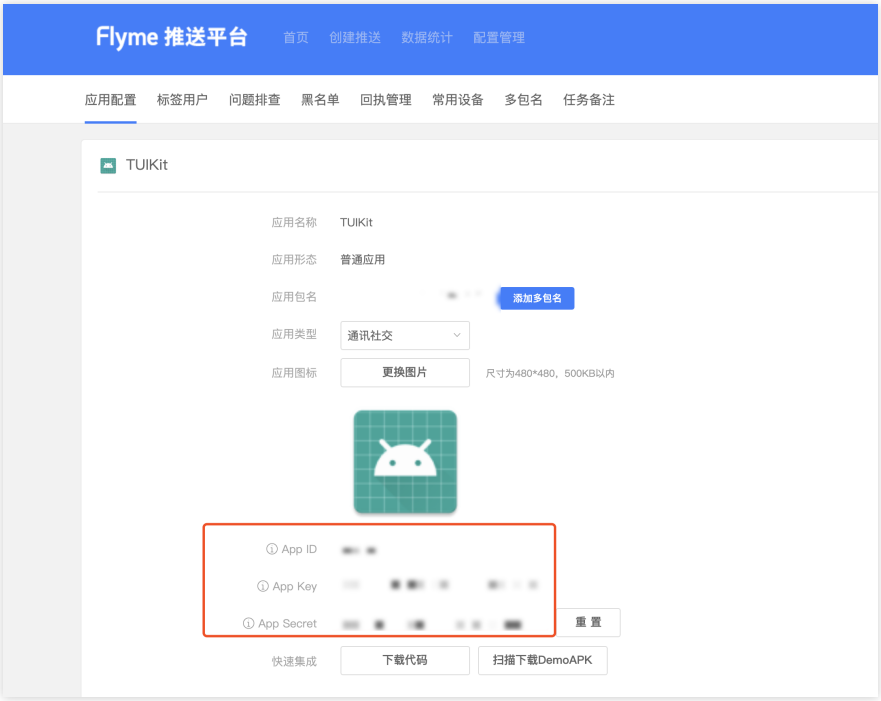
厂商推送平台	IM 控制台配置
	

	说明： Client ID 对应 AppID，Client Secr

厂商推送平台	IM 控制台配置
	

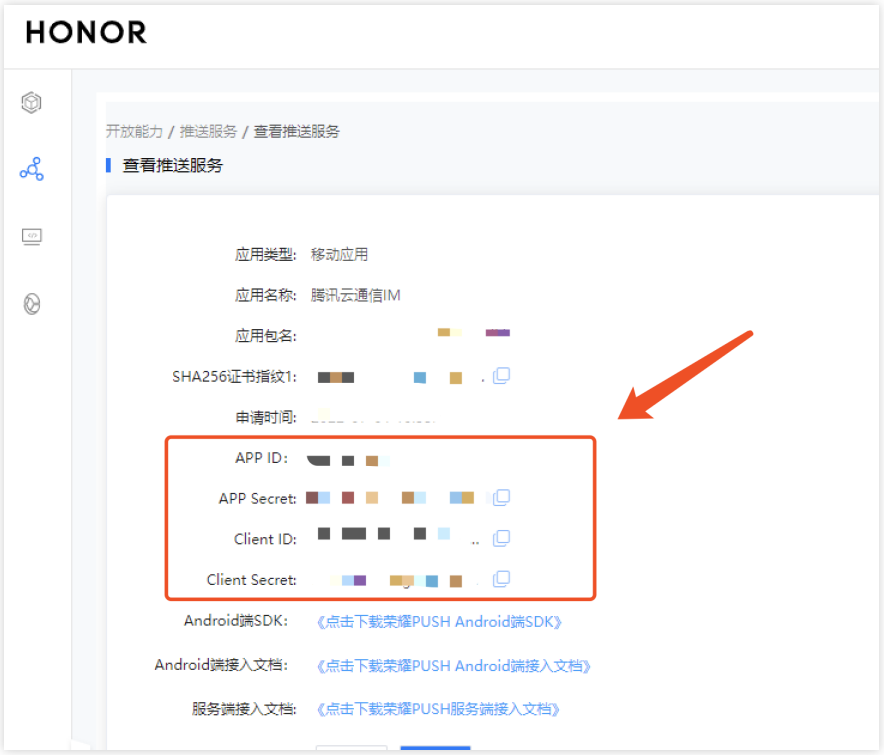
厂商推送平台	IM 控制台配置
	

回执配置请参考：消息触达统计配置-vivo

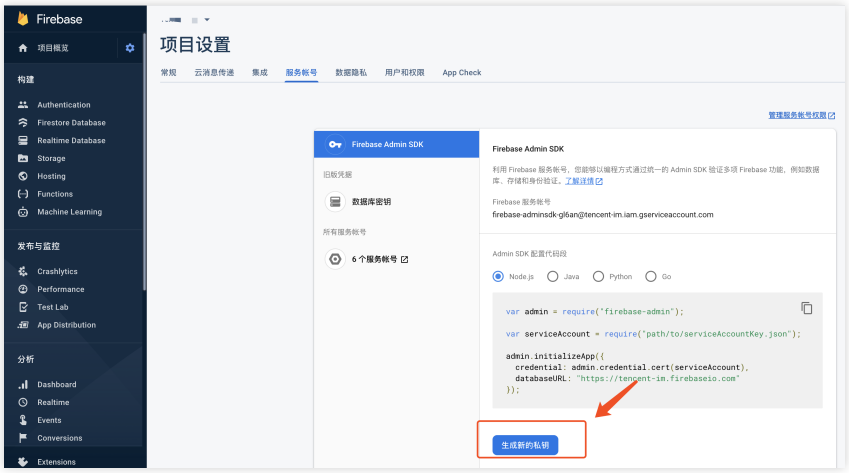
厂商推送平台	IM 控制台配置
	

回执配置请参考：消息触达统计配置-魅族

厂商推送平台	IM 控制台配置



厂商推送平台



IM 控制台配置

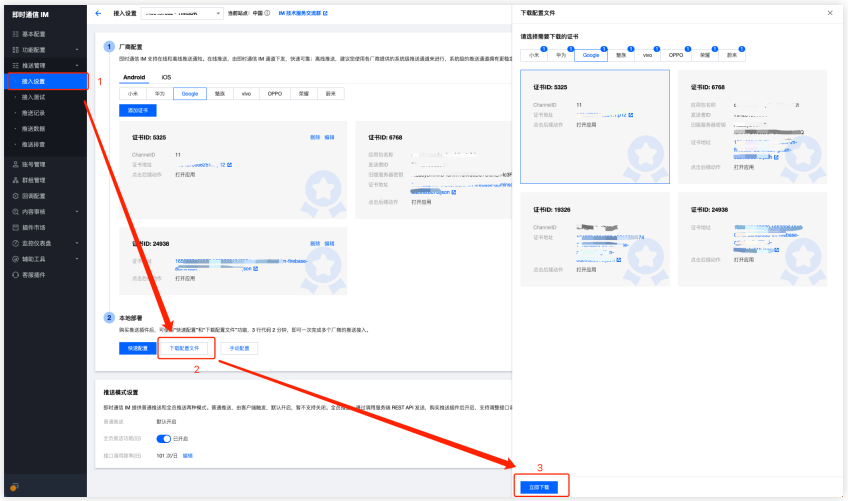
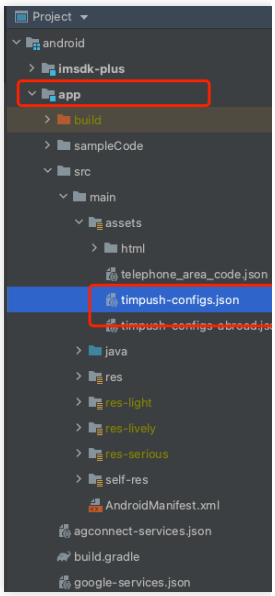
快速接入

Android

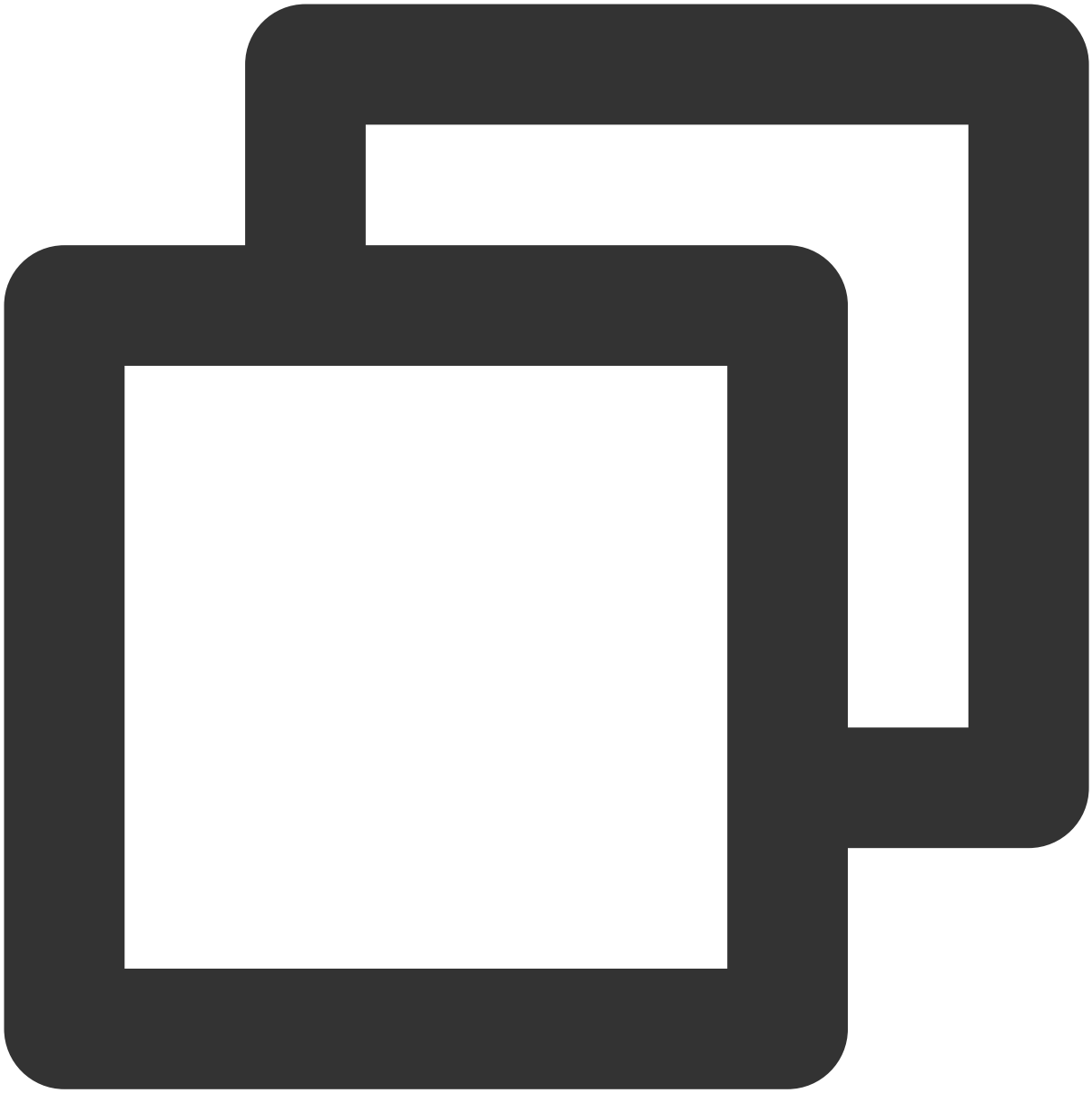
最近更新时间：2024-06-13 10:39:26

步骤1：下载并添加配置文件

完成控制台厂商推送信息填写后，下载并添加配置文件到工程。将下载的 timpush-configs.json 文件添加到应用模块的 assets 目录下：

1.选择下载配置文件 timpush-configs.json	2.添加到工程
	

步骤2：集成 TIMPush 插件

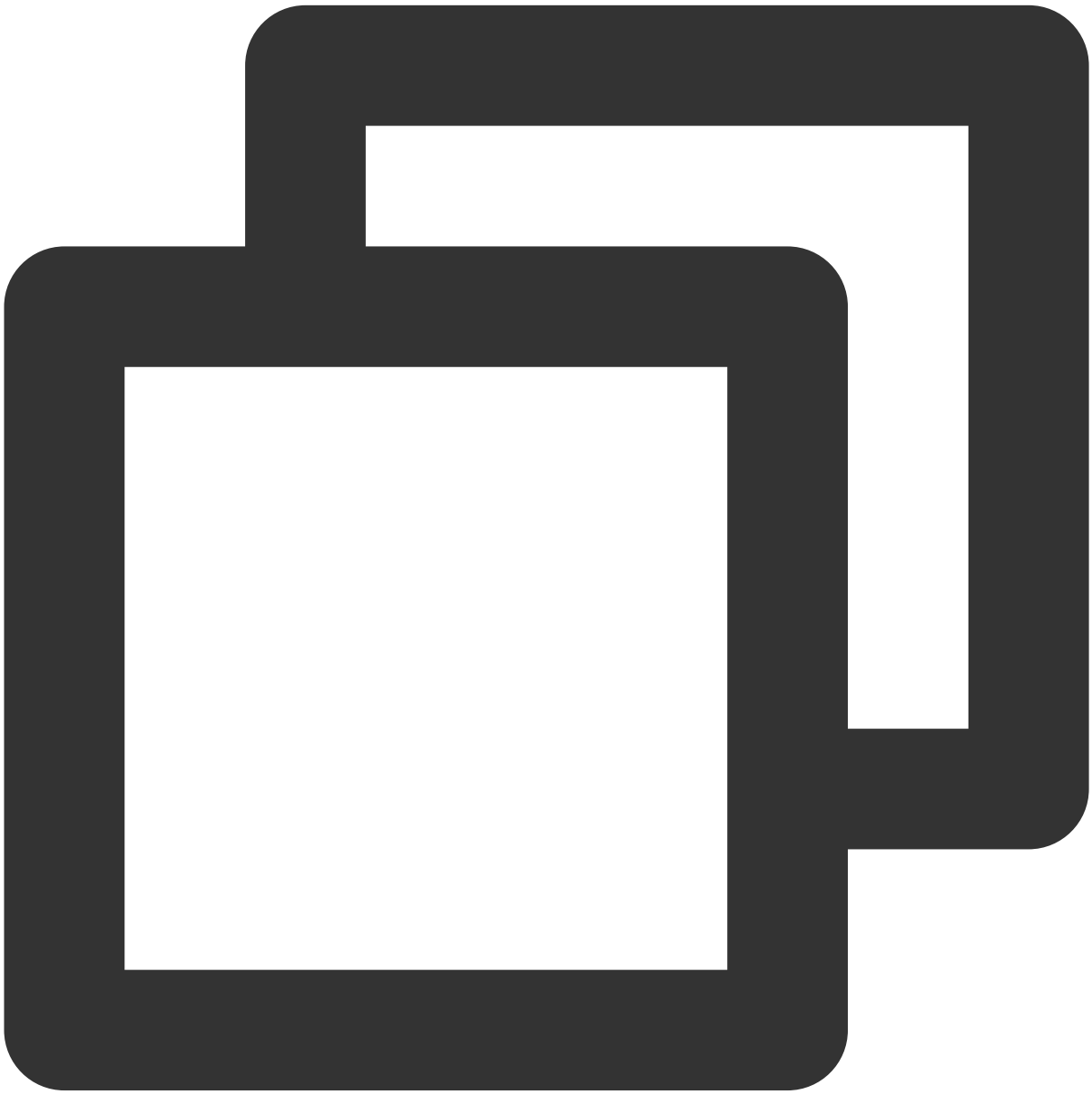


```
// 推送主包必须要集成
implementation 'com.tencent.timpush:timpush:7.9.5668'
// 按照需要集成对应厂商
implementation 'com.tencent.timpush:fcm:7.9.5668'
implementation 'com.tencent.timpush:huawei:7.9.5668'
implementation 'com.tencent.timpush:xiaomi:7.9.5668'
implementation 'com.tencent.timpush:vivo:7.9.5668'
implementation 'com.tencent.timpush:honor:7.9.5668'
implementation 'com.tencent.timpush:meizu:7.9.5668'
//以下二选一
//中国区域选择集成此包
```

```
implementation 'com.tencent.timpush:oppo:7.9.5668'
//其他区域选择集成此包
implementation 'com.tencent.timpush:oppo-intl:7.9.5669'
```

说明：

1. TIMPush 需要集成 IMSDK 在 7.6.5011 版本及以上。
2. 无 UI 或者没有集成其他插件的用户，需要增加集成 TUICore，支持源码和 Maven 集成，方式如下：



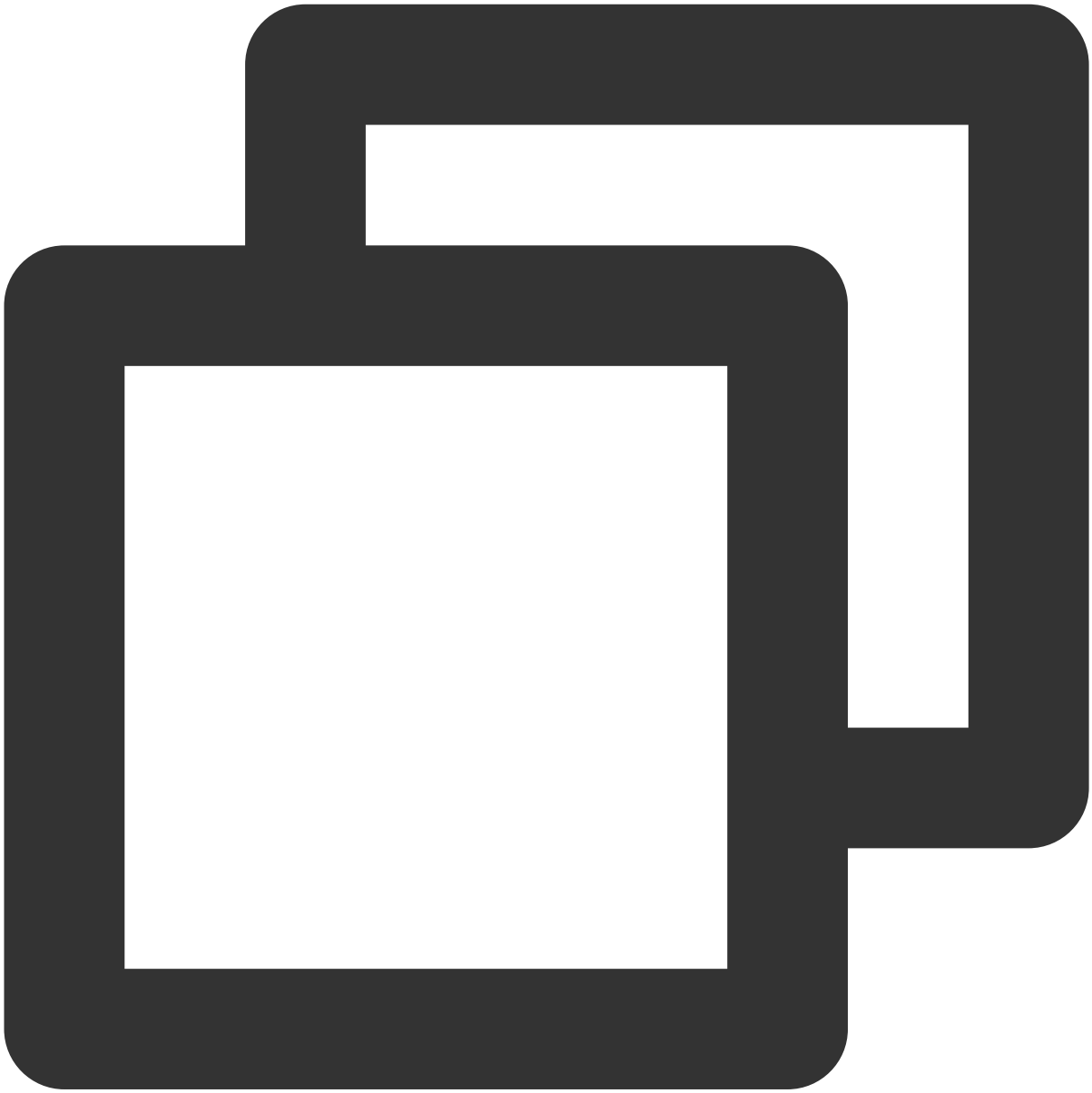
```
def projects = this.rootProject.getAllprojects().stream().map { project -> project.
api projects.contains("tuicore") ? project(':tuicore') : "com.tencent.liteav.tuikit
```

vivo 和荣耀配置

根据 vivo 和荣耀厂商接入指引，需要将 APPID 和 APPKEY 添加到清单文件中，否则会出现编译问题。

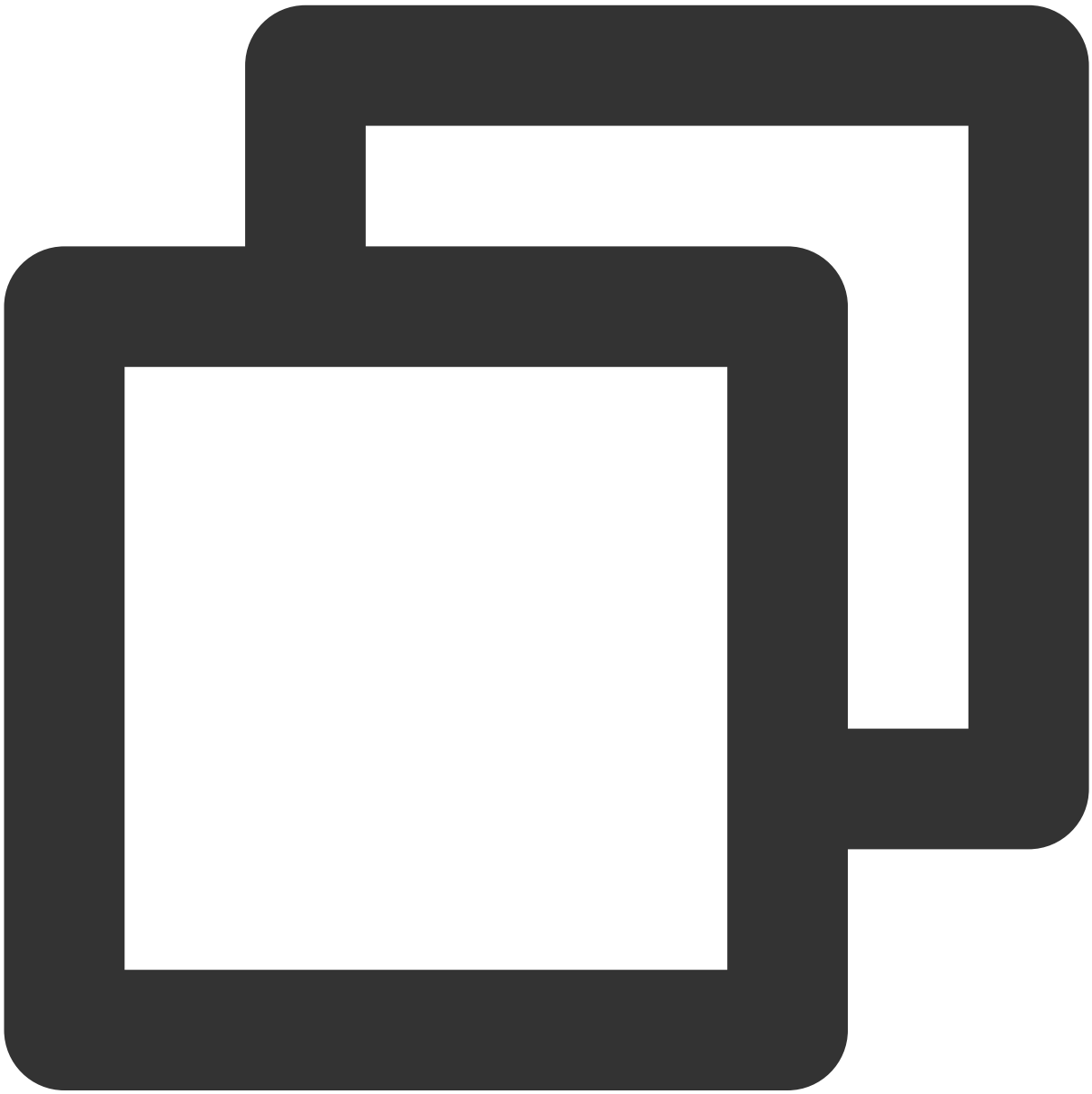
方法1

方法2



```
android {
    ...
    defaultConfig {
        ...
        manifestPlaceholders = [
            "VIVO_APPKEY" : "您应用分配的证书 APPKEY",
```

```
            "VIVO_APPID" : "您应用分配的证书 APPID",
            "HONOR_APPID" : "您应用分配的证书 APPID"
        ]
    }
}
```



```
// vivo begin
<meta-data tools:replace="android:value"
    android:name="com.vivo.push.api_key"
    android:value="您应用分配的证书 APPKEY" />
<meta-data tools:replace="android:value"
```

```
    android:name="com.vivo.push.app_id"
    android:value="您应用分配的证书 APPID" />
// vivo end

// honor begin
<meta-data tools:replace="android:value"
    android:name="com.hihonor.push.app_id"
    android:value="您应用分配的证书 APPID" />
// honor end
```

华为、荣耀和 Google FCM 适配

按照厂商方法，集成对应的 plugin 和 json 配置文件。

注意：

以下荣耀的适配仅 7.7.5283 及以上版本需要配置。

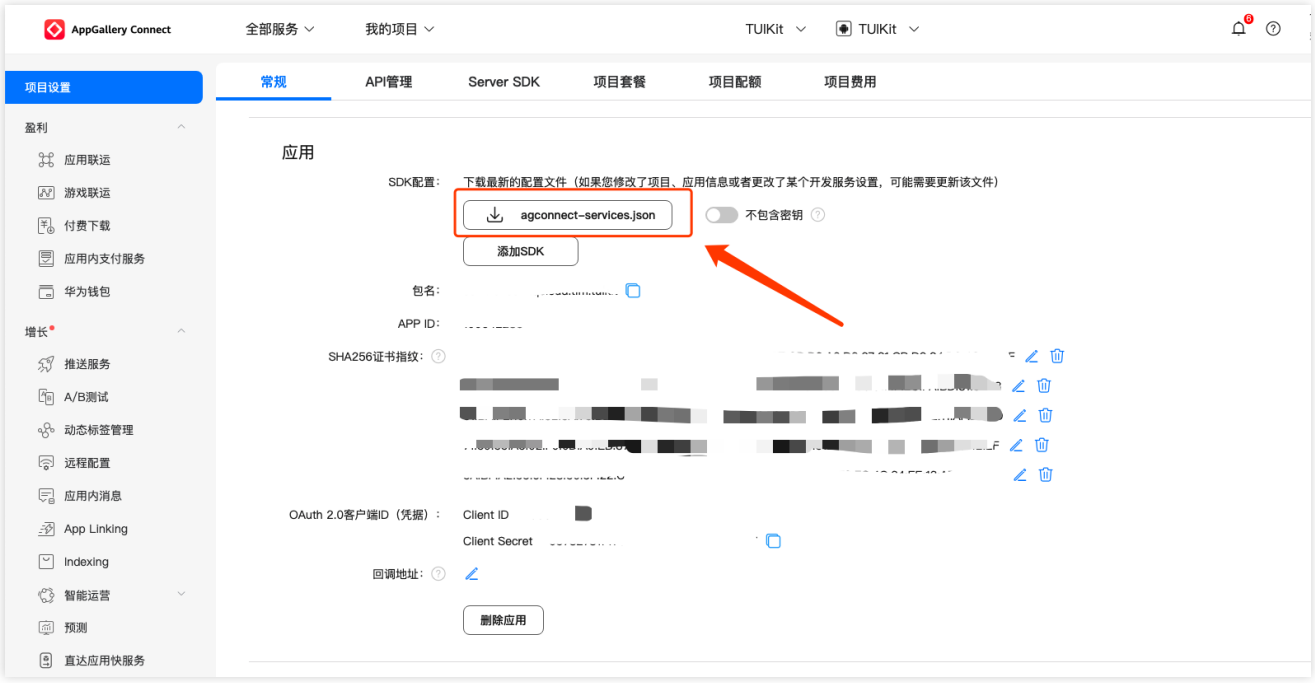
1.1 下载配置文件添加到工程根目录：

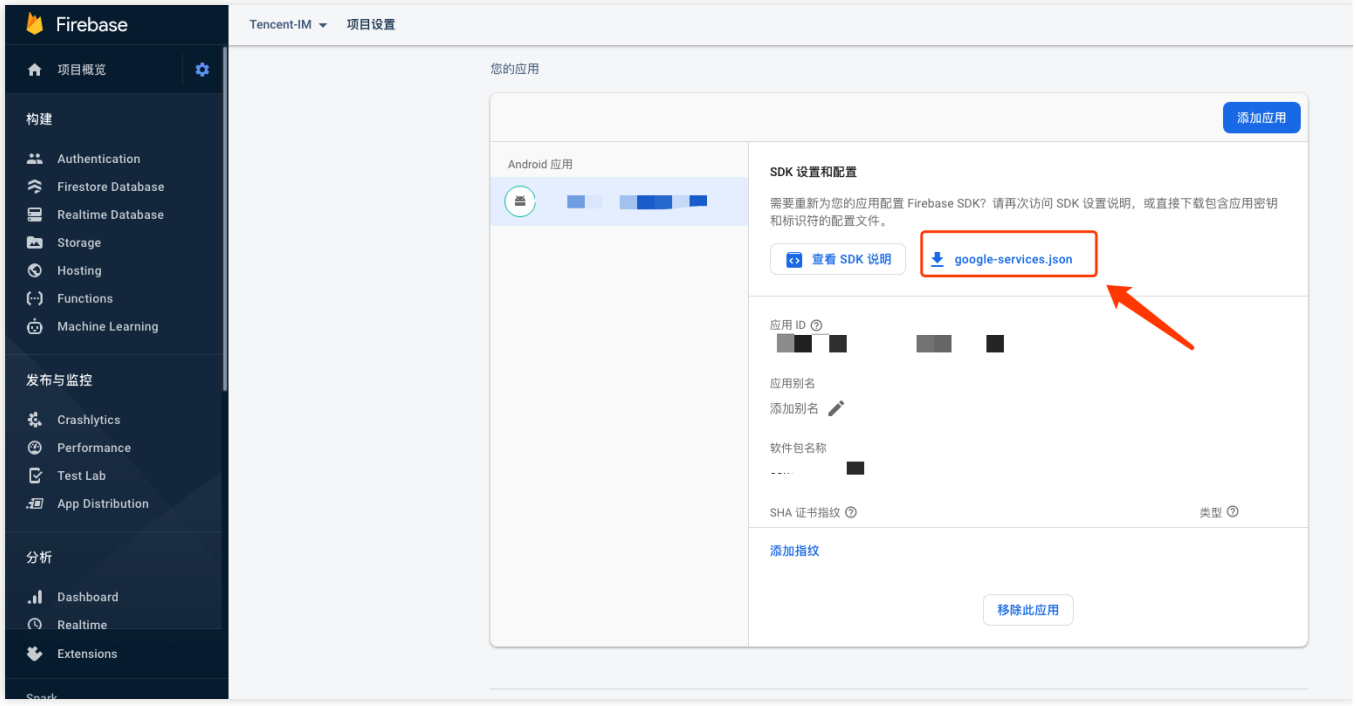
华为

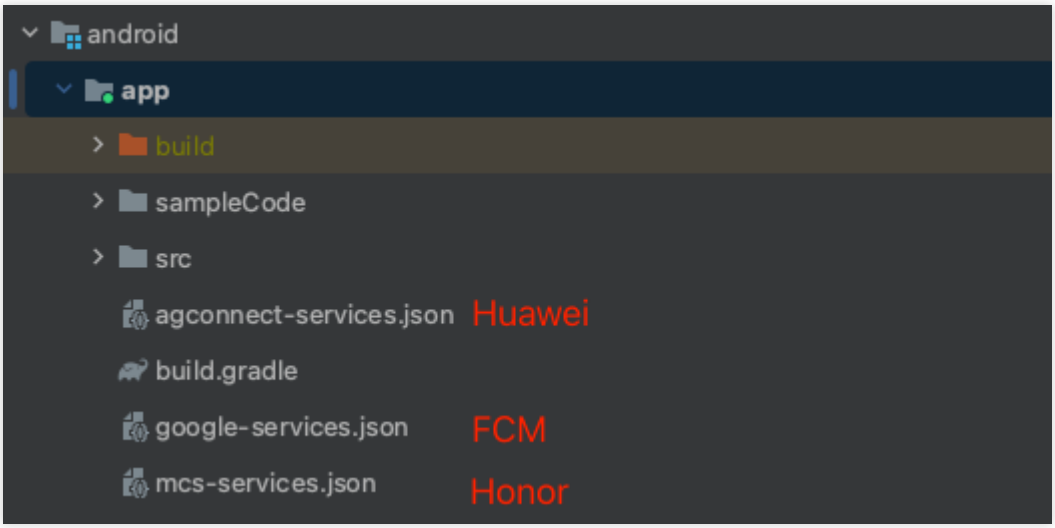
荣耀

Google FCM

操作路径







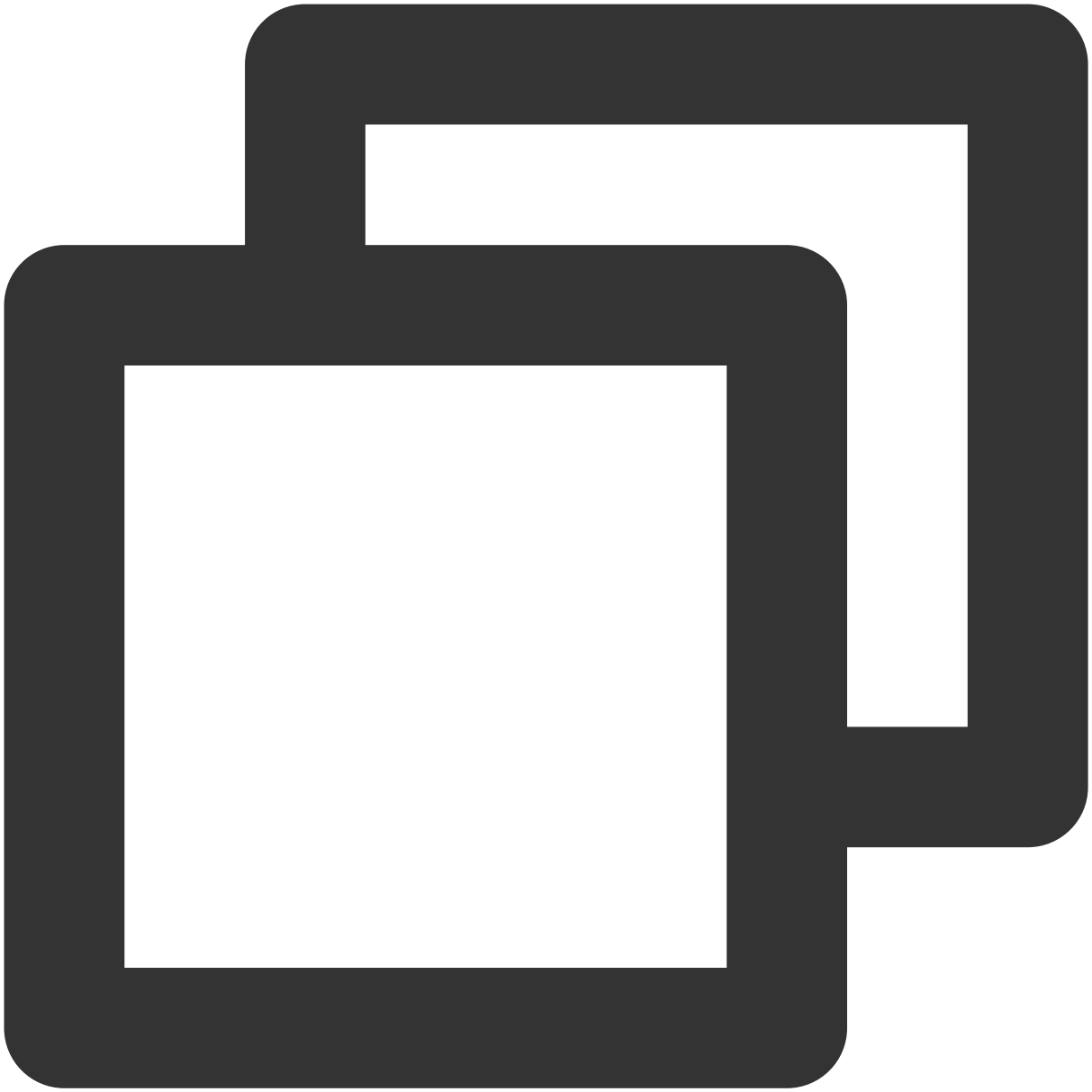
1.2 在项目级 build.gradle 文件中 buildscript -> dependencies 下添加以下配置：

Gradle 7.1 及以上版本

Gradle 7.0 版本

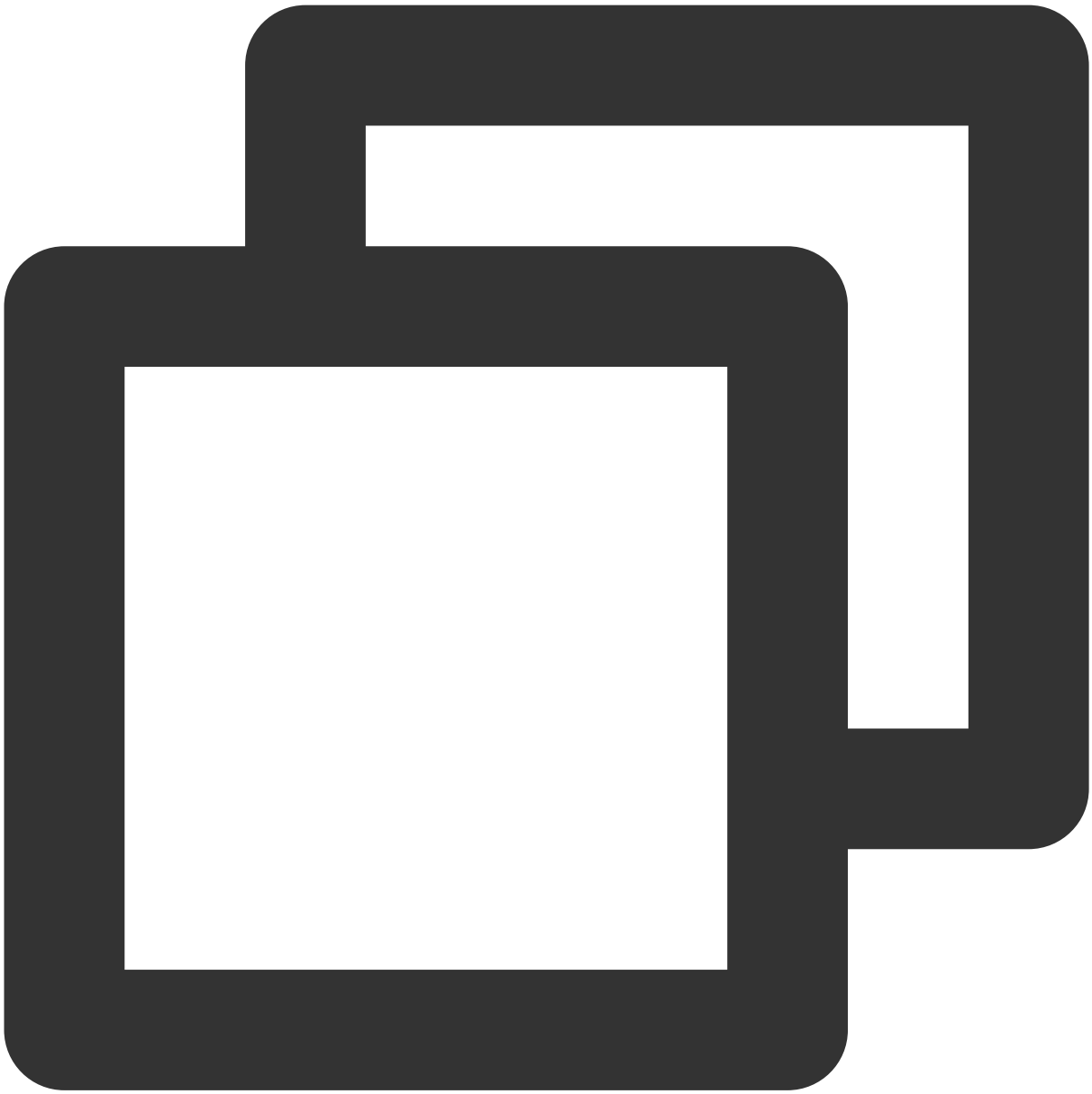
Gradle 7.0 以下版本

在项目级 build.gradle 文件中 buildscript -> dependencies 下添加以下配置：



```
buildscript {
    dependencies {
        ...
        classpath 'com.google.gms:google-services:4.3.15'
        classpath 'com.huawei.agconnect:agcp:1.4.1.300'
        classpath 'com.hihonor.mcs:asplugin:2.0.1.300'
    }
}
```

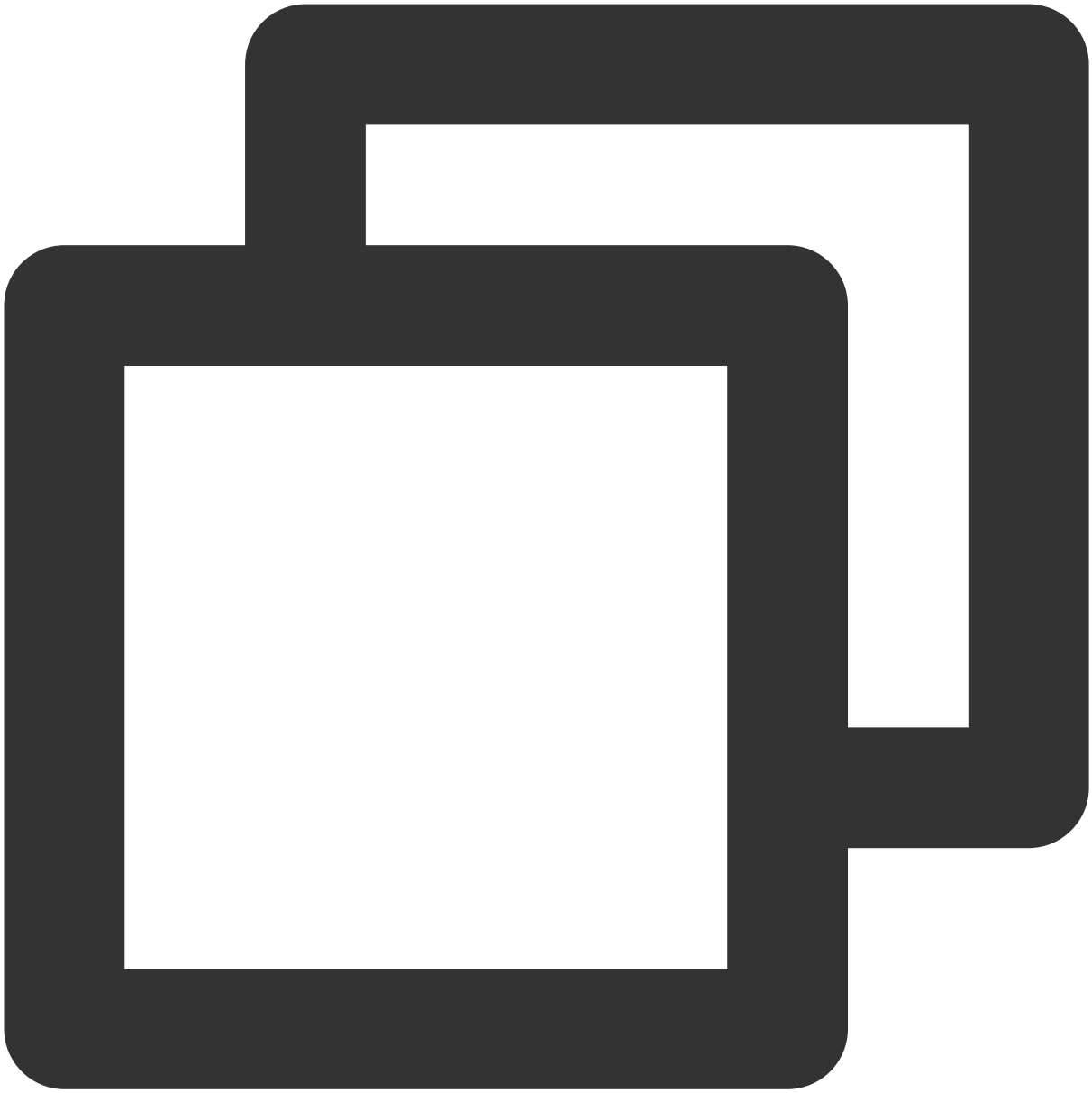
在项目级 settings.gradle 文件中 pluginManagement -> repositories 和 dependencyResolutionManagement -> repositories 下添加以下仓库配置：



```
pluginManagement {
    repositories {
        gradlePluginPortal()
        mavenCentral()
        maven { url "https://mirrors.tencent.com/nexus/repository/maven-public/" }
        // 配置HMS Core SDK的Maven仓地址。
        maven {url 'https://developer.huawei.com/repo/'}
        maven {url 'https://developer.hihonor.com/repo'}
```

```
    }
}
dependencyResolutionManagement {
    ...
    repositories {
        mavenCentral()
        maven { url "https://mirrors.tencent.com/nexus/repository/maven-public/" }
        // 配置HMS Core SDK的Maven仓地址。
        maven {url 'https://developer.huawei.com/repo/'}
        maven {url 'https://developer.hihonor.com/repo'}
    }
    }
}
```

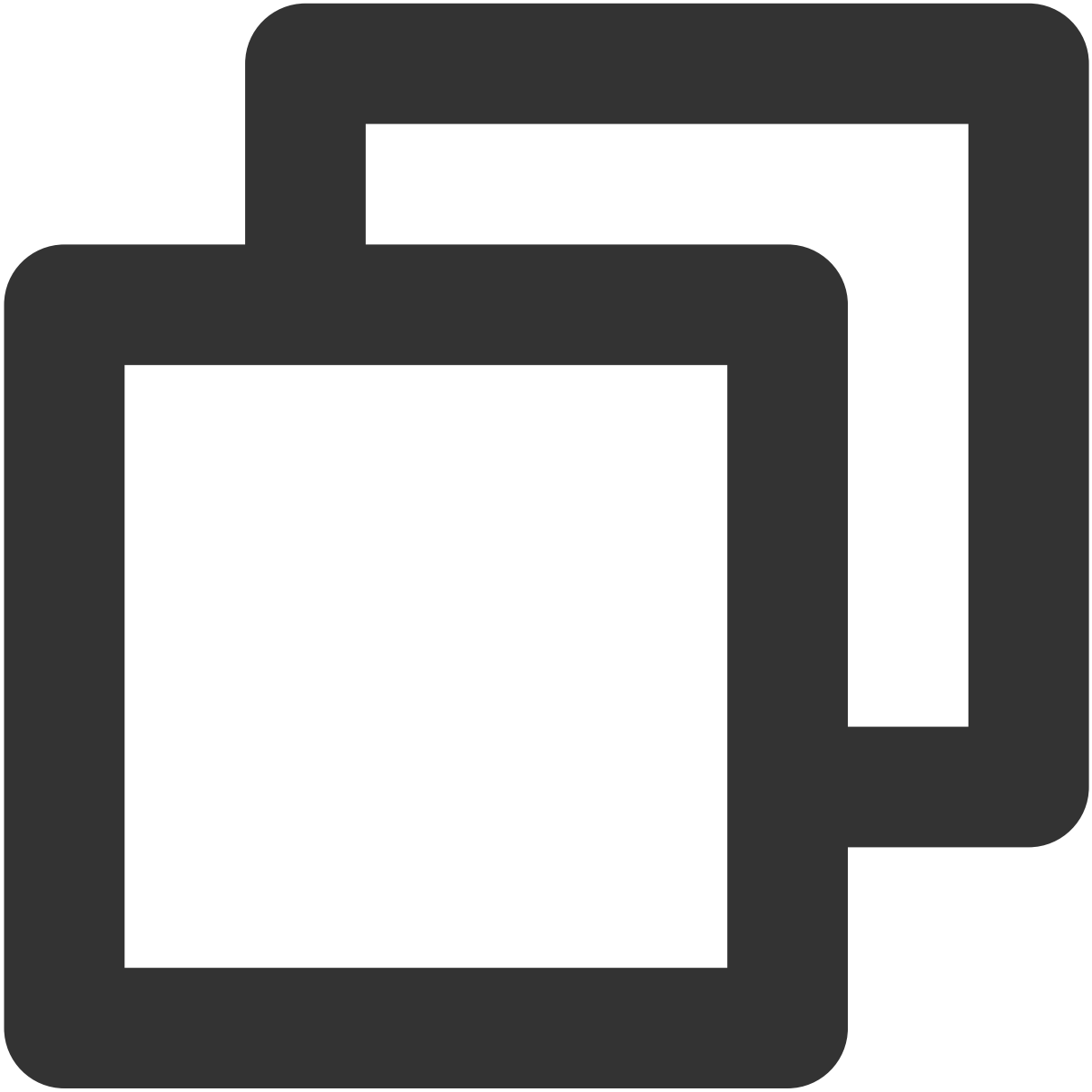
在项目级 build.gradle 文件中 buildscript 下添加以下配置：



```
buildscript {
    repositories {
        mavenCentral()
        maven { url "https://mirrors.tencent.com/nexus/repository/maven-public/" }
        // 配置HMS Core SDK的Maven仓地址。
        maven {url 'https://developer.huawei.com/repo/'}
        maven {url 'https://developer.hihonor.com/repo'}
    }
    dependencies {
        ...
        classpath 'com.google.gms:google-services:4.3.15'
```

```
        classpath 'com.huawei.agconnect:agcp:1.4.1.300'
        classpath 'com.hihonor.mcs:asplugin:2.0.1.300'
    }
}
```

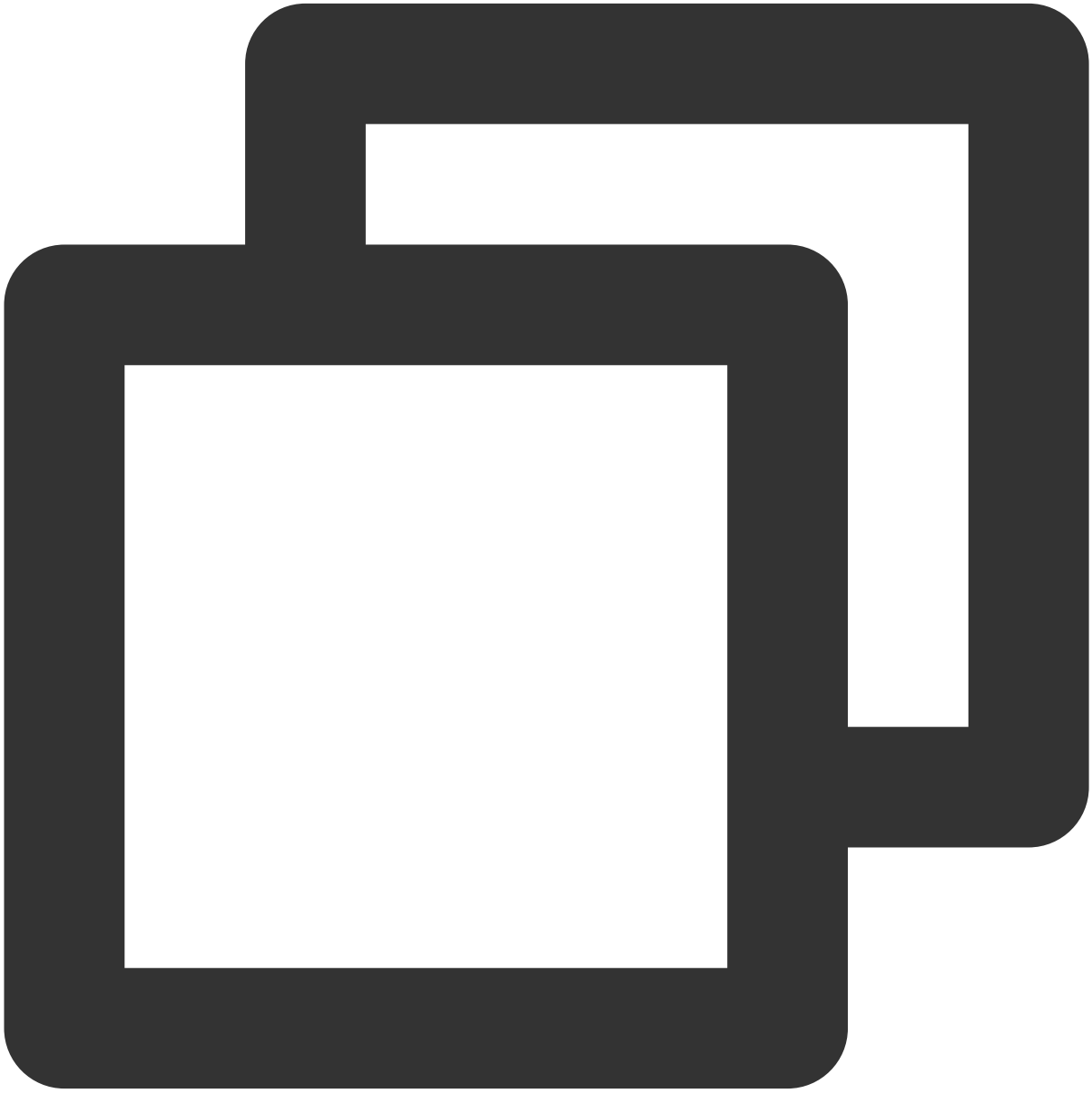
在项目级 settings.gradle 文件中 dependencyResolutionManagement -> repositories 下添加以下仓库配置：



```
dependencyResolutionManagement {
    ...
    repositories {
        mavenCentral()
        maven { url "https://mirrors.tencent.com/nexus/repository/maven-public/" }
```

```
        // 配置HMS Core SDK的Maven仓地址。
        maven {url 'https://developer.huawei.com/repo/'}
        maven {url 'https://developer.hihonor.com/repo'}
    }
}
```

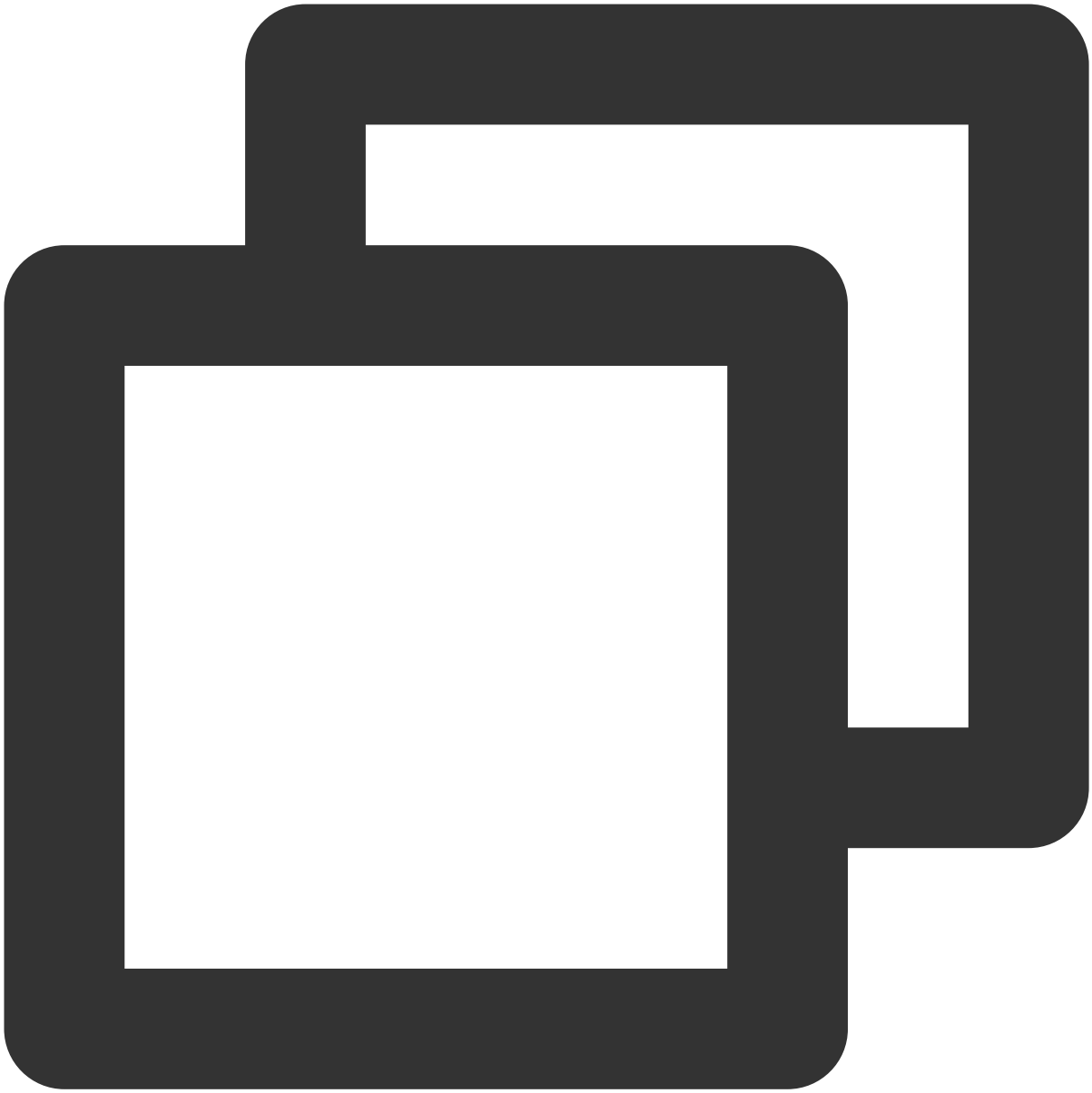
在项目级 build.gradle 文件中 buildscript 和 allprojects 下添加以下配置：



```
buildscript {
    repositories {
```

```
        mavenCentral()
        maven { url "https://mirrors.tencent.com/nexus/repository/maven-public/" }
        // 配置HMS Core SDK的Maven仓地址。
        maven {url 'https://developer.huawei.com/repo/'}
        maven {url 'https://developer.hihonor.com/repo'}
    }
    dependencies {
        ...
        classpath 'com.google.gms:google-services:4.3.15'
        classpath 'com.huawei.agconnect:agcp:1.4.1.300'
        classpath 'com.hihonor.mcs:asplugin:2.0.1.300'
    }
}

allprojects {
    repositories {
        mavenCentral()
        maven { url "https://mirrors.tencent.com/nexus/repository/maven-public/" }
        // 配置HMS Core SDK的Maven仓地址。
        maven {url 'https://developer.huawei.com/repo/'}
        maven {url 'https://developer.hihonor.com/repo'}
    }
}
```

1.3 在应用级 build.gradle 文件中添加下方配置：



```
apply plugin: 'com.google.gms.google-services'
apply plugin: 'com.huawei.agconnect'
apply plugin: 'com.hihonor.mcs.asplugin'
```

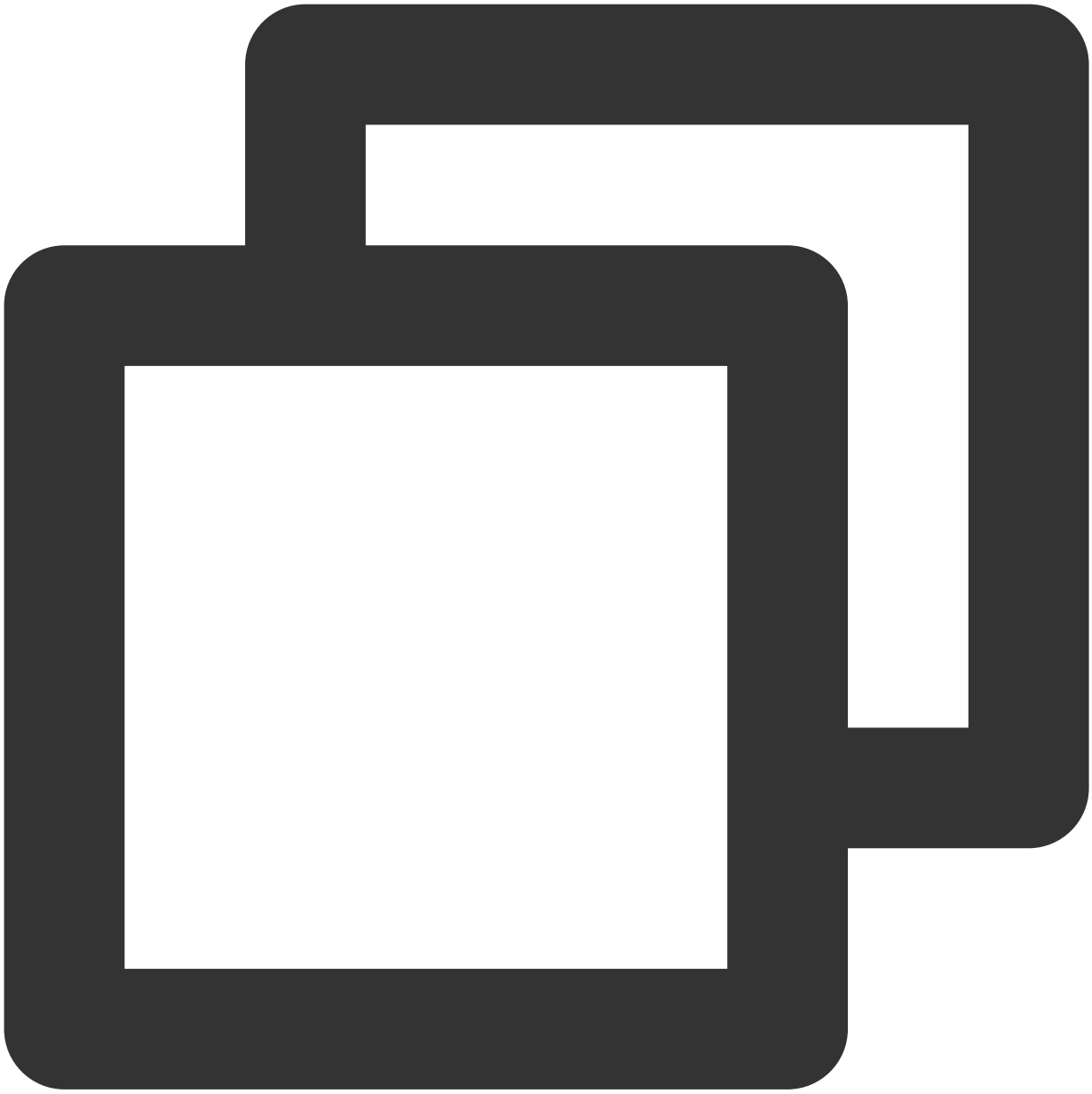
以上步骤完成后，就可以收到离线推送通知了。

注意：

如果您想尽可能简单地接入 TIMPush 组件，您需要使用 TUICore 组件中的 TUILogin 提供的 login/logout 接口登录/登出，此时 TIMPush 组件会自动感知登录/登出事件。如果您不想使用 TUILogin 提供的接口，您需要在完成登录/登出操作后，手动调用 TIMPushManager 的接口 registerPush/unRegisterPush。

步骤3：设置混淆规则

在 proguard-rules.pro 文件，将 TIMPush 相关类加入不混淆名单：



```
-keep class com.tencent.qcloud.** { *; }
-keep class com.tencent.timpush.** { *; }
```

步骤4：消息触达统计配置

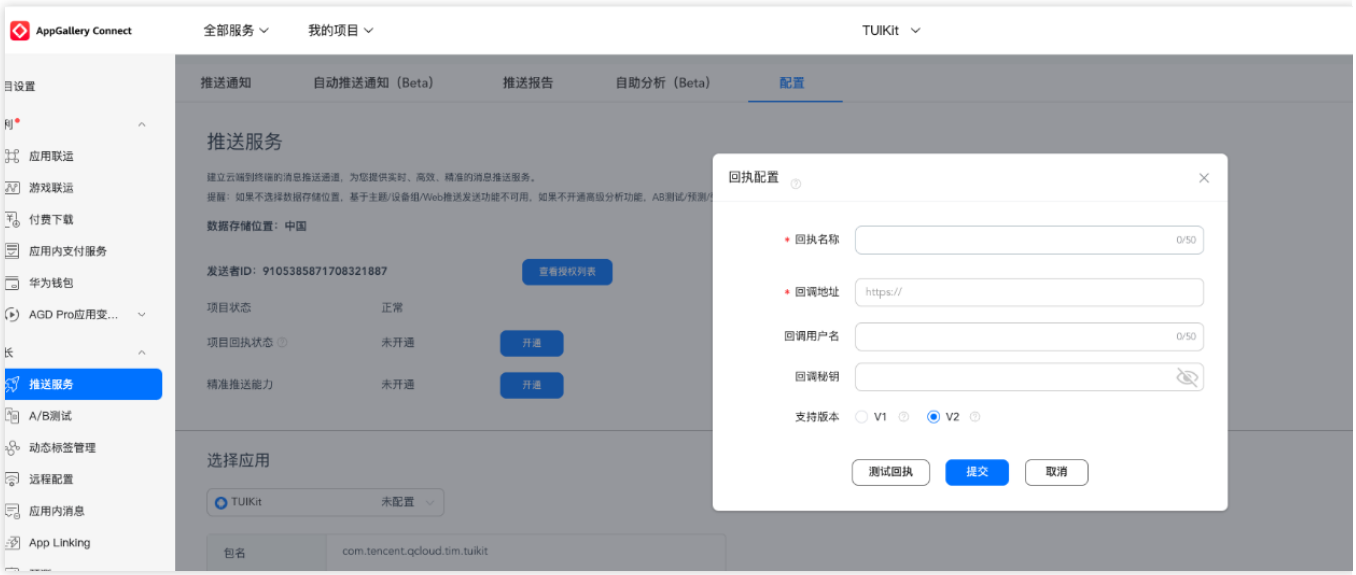
如果您需要统计触达数据，请按照如下完成配置：

华为

荣耀

vivo

魅族

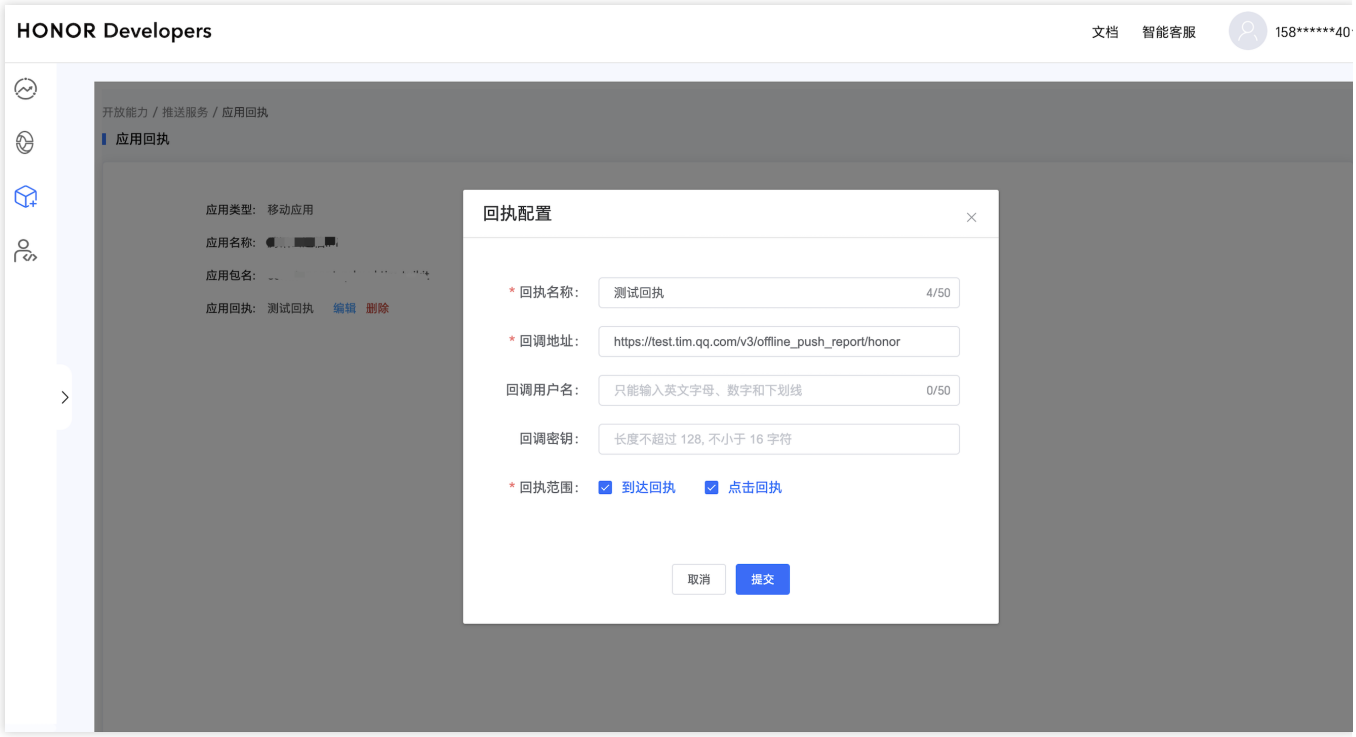


回执地址：

新加坡	`https://apisgp.im.qcloud.com/v3/offline_push_report/huawei`
韩国	`https://apikr.im.qcloud.com/v3/offline_push_report/huawei`
美国	`https://apiusa.im.qcloud.com/v3/offline_push_report/huawei`
德国	`https://apiger.im.qcloud.com/v3/offline_push_report/huawei`
印尼	`https://apiidn.im.qcloud.com/v3/offline_push_report/huawei`
中国	`https://api.im.qcloud.com/v3/offline_push_report/huawei`

注意：

华为推送证书 ID <= 11344 ，使用华为推送 v2 版本接口，不支持触达和点击回执，请重新生成更新证书 ID 。



回执地址：

新加坡	https://apisgp.im.qcloud.com/v3/offline_push_report/honor
韩国	https://apikr.im.qcloud.com/v3/offline_push_report/honor
美国	https://apiusa.im.qcloud.com/v3/offline_push_report/honor
德国	https://apiger.im.qcloud.com/v3/offline_push_report/honor
印尼	https://apiidn.im.qcloud.com/v3/offline_push_report/honor
中国	https://api.im.qcloud.com/v3/offline_push_report/honor

回调地址配置	回执 ID 配置 IM 控制台
	

回执地址：

新加坡 https://apisgp.im.qcloud.com/v3/offline_push_report/vivo 韩国 https://apikr.im.qcloud.com/v3/offline_push_report/vivo 美国 https://apiusa.im.qcloud.com/v3/offline_push_report/vivo 德国 https://apiger.im.qcloud.com/v3/offline_push_report/vivo 印尼 https://apiidn.im.qcloud.com/v3/offline_push_report/vivo 中国 https://api.im.qcloud.com/v3/offline_push_report/vivo	

打开回执开关	配置回执地址
	

回执地址：

新加坡	https://apisgp.im.qcloud.com/v3/offline_push_report/meizu
韩国	https://apikr.im.qcloud.com/v3/offline_push_report/meizu
美国	https://apiusa.im.qcloud.com/v3/offline_push_report/meizu
德国	https://apiger.im.qcloud.com/v3/offline_push_report/meizu
印尼	https://apiidn.im.qcloud.com/v3/offline_push_report/meizu
中国	https://api.im.qcloud.com/v3/offline_push_report/meizu

注意：

打开回执开关后，请务必确保回执地址正确配置。不配置或者配置地址错误，都会影响推送功能。

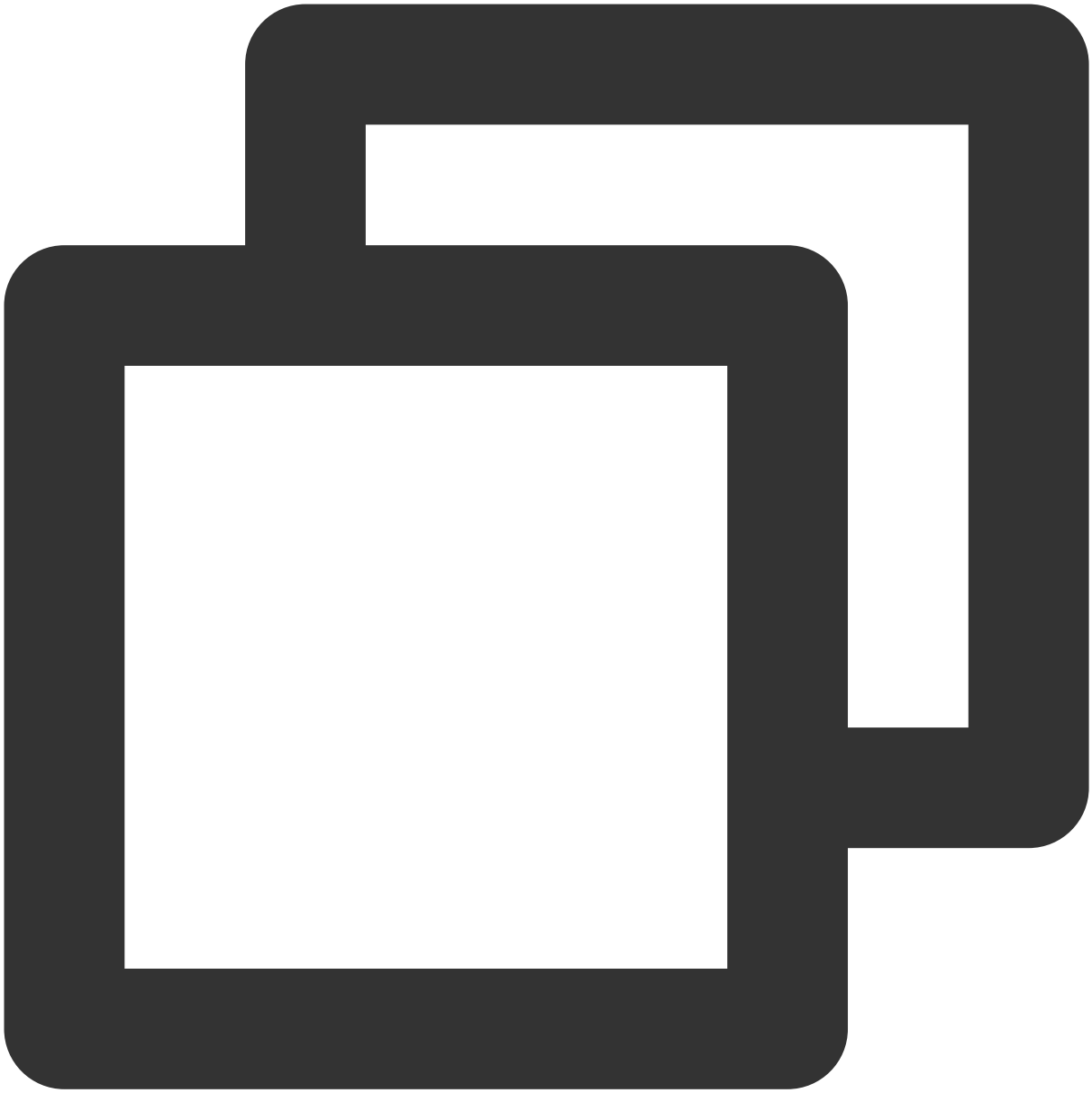
说明：

其他支持厂商无需进行消息触达统计配置。

FCM 暂不支持推送统计功能。

步骤5：发消息时设置离线推送参数

调用 sendMessage 发送消息时，您可以通过 V2TIMOfflinePushInfo 设置离线推送参数。调用 V2TIMOfflinePushInfo 的setExt 设置自定义 ext 数据，当用户收到离线推送启动 App 的时候，可以在单击通知跳转的回调中获取到 ext 字段，然后根据 ext 字段内容跳转到指定的 UI 界面。可以参见 ChatProvider 的 sendMessage() 方法：

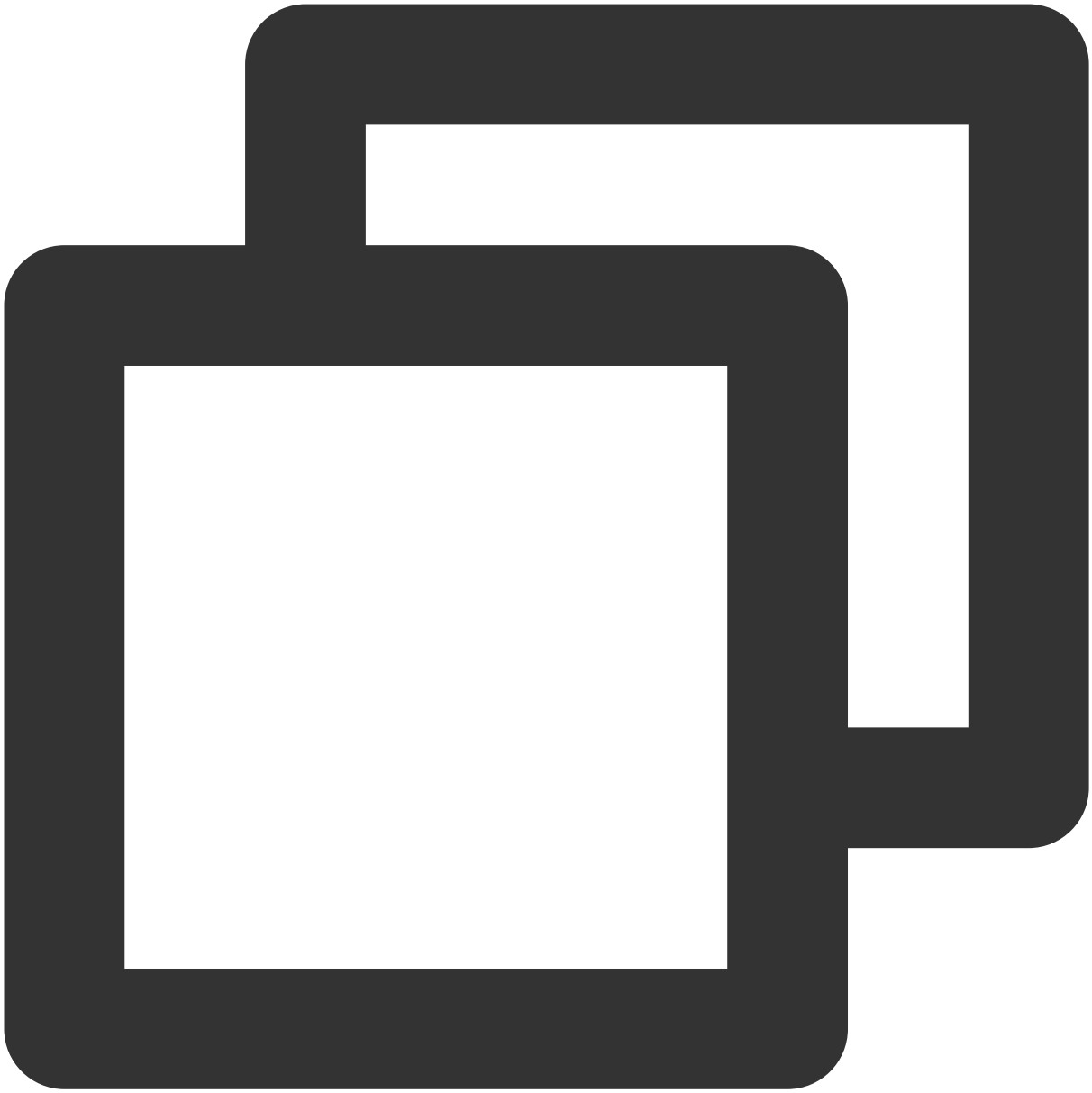


```
V2TIMOfflinePushInfo v2TIMOfflinePushInfo = new V2TIMOfflinePushInfo();
v2TIMOfflinePushInfo.setTitle("推送标题");
v2TIMOfflinePushInfo.setDesc("推送内容");

OfflinePushExtInfo offlinePushExtInfo = new OfflinePushExtInfo();
offlinePushExtInfo.getBusinessInfo().setSenderId("senderID");
offlinePushExtInfo.getBusinessInfo().setSenderNickName("senderNickName");
if (chatInfo.getType() == V2TIMConversation.V2TIM_GROUP) {
    offlinePushExtInfo.getBusinessInfo().setChatType(V2TIMConversation.V2TIM_GROUP)
    offlinePushExtInfo.getBusinessInfo().setSenderId("groupID");
```

```
}
v2TIMOfflinePushInfo.setExt(new Gson().toJson(offlinePushExtInfo).getBytes());

// OPPO必须设置ChannelID才可以收到推送消息，这个channelID需要和控制台一致
v2TIMOfflinePushInfo.setAndroidOPPOChannelID("tuikit");
v2TIMOfflinePushInfo.setAndroidHuaWeiCategory("IM");
v2TIMOfflinePushInfo.setAndroidVIVOCategory("IM");

final V2TIMMessage v2TIMMessage = message.getTimMessage();
String msgID = V2TIMManager.getMessageManager().sendMessage(v2TIMMessage, isGroup ?
    V2TIMMessage.V2TIM_PRIORITY_DEFAULT, false, v2TIMOfflinePushInfo, new V2TIMSend
        @Override
        public void onProgress(int progress) {

        }

        @Override
        public void onError(int code, String desc) {
            TUIChatUtils.callbackOnError(callBack, TAG, code, desc);
        }

        @Override
        public void onSuccess(V2TIMMessage v2TIMMessage) {
            TUIChatLog.v(TAG, "sendMessage onSuccess:" + v2TIMMessage.getMsgID());
            message.setMsgTime(v2TIMMessage.getTimestamp());
            TUIChatUtils.callbackOnSuccess(callBack, message);
        }
    });
```

步骤6：解析离线推送消息

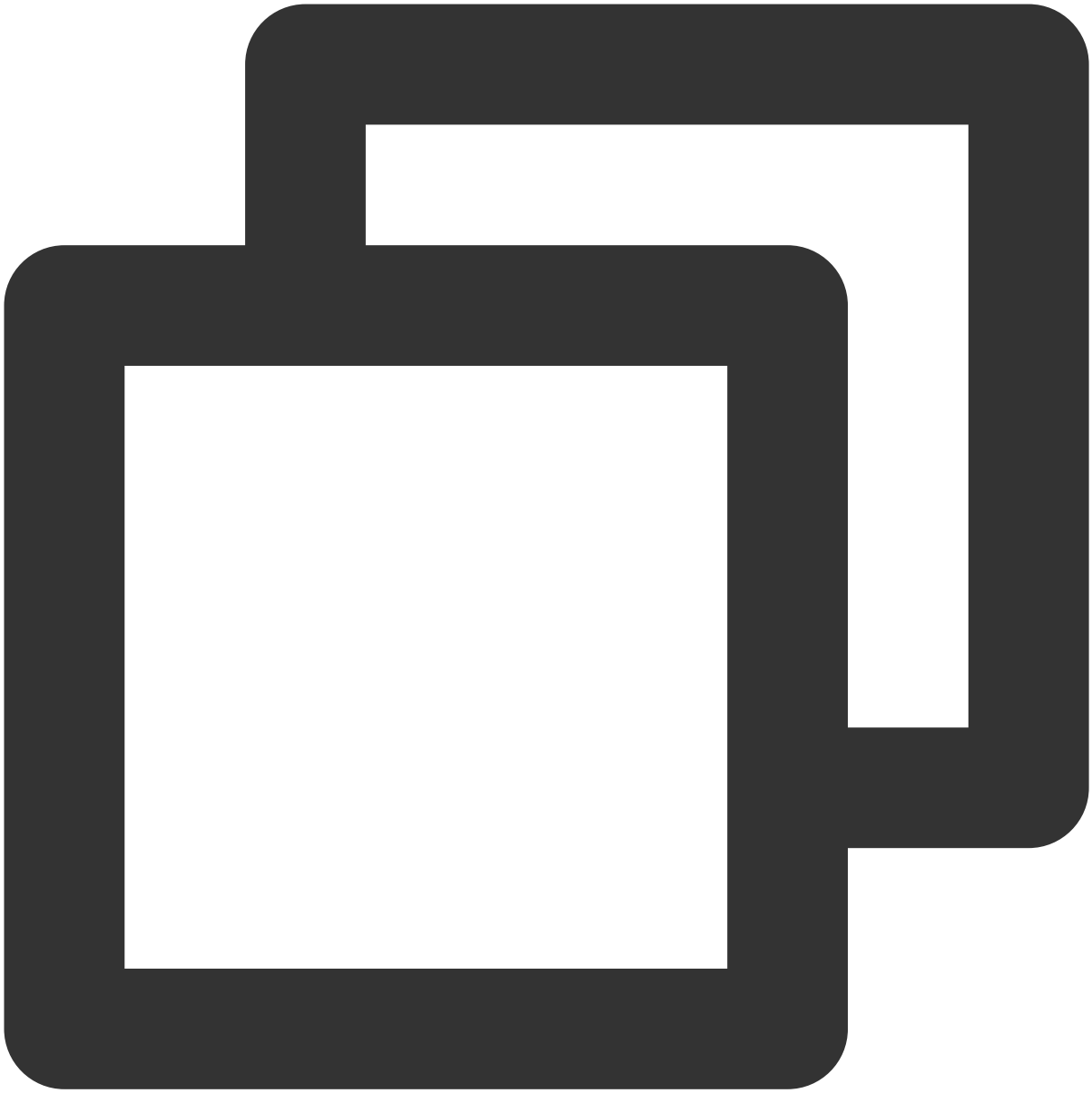
当收到推送时候，点击通知栏的点击事件，组件会以回调或者广播形式通知应用，应用在回调中配置 App 的跳转页面即可。注册回调时机建议放在应用 Application 的 oncreate() 函数中。
回调方式如下：



```
TUICore.registerEvent(TUIConstants.TIMPush.EVENT_NOTIFY, TUIConstants.TIMPush.EVENT
        @Override
        public void onNotifyEvent(String key, String subKey, Map<String, Object> pa
            Log.d(TAG, "onNotifyEvent key = " + key + "subKey = " + subKey);
            if (TUIConstants.TIMPush.EVENT_NOTIFY.equals(key)) {
                if (TUIConstants.TIMPush.EVENT_NOTIFY_NOTIFICATION.equals(subKey))
                    if (param != null) {
                        String extString = (String)param.get(TUIConstants.TIMPush.N
                        // 获取 ext 自定义跳转

                        // 示例：跳转到对应聊天界面
```

```
                        OfflinePushExtInfo offlinePushExtInfo = null;
                        try {
                            offlinePushExtInfo = new Gson().fromJson(extString, Off
                            if (offlinePushExtInfo.getBusinessInfo().getChatAction(
                                String senderId = offlinePushExtInfo.getBusinessInf
                                if (TextUtils.isEmpty(senderId)) {
                                    return;
                                }
                                TUIUtils.startChat(senderId, offlinePushExtInfo.get
                            }
                        } catch (Exception e) {
                            Log.e(TAG, "getOfflinePushExtInfo e: " + e);
                        }
                    }
                }
            }
        }
    });
```

广播方式如下：



```
// 动态注册广播
IntentFilter intentFilter = new IntentFilter();
intentFilter.addAction(TUIConstants.TIMPush.NOTIFICATION_BROADCAST_ACTION);
LocalBroadcastManager.getInstance(context).registerReceiver(localReceiver, intentFi

//广播接收者
public class OfflinePushLocalReceiver extends BroadcastReceiver {
    public static final String TAG = OfflinePushLocalReceiver.class.getSimpleName()

    @Override
    public void onReceive(Context context, Intent intent) {
```

```
        DemoLog.d(TAG, "BROADCAST_PUSH_RECEIVER intent = " + intent);
        if (intent != null) {
            String ext = intent.getStringExtra(TUIConstants.TIMPush.NOTIFICATION_EX
            // 获取 ext 自定义跳转

            // 示例：跳转到对应聊天界面
            OfflinePushExtInfo offlinePushExtInfo = null;
            try {
                offlinePushExtInfo = new Gson().fromJson(extString, OfflinePushExtI
                if (offlinePushExtInfo.getBusinessInfo().getChatAction() == Offline
                    String senderId = offlinePushExtInfo.getBusinessInfo().getSende
                    if (TextUtils.isEmpty(senderId)) {
                        return;
                    }
                    TUIUtils.startChat(senderId, offlinePushExtInfo.getBusinessInfo
                }
             } catch (Exception e) {
                 Log.e(TAG, "getOfflinePushExtInfo e: " + e);
             }
        } else {
            Log.e(TAG, "onReceive ext is null");
        }
    }
}
```

恭喜您已经完成了推送插件的接入，需要提醒您：推送插件**试用或购买到期后，将自动停止提供推送服务（包括普通消息离线推送、全员/标签推送等服务）**。为避免影响您业务正常使用，请提前购买/续费。

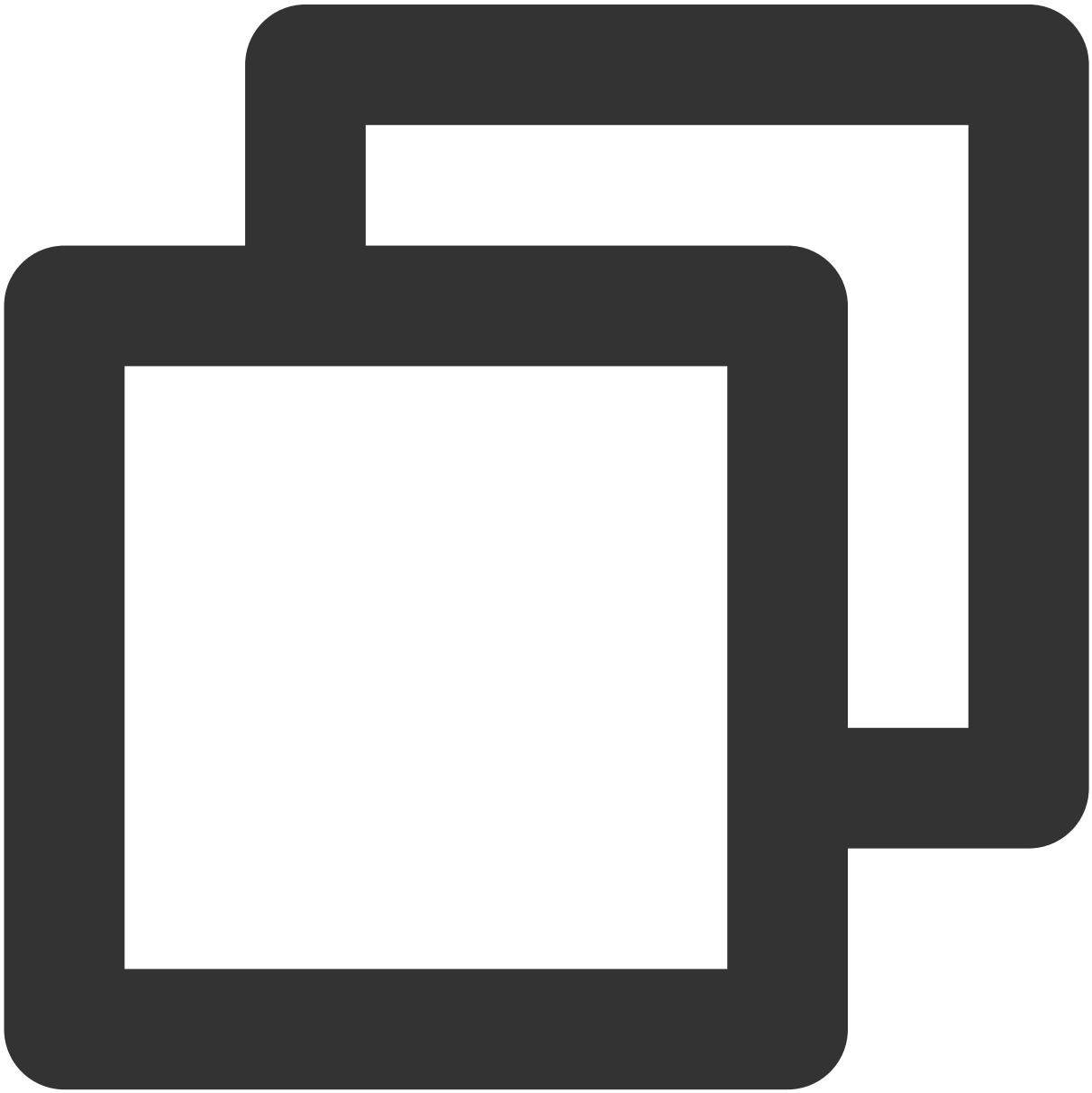
iOS

最近更新时间：2024-06-13 10:39:26

如果您想尽可能简单地接入 TIMPush 组件，您需要使用 TUILogin 的 login/logout 接口进行 IM 账号的登入/登出操作，TIMPush 组件能自动感知登入/登出事件。

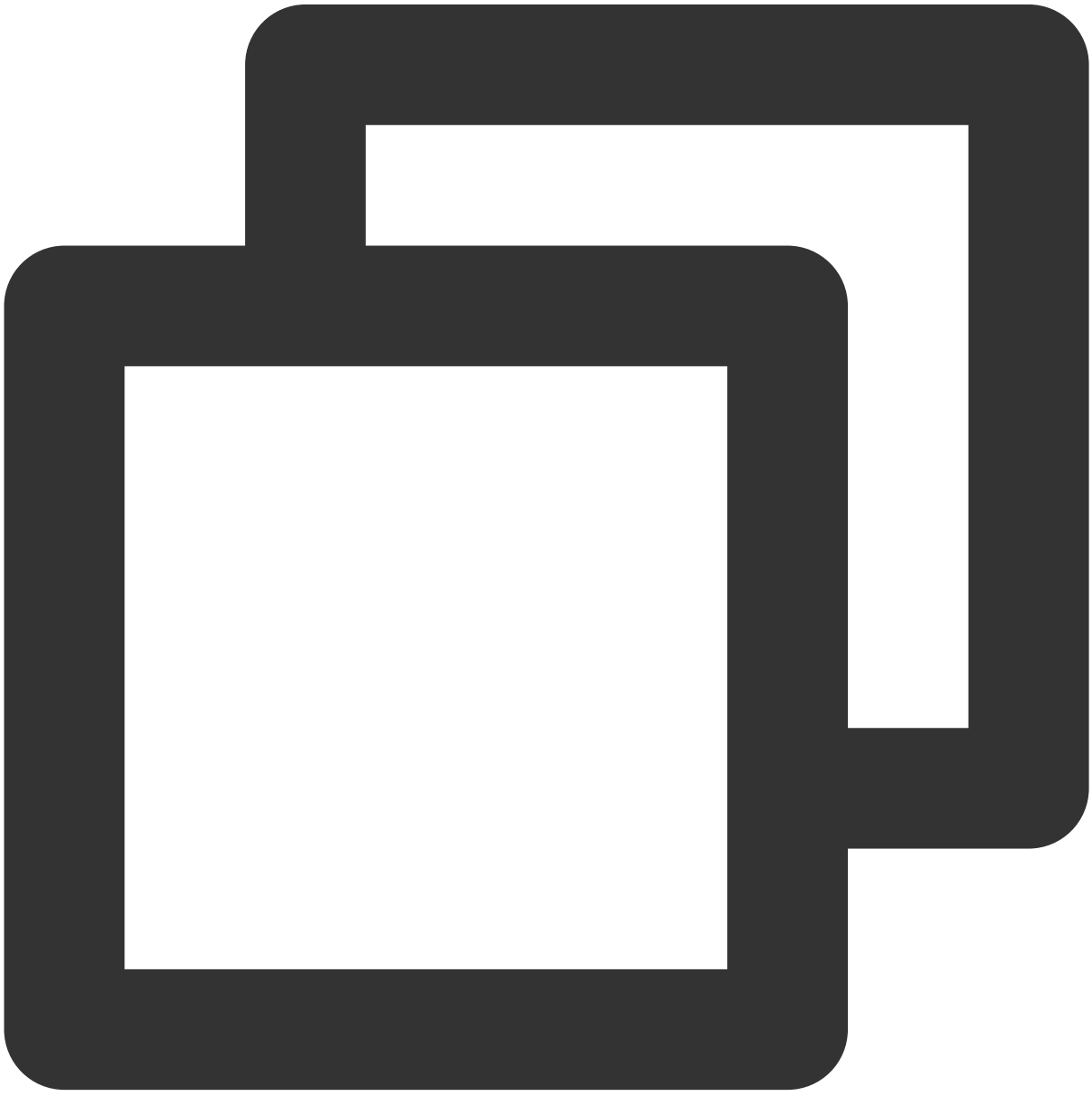
步骤1：集成 TIMPush 组件

1. TIMPush 组件支持 cocoapods 集成，您需要在 Podfile 中添加组件依赖。



```
target 'YourAppName' do
  # Uncommment the next line if you're using Swift or would like to use dynamic fra
  use_frameworks!
  use_modular_headers!

  # Pods for Example
  pod 'TIMPush', '7.9.5668'
end
```

2. 执行以下命令，安装 TIMPush 组件。

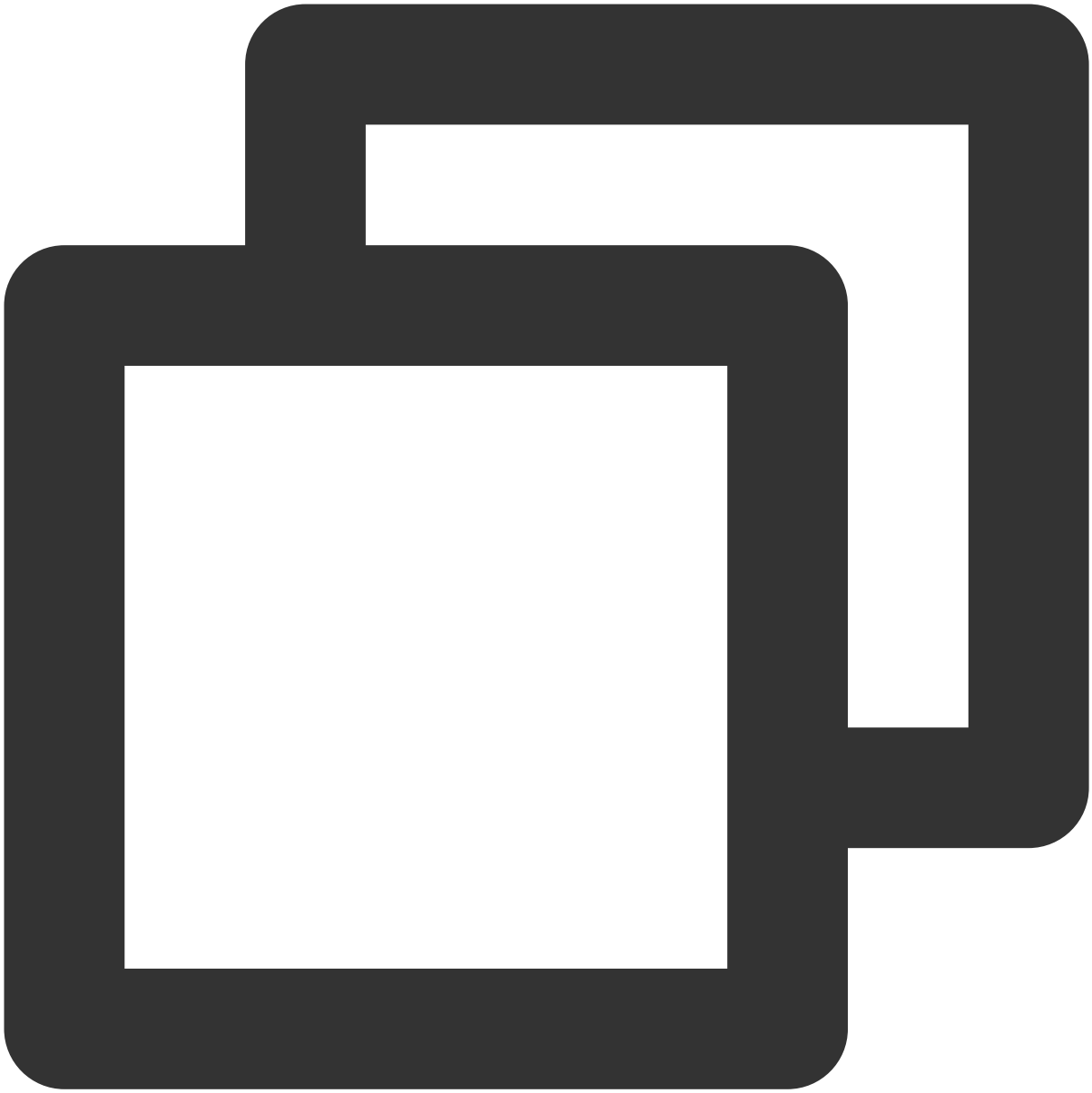


```
pod install
# 如果无法安装 TUIKit 最新版本，执行以下命令更新本地的 CocoaPods 仓库列表。
pod repo update
```

步骤2：配置推送参数

1. 当您上传证书到 IM 控制台后，IM 控制台会为您分配一个证书 ID，见下图：

2. 您需要在 AppDelegate 中，实现 `- offlinePushCertificateID` 协议方法返回证书 ID 即可。



```
#pragma mark - TIMPush

- (int)offlinePushCertificateID {
    return kAPNSBusiId;
}

- (NSString *)applicationGroupID {
    return kTIMPushAppGorupKey;
}

- (void)navigateToBuiltInChatViewController:(NSString *)userID groupID:(NSString *)
```

```
    // custom navigate

}
```

至此，您已经完成了基本推送功能的接入。

注意：

1. 当您登录后，在控制台上看到 APNs configuration success 日志打印时，即表示已成功接入。
2. 如果您的 App 已经获取到了推送权限，此时退入后台或者杀死进程，即可收到远程推送通知。
3. 如果您未接入 TUICore 组件，不需要使用 TUILogin 的登录/登出，但依然想实现离线推送，您只需：

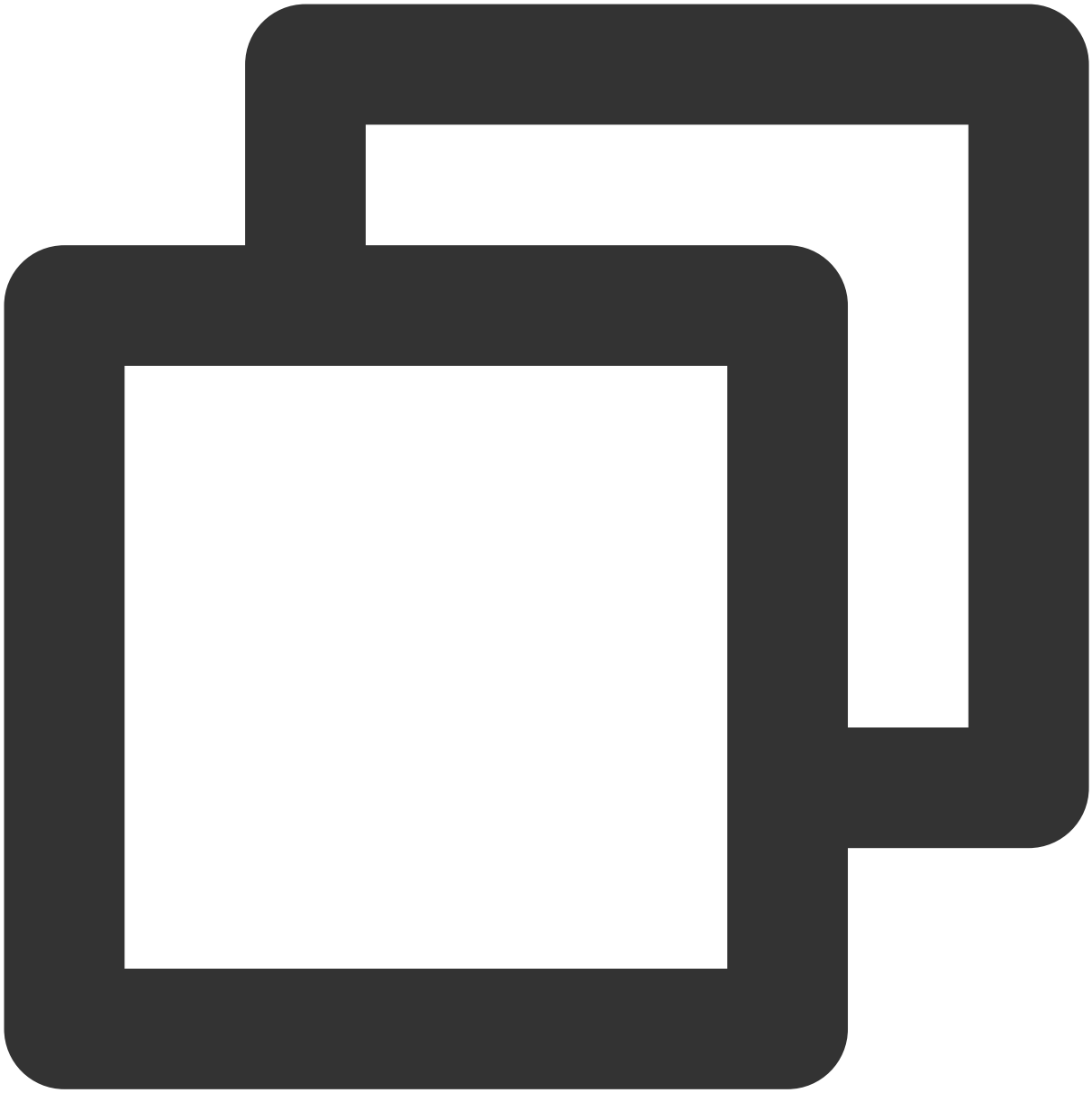
调用 [TIMPush disableAutoRegisterPush]。

在您的 App/IM 登录完成后，主动调用 registerPush 方法注册推送。

退出登录时，主动调用 unRegisterPush 方法 反注册推送。

步骤3：发消息时设置离线推送参数（含 UI 集成时 TUIChat 已默认添加，可跳过此步骤）

调用 sendMessage 发送消息时，您可以通过 V2TIMOfflinePushInfo 设置离线推送参数。调用 V2TIMOfflinePushInfo 的ext 设置自定义 ext 数据，当用户收到离线推送启动 App 的时候，可以在单击通知跳转的回调中获取到 ext 字段，然后根据 ext 字段内容跳转到指定的 UI 界面。可以参见 TUIMessageBaseDataProvider 的 sendMessage: 方法：

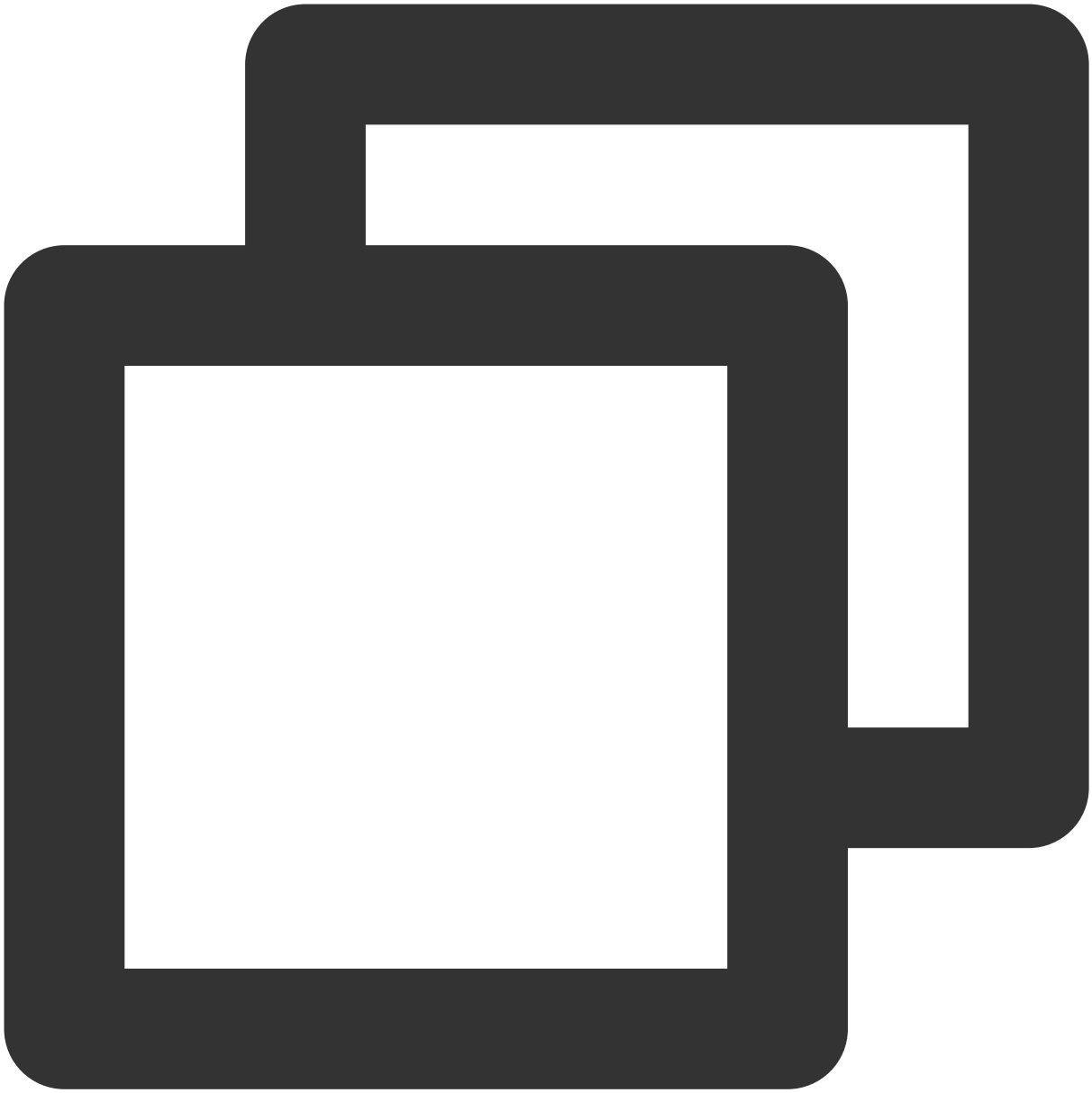


```
#import <TUICore/OfflinePushExtInfo.h>

V2TIMOfflinePushInfo *pushInfo = [[V2TIMOfflinePushInfo alloc] init];
pushInfo.title = @"推送标题";
pushInfo.desc  = @"推送内容";
BOOL isGroup = groupID.length > 0;
NSString *senderId = isGroup ? (groupID) : ([TUILogin getUserID]);
NSString *nickName = isGroup ? (conversationData.title) : ([TUILogin getNickName] ?

OfflinePushExtInfo *extInfo = [[OfflinePushExtInfo alloc] init];
OfflinePushExtBusinessInfo * entity = extInfo.entity;
```

```
entity.action = 1;
entity.content = @"推送内容";
entity.sender = senderId;
entity.nickname = nickName;
entity.faceUrl = [TUILogin getFaceUrl] ?: @"";
entity.chatType = [isGroup ? @(V2TIM_GROUP) : @(V2TIM_C2C) integerValue];
entity.version = kOfflinePushVersion;
pushInfo.ext = [extInfo toReportExtString];
//以下是兼容安卓的字段，需要填写
pushInfo.AndroidOPPOChannelID = @"tuikit";
pushInfo.AndroidSound = TUIConfig.defaultConfig.enableCustomRing ? @"private_ring"
pushInfo.AndroidHuaWeiCategory = @"IM";
pushInfo.AndroidVIVOCategory = @"IM";
```

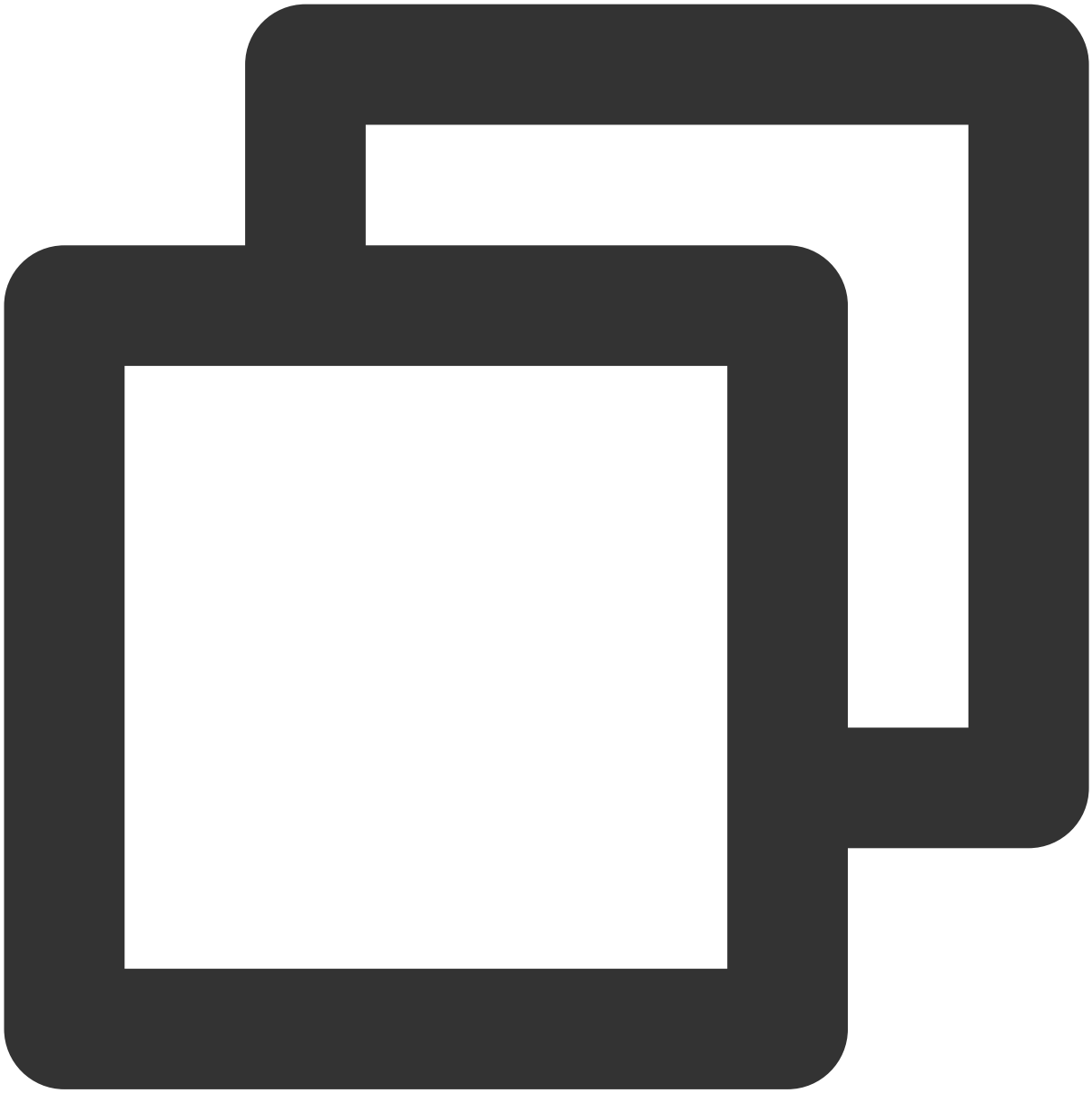
步骤4: 点击离线推送后自定义跳转

1. 如果您需要自定义解析收到的远程推送，您需要在 AppDelegate.m 文件中实现 `-onRemoteNotificationReceived` 方法。



```
#pragma mark - TIMPush

- (BOOL)onRemoteNotificationReceived:(NSString *)notice {
    //- 如果返回 YES， TIMPush 将不在执行内置的 TUIKit 离线推送解析逻辑，完全交由您自行处理；
    //NSString *ext = notice;
    //OfflinePushExtInfo *info = [OfflinePushExtInfo createWithExtString:ext];
    //return YES;

    //- 如果返回 NO，TIMPush 将继续执行内置的 TUIKit 离线推送解析逻辑，继续回调 - navigateTo
    return NO;
}
```

2. 如果您想使用内置的 TUIChat 推送解析逻辑，并跳转到 TUIChat 的聊天页面，您可以实现 `-navigateToBuiltInChatViewController` 方法。

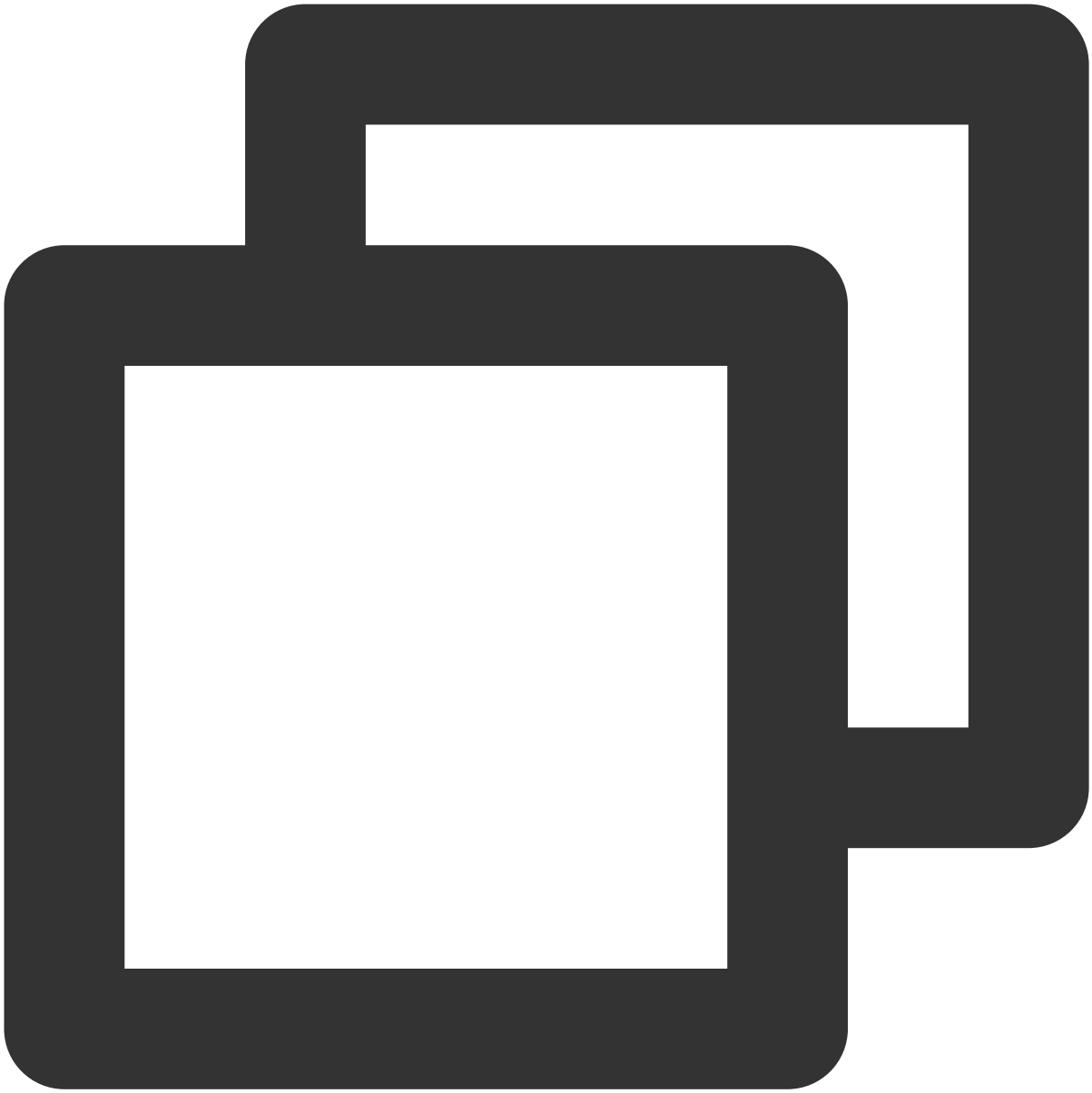


```
#pragma mark - TIMPush

- (void)navigateToBuiltInChatViewController:(NSString *)userID groupID:(NSString *)

    UITabBarController *tab = [self getMainController];

    if (![tab isKindOfClass: UITabBarController.class]) {
```

```
        // 正在登录中
        return;
    }

    if (tab.selectedIndex != 0) {
        [tab setSelectedIndex:0];
    }
    self.window.rootViewController = tab;
    UINavigationController *nav = (UINavigationController *)tab.selectedViewControl
    if (![nav isKindOfClass:UINavigationController.class]) {
        return;
    }
    UIViewController *vc = nav.viewControllers.firstObject;
    if (![vc isKindOfClass:NSClassFromString(@"ConversationController")]
        && ![vc isKindOfClass:NSClassFromString(@"ConversationController_Minimalist
        return;
    }
    if ([vc respondsToSelector:NSSelectorFromString(@"pushToChatViewController:user
        [vc performSelector:NSSelectorFromString(@"pushToChatViewController:userID:
    }
}
```

步骤5: 统计推送抵达率

1. 如果您需要统计推送的抵达和点击数据，您需要在 AppDelegate.m 文件中实现 `- applicationGroupID` 方法，返回 App Group ID （生成方式可参见 厂商配置-生成 App GroupID）。
2. 在 Notification Service Extension 的 `-didReceiveNotificationRequest:withContentHandler:` 方法中调用推送抵达率统计函数:



```
  @implementation NotificationService

- (void)didReceiveNotificationRequest:(UNNotificationRequest *)request withContentH
    //appGroup 标识当前主 APP 和 Extension 之间共享的 APP Group，需要在主 APP 的 Capabili
    //格式为group + [主bundleID]+ key
    //如group.com.tencent.im.pushkey
    NSString * appGroupID = kTIMPushAppGorupKey;
    __weak typeof(self) weakSelf = self;
    [TIMPush onReceiveNotificationRequest:request inAppGroupID:appGroupID callback:
        weakSelf.bestAttemptContent = [content mutableCopy];
        // Modify the notification content here...
```

```
        // self.bestAttemptContent.title = [NSString stringWithFormat:@"%@ [modifie
        weakSelf.contentHandler(weakSelf.bestAttemptContent);
    }];
}
```

注意：

1. 上报推送触达数据，需要开启 mutable-content 开关来支持 iOS 10 的 Extension 功能。

2. 数据详情可在推送数据页面查看，推送数据页面仅限 购买推送插件 后使用。

关于合规

TIMPush 在您未主动调用 registerPush 之前，不会有其他任何操作，符合相关规定。

如果您使用了 TUILogin 的登录登出，TIMPush 会在内部自动调用 registerPush 或 unRegisterPush。

恭喜您已经完成了推送插件的接入，需要提醒您：消息推送插件**试用或购买到期后，将自动停止提供推送服务（包括普通消息离线推送、全员推送等服务）**。为避免影响您业务正常使用，请提前购买/续费。

关于全员/标签推送

全员/标签推送支持发送特定内容，还可根据标签、属性，针对特定用户群体发送个性化内容，例如会员活动、区域通知等，助力拉新、转化、促活等各个阶段运营工作的有效进行，同时支持推送送达报表，自助推送故障排查工具。

更多详细内容建议查阅全员/标签推送

uniapp

最近更新时间：2024-06-13 10:40:32

前置条件

1. 集成 Chat TUIKit，可参见 含 UI 集成方案（荐） -> 集成基础功能 ->uni-app（Vue2/Vue3）

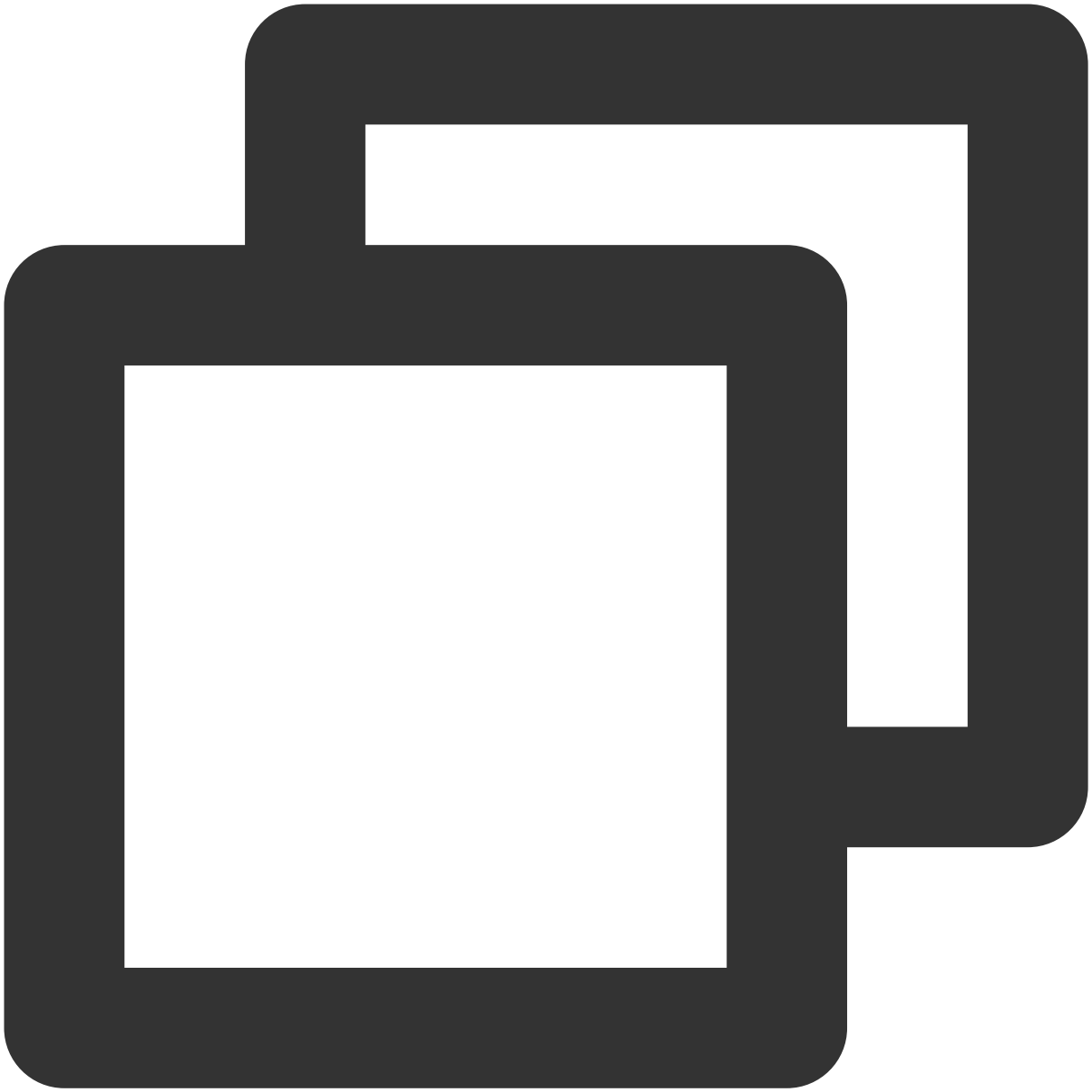
说明：

1. 已集成 Chat TUIKit 可忽略此步骤。
2. 无 UI 集成 Chat 可忽略此步骤。

2. 升级 @tencentcloud/chat 到最新版本。

说明：

@tencentcloud/chat 向下兼容，可放心升级。如果您当前用的是 **tim-js-sdk** 或 **tim-wx-sdk**，请参考我们的升级指引。

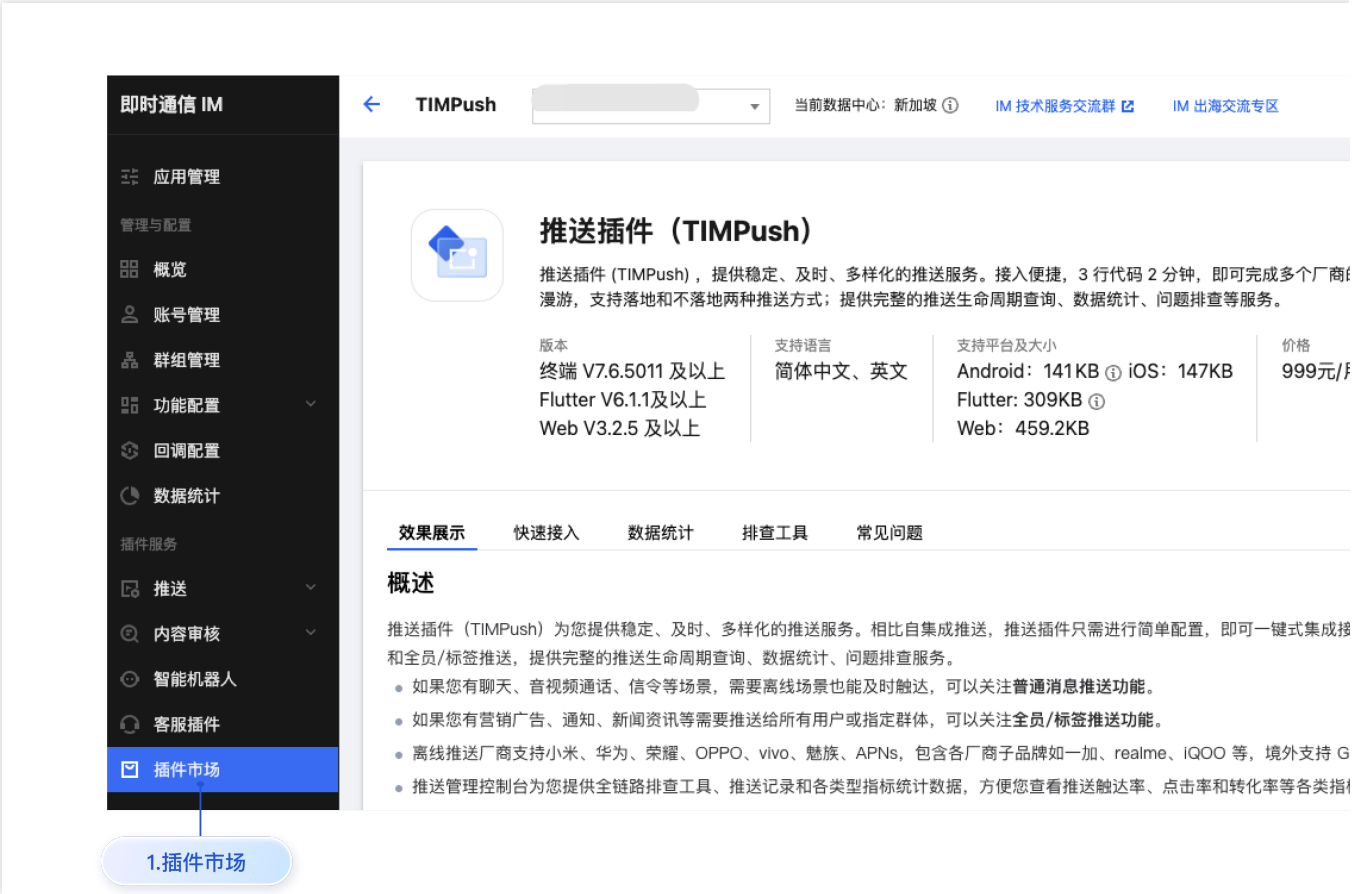


```
npm install @tencentcloud/chat@latest
```

在 HBuilder 日志中查看 TencentCloudChat.VERSION 版本号，来确认 **@tencentcloud/chat ≥ 3.2.5** 如图所示：

3. 开通推送插件

进入 IM 控制台 > 插件市场，单击**立即购买**或**免费试用** 。（每个应用可免费试用一次，有效期7天。）



注意：

推送插件试用或购买到期后，将自动停止提供推送服务（包括普通消息离线推送、全员/标签推送等服务）。为避免影响您业务正常使用，请提前购买/续费。

4. 厂商配置

说明：

uniapp 厂商配置包含 Android 厂商配置 和 iOS 厂商配置， 可参见 厂商配置 - uniapp。

集成 TencentCloud-TIMPush

步骤1：manifest.json App 模块配置

在项目 **【manifest.json】** > **【App 模块配置】** 中配置消息推送模块，如图所示：

步骤2：开通 TencentCloud-TIMPush 云打包服务，填写相关参数。

1. 前往插件市场，开通 TencentCloud-TIMPush 云打包服务。如图所示：

注意：

1. 插件市场开通云打包服务项目的 **appId** 需要和项目 **manifest.json** 中的 **appid** 一致。
2. 开通云打包服务的 **TencentCloud-TIMPush** 仅用于一个项目。仅与项目有关。

2. 在项目的 **【manifest.json】** > **【 App 原生插件配置】** > **【云端插件】**中勾选 **TencentCloud-TIMPush**，并设置相关参数。

注意：

1. 注意在 HBuilderX 中参数可能会出现乱序现象，请仔细认真填写。
2. 各个参数为必填项，否则会编译错误，默认为 0。

com.hihonor.push.app_id

com.vivo.push.app_id && com.vivo.push.api_key

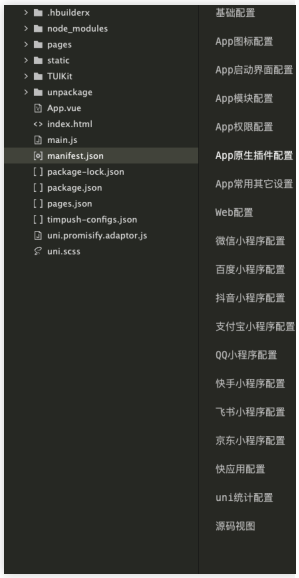
TIMPushAppGroupID

说明：

com.hihonor.push.app_id 对应 hihonor 的 appID。

不启用 hihonor 推送时，com.hihonor.push.app_id 默认设置为 0 即可。

控制台配置	manifest.json 荣耀配置



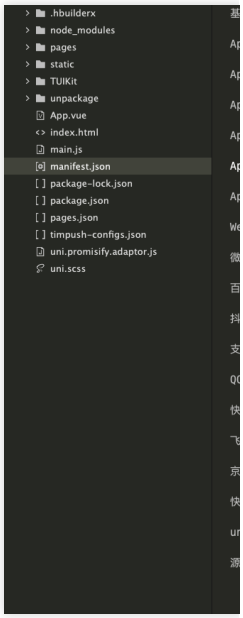
说明：

com.vivo.push.api_id 对应 vivo 的 appID。

com.vivo.push.api_key 对应 vivo 的 appKey。

不启用 vivo 推送时，com.vivo.push.api_id 和 com.vivo.push.api_key 默认设置为 0 即可。

控制台配置	manifest.json vivo 配置



TIMPushAppGroupID 对应 iOS 的 appGroupID，它是 iOS 的触达上报的配置项，可参考 configuring-app-groups 生成 appGroupID。

注意：

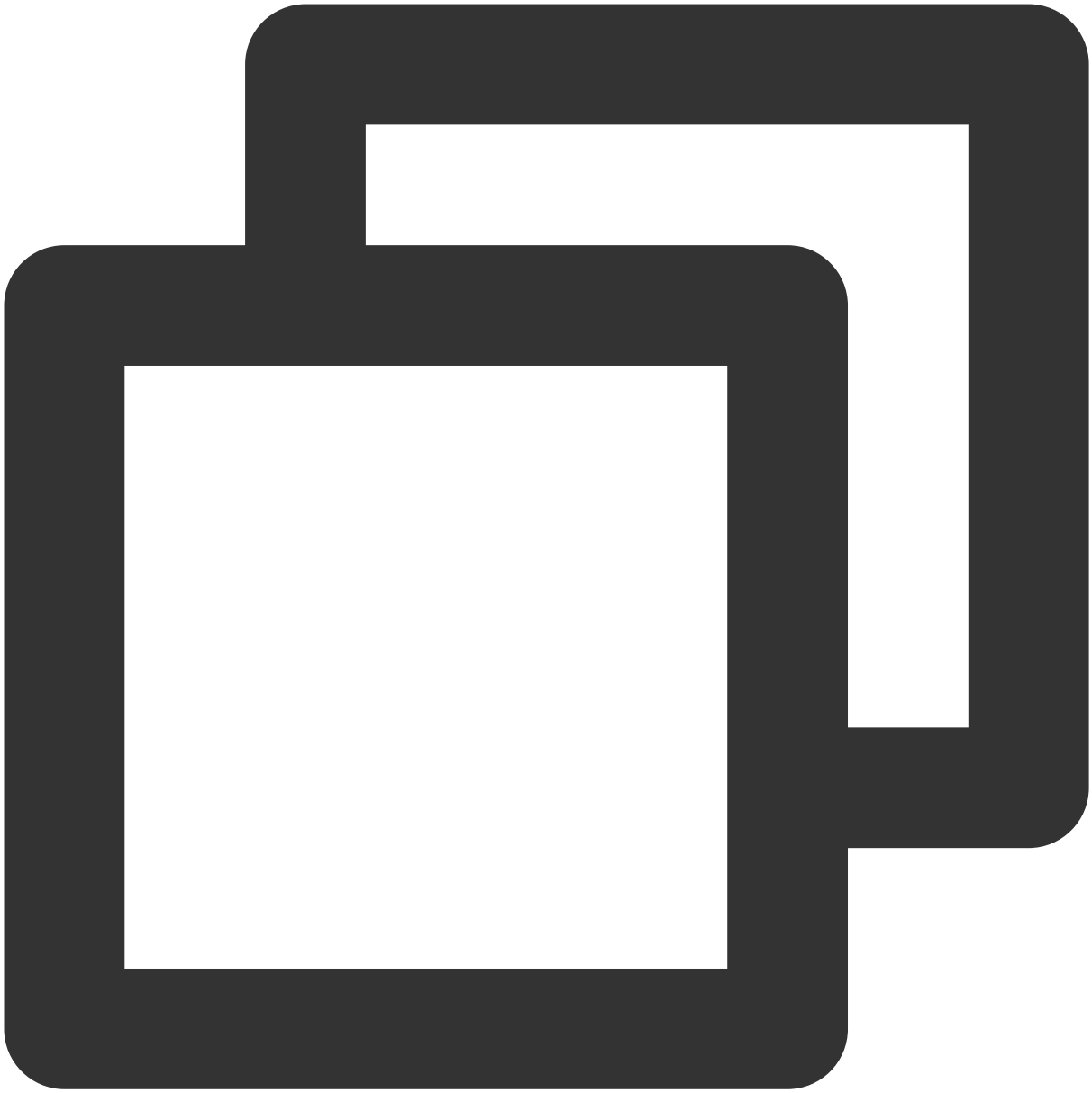
不配置 TIMPushAppGroupID 不会影响正常推送功能。

不启动 iOS 的触达上报时，TIMPushAppGroupID 默认设置为 0 即可；

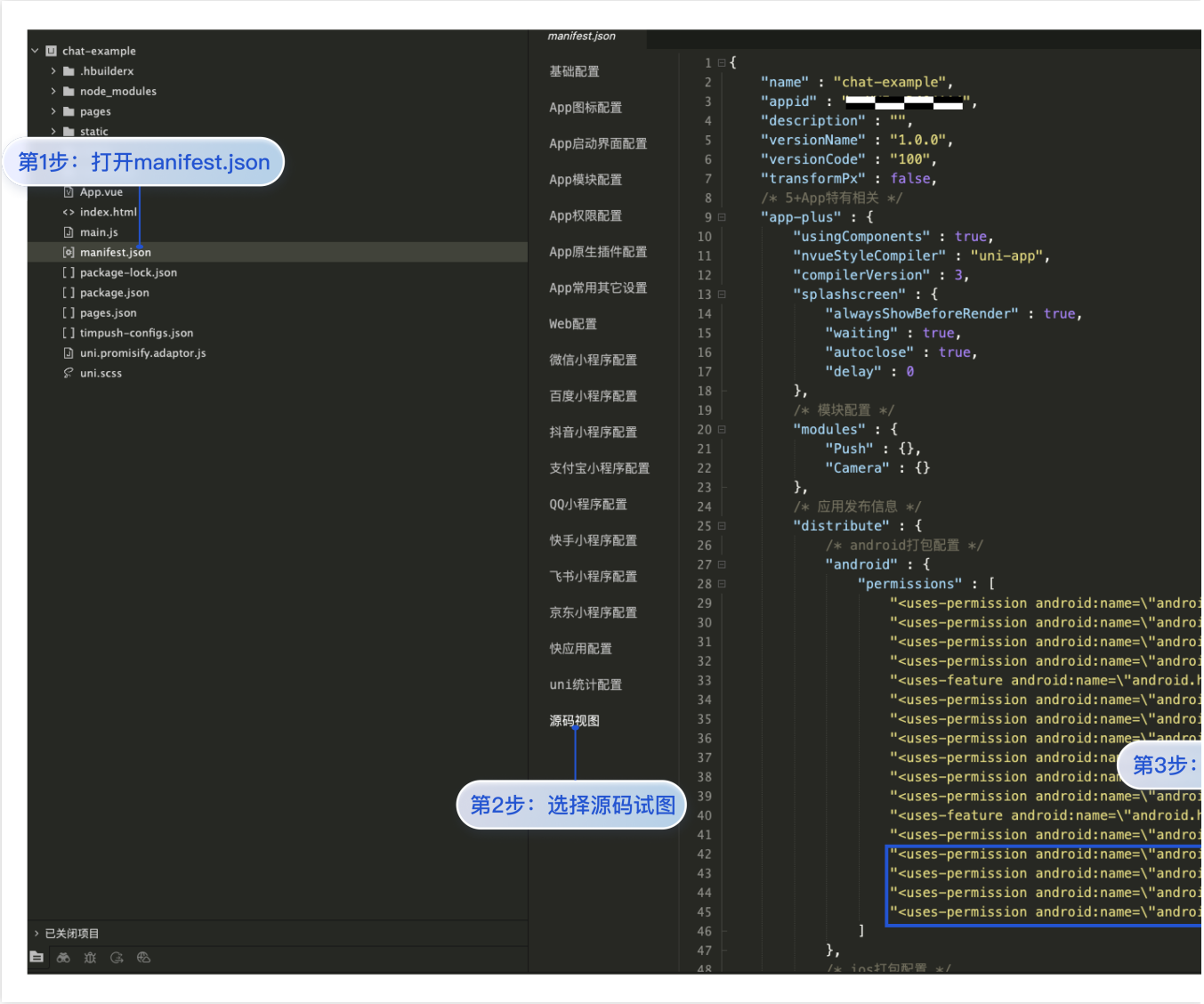
iOS appGroupID 生成指引	manifest.json iOS 配置
可参考 configuring-app-groups 生成 appGroupID	

步骤3: manifest.json Android 权限配置

在项目的 **manifest.json > 源码视图 > app-plus > distribute > android > permissions** 中追加以下权限项，如图所示：



```
"<uses-permission android:name=\\"android.permission.INTERNET\\" />",
"<uses-permission android:name=\\"android.permission.ACCESS_NETWORK_STATE\\" />",
"<uses-permission android:name=\\"android.permission.ACCESS_WIFI_STATE\\" />",
"<uses-permission android:name=\\"android.permission.WRITE_EXTERNAL_STORAGE\\" />"
```



步骤4. 注册 TencentCloud-TIMPush

注意：

@tencentcloud/chat ≥ 3.2.5 支持 TencentCloud-TIMPush。

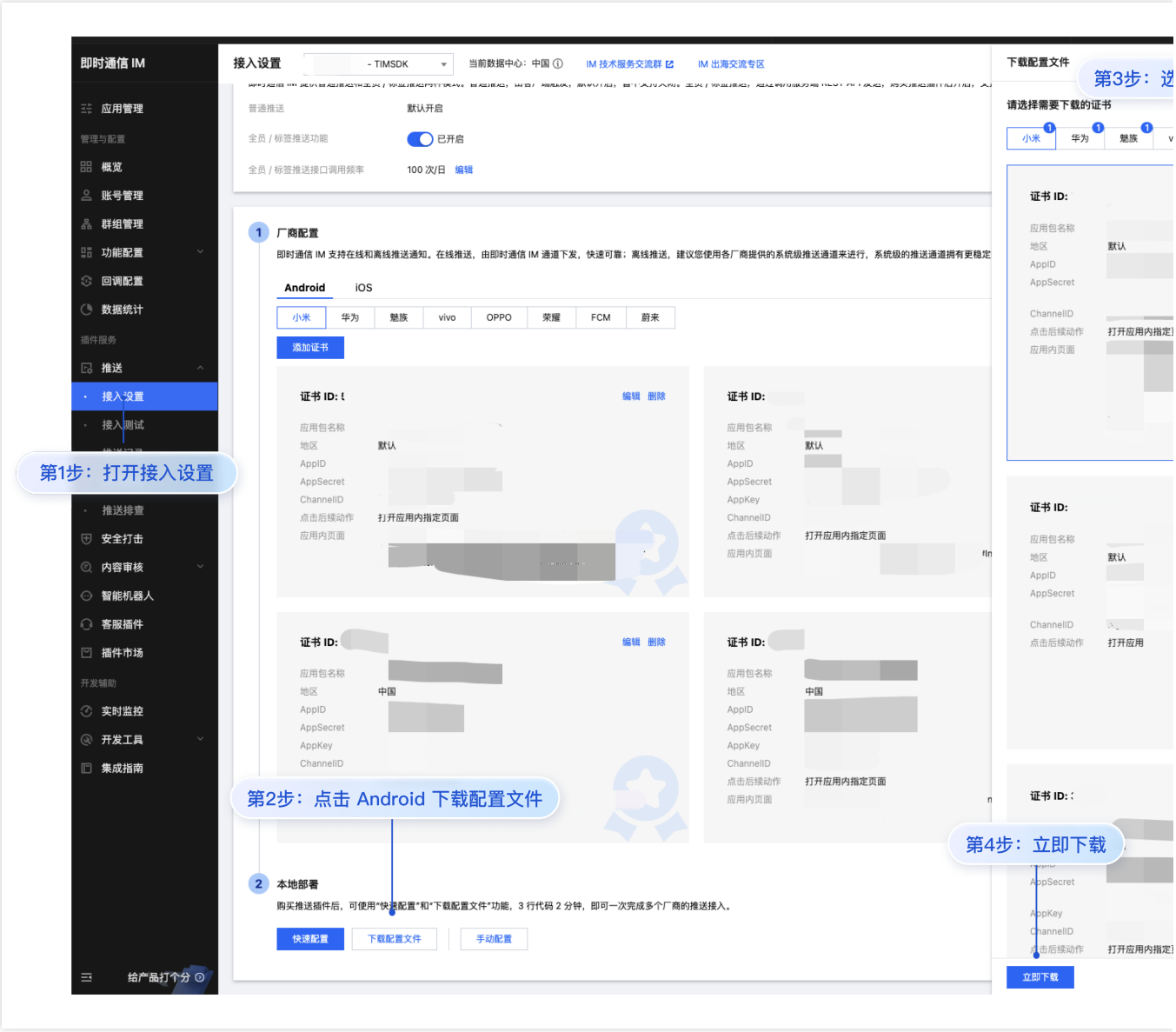
androidConfig 是 Android 推送配置，如不需要打包 Android App，可传空。

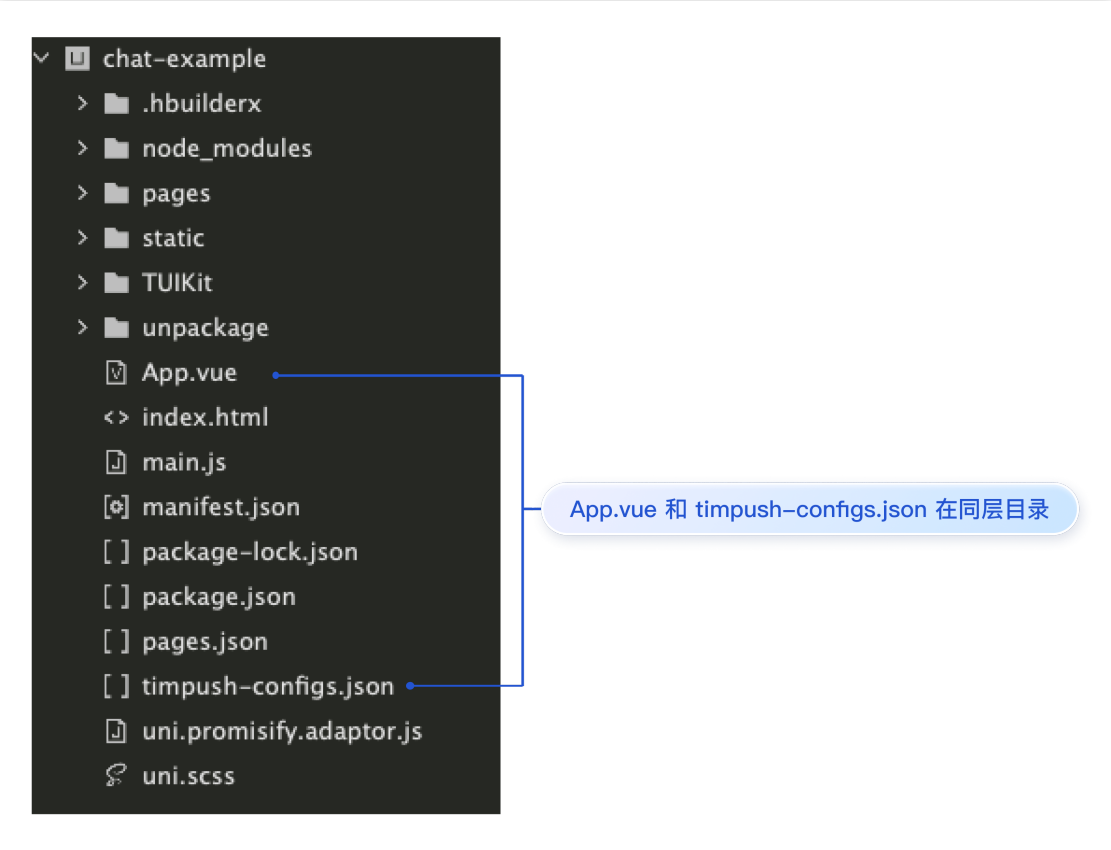
iOSConfig 是 iOS 推送配置，如不需要打包 iOS App， 可传空。

获取 Android 配置 timpush-configs.json

获取 iOS 配置 iOSBusinessID

timpush-configs.json 可以在 IM控制台 > 推送 > 接入设置 进行下载, 并将 timpush-configs.json 放入 `App.vue` 同级目录中， 如图所示：


第1步：打开接入设置
第2步：点击 Android 下载配置文件
第3步：选
第4步：立即下载
即时通信 IM
接入设置
应用管理
概览
账号管理
群组管理
功能配置
回调配置
数据统计
推送
接入设置
接入测试
推送排查
安全打击
内容审核
智能机器人
客服插件
插件市场
实时监控
开发工具
集成指南
给产品打个分
普通推送
默认开启
全员 / 标签推送功能
已开启
全员 / 标签推送接口调用频率
100 次/日
编辑
1 厂商配置
Android
iOS
小米
华为
魅族
vivo
OPPO
荣耀
FCM
蔚来
添加证书
证书 ID:
编辑
删除
应用包名称
地区
默认
AppID
AppSecret
AppKey
ChannelID
点击后续动作
打开应用内指定页面
应用内页面
中国
2 本地部署
购买推送插件后，可使用"快速配置"和"下载配置文件"功能，3 行代码 2 分钟，即可一次完成多个厂商的推送接入。
快速配置
下载配置文件
手动配置
下载配置文件
请选择需要下载的证书
立即下载



iOS iOSBusinessID 可以在 IM 控制台 > 推送 > 接入设置获取，如图所示：


即时通信 IM
应用管理
管理与配置
概览
账号管理
群组管理
功能配置
回调配置
数据统计
插件服务
推送
接入设置
接入测试
推送数据
推送排查
安全打击
内容审核
智能机器人
客服插件
插件市场
开发辅助
接入设置
TIMSDK
当前数据中心：
购买推送插件后，可使用快速配置、全员 / 标签推送、推送排查功
推送模式设置
即时通信 IM 提供普通推送和全员 / 标签推送两种模式。普通推送，由客
普通推送
默认开启
全员 / 标签推送功能
已开启
全员 / 标签推送接口调用频率
100 次/日
编辑
第2步：切换到 iOS 厂商配置
1
厂商配置
即时通信 IM 支持在线和离线推送通知。在线推送，由即时通信 IM
Android
iOS
添加证书
证书 ID:
证书信息
证书类型
开发环境
mutable-content
未开启
证书密码
到期时间
2022-08-30 02:39:35
第1步：打开接入设置
第3步：获取证书ID（iOSBusinessID就是证书ID）

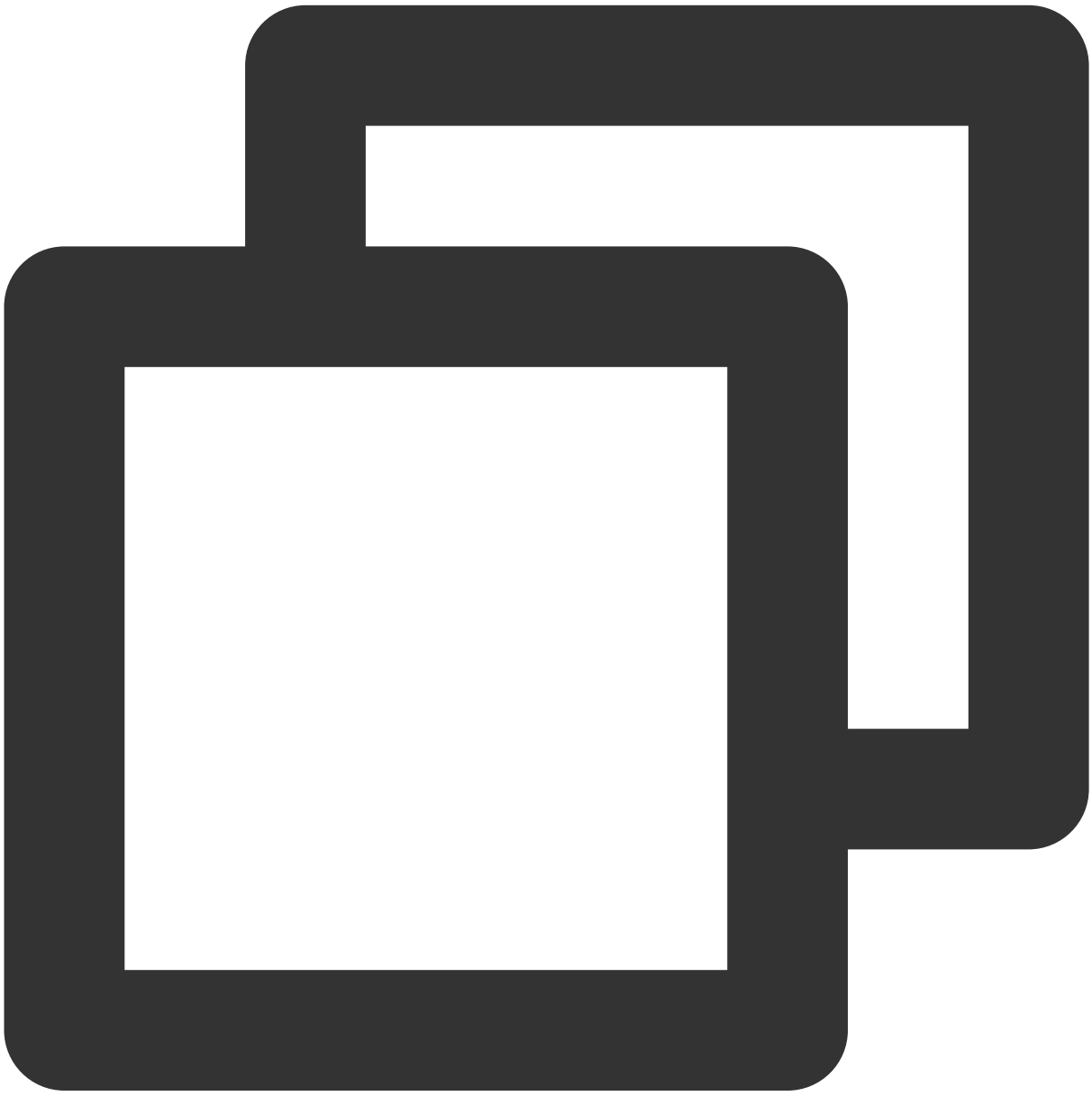
App.vue

含 UI 集成

无 UI 集成

在 App.vue 中引入 TencentCloud-TIMPush，并挂载到 uni 上。

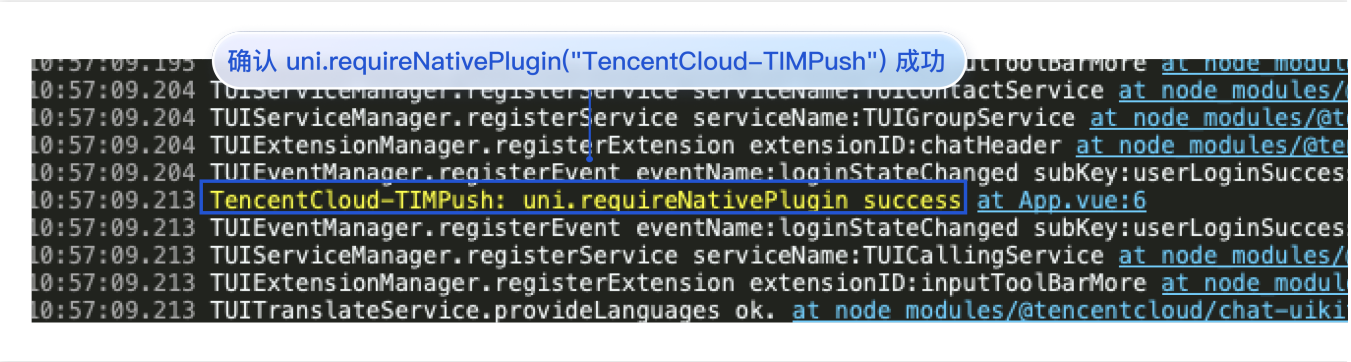
在 App.vue 中引入 timpush-configs.json，并挂载到 uni 上。



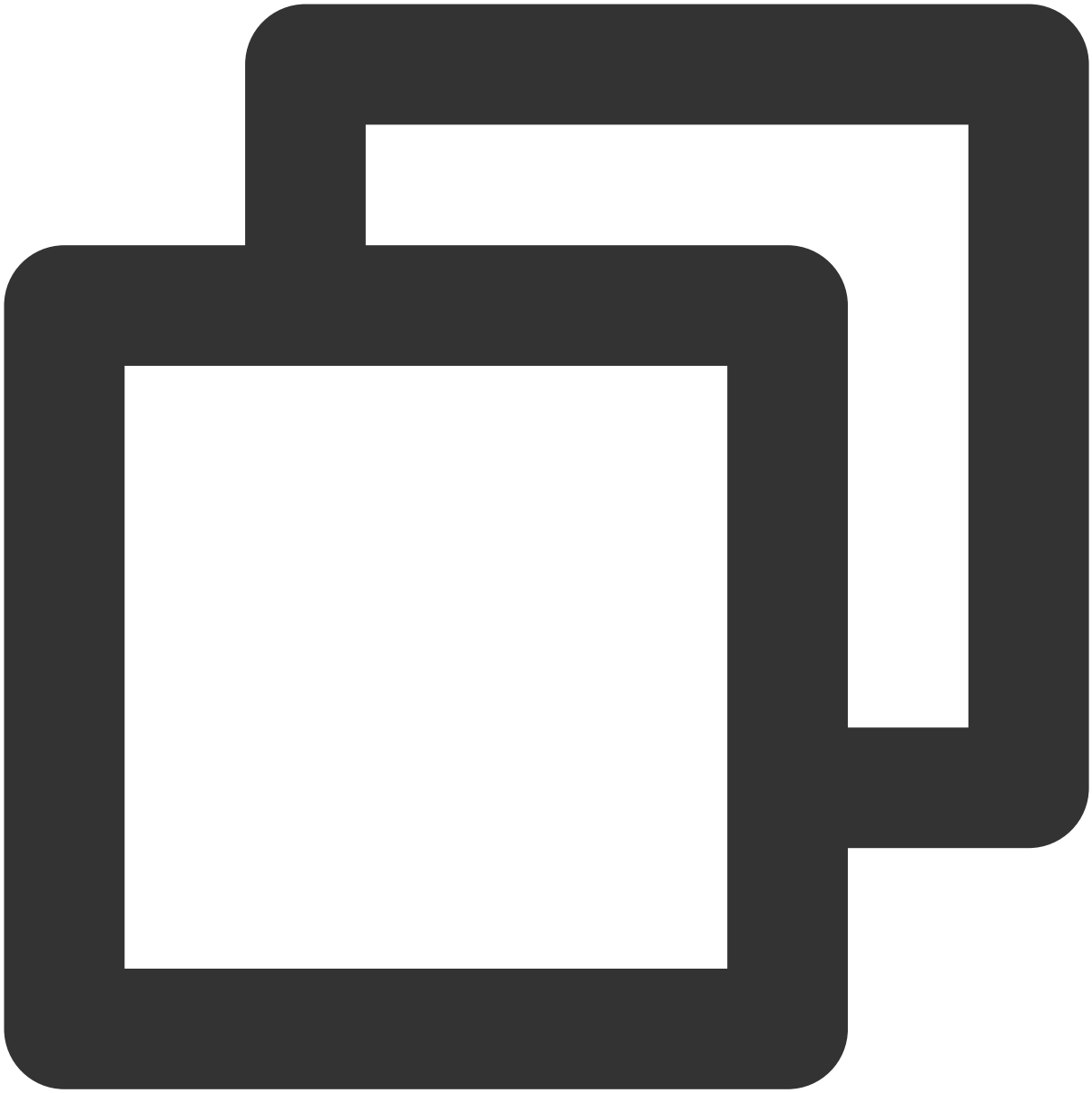
```
// #ifdef APP-PLUS
import TIMPushConfigs from "./timpush-configs.json";
const TIMPush = uni.requireNativePlugin("TencentCloud-TIMPush");
console.warn(`TencentCloud-TIMPush: uni.requireNativePlugin ${!!TIMPush ? 'success'
```

```
uni.$TIMPush = TIMPush;
uni.$TIMPushConfigs = TIMPushConfigs || {};
// #endif
```

确认 uni.requireNativePlugin 引入 TencentCloud-TIMPush 成功，如图所示：



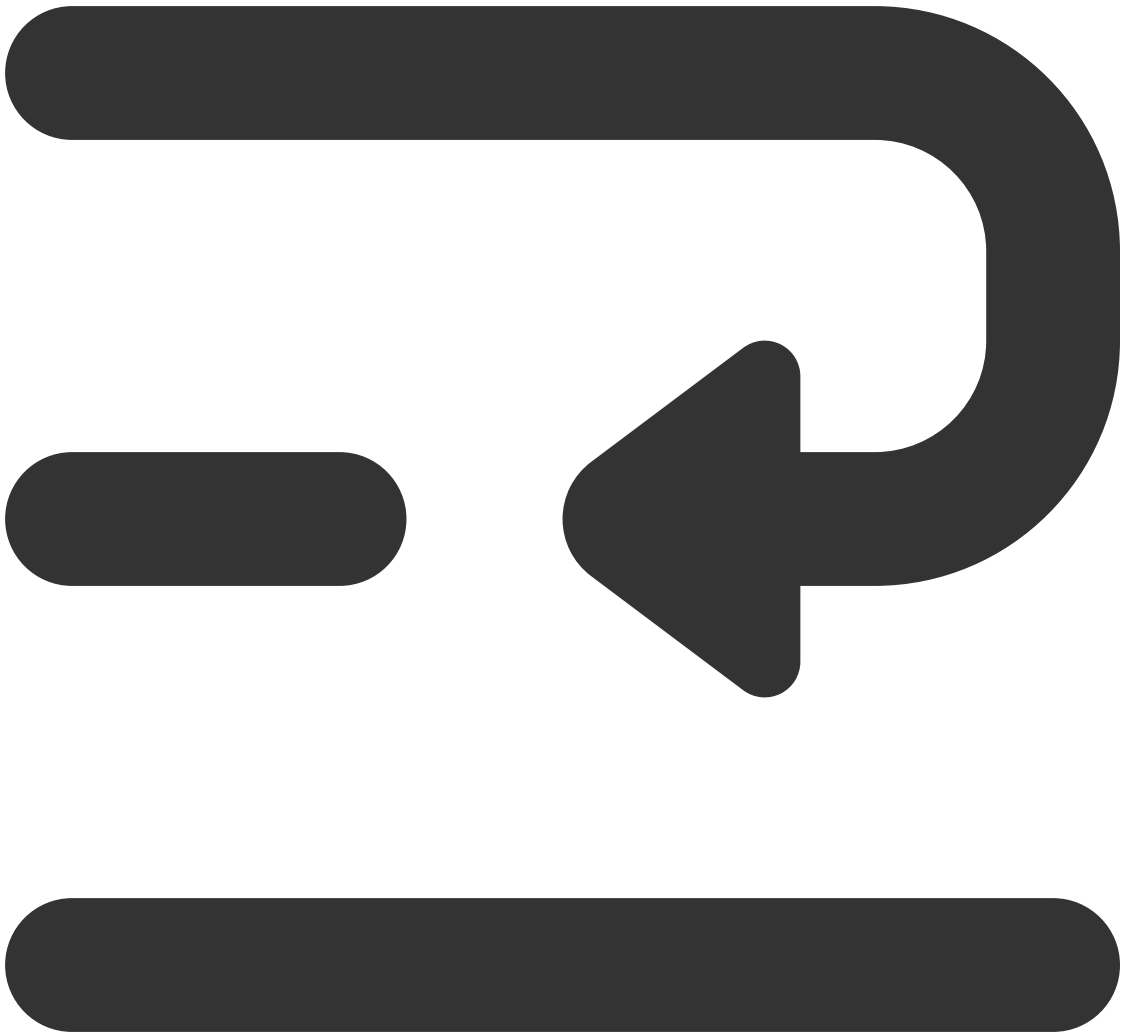
在 uikit 登录时，将 TencentCloud-TIMPush 注册进 uikit 中.

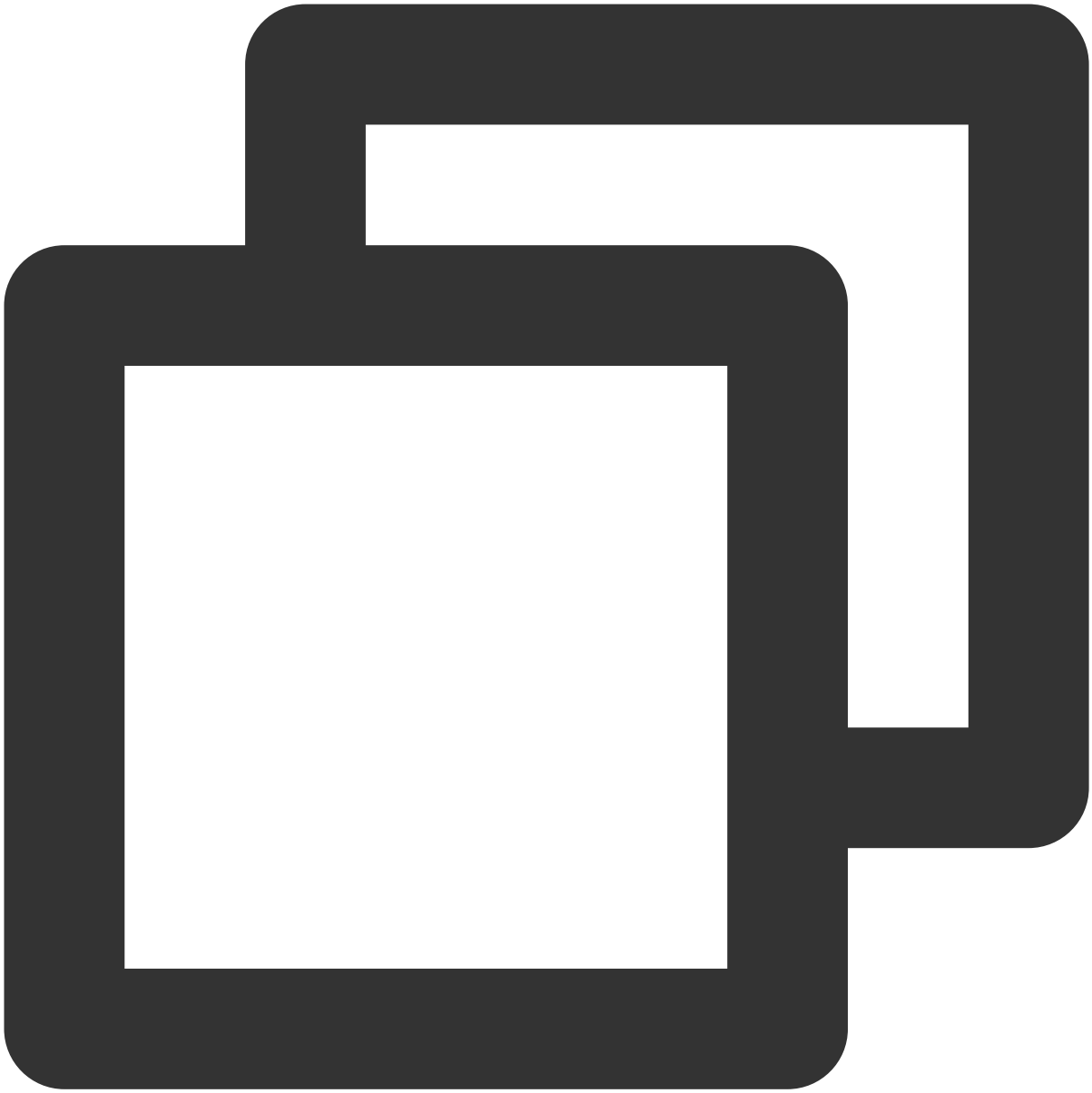


```
import { TUILogin } from "@tencentcloud/tui-core";
TUILogin.login({
    SDKAppID: 0, // 接入时需要将0替换为您的即时通信应用的 SDKAppID
    userID: '',
    // UserSig 是用户登录即时通信 IM 的密码，其本质是对 UserID 等信息加密后得到的密文。
    // 该方法仅适合本地跑通 Demo 和功能调试，详情请参见 https://cloud.tencent.com/document
    userSig: '',
    // 如果您需要发送图片、语音、视频、文件等富媒体消息，请设置为 true
    useUploadPlugin: true,
    framework: `vue${vueVersion}` // 当前开发使用框架 vue2 / vue3
```

```
    TIMPush: uni.$TIMPush, // APP 注册推送插件
    pushConfig: {
        androidConfig: uni.$TIMPushConfigs,    // Android 推送配置，如不需要可传空。
        iOSConfig: {
          "iOSBusinessID": "" // iOS 推送配置，如不需要可传空。
        }
    }
})
```

在 chat 登录前，需要将 TencentCloud-TIMPush 注册进 chat 中.





```
import TencentCloudChat from '@tencentcloud/chat';
const chat = TencentCloudChat.create({SDKAppID: 0}) // 接入时需要将0替换为您的即时通信应
chat.registerPlugin({
  'tim-push': uni.$TIMPush,
  pushConfig: {
    androidConfig: uni.$TIMPushConfigs,    // Android 推送配置，如不需要可传空。
    iOSConfig: {
      "iOSBusinessID": "xxx" // iOS 推送配置，如不需要可传空。
     }
  }
})
```

```
chat.login({
    userID: '', // 用户 ID。
    userSig: '' // 用户登录即时通信 IM 的密码，其本质是对 UserID 等信息加密后得到的密文。具体
})
```

步骤5. 生成自定义基座

单击 HBuilderX 的 **运行 > 运行到手机或模拟器 > 制作自定义调试基座**制作自定义基座。

注意：

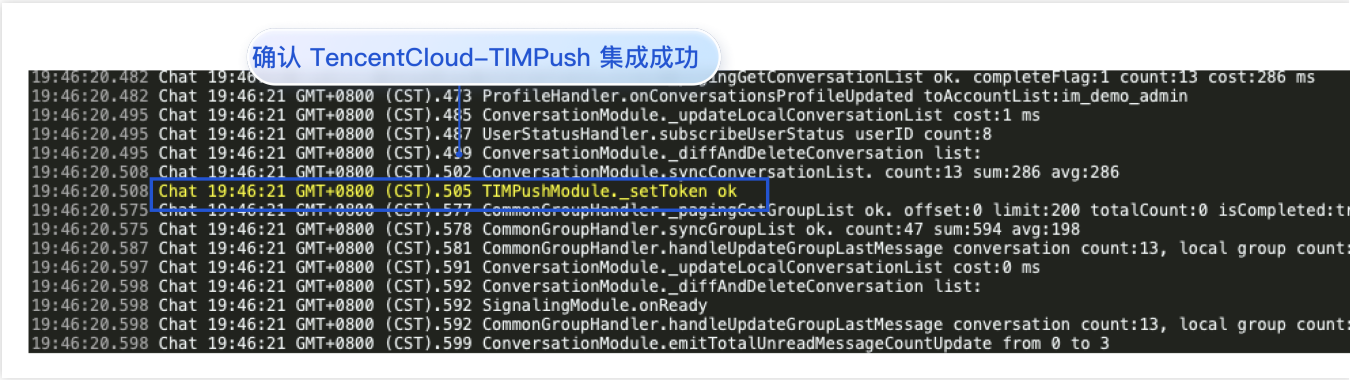
配置原生插件，必须打包自定义基座进行测试。

制作基座时，Android 包名为插件开通云打包时绑定的应用包名。

证书使用云端证书。

步骤6. 运行并登录项目，确认 TencentCloud-TIMPush 集成成功。

运行时，选择自定义基座运行，在 HBuilder 的日志中，确认有 `TIMPushModule._setToken ok` 打印，表示 TencentCloud-TIMPush 集成成功，如图所示：



步骤7. 接收您的第一条推送。

进入 IM 控制台推送 > 推送测试发送您的第一条推送。如图所示：

注意：

1. 接收端的应用必须置于后台，或者结束进程的状态。
2. 直播群不支持离线消息推送。

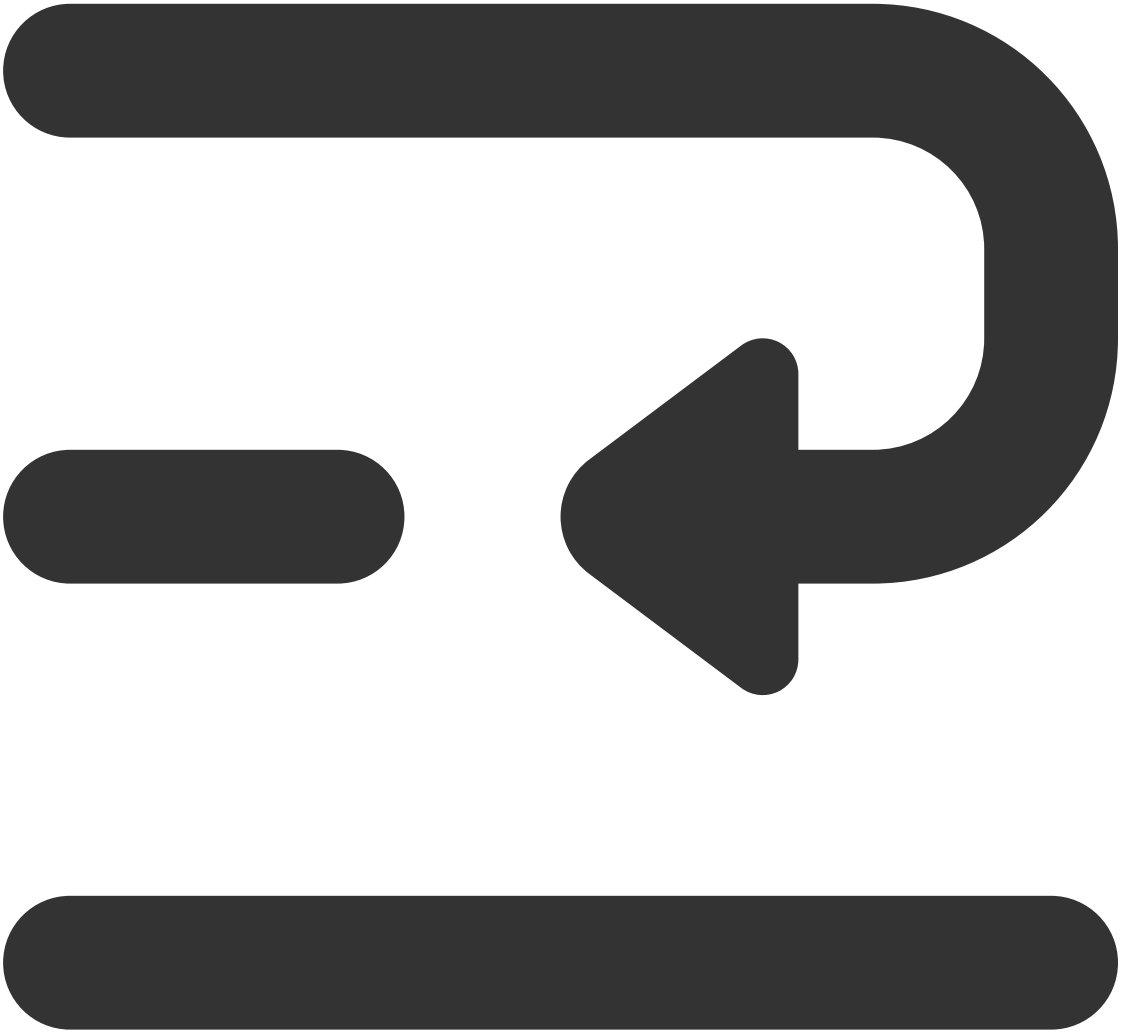
更多高级特性（强烈推荐）

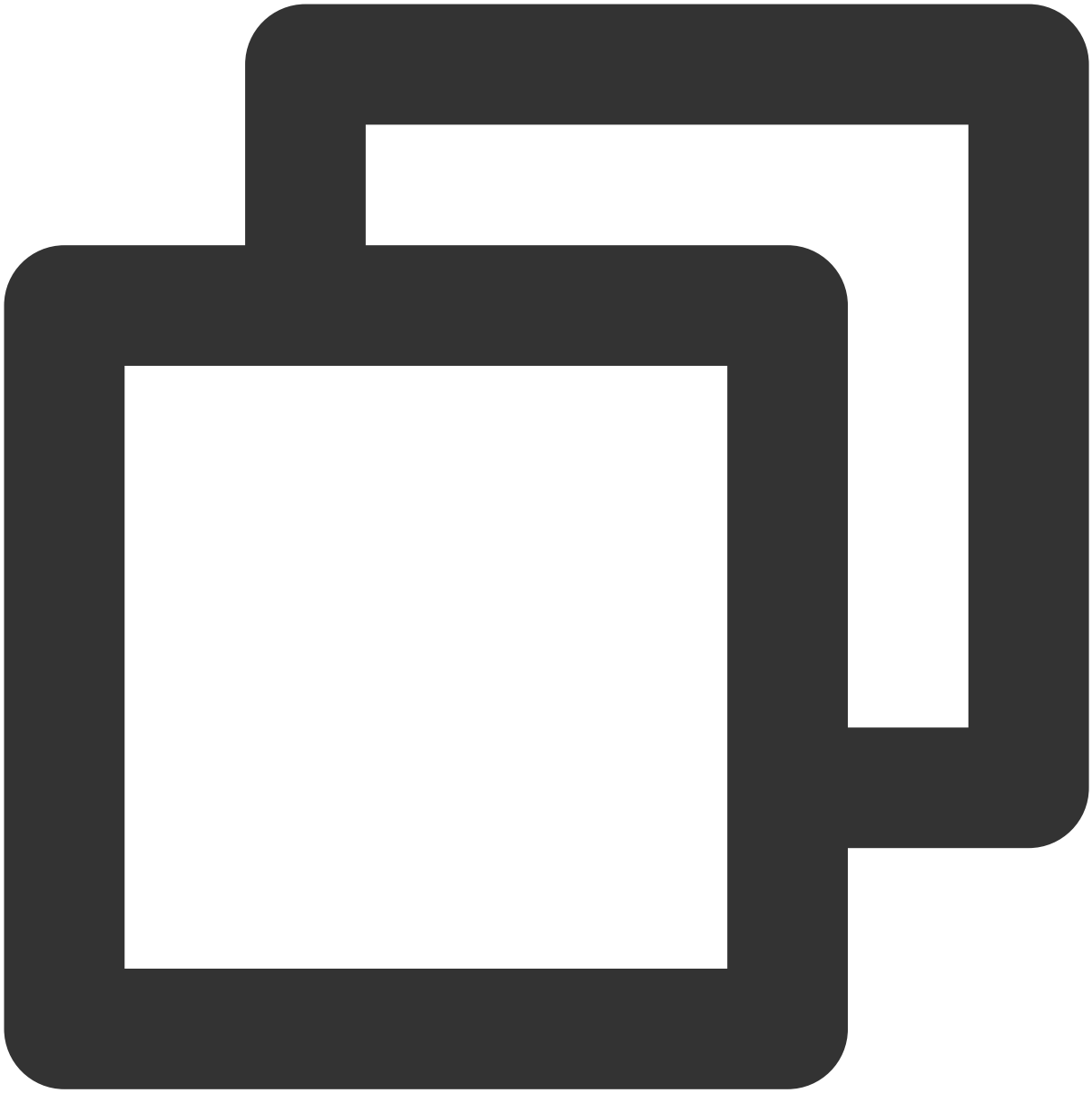
1. 设置推送内容

含 UI 集成

无 UI 集成

在 uikit 中使用 TUIChatService 发送消息时，设置 offlinePushInfo 相关参数，如发送普通文本消息，代码如下：



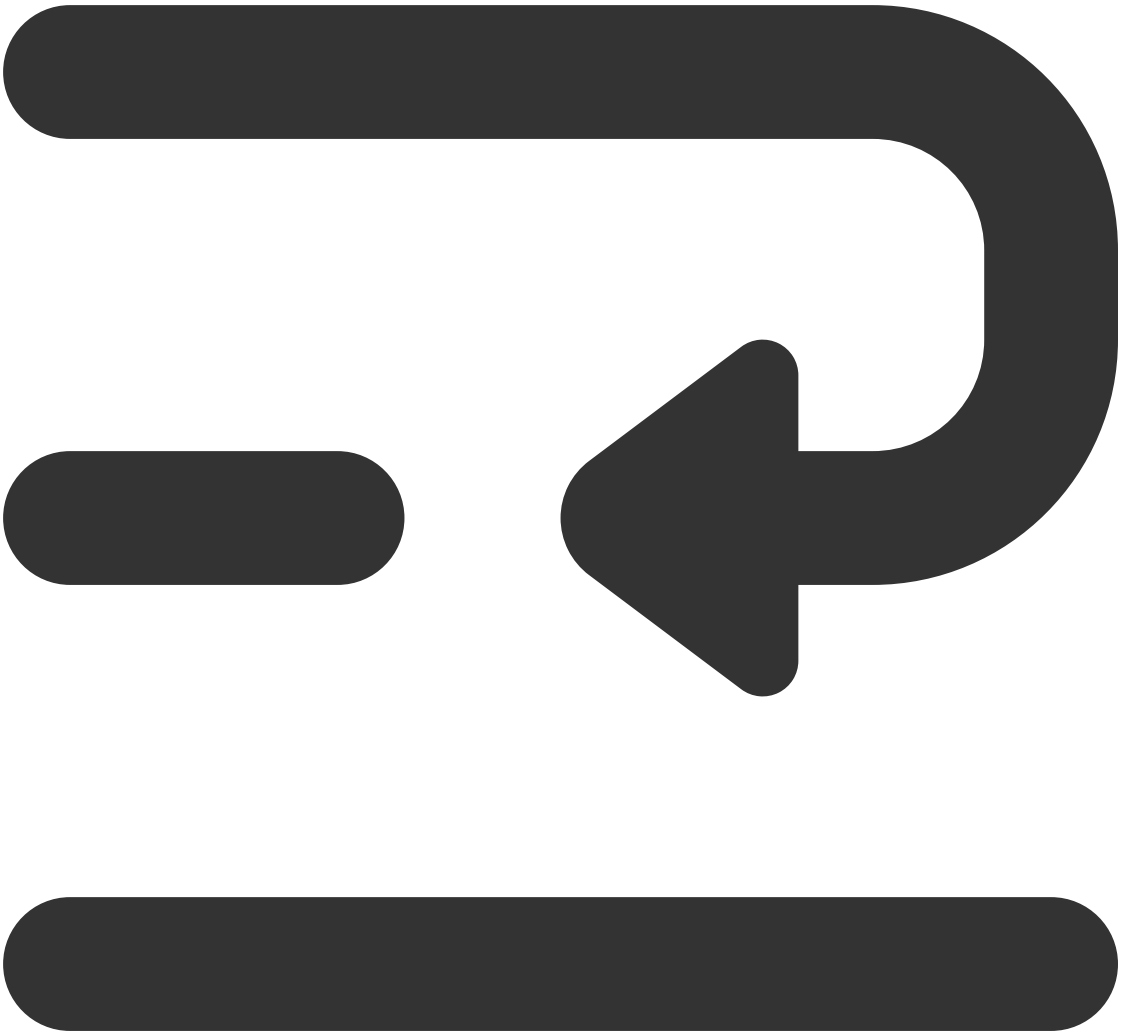


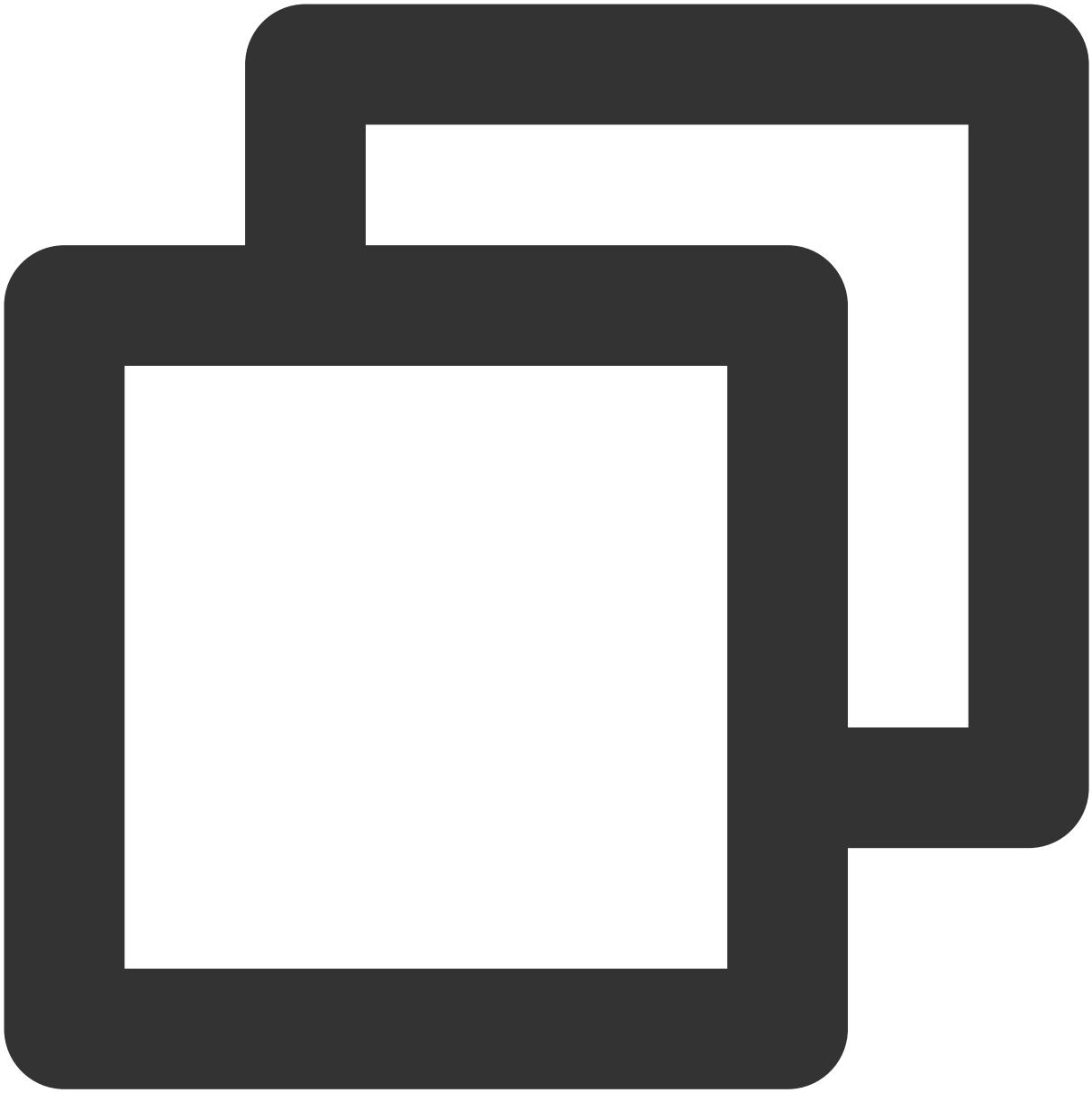
```
// 发送普通文本消息
let promise = TUIChatService.sendTextMessage(
{
   payload: {   text: 'Hello world!' }
},
{
  // 如果接收方不在线，则消息将存入漫游，且进行离线推送（在接收方 App 退后台或者进程被 kill 的情
   offlinePushInfo: {
     title: '', // 离线推送标题。
     description: '', // 离线推送内容。
     extension: '', // 离线推送透传内容
```

```
    }
}
);
promise.catch((error) => {
    // 调用异常时业务侧可以通过 promise.catch 捕获异常进行错误处理
});
```

参考文档：uikit - TUIChatService 发送消息相关文档

在 chat 发送消息时，设置 offlinePushInfo 相关字段，代码如下:





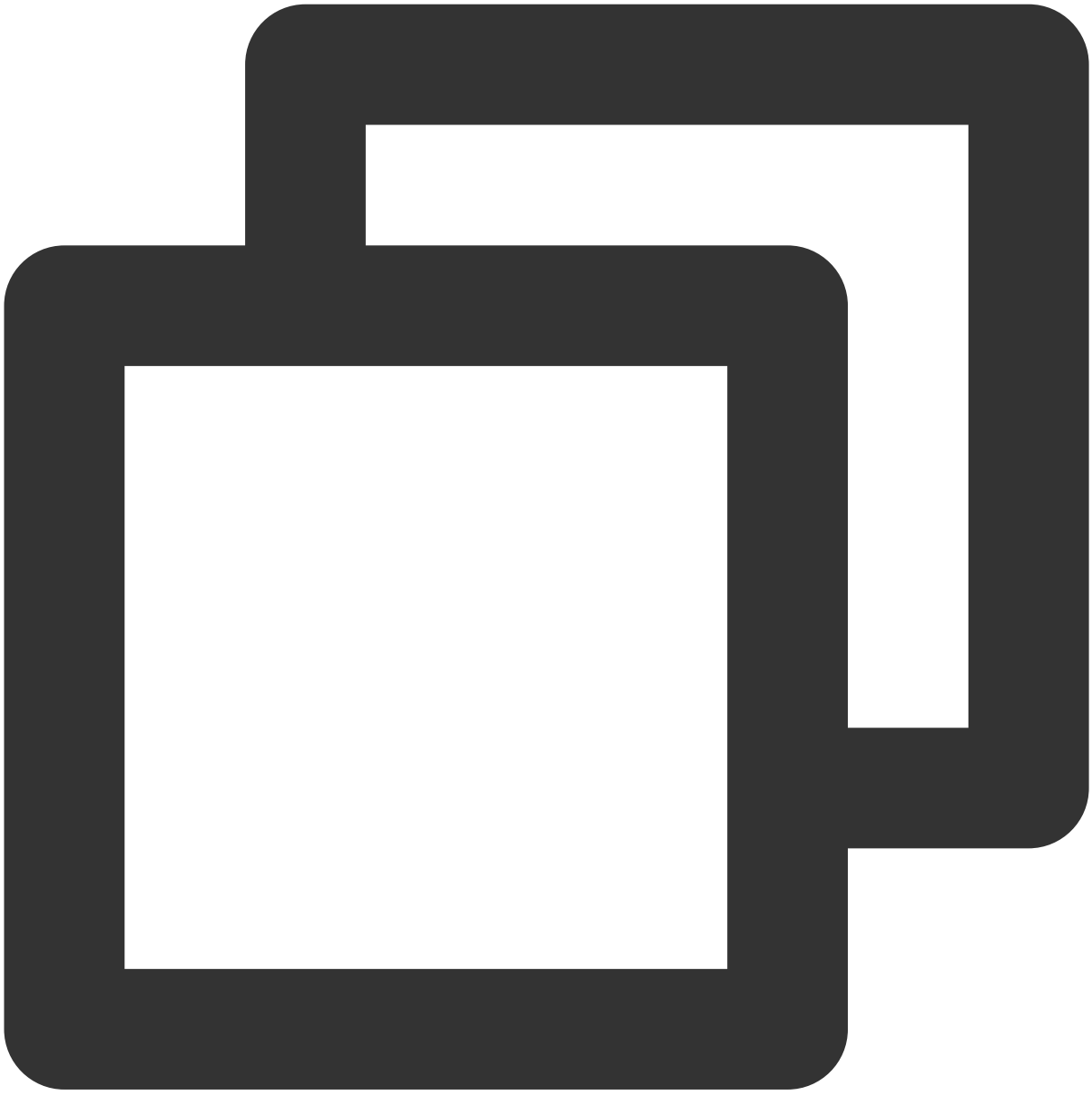
```
// 消息发送选项
chat.sendMessage(message, {
  // 如果接收方不在线，则消息将存入漫游，且进行离线推送（在接收方 App 退后台或者进程被 kill 的情
  offlinePushInfo: {
     title: '', // 离线推送标题。
     description: '', // 离线推送内容。
     extension: '', // 离线推送透传内容
  }
});
```

参考文档：

Chat SDK - sendMessage 文档

2. 获取点击透传的内容

在 `App.vue` 文件中获取透传内容，配置指定跳转页面。



```
onLaunch: function () {
    // #ifdef APP-PLUS
    // 在 App.vue, 生命钩子 onLaunch 中监听
    if (uni.$TIMPush) {
     uni.$TIMPush.setOfflinePushListener((data) => {
         // 透传 entity 中的内容，不包含推送的 Message
```

```
            console.log('setOfflinePushListener', data);
            // 跳转到应用内指定界面
            uni.navigateTo({
                url: "/pages/xxx/xxx",
            });
        });
    }
    // #endif
}
```

注意：

可在此获取推送时配置的透传内容。

可在此配置应用跳转界面。

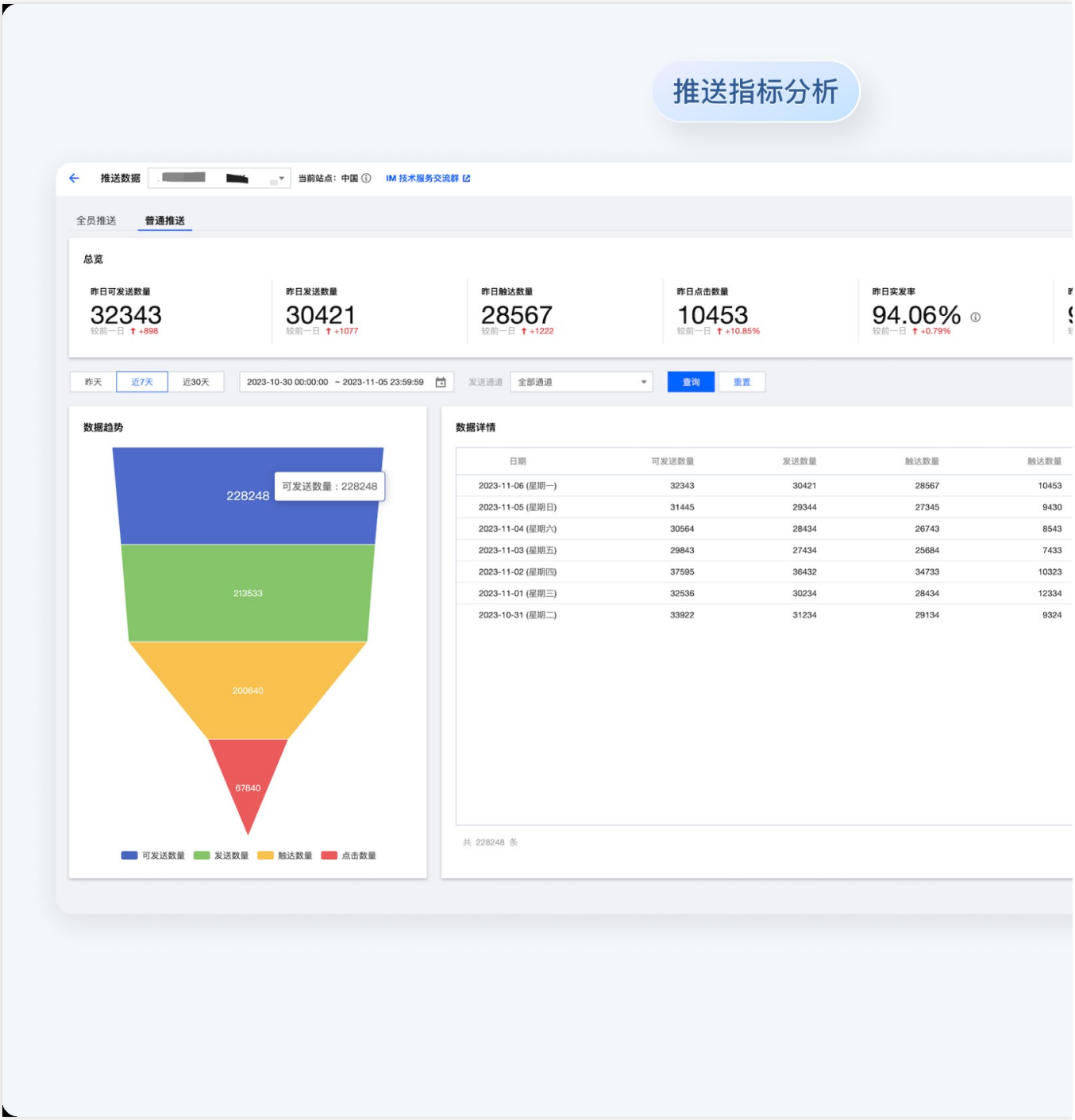
各端互通时，要确保 `extension` 保持一致， `extension` 中需要包含 `entity` 字段。

异常排查

如果用户在 App 离线时，未收到离线推送消息，可以通过排查工具来全链路排查推送详情。

数据统计

查询用户所有推送数据，整理和分析用户每日推送各类型指标数据，生成近日**发送 > 触达 > 点击**漏斗转化图表，支持按照厂商通道分类查看。



设备通知栏设置

推送的直观表现就是通知栏提示，所以同其他通知一样受设备通知相关设置的影响，以华为为例：

“手机设置-通知-锁屏通知-隐藏或者不显示通知”，会影响锁屏状态下离线推送通知显示。

“手机设置-通知-更多通知设置-状态栏显示通知图标”，会影响状态栏下离线推送通知的图标显示。

“手机设置-通知-应用的通知管理-允许通知”，打开关闭会直接影响离线推送通知显示。

“手机设置-通知-应用的通知管理-通知铃声” 和 “手机设置-通知-应用的通知管理-静默通知”，会影响离线推送通知铃音的效果。

厂商推送限制

1. 国内厂商都有消息分类机制，不同类型也会有不同的推送策略。如果想要推送及时可靠，需要按照厂商规则设置自己应用的推送类型为高优先级的系统消息类型或者重要消息类型。反之，离线推送消息会受厂商推送消息分类影响，与预期会有差异。
2. 另外，一些厂商对于应用每天的推送数量也是有限制的，可以在厂商控制台查看应用每日限制的推送数量。 如果离线推送消息出现推送不及时或者偶尔收不到情况，需要考虑下这里：

华为	vivo	OPPO	小米	魅族	FCM
将推送消息分为服务与通讯	将推送消息分为系统消息类	将推送消息分为私信消息类	将推送消息分为重要消息类	推送消息数量有限制。 具体	推送上行消息频率有限制。

类和资讯营销类，推送效果和策略不同。另外，消息分类还和自分类权益有关：无自分类权益，推送消息厂商还会进行二次智能分类。有申请自分类权益，消息分类会按照自定义的分类进行推送。 具体请参见 厂商描述。	和运营消息类，推送效果和策略不同。系统消息类型还会进行厂商的智能分类二次修正，若智能分类识别出不是系统消息，会自动修正为运营消息，如果误判可邮件申请反馈。另外，消息推送也受日推总数量限制，日推送量由应用在厂商订阅数统计决定。 具体请参见 厂商描述1 或 厂商描述2。	和公信消息类，推送效果和策略不同。其中私信消息是针对用户有一定关注度，且希望能及时接收的信息，私信通道权益需要邮件申请。公信通道推送数量有限制。 具体请参见 厂商描述1 或 厂商描述2。	和普通消息类，推送效果和策略不同。其中重要消息类型仅允许即时通讯消息、个人关注动态提醒、个人事项提醒、个人订单状态变化、个人财务提醒、个人状态变化、个人资源变化、个人设备提醒这8类消息推送，可以在厂商控制台申请开通。普通消息类型推送数量有限制。 具体请参见 厂商描述1 或 厂商描述2。	请参见 厂商描述。	具体请参见 厂商描述。

华为：将推送消息分为服务与通讯类和资讯营销类，推送效果和策略不同。另外，消息分类还和自分类权益有关：无自分类权益，推送消息厂商还会进行二次智能分类。
有申请自分类权益，消息分类会按照自定义的分类进行推送。 具体请参见 厂商描述。
vivo：将推送消息分为系统消息类和运营消息类，推送效果和策略不同。系统消息类型还会进行厂商的智能分类二次修正，若智能分类识别出不是系统消息，会自动修正为运营消息，如果误判可邮件申请反馈。另外，消息推送也受日推总数量限制，日推送量由应用在厂商订阅数统计决定。 具体请参见 厂商描述1 或 厂商描述2。
OPPO：将推送消息分为私信消息类和公信消息类，推送效果和策略不同。其中私信消息是针对用户有一定关注度，且希望能及时接收的信息，私信通道权益需要邮件申请。公信通道推送数量有限制。 具体请参见 厂商描述1 或 厂商描述2。
小米：将推送消息分为重要消息类和普通消息类，推送效果和策略不同。其中重要消息类型仅允许即时通讯消息、个人关注动态提醒、个人事项提醒、个人订单状态变化、个人财务提醒、个人状态变化、个人资源变化、个人设备提醒这8类消息推送，可以在厂商控制台申请开通。普通消息类型推送数量有限制。 具体请参见 厂商描述1 或 厂商描述2。
魅族：推送消息数量有限制。 具体请参见 厂商描述。
FCM：推送上行消息频率有限制。 具体请参见 厂商描述。

常见问题

1. TencentCloud-TIMPush 和 uniPush2 冲突了，不能共用该如何处理？

原因：TencentCloud-TIMPush 不支持与其他离线推送通道共用。
解决方案：推荐仅使用 TencentCloud-TIMPush 即可。

2. 推送有报错 "TIMPushModule._getDeviceToken failed. error: { "code": -1, "msg": "huawei ApiException: com.huawei.hms.common.ApiException: 907135000: arguments invalid"}"。

原因：华为 agconnect-services.json 文件未引入。
解决方案：检查华为 agconnect-services.json 文件是否引入。
注意：
华为推送需要将官网下载的 agconnect-services.json 文件放到 `nativeResources/android/assets` 路径下。
详细可参见 uniapp 厂商配置 - Android - 华为。

3. 推送有报错 “TIMPushModule._getDeviceToken failed. error:{"code":-1, "msg": "callback is not String"}”。

原因：华为版本过低，EMUI 版本需要大于 10。
解决方案：升级或换EMUI 版本 > 10 的华为进行推送。

4. 推送有报错 "TIMPushModule._getDeviceToken failed. error:{"code":22022,"msg":"xiaomi error code: 22022"}"。

原因：应用程序 package name 不合法。
解决方案：前往小米推送平台检查应用的包名、appId、appKey 是否匹配 。具体可参见：厂商通道注册失败排查指南。

5. OPPO 手机收不到推送？

原因: OPPO 安装应用通知栏显示默认关闭。
解决方案：开启通知栏显示状态。

6. 按照流程接入完成，还是收不到推送？

原因：
设备状态异常、 IM 控制台配置未配置、初始化未注册等。
推送依赖厂商能力，一些简单的字符可能会被厂商过滤不能透传推送。
如果推送消息出现推送不及时或者偶尔收不到情况，需要看下厂商的推送限制。
解决方案:
可通过排查工具来进行排查。
检查发送内容，避免使用简单字符。
检查厂商推送限制，详情可见厂商推送限制。

7. iOS 普通消息为什么收不到离线推送?

原因：App 的运行环境和证书的环境是否不一致。
解决方案：
检查 App 的运行环境和证书的环境是否一致，如果不一致，收不到离线推送。
检查下 App 和证书的环境是否为生产环境。如果是开发环境，向苹果申请 deviceToken 可能会失败，生产环境暂时没有发现这个问题，请切换到生产环境测试。

8. iOS 开发环境下，注册偶现不返回 deviceToken 或提示 APNs 请求 token 失败？

原因：此问题现象是由于 APNs 服务不稳定导致的。
解决方案：
给手机插入 SIM 卡后使用4G网络测试。
卸载重装、重启 App、关机重启后测试。
打生产环境的包测试。
更换其它 iOS 系统的手机测试。

9. iOS 没有 token的原因？

原因：
模拟器不产生 token。
真机，需要在手机上开启推送的权限。
真机，需要添加 push notification 的 enetitemenet。
解决方案：使用真机开启推送权限，并添加 push notification 的 enetitemenet。

10. tim-js-sdk 可以使用 TIMPush 吗？

原因: 不支持 TencentCloud-TIMPush ；
解决方案：升级 @tencentcloud/chat 接入，@tencentcloud/chat 是向下兼容的，不会影响 tim-js-sdk 实现的业务。

技术咨询

点此进入IM社群，享有专业工程师的支持，解决您的难题。

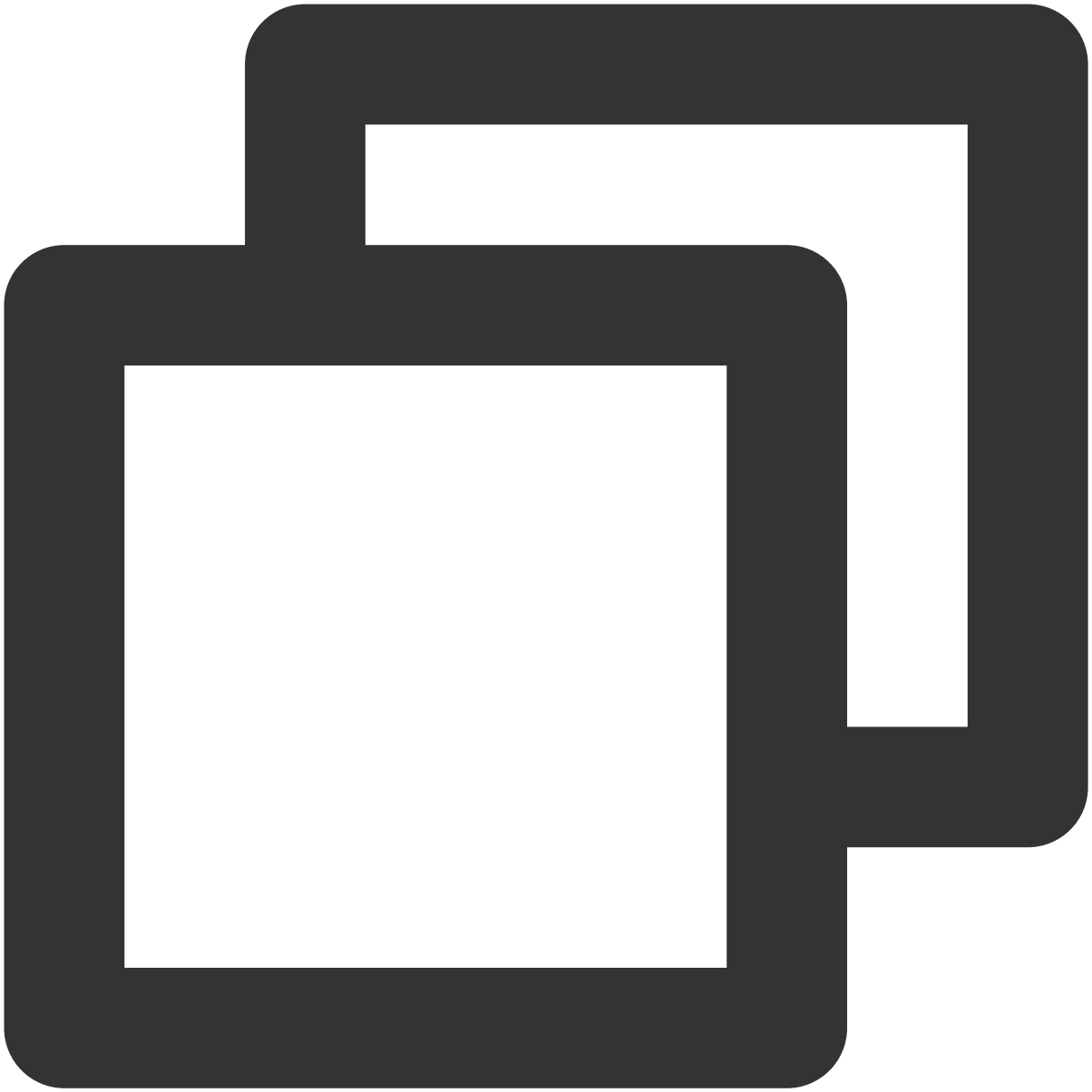
Flutter

最近更新时间：2024-06-13 10:39:26

操作步骤

步骤1: 集成消息推送插件

本插件在 pub.dev 的包名为: `tencent_cloud_chat_push` , 您可以收到将其引入 pubspec.yaml 依赖目录中, 也可以执行下列命令, 自动安装。



```
flutter pub add tencent_cloud_chat_push
```

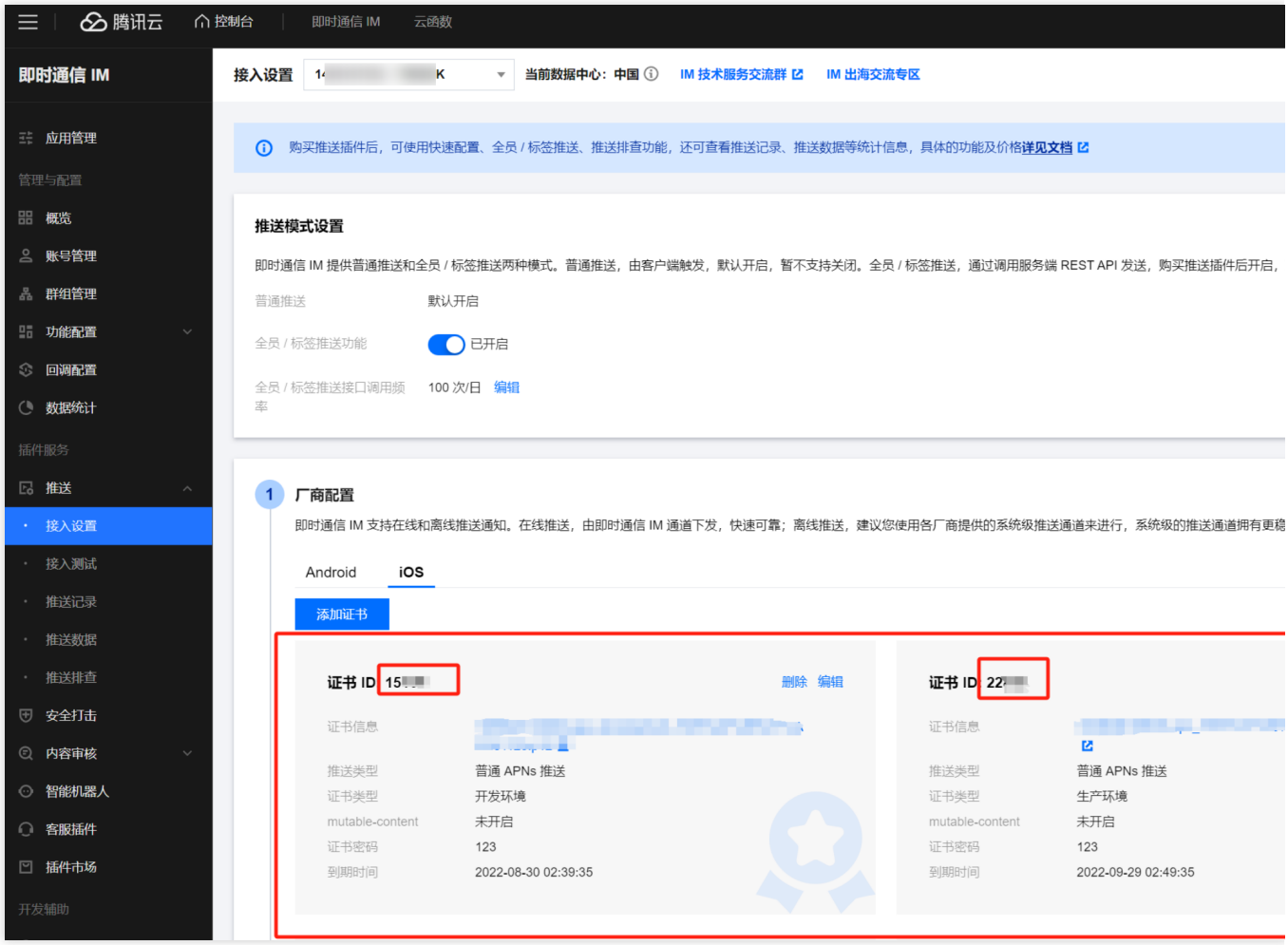
步骤2: 推送参数配置

iOS

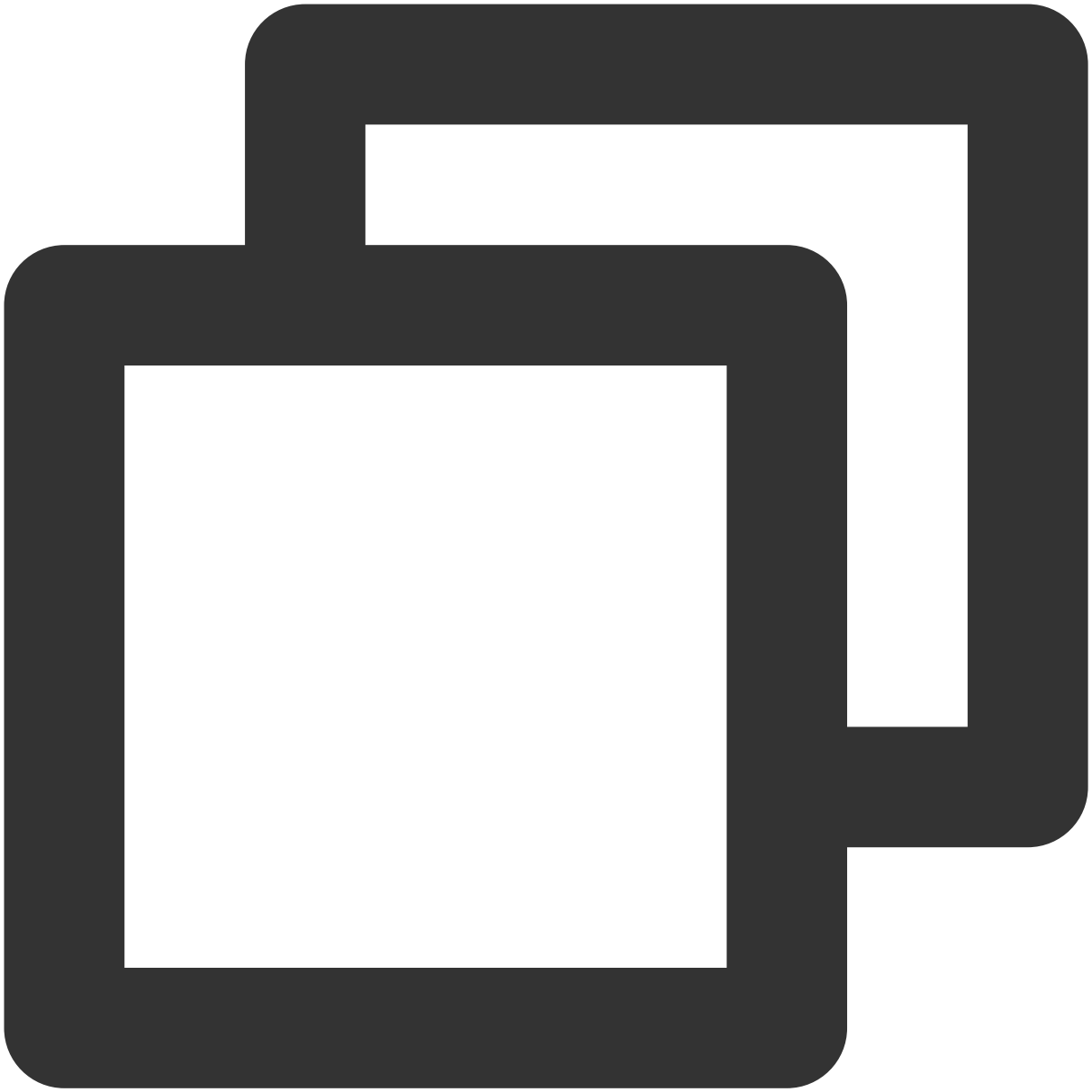
Android

请将您在厂商配置步骤中, 获取到的 iOS APNs 推送证书, 上传至 IM 控制台。

IM 控制台会为您分配一个证书 ID，见下图：



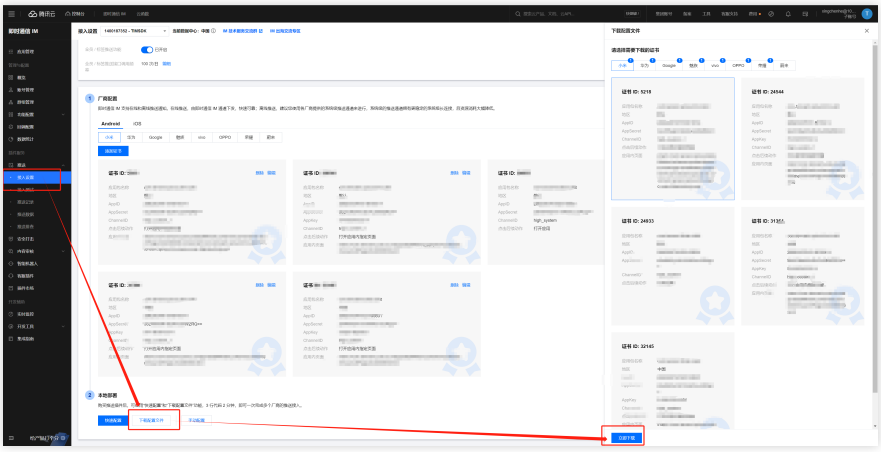
在您应用的启动后尽可能早的位置，调用 `TencentCloudChatPush().setApnsCertificateID` 方法，将此证书 ID 传入。

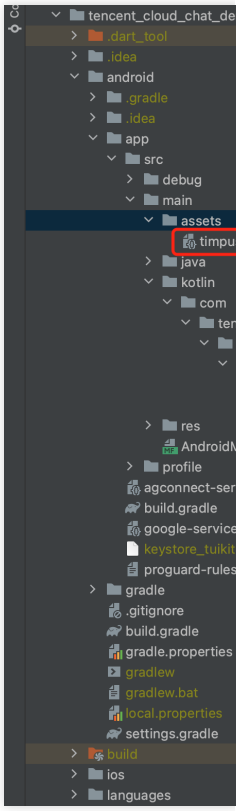


```
TencentCloudChatPush().setApnsCertificateID(apnsCertificateID: 证书ID);
```

完成控制台厂商推送信息填写后，下载并添加配置文件到工程。将下载的 timpush-configs.json 文件添加到项目的 `android/app/src/main/assets` 目录下, 如果该目录不存在, 请手动创建.

1.选择下载配置文件 timpush-configs.json	2.添加到工程





步骤3: 客户端代码配置

本步骤，需编写若干原生代码，例如：Swift, Java, XML 等。

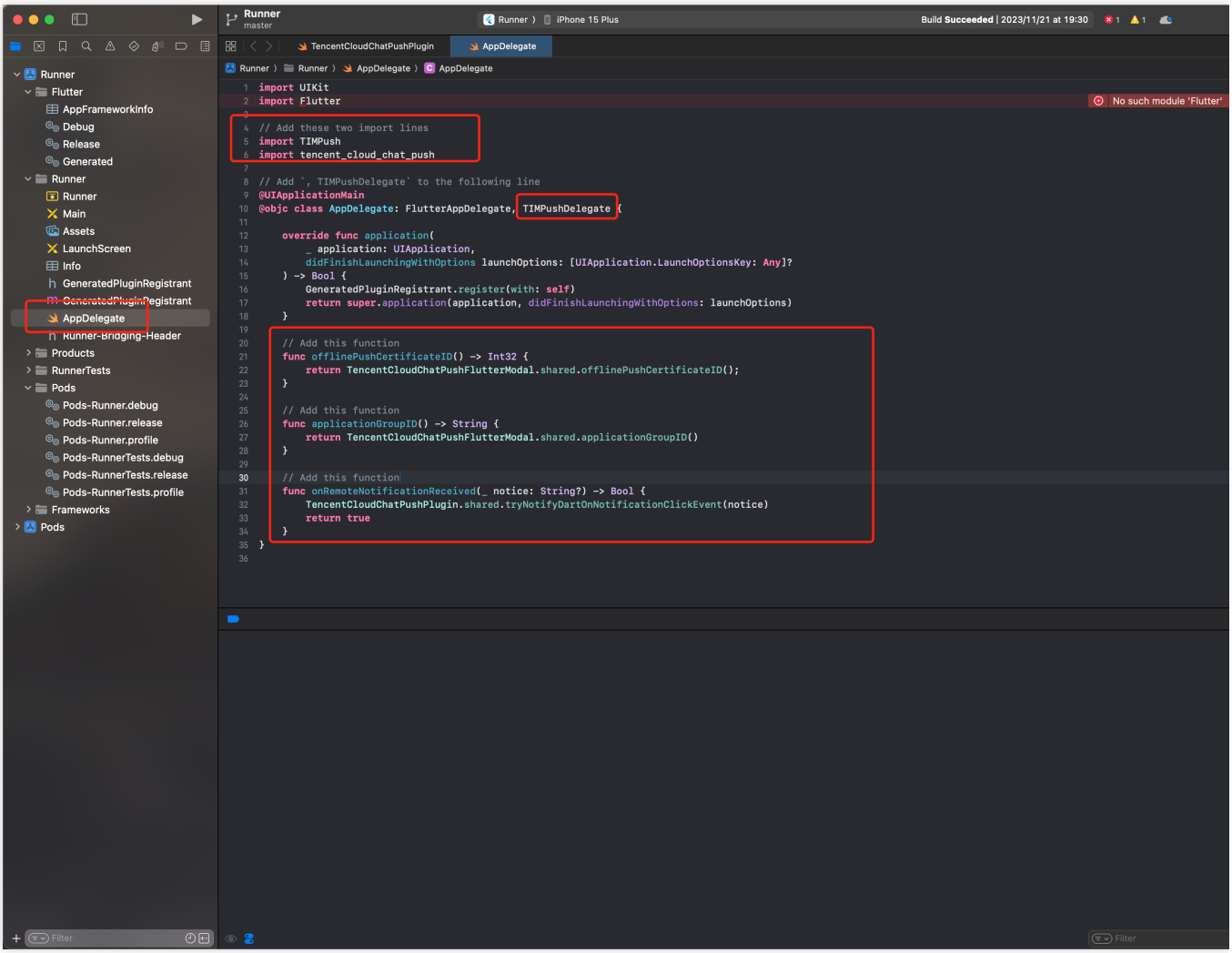
请不要担心, 直接根据说明, 复制我们提供的代码到指定文件即可。

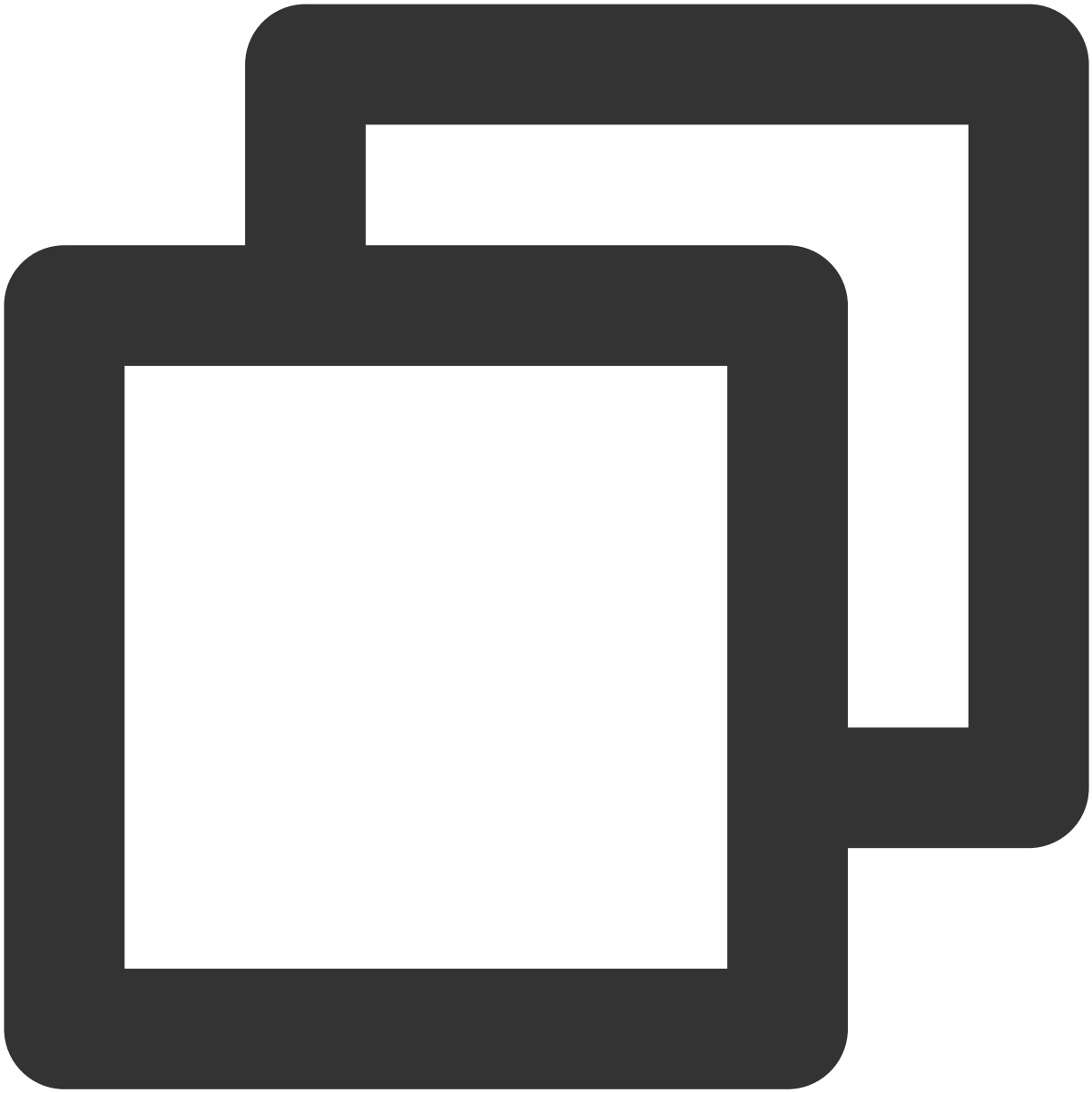
iOS

Android

您可使用 Xcode 编辑，也可直接在 Visual Studio Code 或 Android Studio 中编辑。

打开 `ios/Runner/AppDelegate.swift` 文件，将下列圈出的代码粘贴进入，效果如图所示. 代码附在图片后。


Runner
master
Runner 〉 iPhone 15 Plus
Build Succeeded | 2023/11/21 at 19:30
TencentCloudChatPushPlugin
AppDelegate
Runner 〉 Runner 〉 AppDelegate 〉 AppDelegate
No such module 'Flutter'
Runner
Flutter
AppFrameworkInfo
Debug
Release
Generated
Runner
Runner
Main
Assets
LaunchScreen
Info
GeneratedPluginRegistrant
AppDelegate
Runner-Bridging-Header
Products
RunnerTests
Pods
Pods-Runner.debug
Pods-Runner.release
Pods-Runner.profile
Pods-RunnerTests.debug
Pods-RunnerTests.release
Pods-RunnerTests.profile
Frameworks
Pods
import UIKit
import Flutter
// Add these two import lines
import TIMPush
import tencent_cloud_chat_push
// Add `, TIMPushDelegate` to the following line
@UIApplicationMain
@objc class AppDelegate: FlutterAppDelegate, TIMPushDelegate {
override func application(
_ application: UIApplication,
didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey: Any]?
) -> Bool {
GeneratedPluginRegistrant.register(with: self)
return super.application(application, didFinishLaunchingWithOptions: launchOptions)
}
// Add this function
func offlinePushCertificateID() -> Int32 {
return TencentCloudChatPushFlutterModal.shared.offlinePushCertificateID();
}
// Add this function
func applicationGroupID() -> String {
return TencentCloudChatPushFlutterModal.shared.applicationGroupID()
}
// Add this function
func onRemoteNotificationReceived(_ notice: String?) -> Bool {
TencentCloudChatPushPlugin.shared.tryNotifyDartOnNotificationClickEvent(notice)
return true
}
}



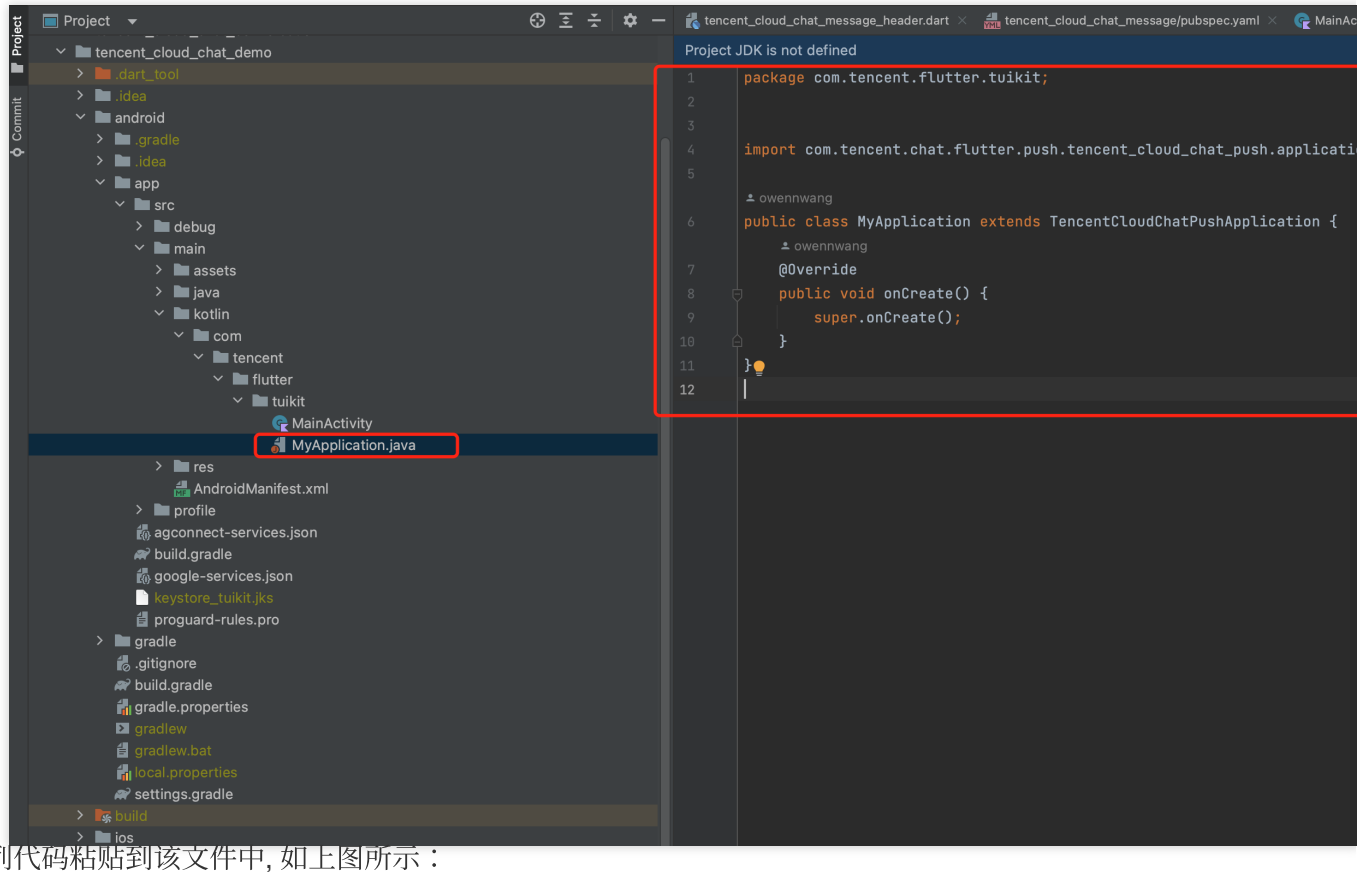
```
import UIKit
import Flutter

// Add these two import lines
import TIMPush
import tencent_cloud_chat_push

// Add `, TIMPushDelegate` to the following line
@UIApplicationMain
@objc class AppDelegate: FlutterAppDelegate, TIMPushDelegate {
    override func application(
```

```
        _ application: UIApplication,
        didFinishLaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKe
    ) -> Bool {
        GeneratedPluginRegistrant.register(with: self)
        return super.application(application, didFinishLaunchingWithOptions: launch
    }

    // Add this function
    func offlinePushCertificateID() -> Int32 {
        return TencentCloudChatPushFlutterModal.shared.offlinePushCertificateID();
    }

    // Add this function
    func applicationGroupID() -> String {
        return TencentCloudChatPushFlutterModal.shared.applicationGroupID()
    }

    // Add this function
    func onRemoteNotificationReceived(_ notice: String?) -> Bool {
        TencentCloudChatPushPlugin.shared.tryNotifyDartOnNotificationClickEvent(not
        return true
    }
}
```

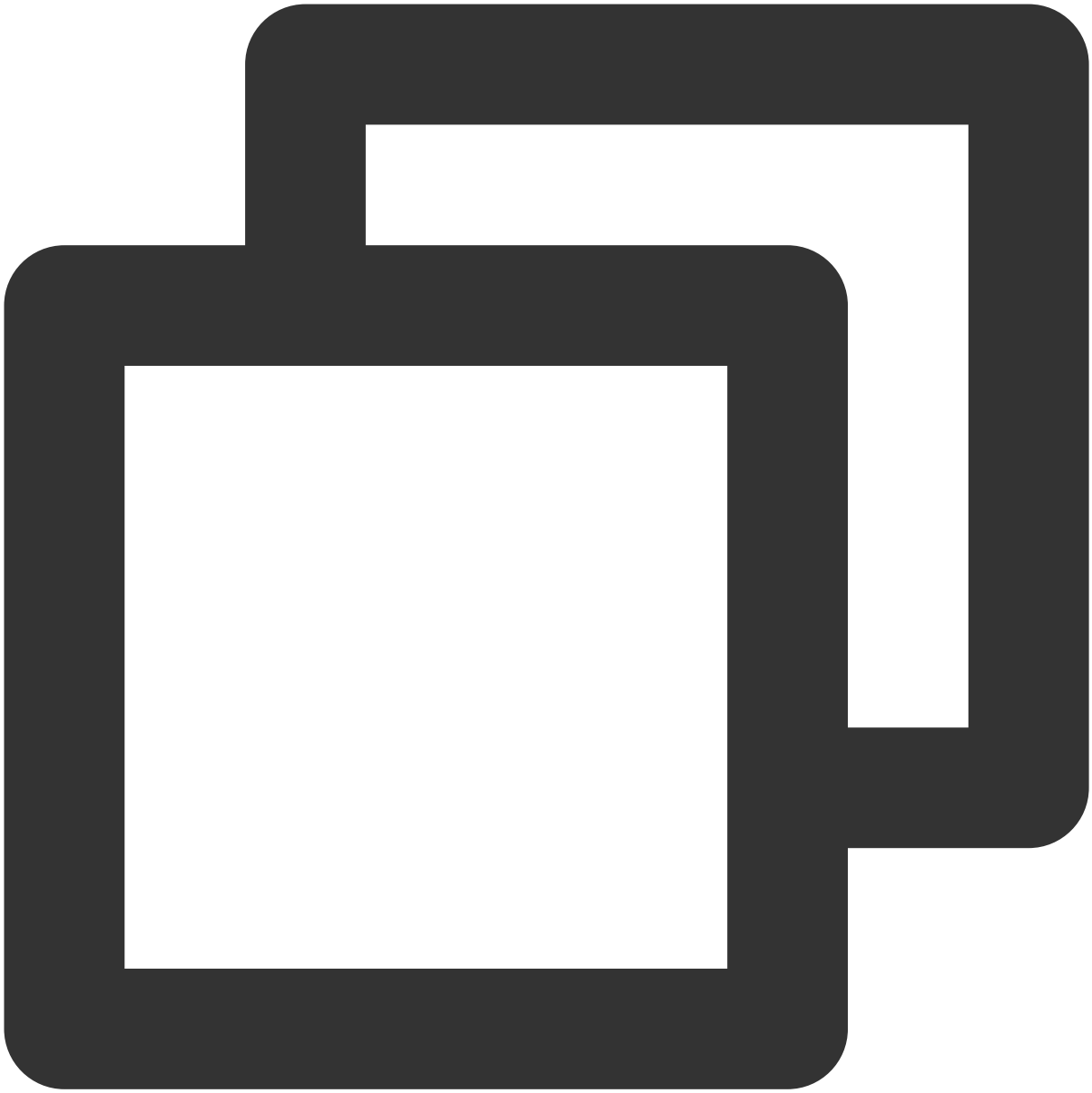
建议使用 Android Studio 完成本部分编辑。

在您项目 android 路径下 `MainActivity` 同级目录中，新建一个新的 Application 文件类, 例如可命名为 `MyApplication.java` 。

如果您已经自定义了一个 Application 类，则可直接复用，不需要再次创建。



将下列代码粘贴到该文件中，如上图所示：

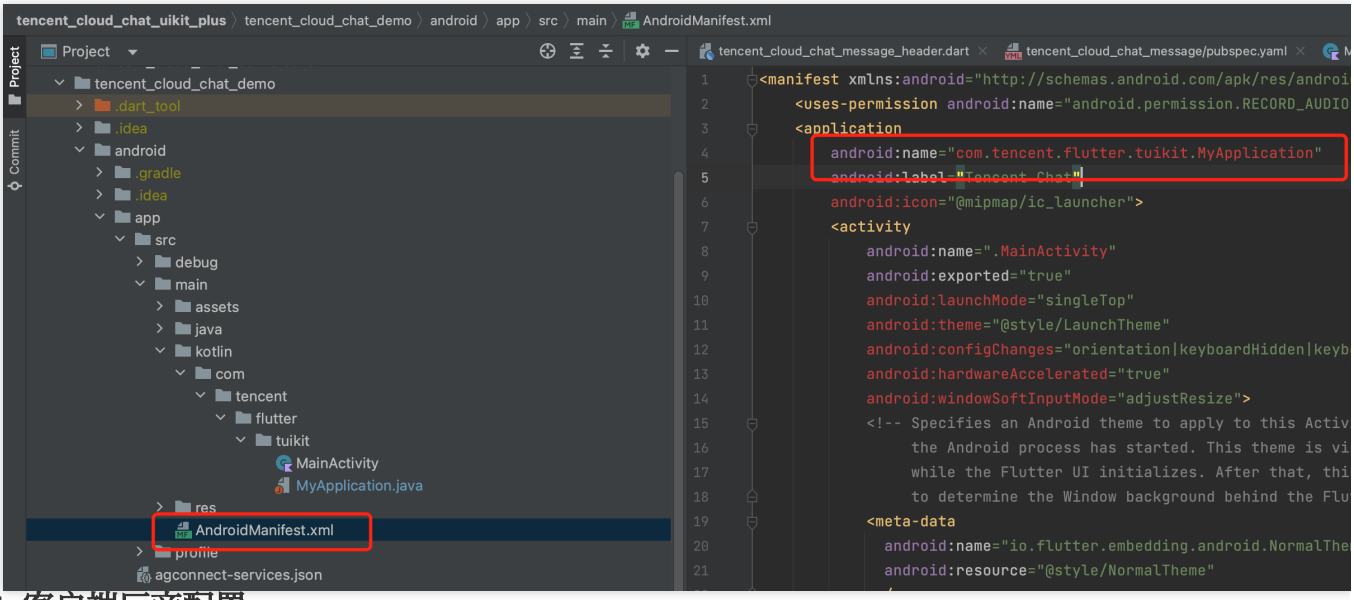


```
package 替换成您自己的, 一般 Android Studio 会自动生成;

import com.tencent.chat.flutter.push.tencent_cloud_chat_push.application.TencentClo

public class MyApplication extends TencentCloudChatPushApplication {
    @Override
    public void onCreate() {
        super.onCreate();
    }
}
```

说明：

如果您已经创建了自己的 Application 为了其他用途，请直接 `extends TencentCloudChatPushApplication` 并保证 `onCreate()` 函数中，调用了 `super.onCreate();` 即可。

打开 `android/app/src/main/AndroidManifest.xml` 文件，为 `<application>` 标签，新增指定一个 `android:name` 参数即可，指向刚制作的自定义 `Application` 类。如图所示：



步骤4: 客户端厂商配置

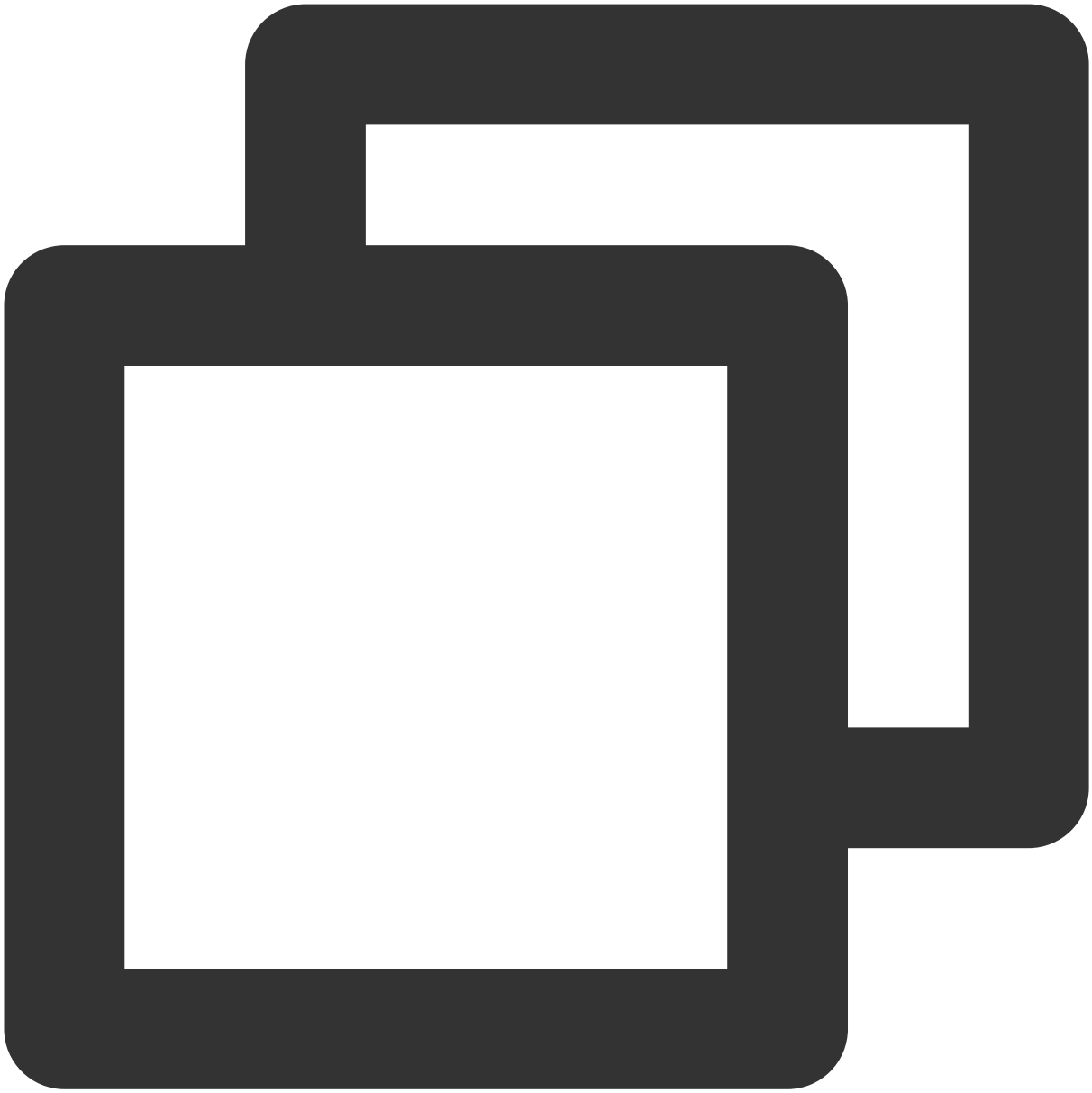
iOS

Android

iOS 端无需进行此步骤。

打开 android/app/build.gradle 文件，在最后，新增 dependencies 配置，并根据需要，引入下列全部或部分厂商的推送包。只有引入对应厂商的推送包，才能启用该厂商的原生推送能力。

以下所述的版本号，和本 Flutter 推送插件（tencent_cloud_chat_push）的版本号，保持一致。



```
dependencies {
      // Huawei
      implementation 'com.tencent.timpush:huawei:${推送插件的版本号}'

      // XiaoMi
      implementation 'com.tencent.timpush:xiaomi:${推送插件的版本号}'

      // vivo
      implementation 'com.tencent.timpush:vivo:${推送插件的版本号}'

      // Honor
```

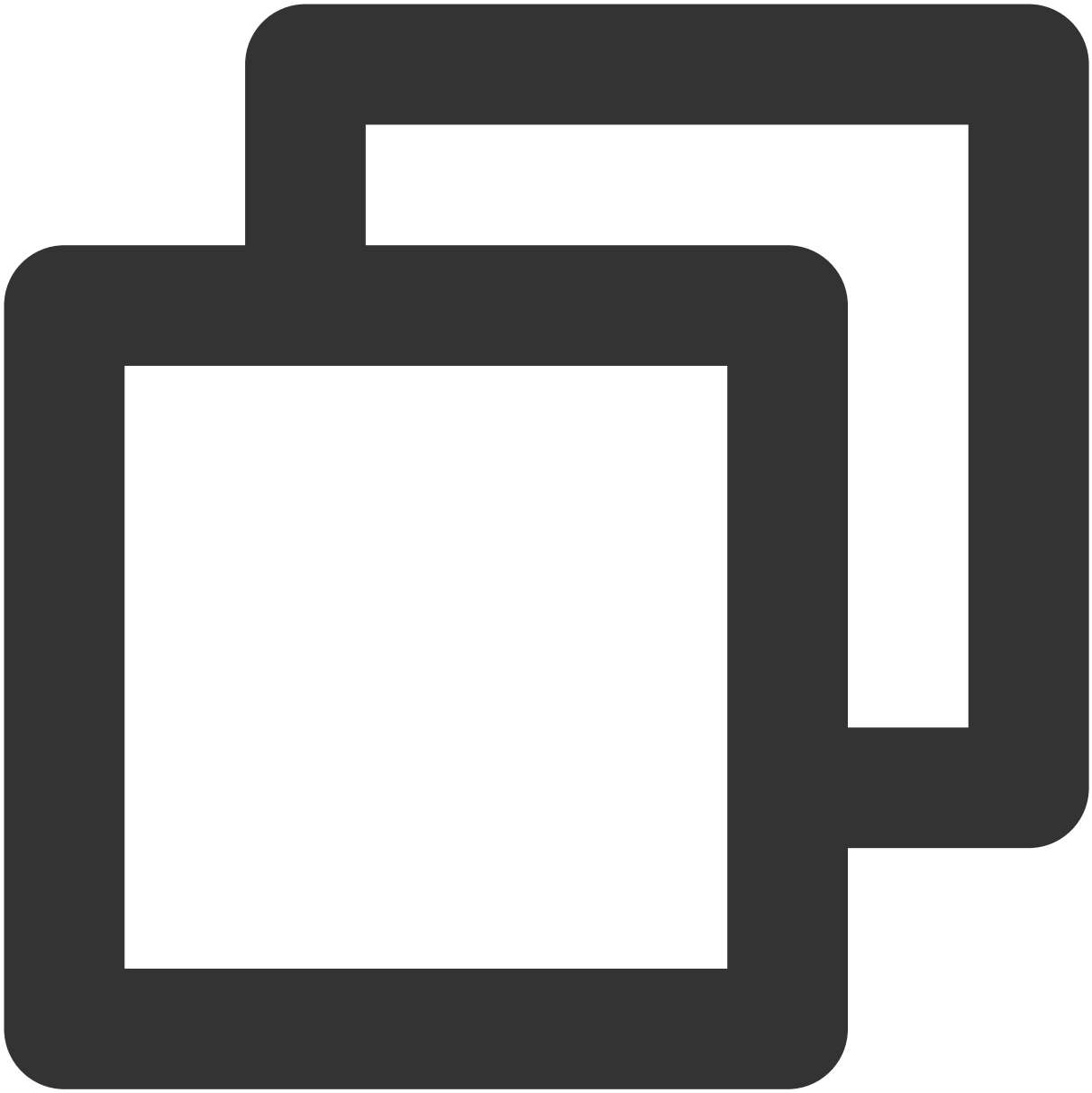
```
        implementation 'com.tencent.timpush:honor:${推送插件的版本号}'

        // Meizu
        implementation 'com.tencent.timpush:meizu:${推送插件的版本号}'

        // Google Firebase Cloud Messaging (Google FCM)
        implementation 'com.tencent.timpush:fcm:${推送插件的版本号}'

        // OPPO 以下二选一
        //中国区域选择集成此包
        implementation 'com.tencent.timpush:oppo:${推送插件的版本号}'
        //其他区域选择集成此包
        implementation 'com.tencent.timpush:oppo-intl:${推送插件的版本号}'
    }
```

Vivo 和荣耀适配

根据 vivo 和荣耀厂商接入指引，需要将 APPID 和 APPKEY 添加到清单文件中。

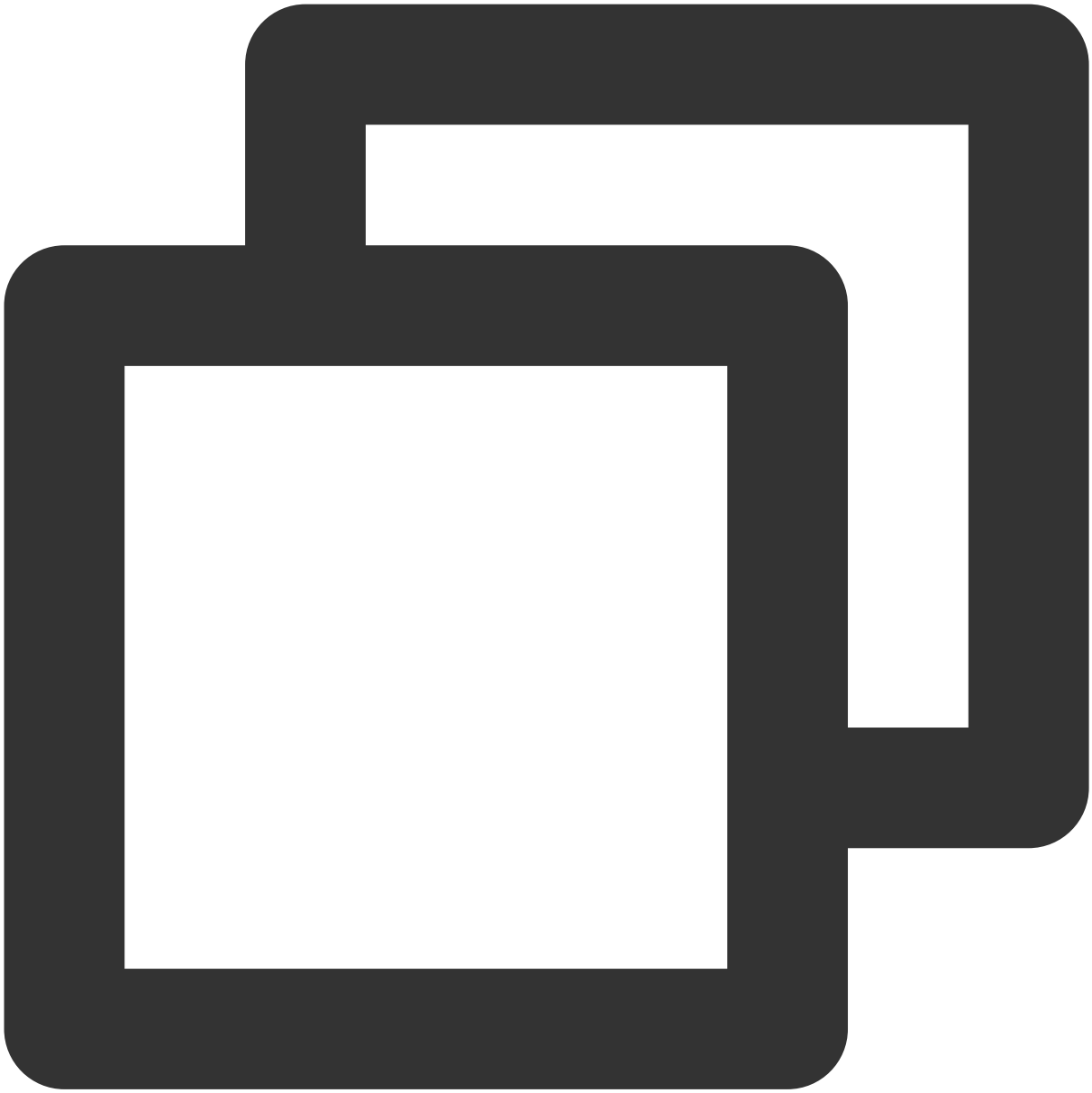
方法1

方法2



```
// android/app/build.gradle

android {
    ...
    defaultConfig {
        ...
        manifestPlaceholders = [
            "VIVO_APPKEY" : "您应用分配的证书 APPKEY",
            "VIVO_APPID" : "您应用分配的证书 APPID",
            "HONOR_APPID" : "您应用分配的证书 APPID"
        ]
```

```
    }
}
```



```
// android/app/src/main/AndroidManifest.xml

// Vivo begin
<meta-data tools:replace="android:value"
    android:name="com.vivo.push.api_key"
    android:value="您应用分配的证书 APPKEY" />
<meta-data tools:replace="android:value"
    android:name="com.vivo.push.app_id"
```

```
    android:value="您应用分配的证书 APPID" />
// Vivo end

// Honor begin
<meta-data tools:replace="android:value"
    android:name="com.hihonor.push.app_id"
    android:value="您应用分配的证书 APPID" />
// Honor end
```

华为、荣耀和 Google FCM 适配

按照厂商方法，集成对应的 plugin 和 json 配置文件。

注意：

以下荣耀的适配仅 7.7.5283 及以上版本需要配置。

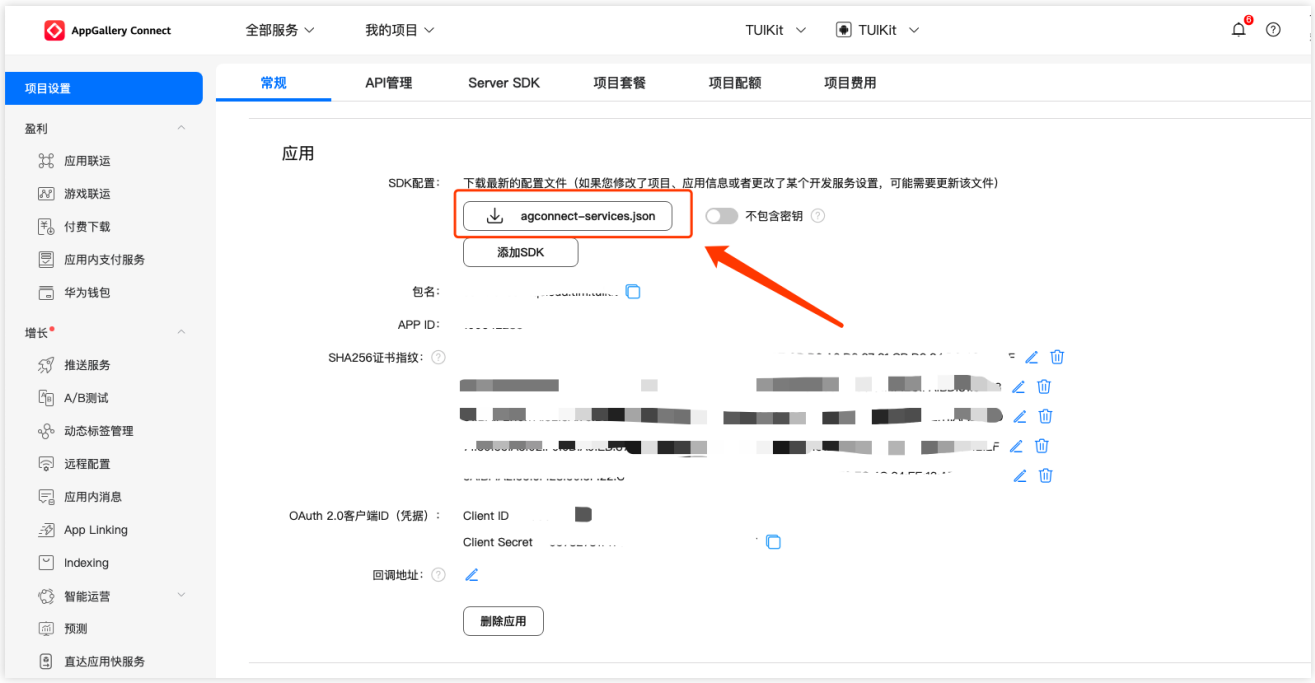
1.1 下载配置文件添加到工程根目录/Android/app。

华为

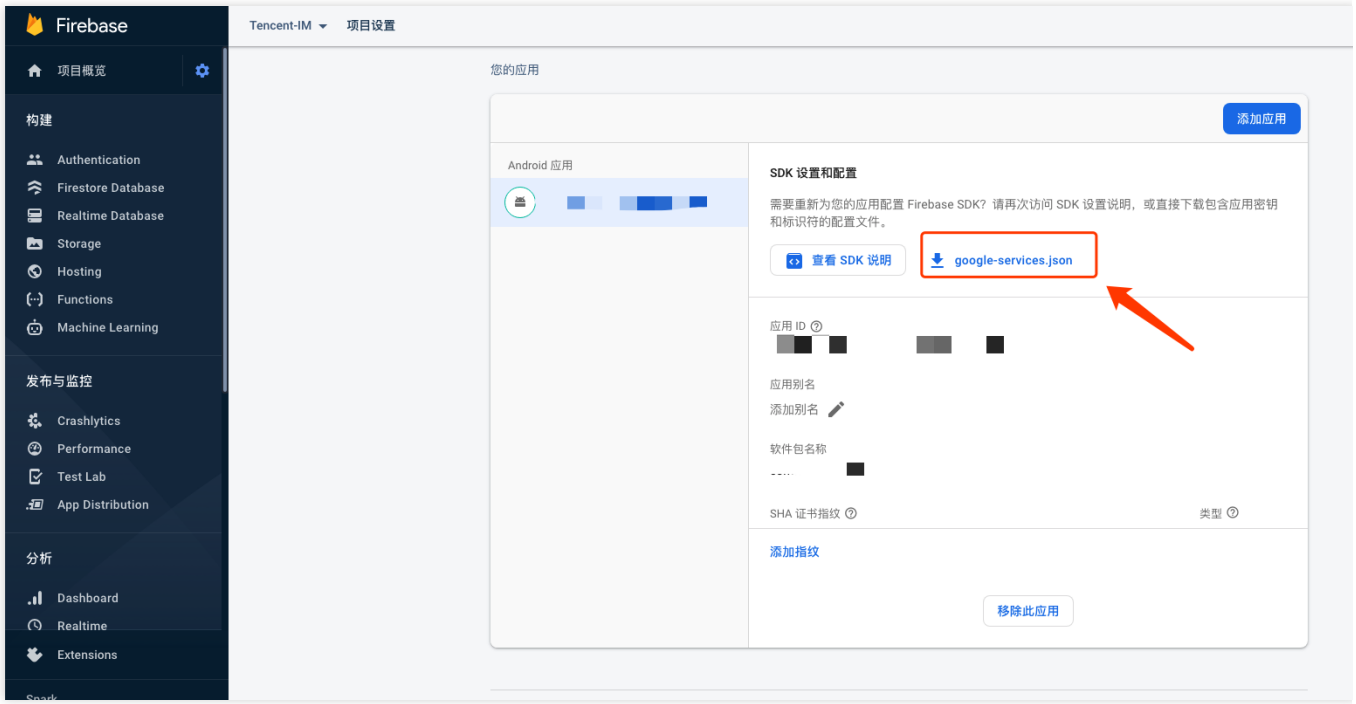
荣耀

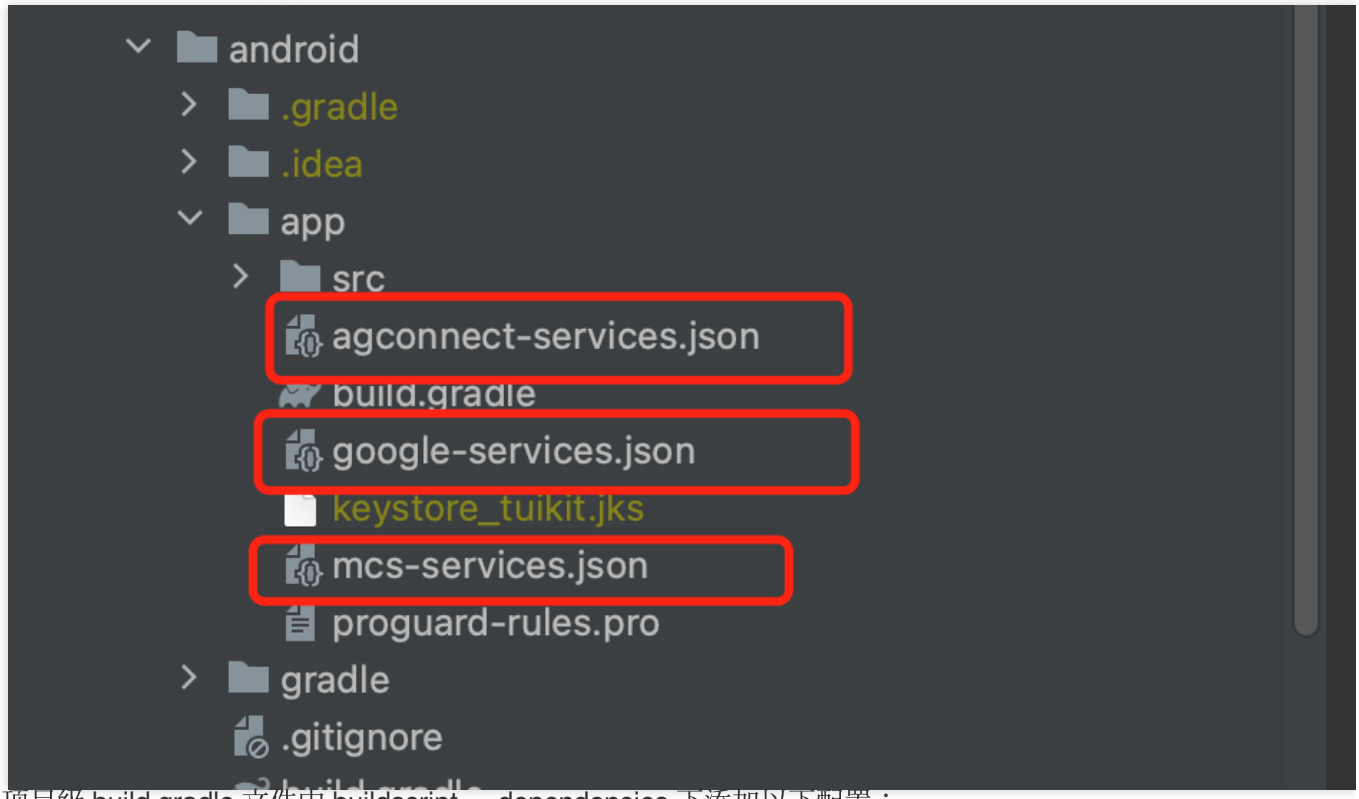
Google FCM

操作路径




HONOR Developers
生态服务
应用服务
游戏服务
智慧服务
内容服务
应用管理
HONOR Connect
开放能力
开发者中心
生态服务 > 应用管理 > 应用基础信息查看
应用基础信息
应用基础信息做任何更改，将在提交保存后生效
应用名称：
更新时间：
App ID：
应用包名：
平台类型：安卓
应用类型：应用
支持设备：手机/平板
默认语言：简体中文
SDK 配置：下载最新的配置文件(如果您修改了应用信息或者更改了某个开发服务设置，可能需要更新该文件)
mcs-services.json
添加 SDK


Firebase
Tencent-IM
项目设置
项目概览
构建
Authentication
Firestore Database
Realtime Database
Storage
Hosting
Functions
Machine Learning
发布与监控
Crashlytics
Performance
Test Lab
App Distribution
分析
Dashboard
Realtime
Extensions
您的应用
添加应用
Android 应用
SDK 设置和配置
需要重新为您的应用配置 Firebase SDK？请再次访问 SDK 设置说明，或直接下载包含应用密钥和标识符的配置文件。
查看 SDK 说明
google-services.json
应用 ID
应用别名
添加别名
软件包名称
SHA 证书指纹
类型
添加指纹
移除此应用



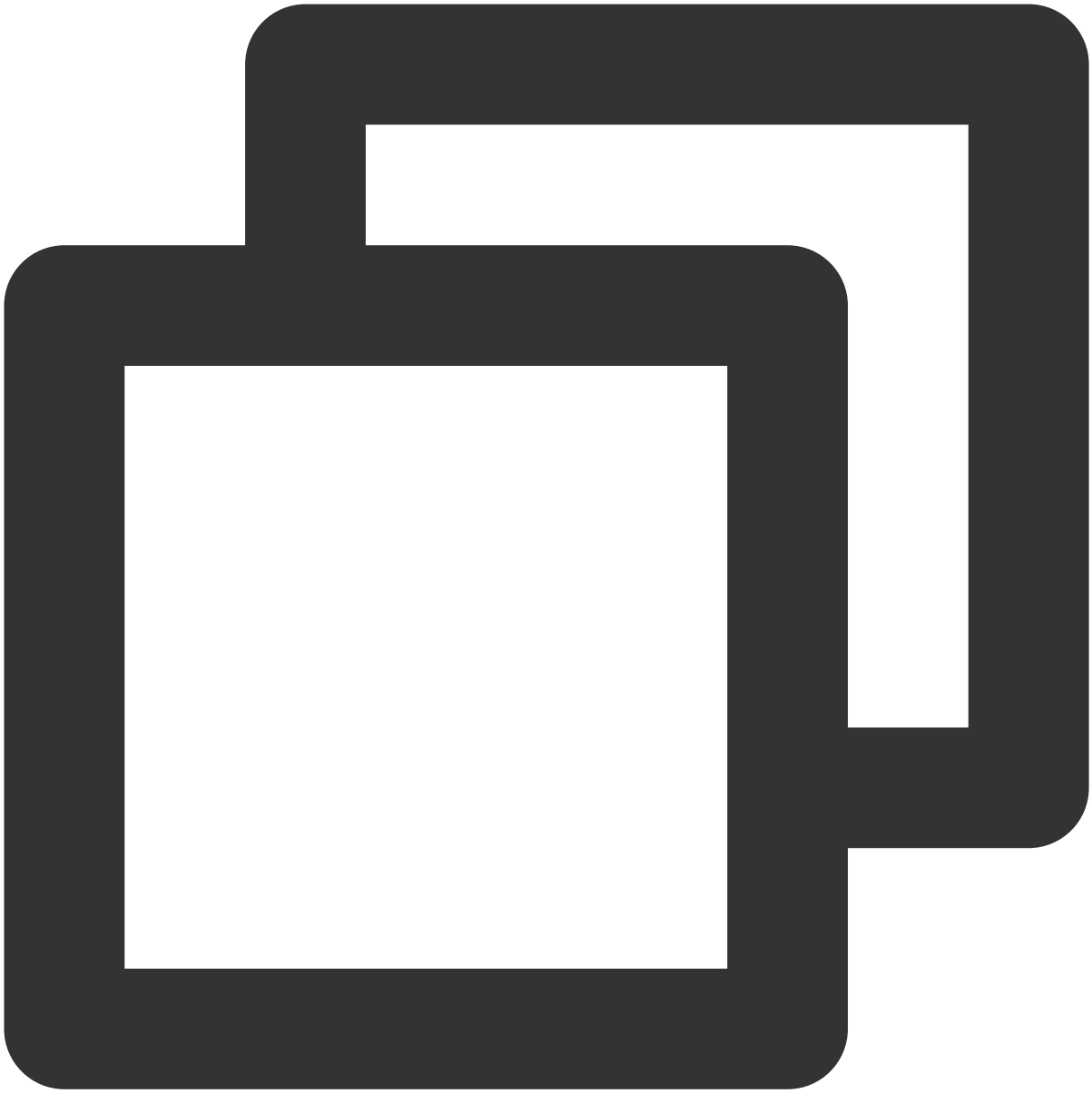
1.2 在项目级 build.gradle 文件中 buildscript -> dependencies 下添加以下配置：

Gradle 7.1 及以上版本

Gradle 7.0 版本

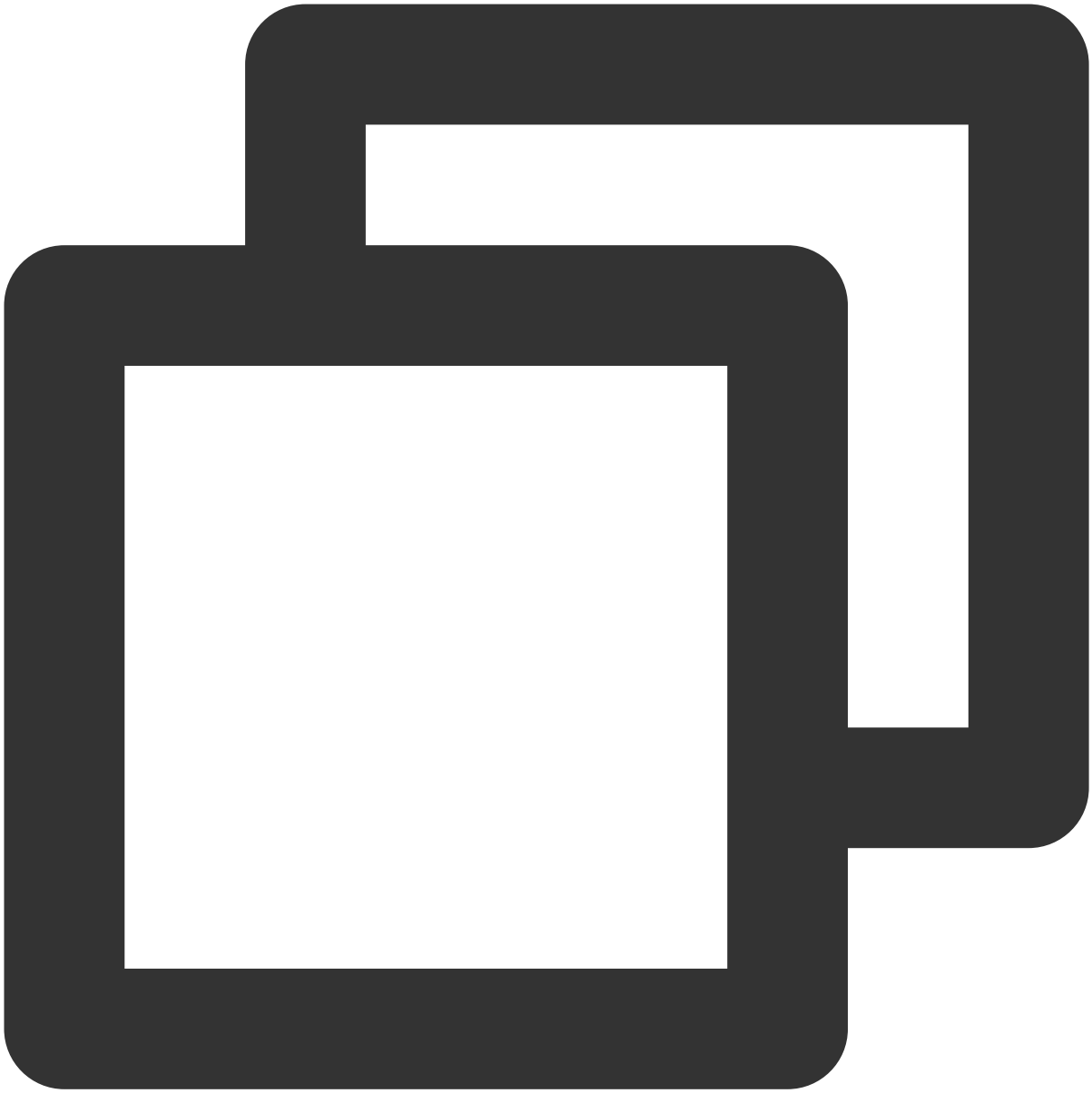
Gradle 7.0 以下版本

在项目级 build.gradle 文件中 buildscript -> dependencies 下添加以下配置：



```
buildscript {
    dependencies {
        ...
        classpath 'com.huawei.agconnect:agcp:1.6.0.300'
        classpath 'com.hihonor.mcs:asplugin:2.0.1.300'
        classpath 'com.google.gms:google-services:4.4.0'
    }
}
```

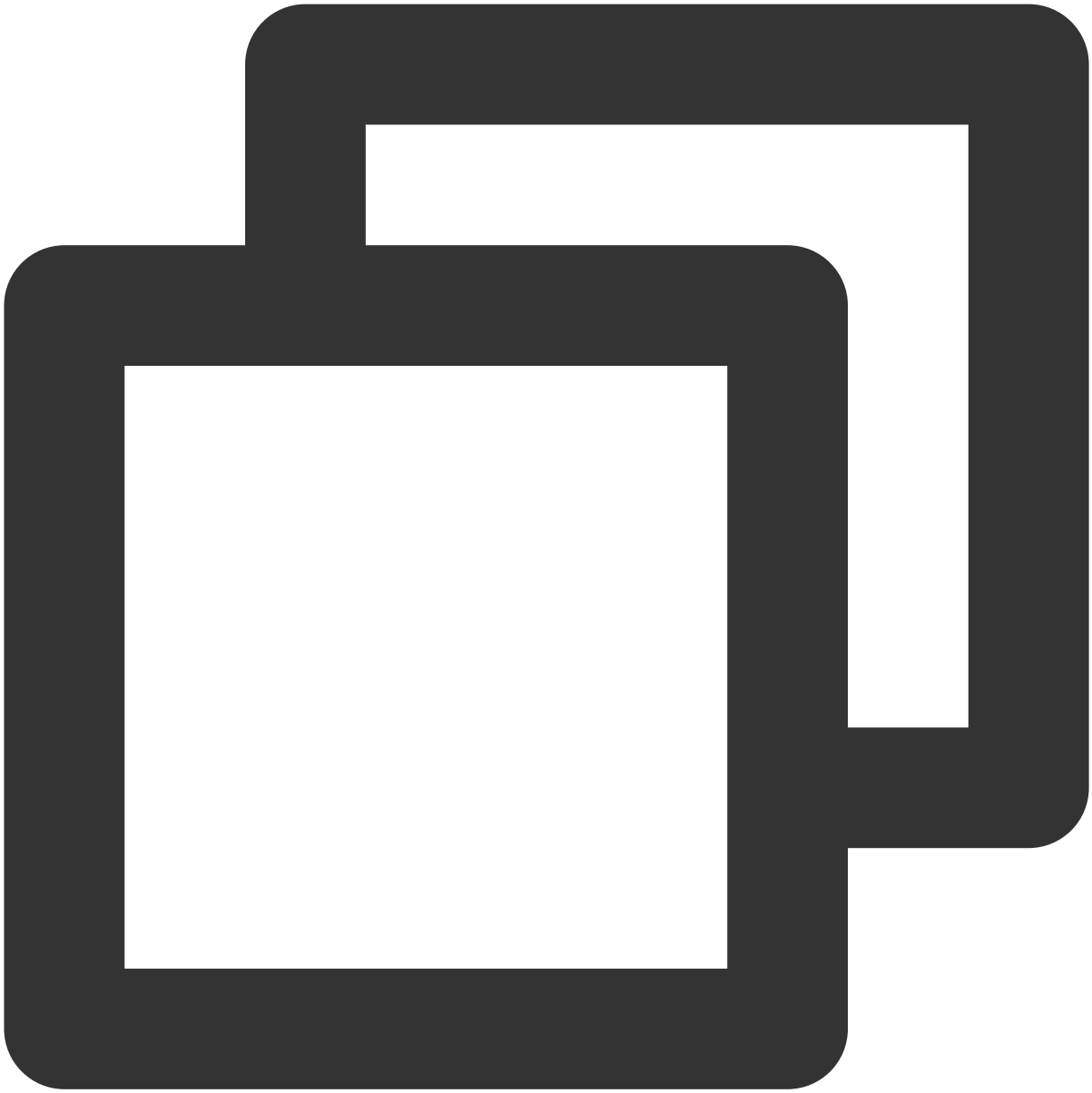
在项目级 settings.gradle 文件中 buildscript -> repositories 和 allprojects -> repositories 下添加以下仓库配置：



```
pluginManagementbuildscript {
    repositories {
        gradlePluginPortal()
        mavenCentral()
        maven { url "https://mirrors.tencent.com/nexus/repository/maven-public/" }
        // 配置HMS Core SDK的Maven仓地址。
        maven {url 'https://developer.huawei.com/repo/'}
        maven {url 'https://developer.hihonor.com/repo'}
    }
}
allprojects {
```

```
    ...
    repositories {
        mavenCentral()
        maven { url "https://mirrors.tencent.com/nexus/repository/maven-public/" }
        // 配置HMS Core SDK的Maven仓地址。
        maven {url 'https://developer.huawei.com/repo/'}
        maven {url 'https://developer.hihonor.com/repo'}
    }
    }
}
```

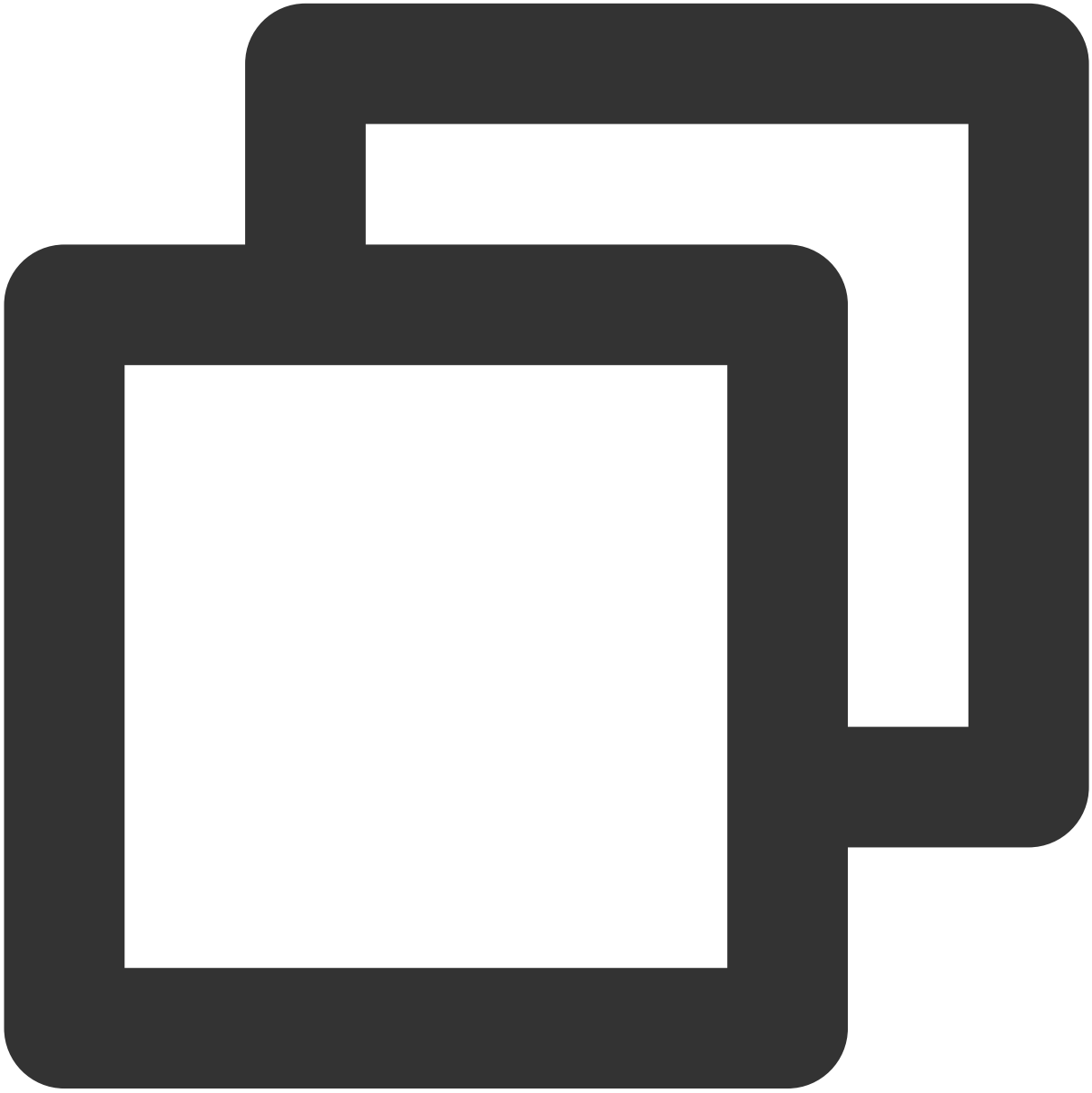
在项目级 build.gradle 文件中 buildscript 下添加以下配置：



```
buildscript {
    repositories {
        mavenCentral()
        maven { url "https://mirrors.tencent.com/nexus/repository/maven-public/" }
        // 配置HMS Core SDK的Maven仓地址。
        maven {url 'https://developer.huawei.com/repo/'}
        maven {url 'https://developer.hihonor.com/repo'}
    }
    dependencies {
        ...
        classpath 'com.google.gms:google-services:4.2.0'
```

```
        classpath 'com.huawei.agconnect:agcp:1.4.1.300'
        classpath 'com.hihonor.mcs:asplugin:2.0.1.300'
    }
}
```

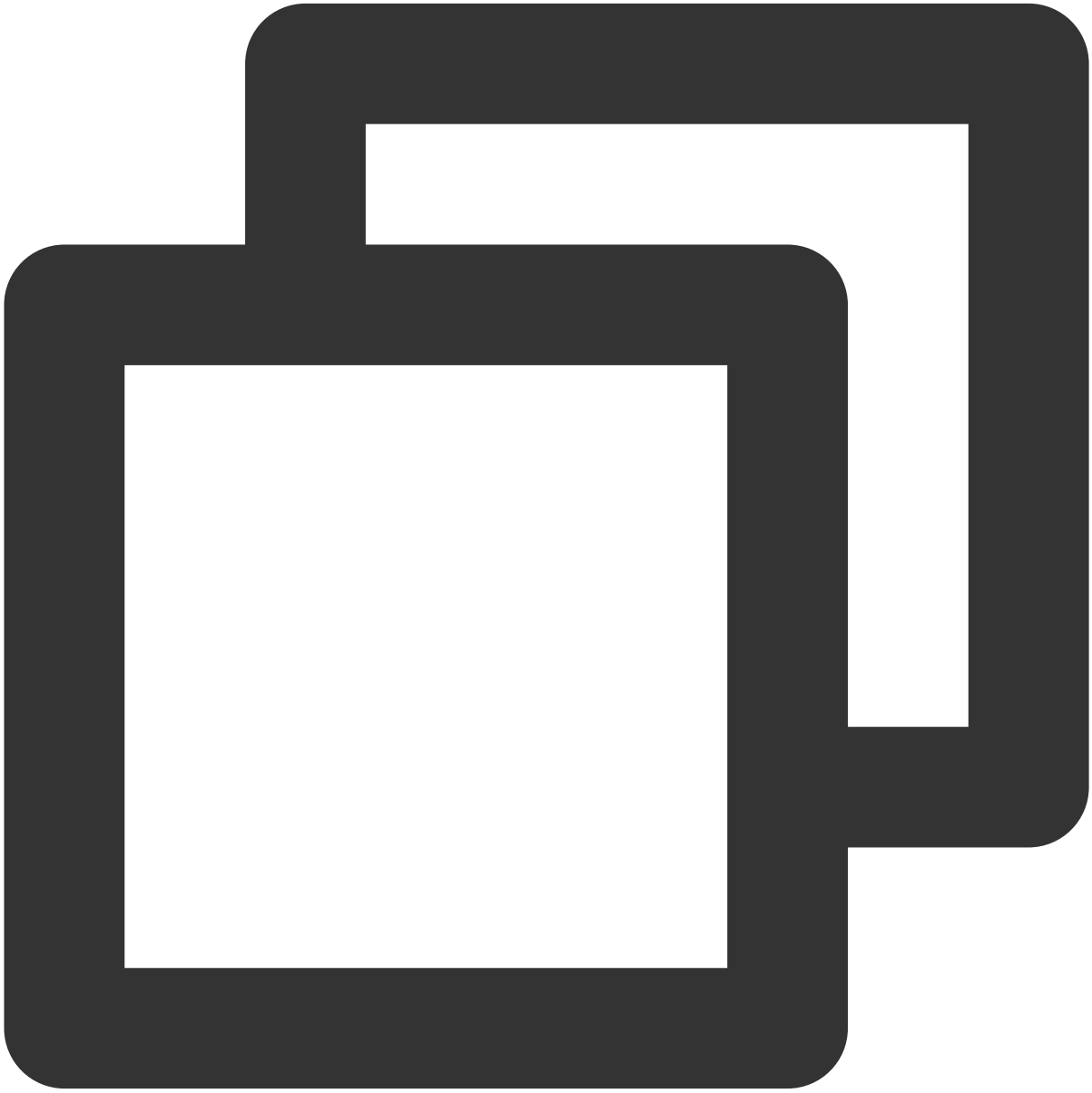
在项目级 settings.gradle 文件中 allprojects -> repositories 下添加以下仓库配置：



```
allprojects {
    ...
    repositories {
        mavenCentral()
        maven { url "https://mirrors.tencent.com/nexus/repository/maven-public/" }
```

```
        // 配置HMS Core SDK的Maven仓地址。
        maven {url 'https://developer.huawei.com/repo/'}
        maven {url 'https://developer.hihonor.com/repo'}
    }
}
```

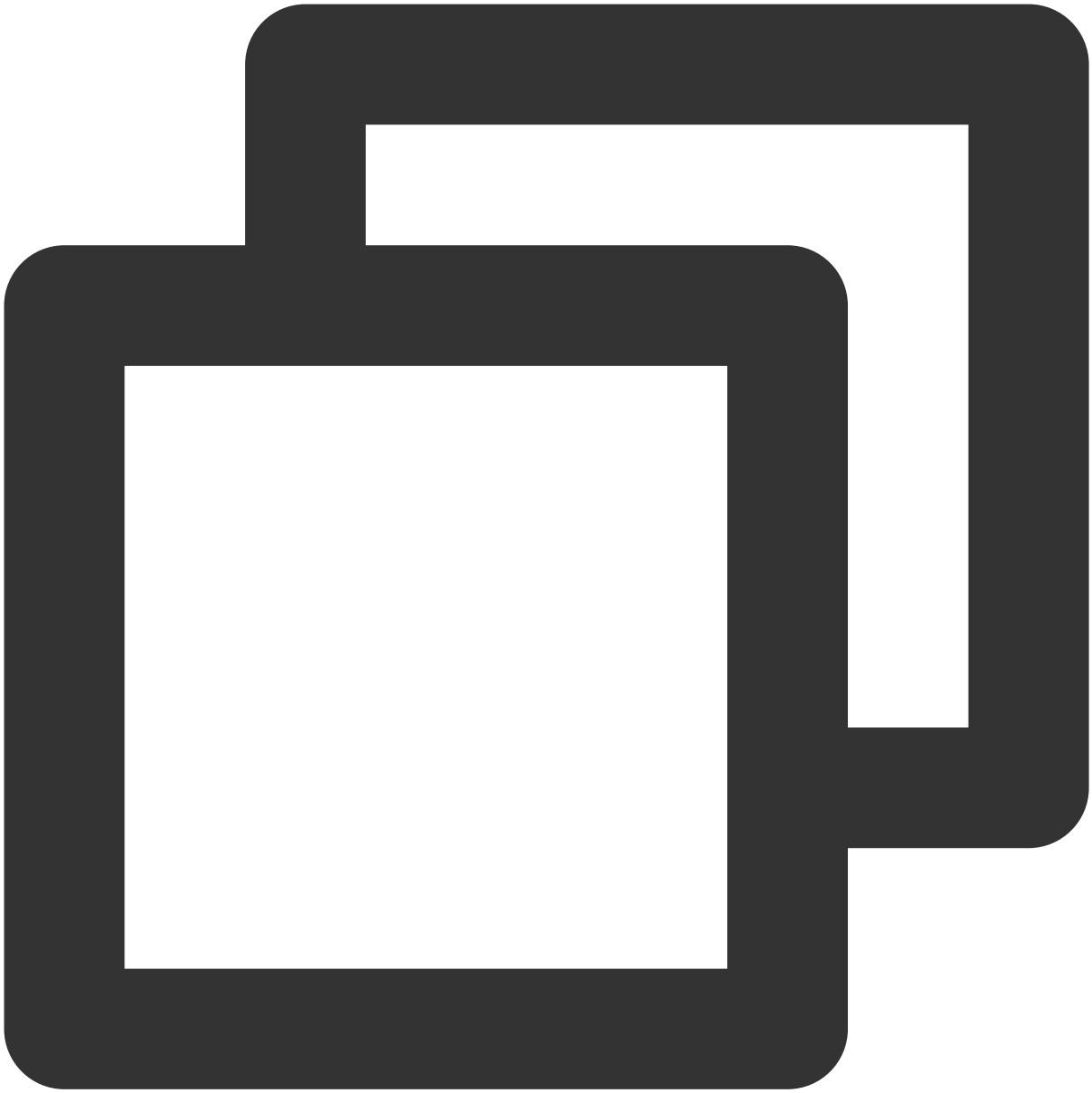
在项目级 build.gradle 文件中 buildscript 和 allprojects 下添加以下配置：



```
buildscript {
    repositories {
```

```
        mavenCentral()
        maven { url "https://mirrors.tencent.com/nexus/repository/maven-public/" }
        // 配置HMS Core SDK的Maven仓地址。
        maven {url 'https://developer.huawei.com/repo/'}
        maven {url 'https://developer.hihonor.com/repo'}
    }
    dependencies {
        ...
        classpath 'com.google.gms:google-services:4.2.0'
        classpath 'com.huawei.agconnect:agcp:1.4.1.300'
        classpath 'com.hihonor.mcs:asplugin:2.0.1.300'
    }
}

allprojects {
    repositories {
        mavenCentral()
        maven { url "https://mirrors.tencent.com/nexus/repository/maven-public/" }
        // 配置HMS Core SDK的Maven仓地址。
        maven {url 'https://developer.huawei.com/repo/'}
        maven {url 'https://developer.hihonor.com/repo'}
    }
}
```

1.3 在应用级 build.gradle 文件中添加下方配置：



```
apply plugin: 'com.google.gms.google-services'
apply plugin: 'com.huawei.agconnect'
apply plugin: 'com.hihonor.mcs.asplugin'
```

步骤5: 处理消息点击回调, 并解析参数

请定义一个函数, 用于接受推送消息点击回调事件.

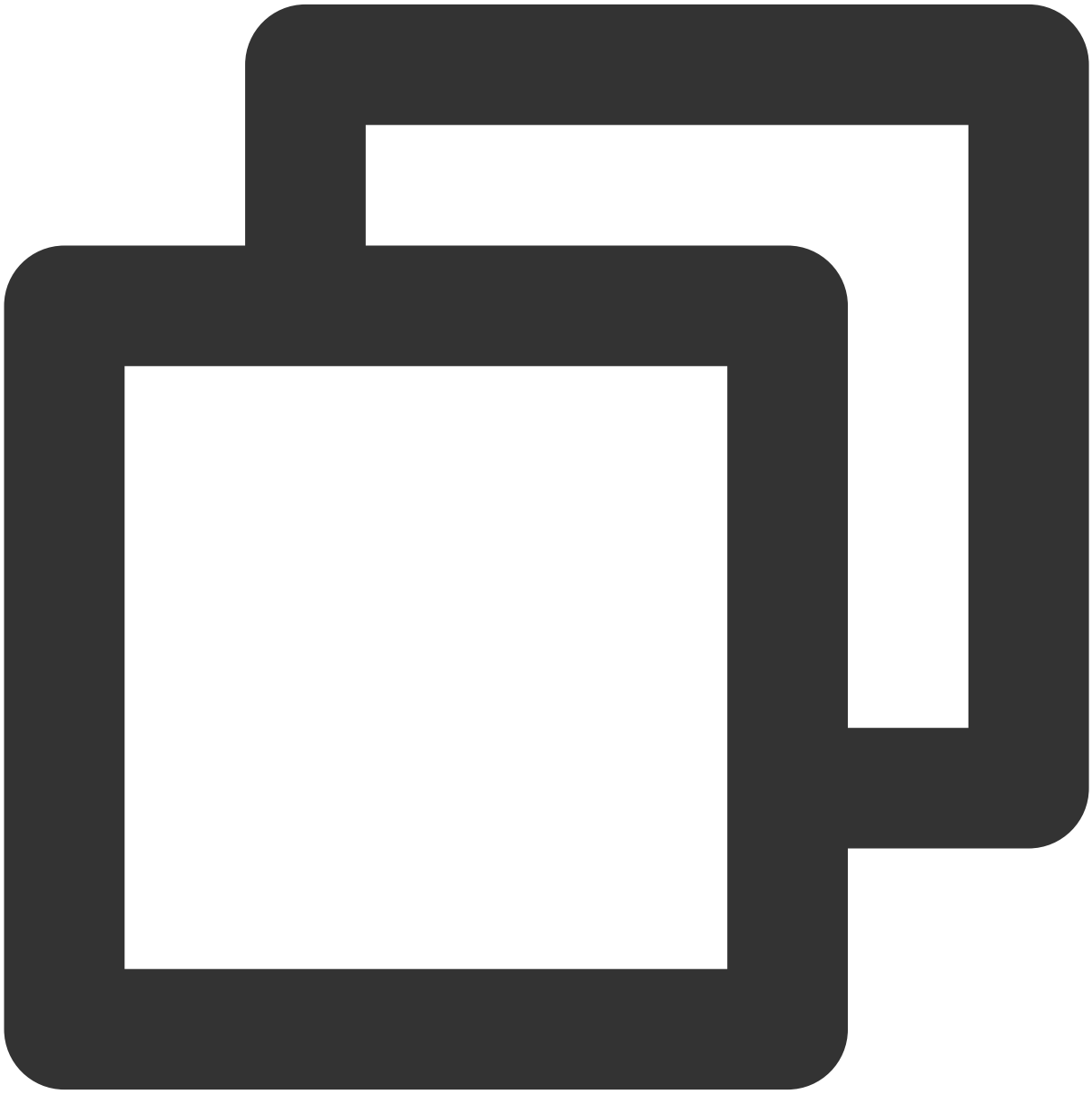
该函数请定义成 `{required String ext, String? userID, String? groupID}` 的入参形式。

其中, ext字段, 为该消息所携带的完整 ext 信息, 由发送方指定, 如果未指定, 则有默认值. 您可根据解析该字段, 跳转至对应页面。

userID 和 groupID 字段，为本插件，自动尝试解析 ext Json String， 获取里面携带的单聊对方 userID 和 群聊 groupID 信息。如果您未自定义 ext 字段，ext 字段由 SDK 或 UIKit 默认指定，则可使用此处的默认解析。如果尝试解析失败, 则为 null 空。

您可定义一个函数来接收该回调，并据此跳转至对应会话页面或您的业务页面。

示例如下:



```
void _onNotificationClicked({required String ext, String? userID, String? groupID})
  print("_onNotificationClicked: $ext, userID: $userID, groupID: $groupID");
```

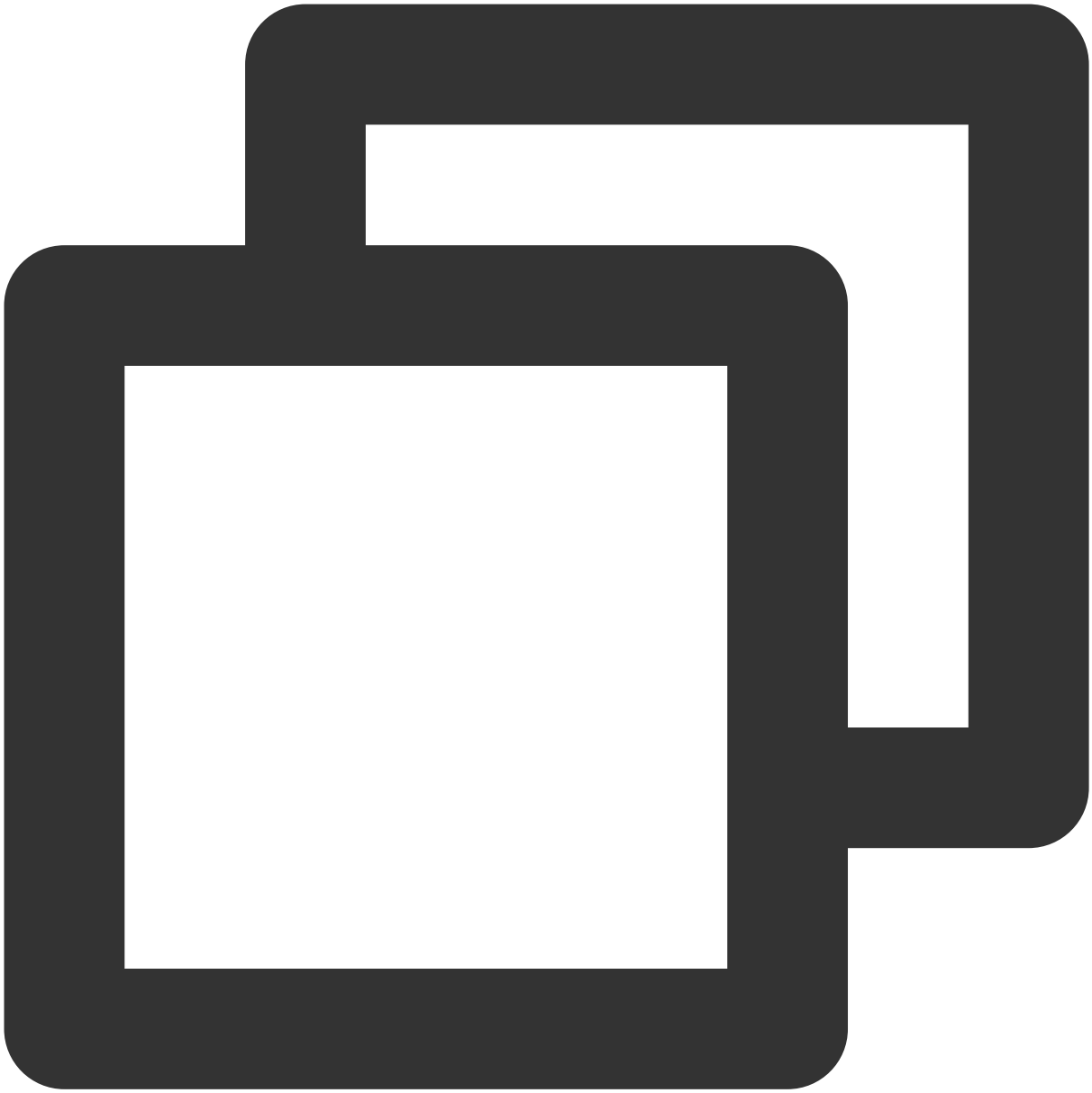
```
    if (userID != null || groupID != null) {
      // 根据 userID 或 groupID 跳转至对应 Message 页面.
    } else {
      // 根据 ext 字段，自己写解析方式，跳转至对应页面.
    }
  }
```

步骤6: 注册推送插件

请在 IM 登录完成后，且在其他插件 (如 CallKit) 使用前，立即注册推送插件。

调用 `TencentCloudChatPush().registerPush` 方法，需传入上一步定义的点击回调函数。

此外，您还可选传入 `apnsCertificateID` iOS 推送证书 ID 及 `androidPushOEMConfig` Android 推送厂商配置。此二项配置已在前序步骤指定，若无修改必要，可不再传入。

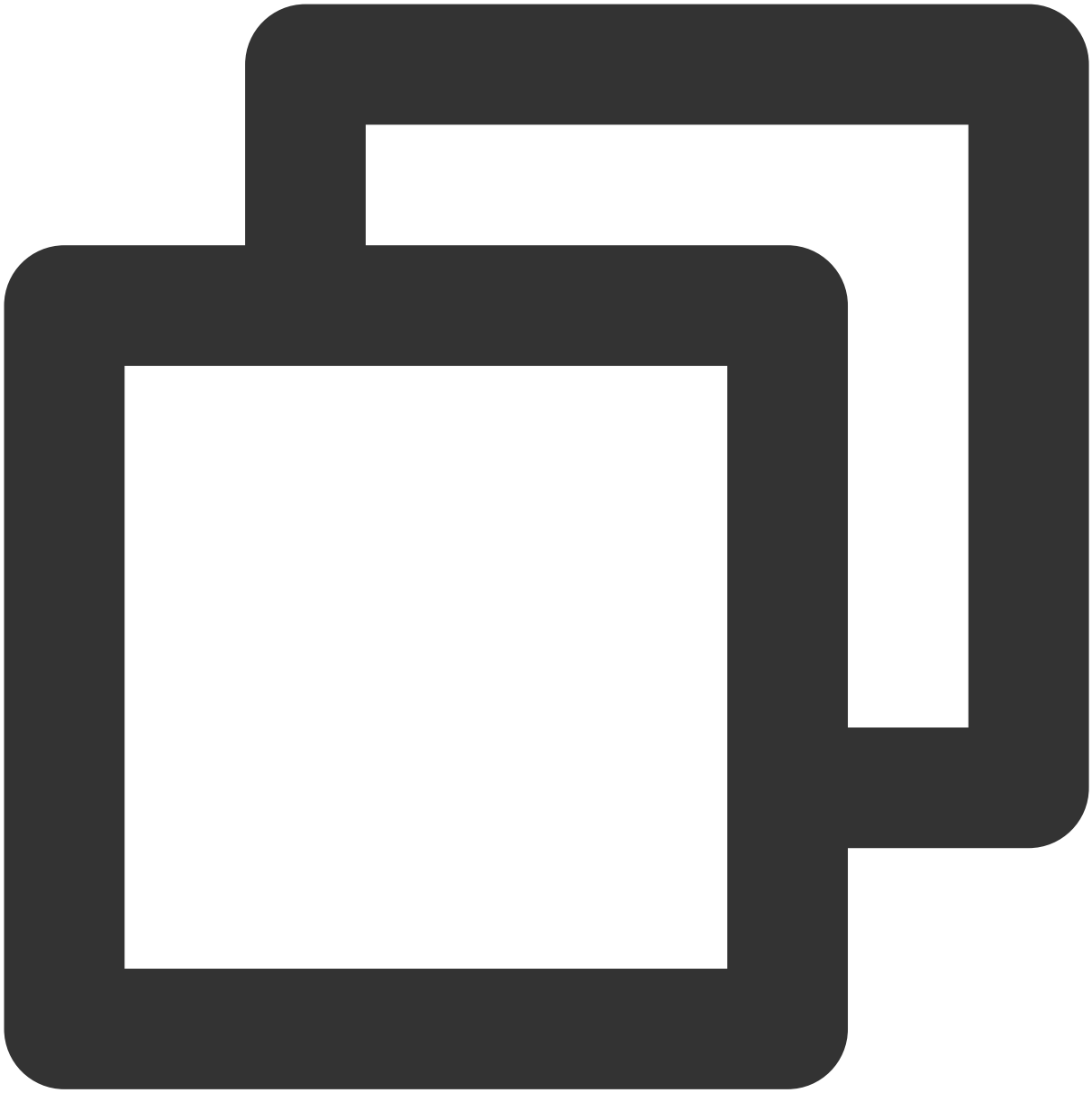


```
TencentCloudChatPush().registerPush(onNotificationClicked: _onNotificationClicked);
```

说明：

如果您的应用需要使用**推送插件进行业务消息通知**，并且在**启动后不会立即启动并登录 IM 模块**，或者在**登录 IM 模块之前需要通过获取消息点击回调来处理业务导航**，建议您尽早调用 `TencentCloudChatPush().registerOnNotificationClickedEvent` 方法，手动挂载消息单击回调，以便及时获取消息参数。

在这种场景下，您可以在调用 `TencentCloudChatPush().registerPush` 之前执行此函数，并尽可能提前将其放置在代码中。



```
TencentCloudChatPush().registerOnNotificationClickedEvent(onNotificationClicked: _o
```

步骤7: 消息推送触达统计

如果您需要统计触达数据，请按照如下完成配置：

华为

荣耀

vivo

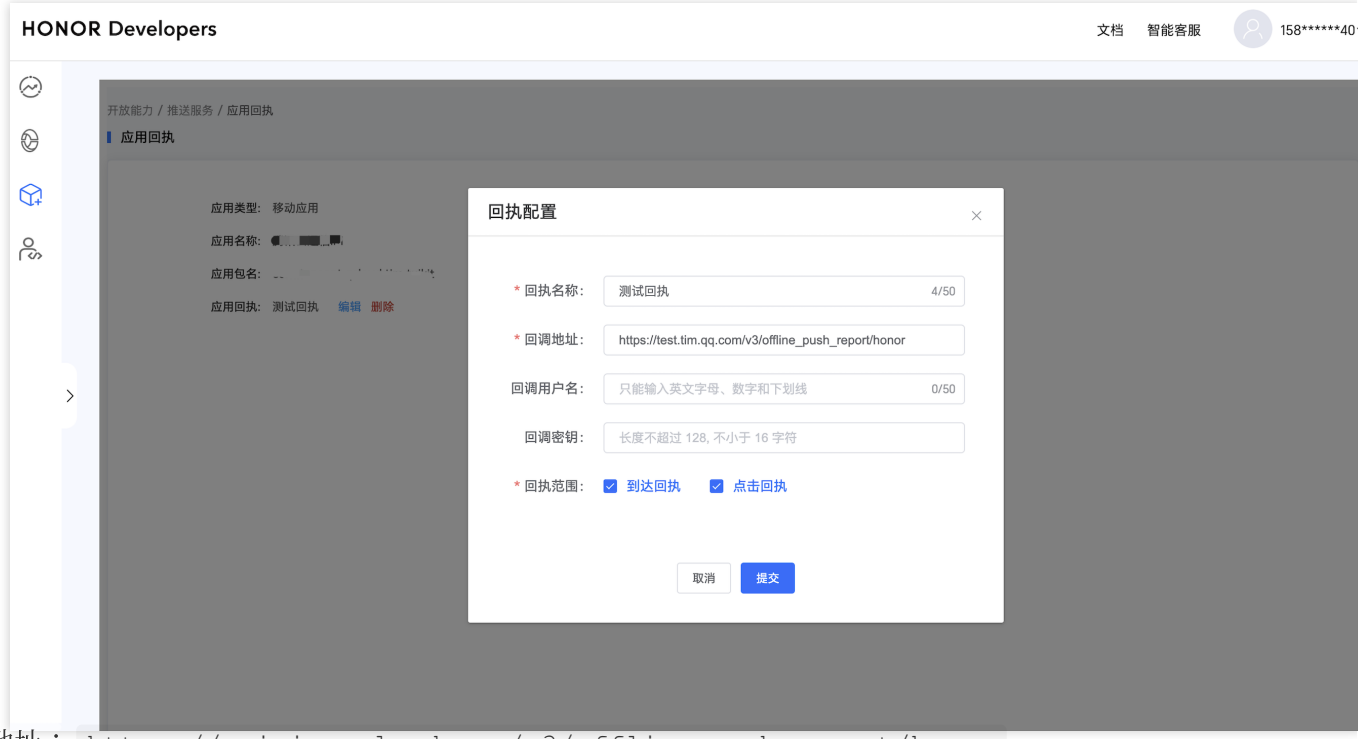
魅族

iOS

回执地址： `https://api.im.qcloud.com/v3/offline_push_report/huawei`

注意：

华为推送证书 ID <= 11344 ，使用华为推送 v2 版本接口，不支持触达和点击回执，请重新生成更新证书 ID。

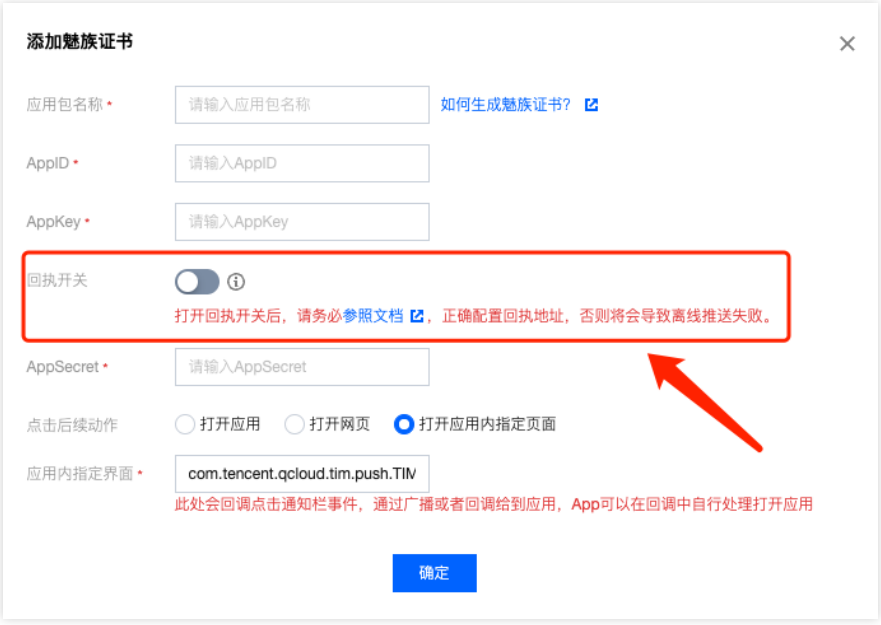


回执地址： `https://api.im.qcloud.com/v3/offline_push_report/honor`

回调地址配置	回执 ID 配置 IM 控制台

回执地址：`https://api.im.qcloud.com/v3/offline_push_report/vivo`

打开回执开关



配置回执地址

回执地址：`https://api.im.qcloud.com/v3/offline_push_report/meizu`

注意：

打开回执开关后，请务必确保回执地址正确配置。不配置或者配置地址错误，都会影响推送功能。

iOS 端推送触达统计配置，请参见 **统计推送抵达率**。

其余支持厂商不需要配置，FCM 暂不支持推送统计功能。

恭喜您已经完成了推送插件的接入，需要提醒您：推送插件**试用或购买到期后，将自动停止提供推送服务（包括普通消息离线推送、全员/标签推送等服务）**。为避免影响您业务正常使用，请提前购买/续费。

React-Native

最近更新时间：2024-06-24 15:48:08

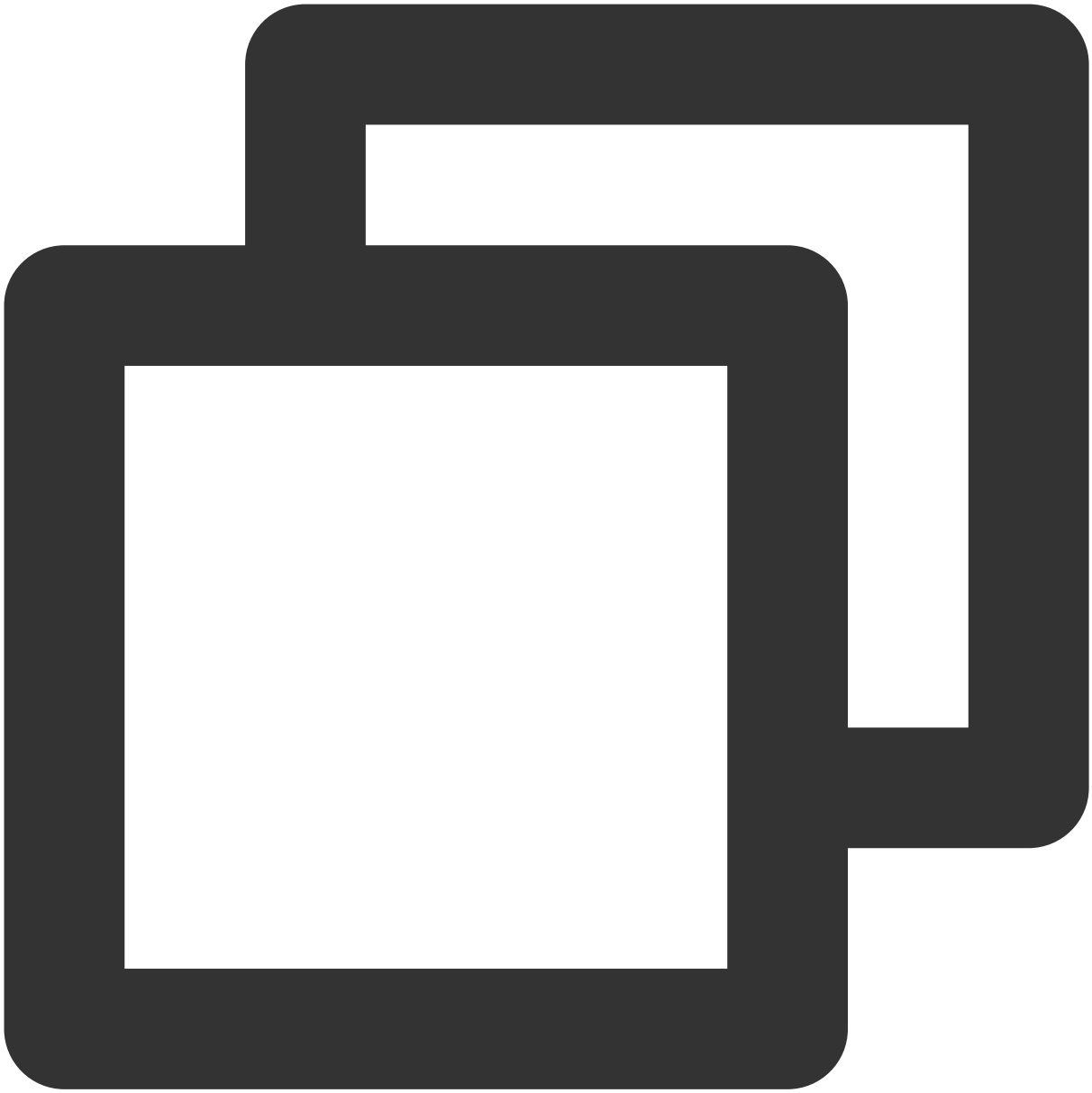
注意：

react-native-tim-push 版本低于1.0.0将不适用于该文档。

操作步骤

步骤1: 集成消息推送包

在控制台执行如下代码安装 react-native-tim-push 包。



```
// yarn
yarn add react-native-tim-push
// npm
npm install react-native-tim-push
```

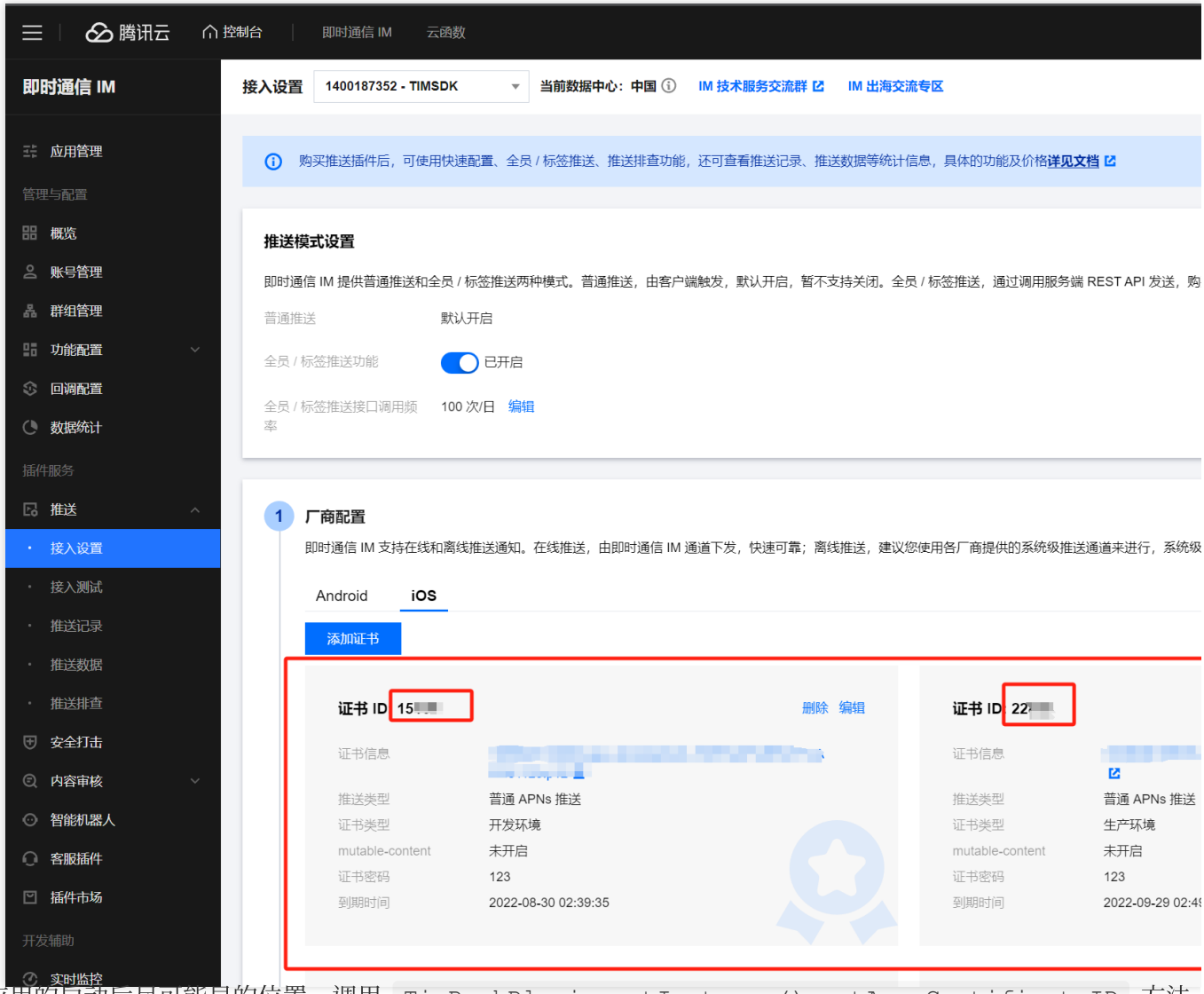
步骤2: 推送参数配置

iOS

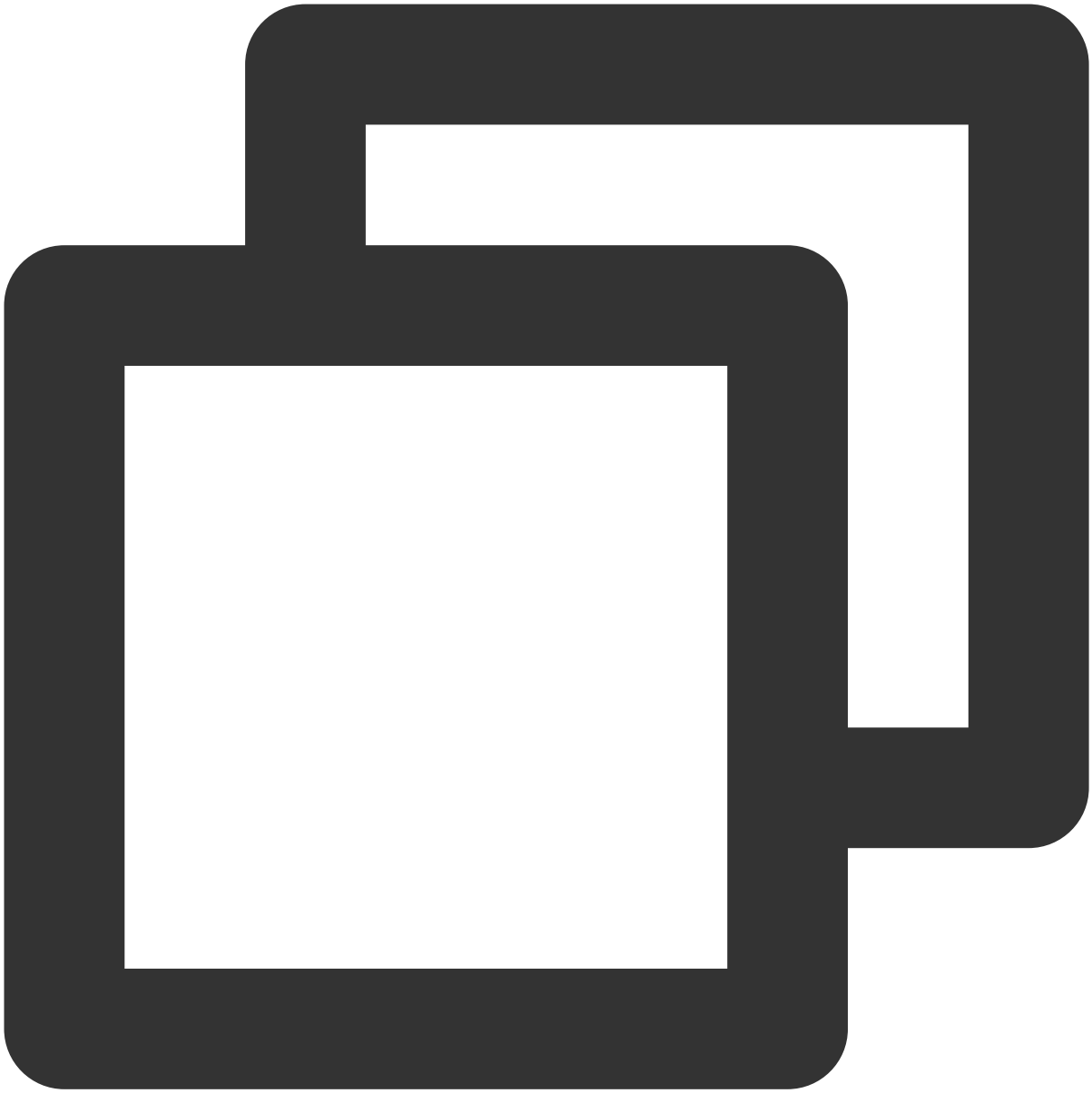
Android

请将您在厂商配置步骤中, 获取到的 iOS APNs 推送证书，上传至 IM 控制台。

IM 控制台会为您分配一个证书 ID，见下图：



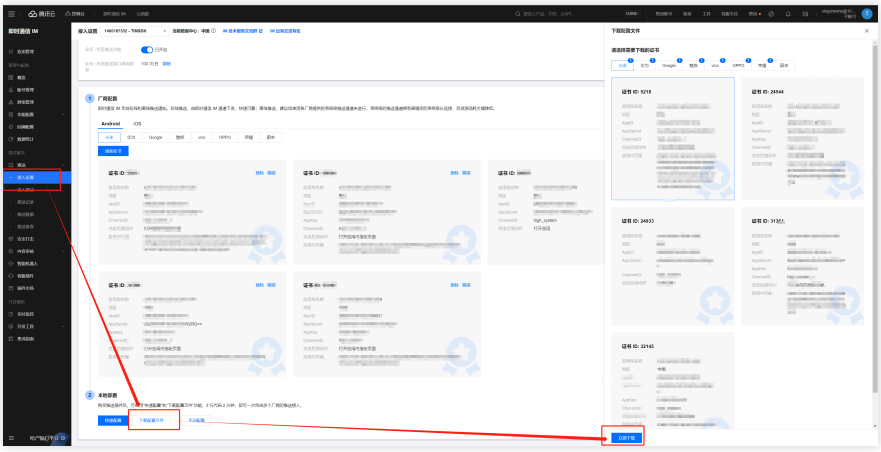
在您应用的启动后尽可能早的位置，调用 `TimPushPlugin.getInstance().setApnsCertificateID` 方法，将此证书 ID 传入。

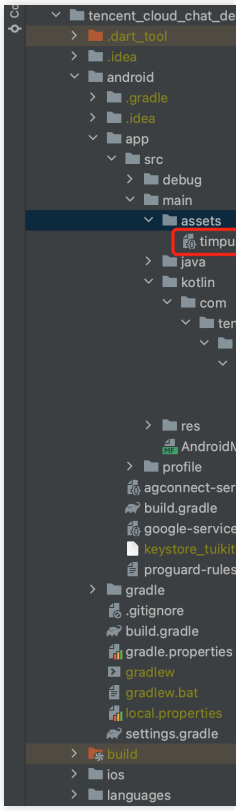


```
import { TimPushPlugin } from 'react-native-tim-push';
const certificateID = '证书ID';
TimPushPlugin.getInstance().setApnsCertificateID(certificateID);
```

完成控制台厂商推送信息填写后，下载并添加配置文件到工程。将下载的 timpush-configs.json 文件添加到项目的 `android/app/src/main/assets` 目录下，如果该目录不存在，请手动创建。

1.选择下载配置文件 timpush-configs.json	2.添加到工程





步骤3: 客户端代码配置

本操作步骤中，您将需要编写若干原生代码，例如：Swift, Java 等。

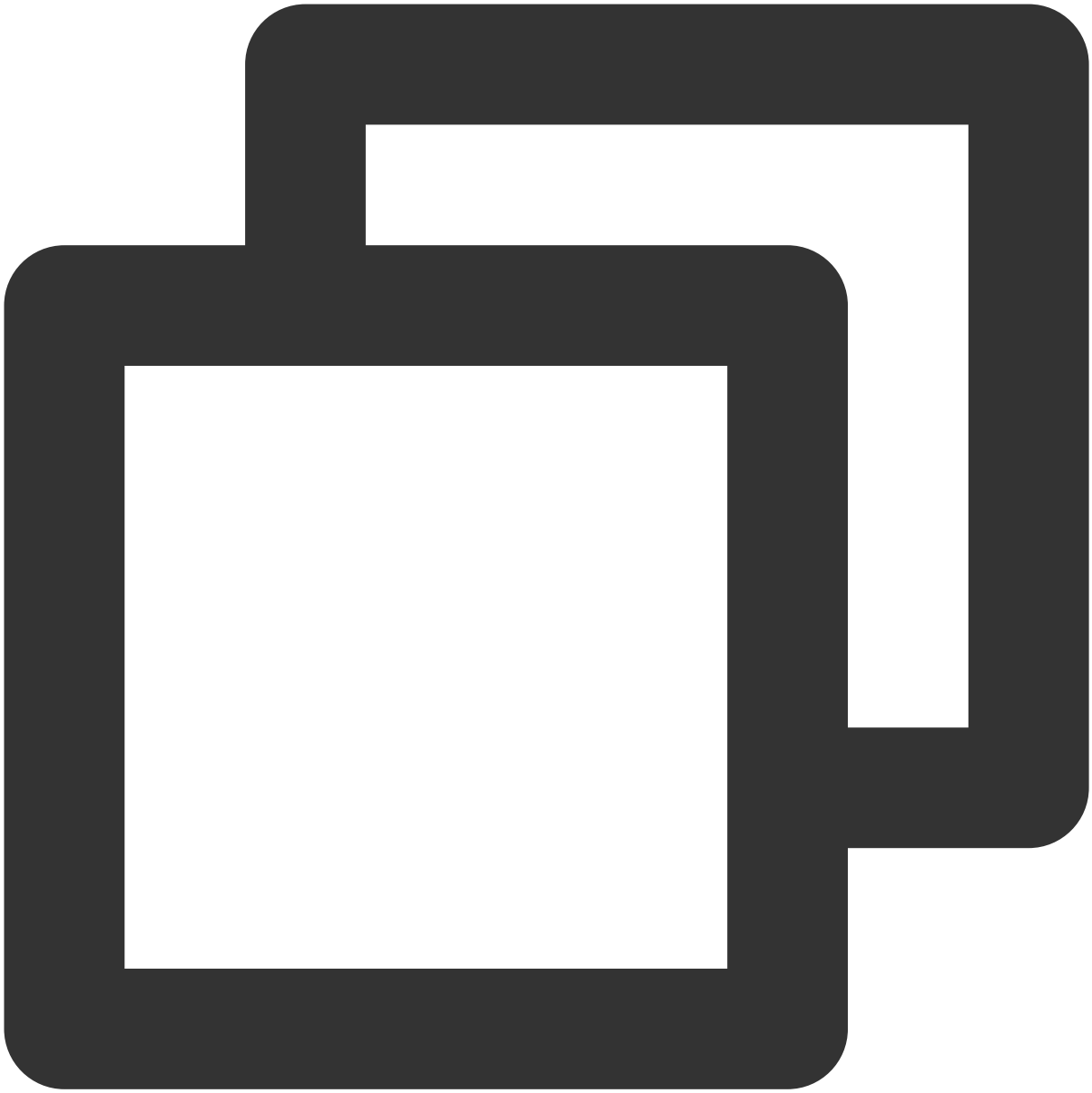
但请不要担心，您可以直接根据说明，复制我们提供的代码到指定文件即可。

iOS

Android

您可使用 Xcode 编辑，也可直接在 Visual Studio Code 或 Android Studio 中编辑。

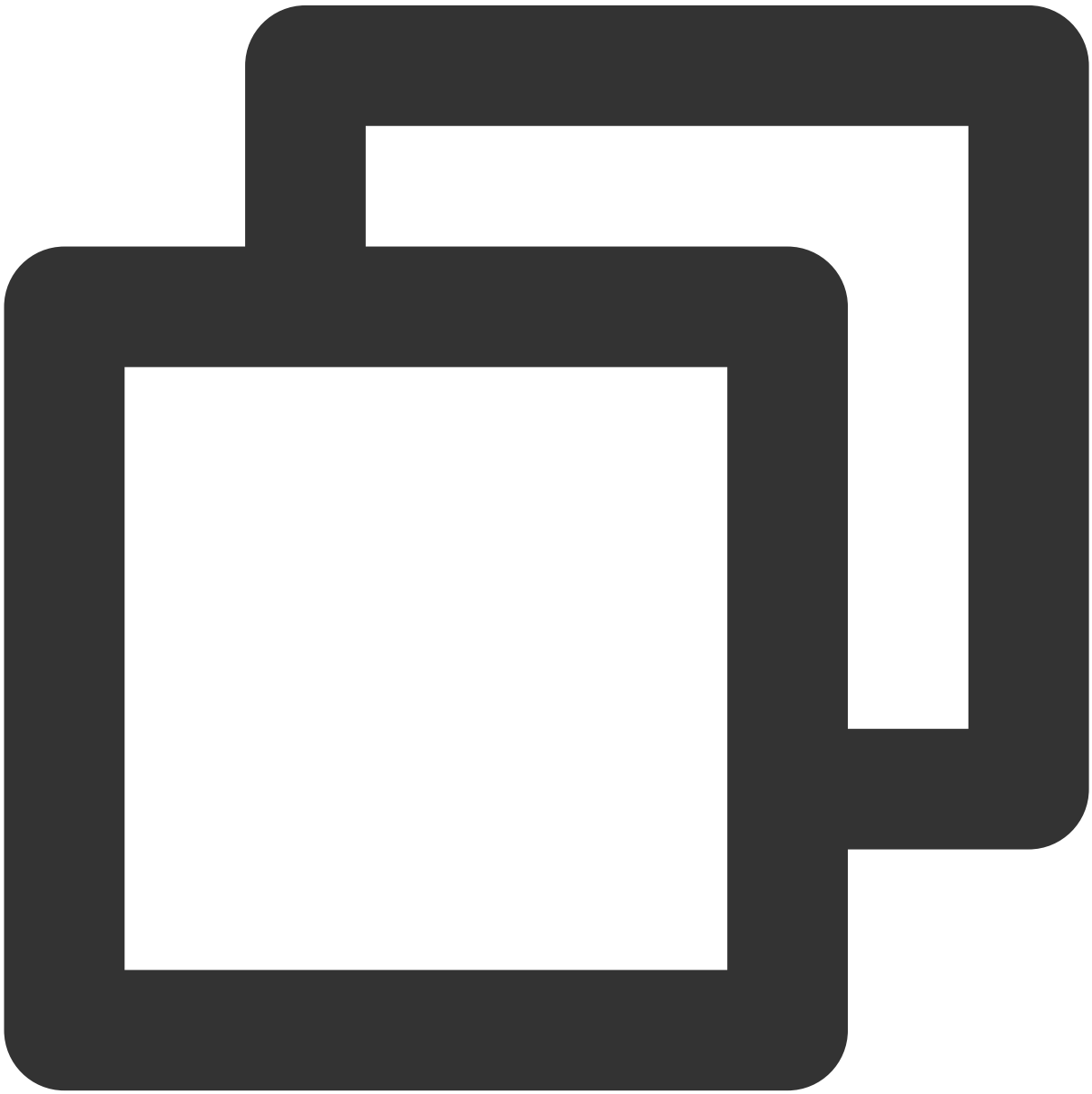
第一步：在 ios 目录下创建 `TencentIMPush.swift` 文件并将以下的代码复制进去：



```
import Foundation
import react_native_tim_push


@objc class TencentImPush: NSObject{


  @objc func getOfflinePushCertificatedID() -> Int32 {
    return TencentCloudPushModal.shared.offlinePushCertificateID();
  }
```

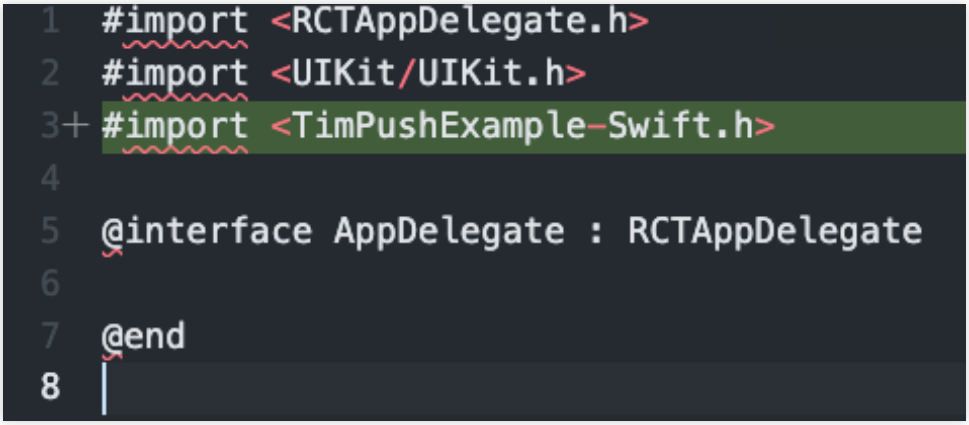
```
    @objc func getApplicationGroupID() -> String {
      return TencentCloudPushModal.shared.applicationGroupID();
    }

    @objc func onRemoteNotificationReceived(_ notice: String?) -> Void {
      TencentCloudPushModal.shared.onRemoteNotificationReceived(notice);
    }
}
```

第二步: 找到 `AppDelegate.h` 文件并在该文件中添加如下代码：

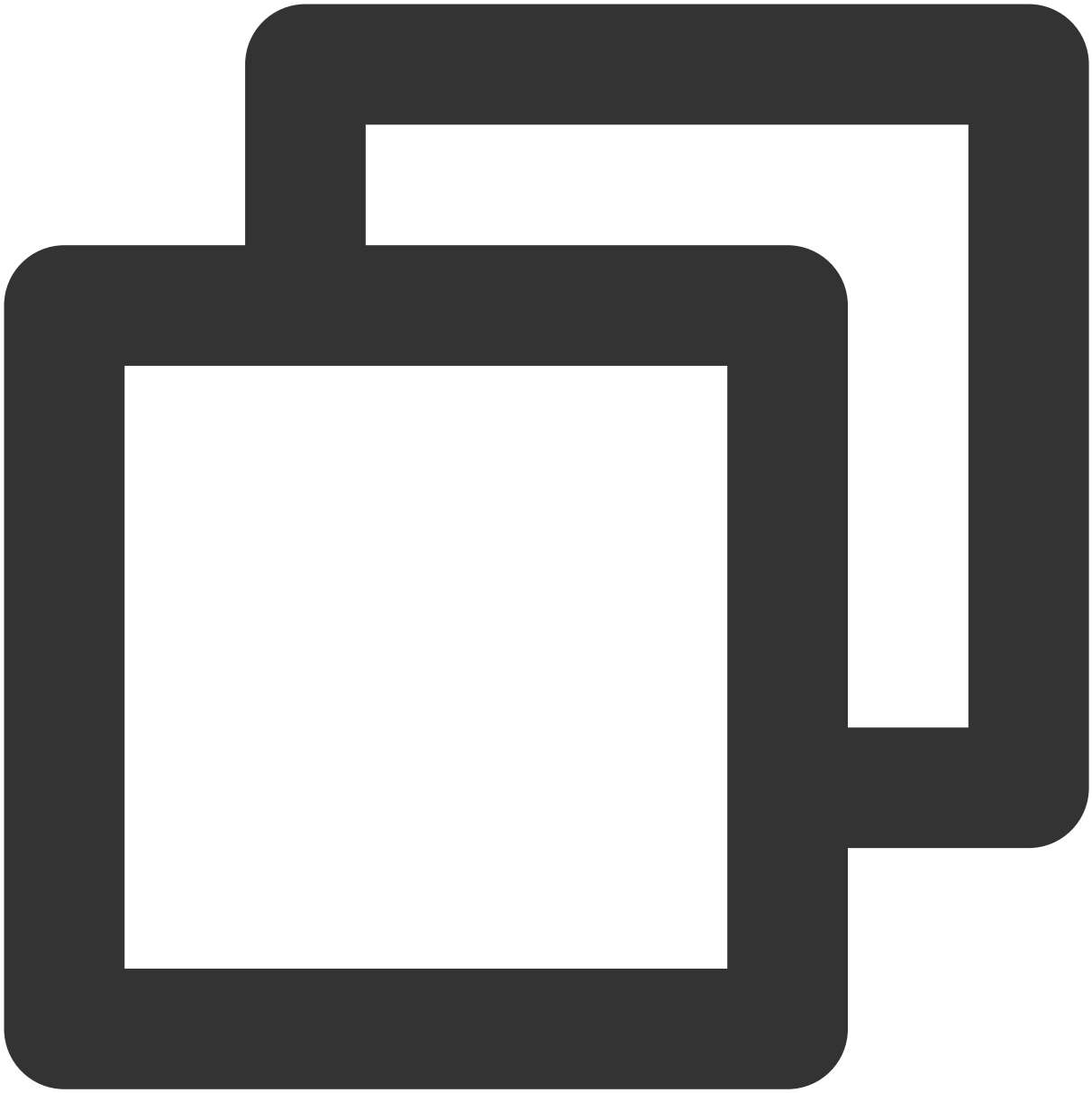


```
...
#import <Your-Project-Name-Swift.h>
// My project Name is `TimPushExample`. So it should be `#import <TimPushExample-Sw
...
```

添加完代码如下:



第三步: 找到 `AppDelegate.mm` 文件并在该文件中添加如下代码：



```
- (int)offlinePushCertificateID {
    TencentImPush *instance = [[TencentImPush alloc] init];
    return [instance getOfflinePushCertificatedID];
}


- (NSString *)applicationGroupID {
    TencentImPush *instance = [[TencentImPush alloc] init];
    return [instance getApplicationGroupID];
}
```

```
- (BOOL)onRemoteNotificationReceived:(NSString *)notice {
    TencentImPush *instance = [[TencentImPush alloc] init];
    [instance onRemoteNotificationReceived:notice];
    return YES;
}
```

添加完代码如下：

```
#import "AppDelegate.h"

#import <React/RCTBundleURLProvider.h>

@implementation AppDelegate

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(N
{
  self.moduleName = @"TimPushExample";
  // You can add your custom initial props in the dictionary below.
  // They will be passed down to the ViewController used by React Native.
  self.initialProps = @{};

  return [super application:application didFinishLaunchingWithOptions:launchOptio
}

- (NSURL *)sourceURLForBridge:(RCTBridge *)bridge
{
  return [self getBundleURL];
}

- (int)offlinePushCertificateID {
    TencentImPush *instance = [[TencentImPush alloc] init];
    return [instance getOfflinePushCertificatedID];
}

- (NSString *)applicationGroupID {
    TencentImPush *instance = [[TencentImPush alloc] init];
    return [instance getApplicationGroupID];
}

- (BOOL)onRemoteNotificationReceived:(NSString *)notice {
    TencentImPush *instance = [[TencentImPush alloc] init];
    [instance onRemoteNotificationReceived:notice];
    return YES;
}
```

```
- (NSURL *)getBundleURL
{
#if DEBUG
  return [[RCTBundleURLProvider sharedSettings] jsBundleURLForBundleRoot:@"index"
#else
  return [[NSBundle mainBundle] URLForResource:@"main" withExtension:@"jsbundle"]
#endif
}

@end
```

在 `MainApplication.java` 中引入 `com.timpush.RNTencentIMPushApplication`，并将 `Application` 替换成 `RNTencentIMPushApplication`。如下图所示：

```
import com.facebook.react.flipper.ReactNativeFlipper
import com.facebook.soloader.SoLoader
import com.timpush.RNTencentIMPushApplication

class MainApplication : Application(), ReactApplication {
class MainApplication : RNTencentIMPushApplication(), ReactApplication {

  override val reactNativeHost: ReactNativeHost =
      object : DefaultReactNativeHost(this) {
        override fun getPackages(): List<ReactPackage> =
            PackageList(this).packages.apply {
              // Packages that cannot be autolinked yet can be added mar
              // add(MyReactNativePackage())
            }
```

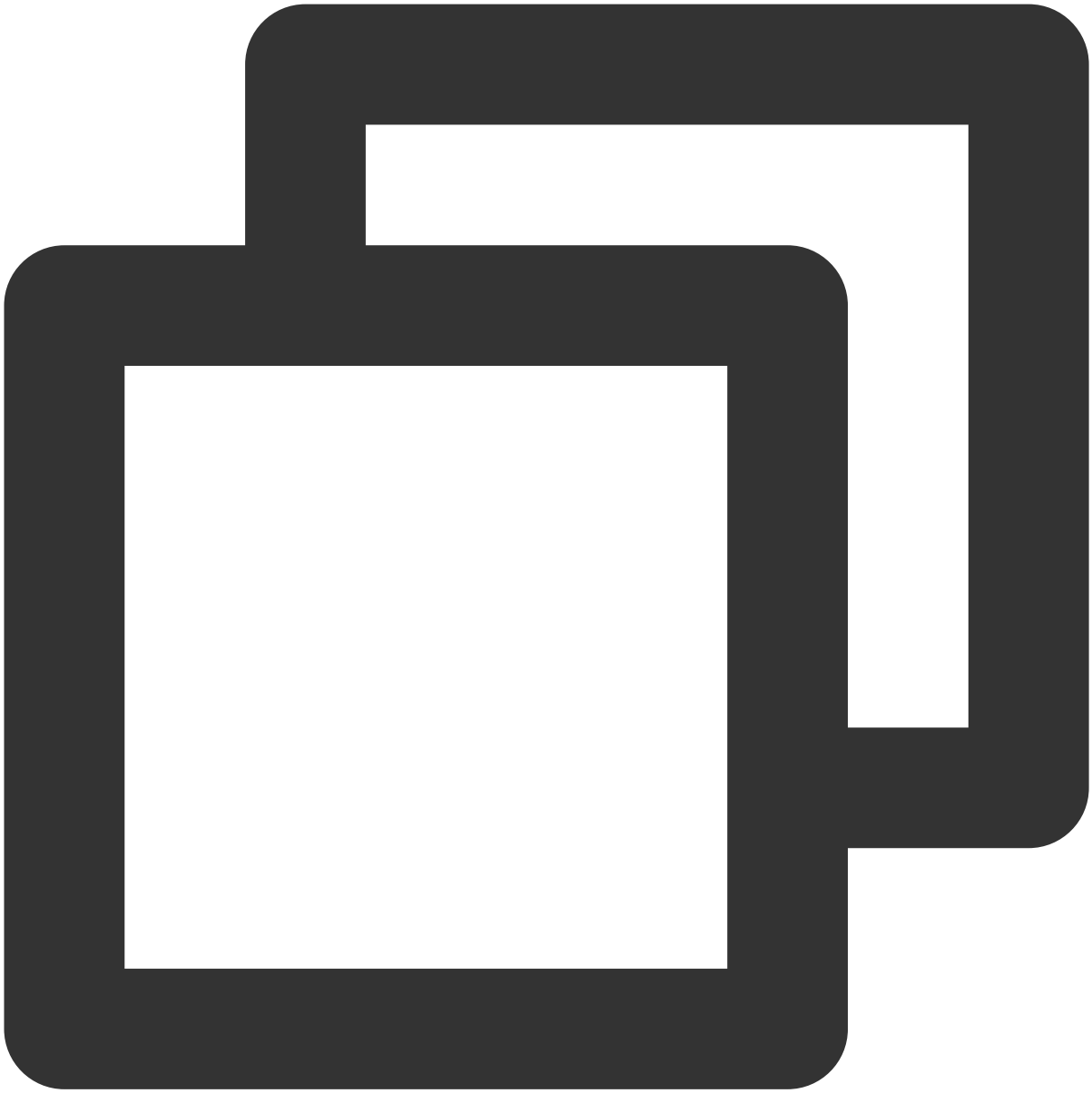
步骤4: 客户端厂商配置

iOS

Android

iOS 端无需进行此步骤。

打开 android/app/build.gradle 文件，在最后，新增 dependencies 配置，并根据需要，引入下列全部或部分厂商的推送包。只有引入对应厂商的推送包，才能启用该厂商的原生推送能力。



```
dependencies {
      // Huawei
      implementation 'com.tencent.timpush:huawei:${推送插件的版本号}'

      // XiaoMi
      implementation 'com.tencent.timpush:xiaomi:${推送插件的版本号}'

      // vivo
      implementation 'com.tencent.timpush:vivo:${推送插件的版本号}'

      // Honor
```

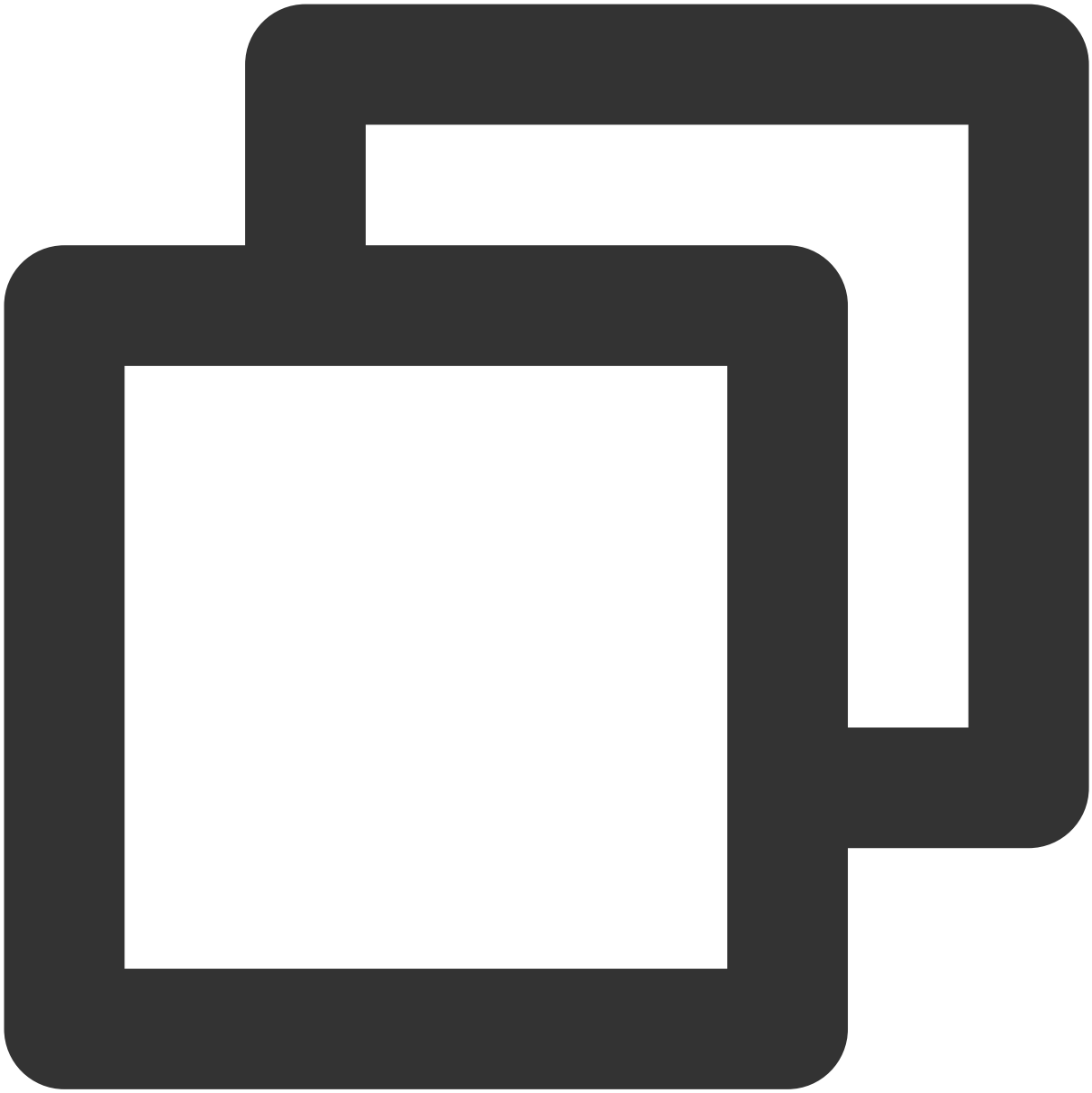
```
        implementation 'com.tencent.timpush:honor:${推送插件的版本号}'

        // Meizu
        implementation 'com.tencent.timpush:meizu:${推送插件的版本号}'

        // Google Firebase Cloud Messaging (Google FCM)
        implementation 'com.tencent.timpush:fcm:${推送插件的版本号}'

        //OPPO 以下二选一
        //中国区域选择集成此包
        implementation 'com.tencent.timpush:oppo:${推送插件的版本号}'
        //其他区域选择集成此包
        implementation 'com.tencent.timpush:oppo-intl:${推送插件的版本号}'
    }
```

vivo 和荣耀适配

根据 vivo 和荣耀厂商接入指引，需要将 APPID 和 APPKEY 添加到清单文件中。

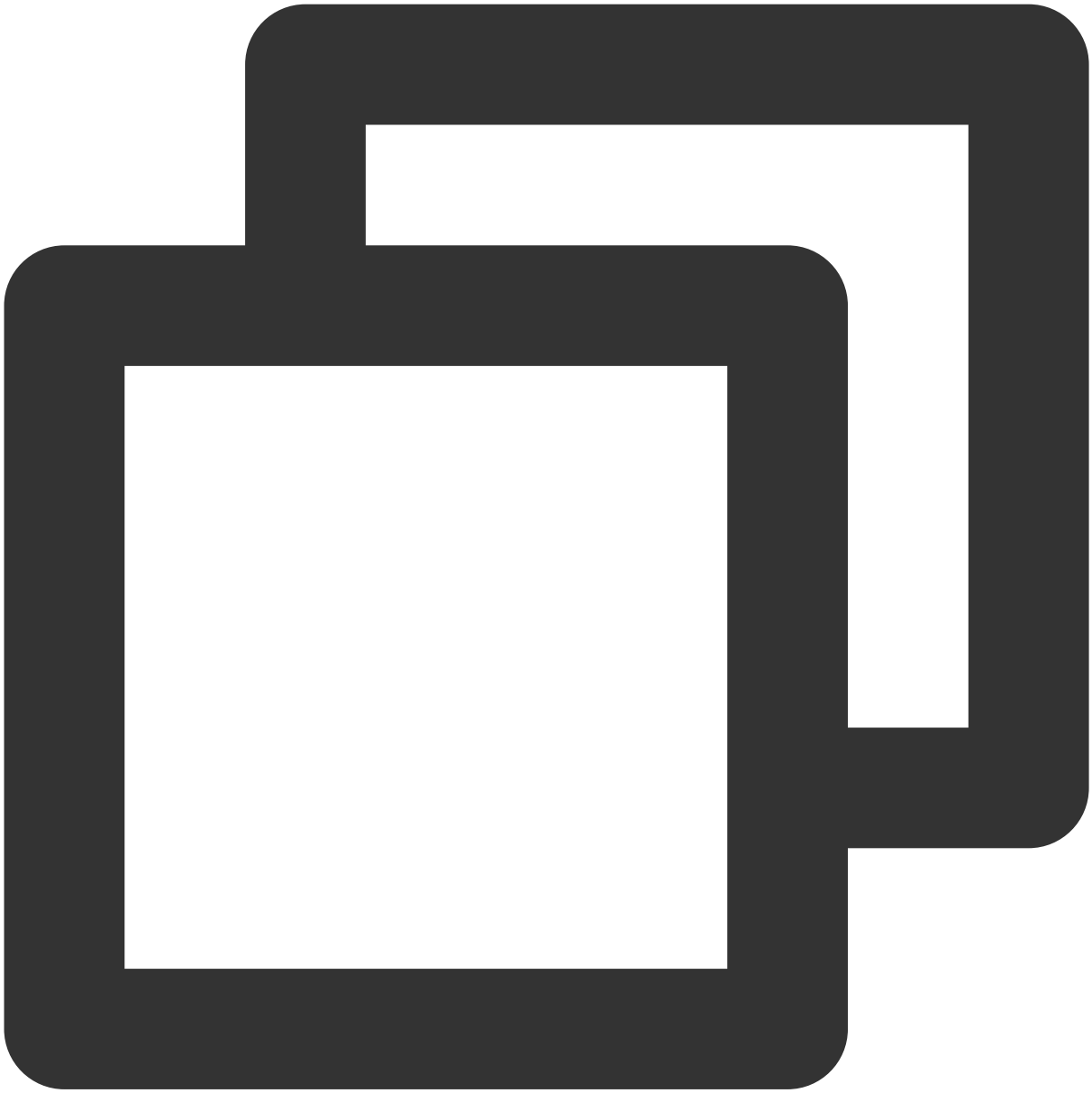
方法1

方法2



```
// android/app/build.gradle

android {
    ...
    defaultConfig {
        ...
        manifestPlaceholders = [
            "VIVO_APPKEY" : "您应用分配的证书 APPKEY",
            "VIVO_APPID" : "您应用分配的证书 APPID",
            "HONOR_APPID" : "您应用分配的证书 APPID"
        ]
```

```
    }
}
```



```
// android/app/src/main/AndroidManifest.xml

// Vivo begin
<meta-data tools:replace="android:value"
    android:name="com.vivo.push.api_key"
    android:value="您应用分配的证书 APPKEY" />
<meta-data tools:replace="android:value"
    android:name="com.vivo.push.app_id"
```

```
    android:value="您应用分配的证书 APPID" />
// Vivo end

// Honor begin
<meta-data tools:replace="android:value"
    android:name="com.hihonor.push.app_id"
    android:value="您应用分配的证书 APPID" />
// Honor end
```

华为、荣耀和 Google FCM 适配

按照厂商方法，集成对应的 plugin 和 json 配置文件。

注意：

以下荣耀的适配仅 7.7.5283 及以上版本需要配置。

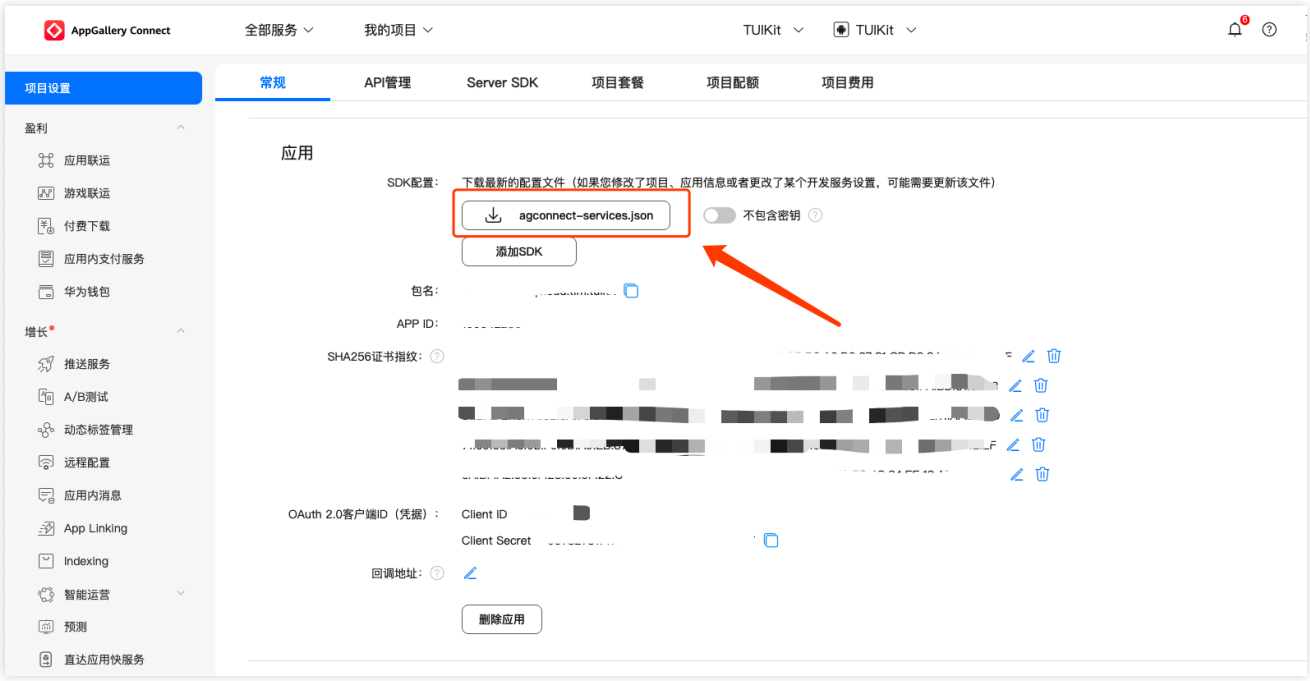
1.1 下载配置文件添加到工程根目录/Android/app。

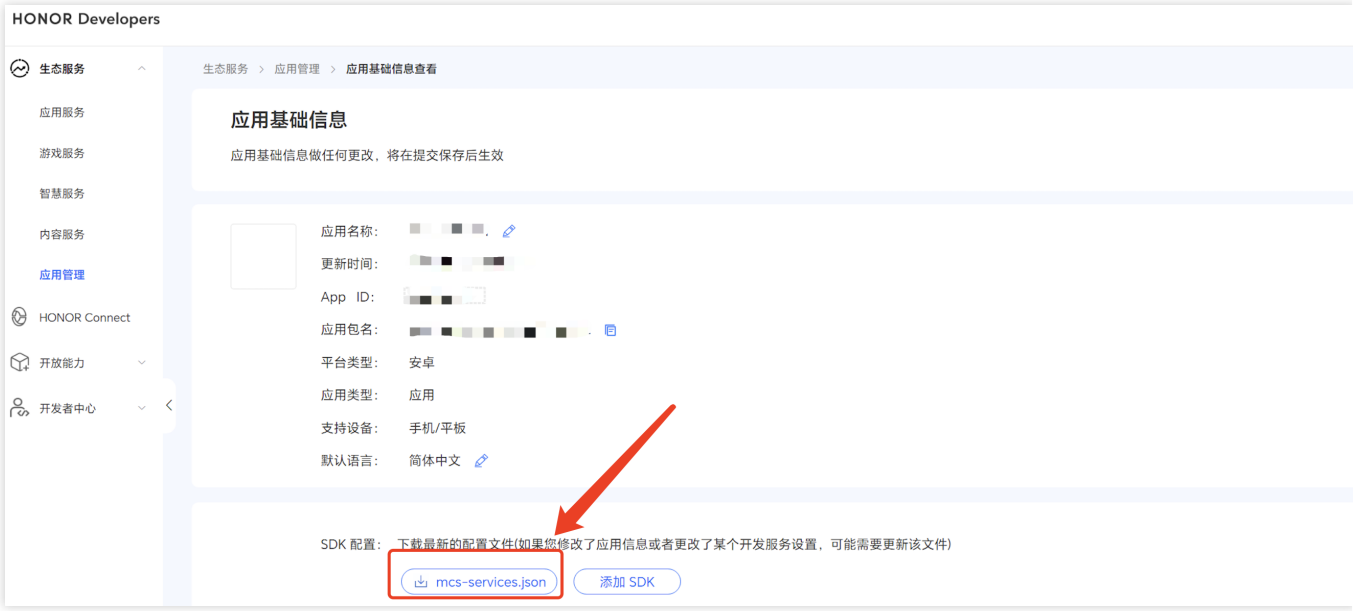
华为

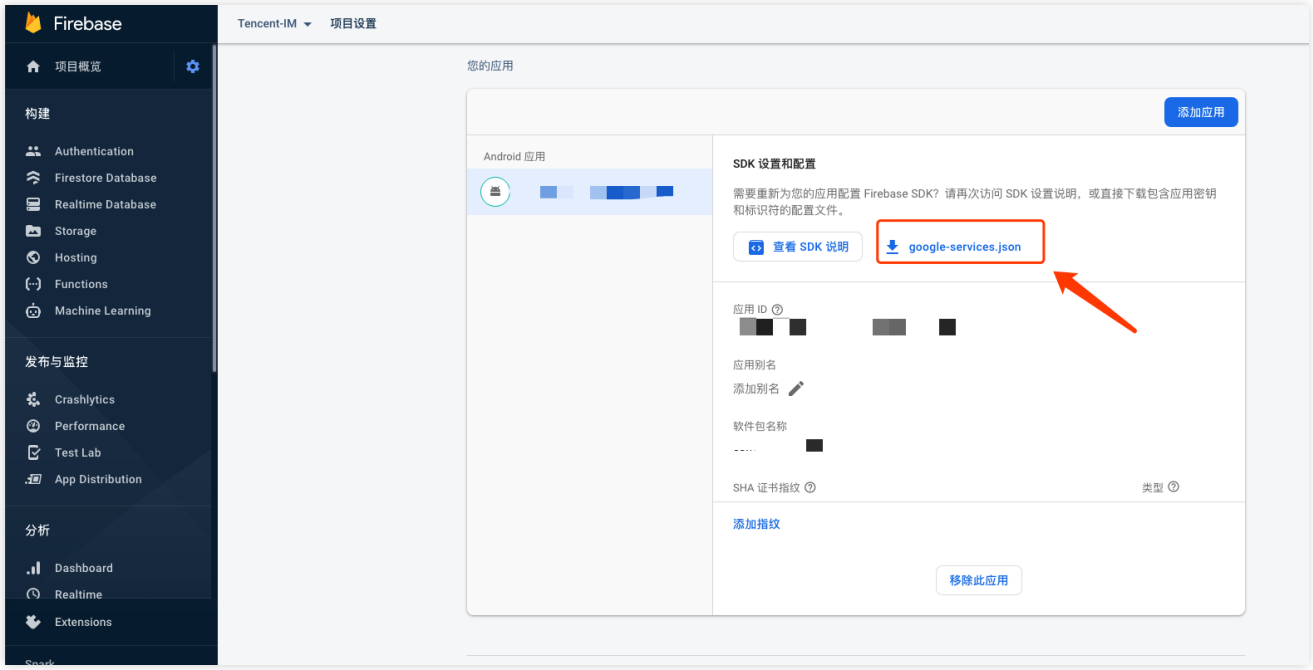
荣耀

Google FCM

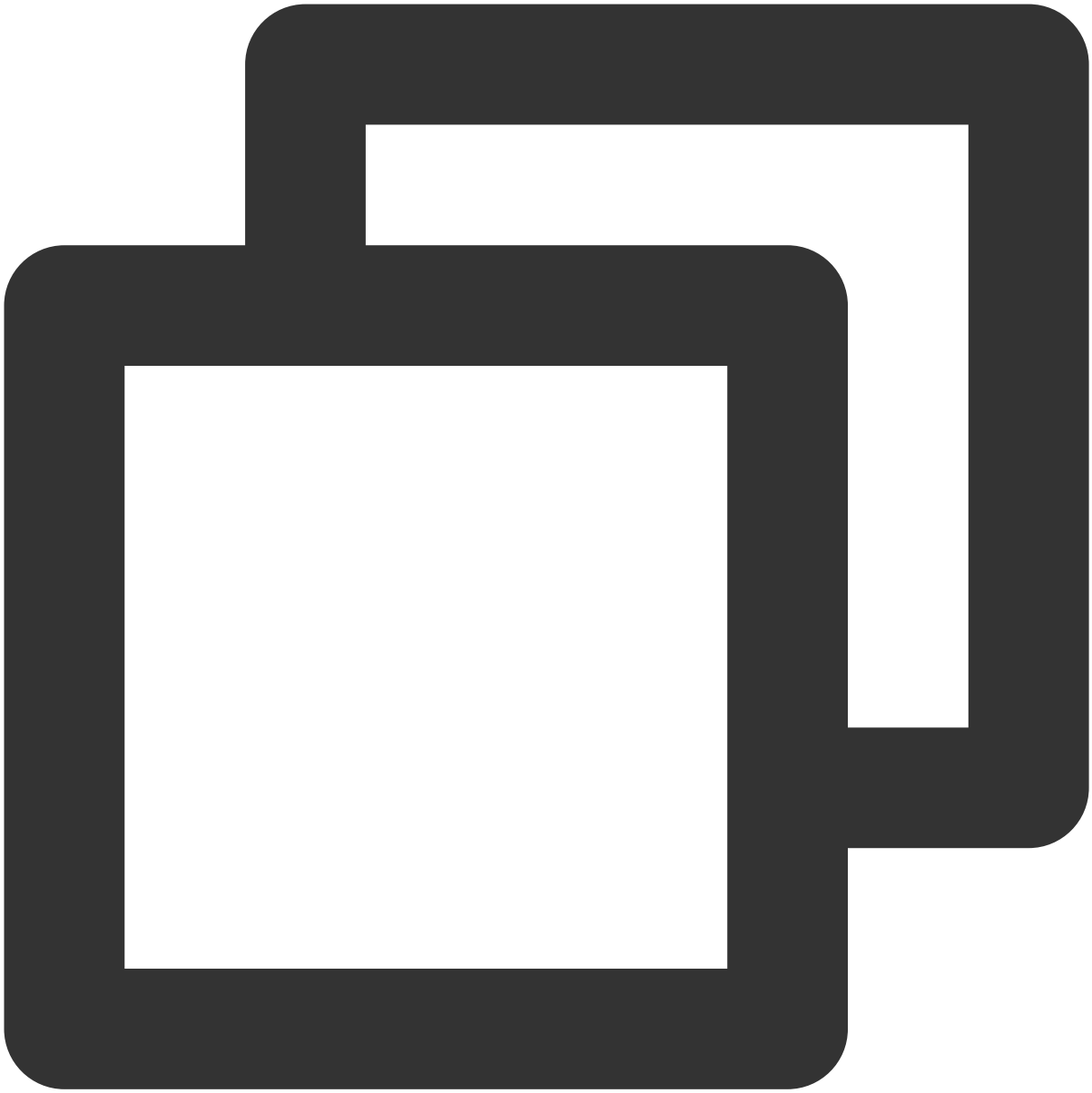
操作路径




HONOR Developers
生态服务
应用服务
游戏服务
智慧服务
内容服务
应用管理
HONOR Connect
开放能力
开发者中心
生态服务 > 应用管理 > 应用基础信息查看
应用基础信息
应用基础信息做任何更改，将在提交保存后生效
应用名称:
更新时间:
App ID:
应用包名:
平台类型: 安卓
应用类型: 应用
支持设备: 手机/平板
默认语言: 简体中文
SDK 配置: 下载最新的配置文件(如果您修改了应用信息或者更改了某个开发服务设置，可能需要更新该文件)
mcs-services.json
添加 SDK


Firebase
Tencent-IM
项目设置
项目概览
构建
Authentication
Firestore Database
Realtime Database
Storage
Hosting
Functions
Machine Learning
发布与监控
Crashlytics
Performance
Test Lab
App Distribution
分析
Dashboard
Realtime
Extensions
您的应用
添加应用
Android 应用
SDK 设置和配置
需要重新为您的应用配置 Firebase SDK？请再次访问 SDK 设置说明，或直接下载包含应用密钥和标识符的配置文件。
查看 SDK 说明
google-services.json
应用 ID
应用别名
添加别名
软件包名称
SHA 证书指纹
类型
添加指纹
移除此应用

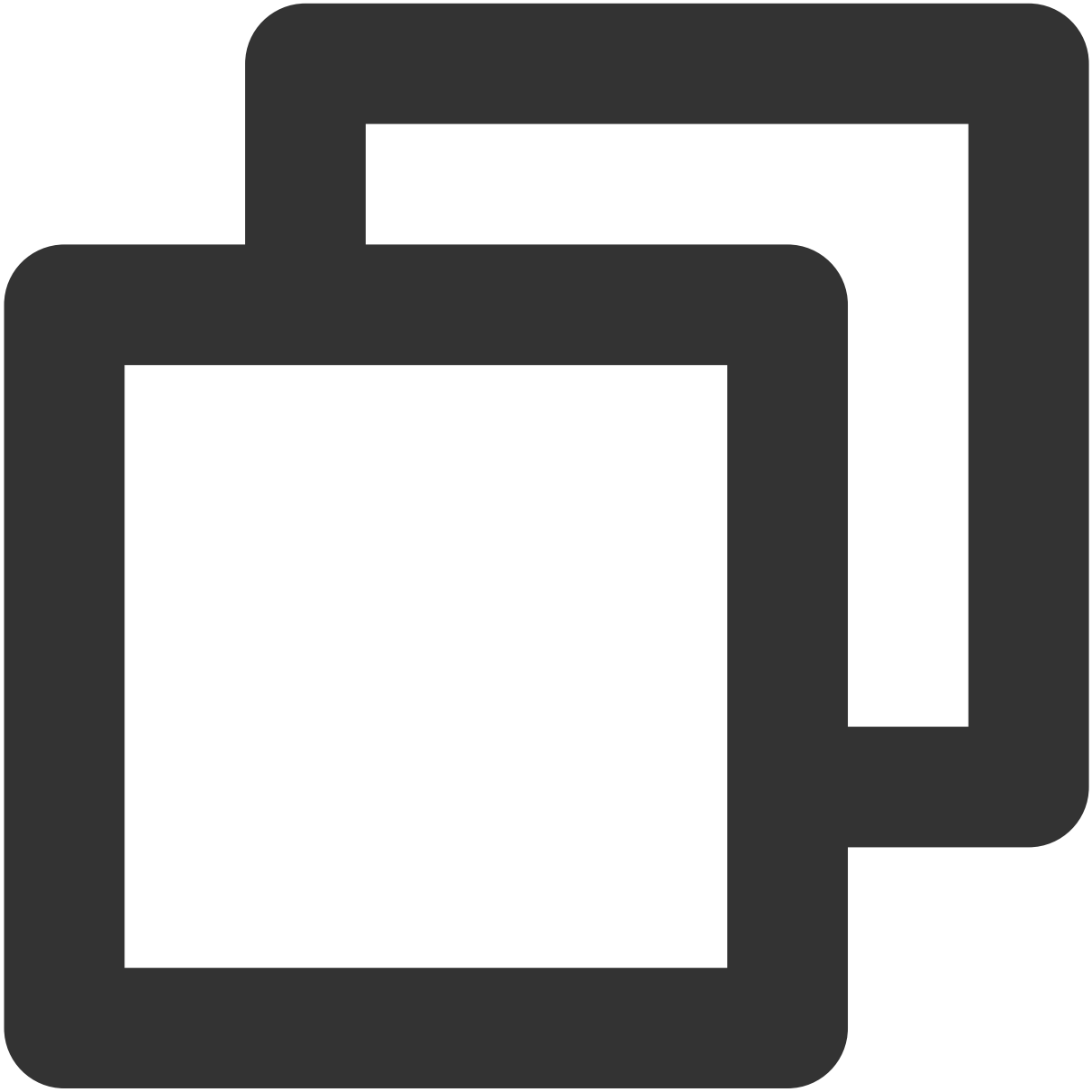
1.2 在项目级 build.gradle 文件中 buildscript -> dependencies 下添加以下配置：



```
buildscript {
 repositories {
     mavenCentral()
     maven { url "https://mirrors.tencent.com/nexus/repository/maven-public/" }
     // 配置HMS Core SDK的Maven仓地址。
     maven {url 'https://developer.huawei.com/repo/'}
     maven {url 'https://developer.hihonor.com/repo'}
 }
 dependencies {
     ...
     classpath 'com.google.gms:google-services:4.2.0'
```

```
        classpath 'com.huawei.agconnect:agcp:1.4.1.300'
        classpath 'com.hihonor.mcs:asplugin:2.0.1.300'
    }
}
```

1.3 在android/app/build.gradle 文件中添加下方配置：



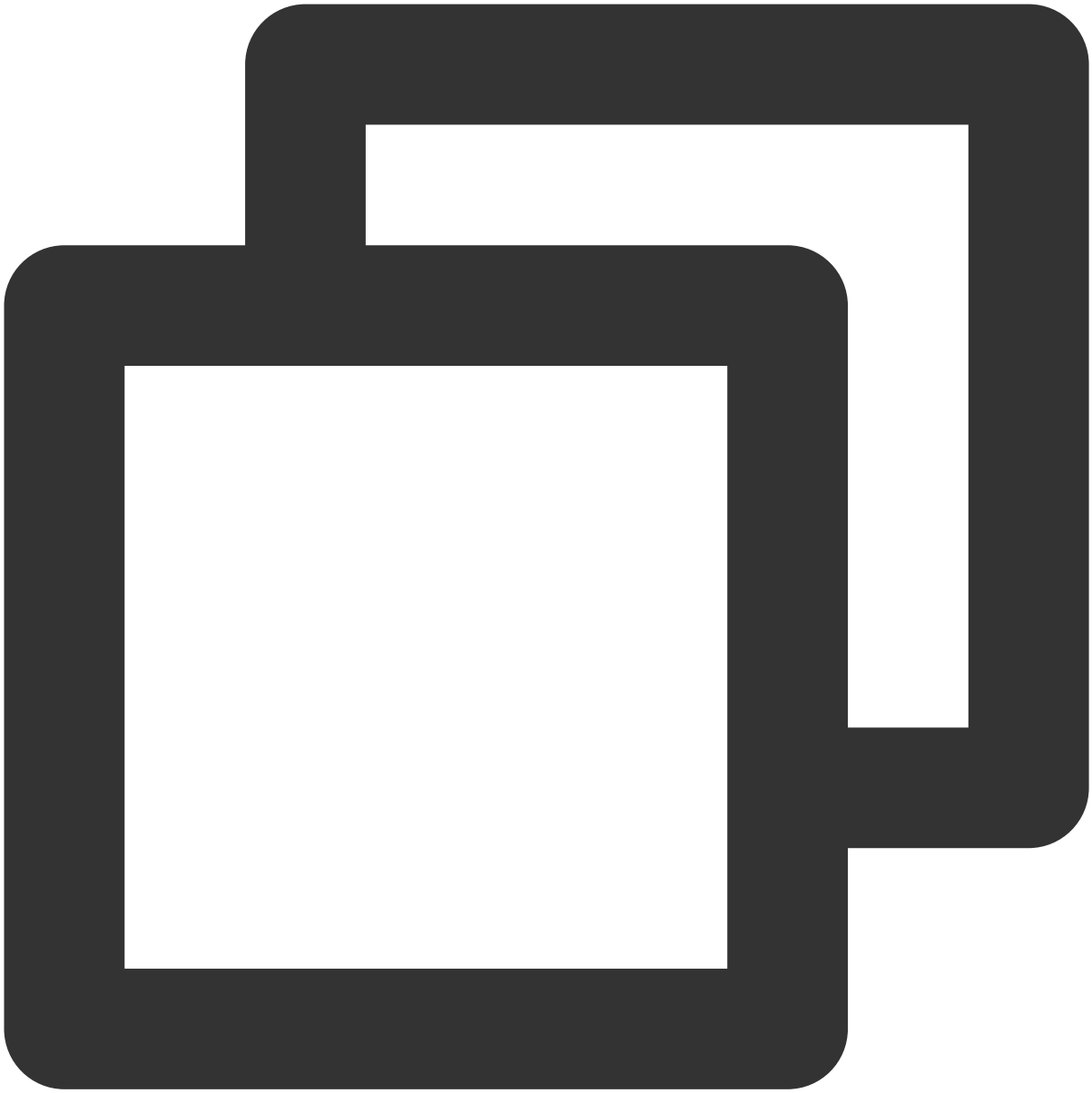
```
apply plugin: 'com.google.gms.google-services'
apply plugin: 'com.huawei.agconnect'
apply plugin: 'com.hihonor.mcs.asplugin'
```

步骤5: 处理消息点击回调, 并解析参数

请定义一个函数, 用于接受推送消息点击回调事件.
该函数请定义成 `(ext: string, userID?: string, groupID?: string): void;` 的入参形式。
其中, ext字段, 为该消息所携带的完整 ext 信息, 由发送方指定, 如果未指定, 则有默认值. 您可根据解析该字段, 跳转至对应页面。
userID 和 groupID 字段，为本插件，自动尝试解析 ext Json String， 获取里面携带的单聊对方 userID 和 群聊 groupID 信息。如果您未自定义 ext 字段，ext 字段由 SDK 默认指定，则可使用此处的默认解析。如果尝试解析失败, 则为 null 空。
您可定义一个函数来接收该回调，并据此跳转至对应会话页面或您的业务页面。

步骤6: 注册推送插件

请在 IM 登录完成后, 且在其他插件 (如 CallKit) 使用前, 立即注册推送插件。
调用 `TimPushPlugin.getInstance().registerPush` 方法, 需传入上一步定义的点击回调函数。
此外，您还可选传入 `apnsCertificateID` iOS 推送证书 ID 及 `androidPushOEMConfig` Android 推送厂商配置。此二项配置已在前序步骤指定，若无修改必要，可不再传入。

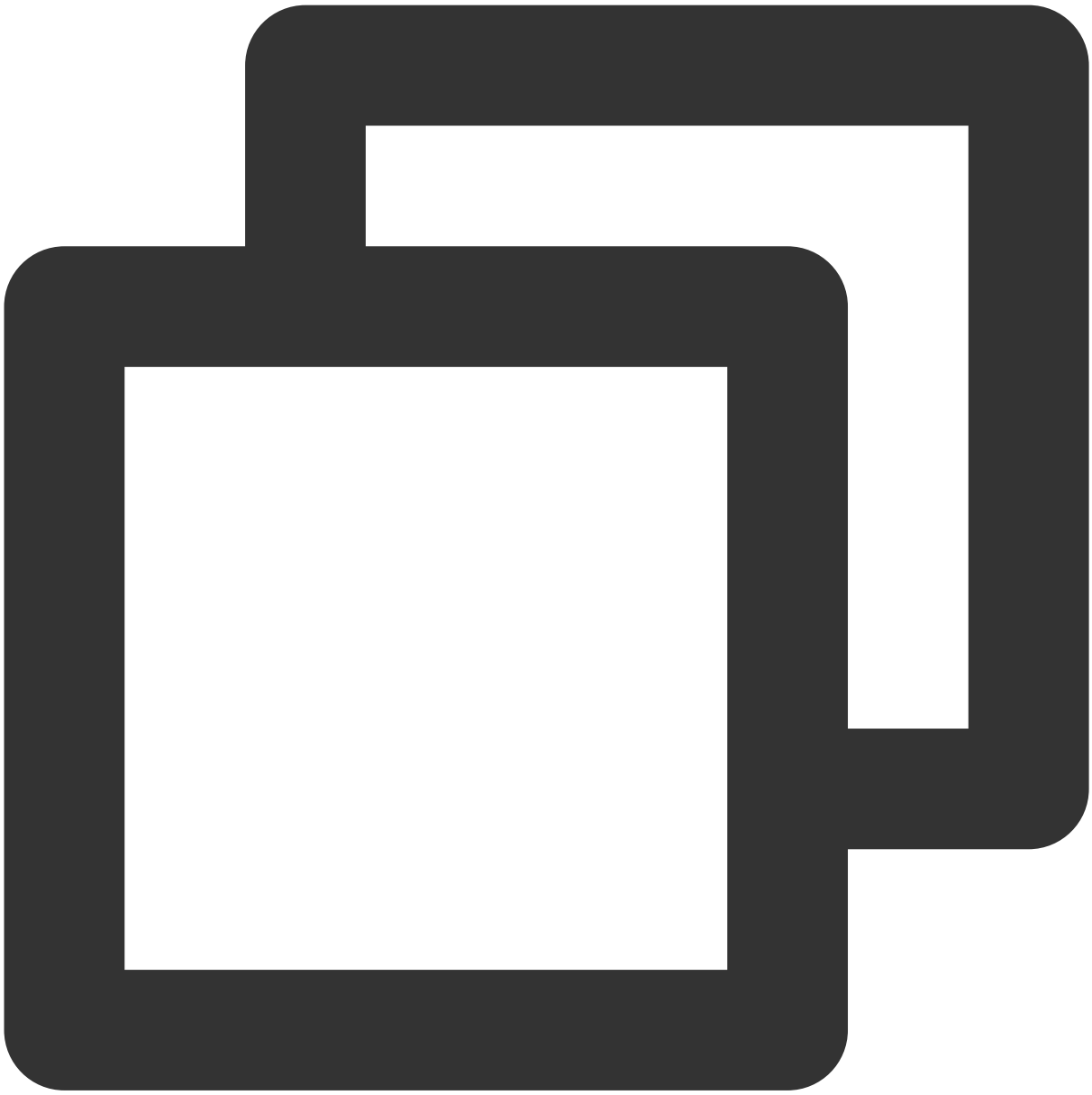


```
TimPushPlugin.getInstance().registerPush({
      onNotificationClicked: (extString) => {},
    })
```

说明：

如果您的应用需要使用**推送插件进行业务消息通知**，并且在**启动后不会立即启动并登录 IM 模块**，或者在**登录 IM 模块之前需要通过获取消息点击回调来处理业务导航**，建议您尽早调用 `TimPushPlugin.getInstance().registerOnNotificationClickedEvent` 方法，手动挂载消息单击回调，以便及时获取消息参数。

在这种场景下，您可以在调用 `TimPushPlugin.getInstance().registerPush` 之前执行此函数，并尽可能提前将其放置在代码中。



```
TimPushPlugin.getInstance().registerOnNotificationClickedEvent({onNotificationClick
```

步骤7: 消息推送触达统计

如果您需要统计触达数据，请按照如下完成配置：

华为

荣耀

vivo

魅族

IOS

回执地址： `https://api.im.qcloud.com/v3/offline_push_report/huawei`

注意：

华为推送证书 ID <= 11344 ，使用华为推送 v2 版本接口，不支持触达和点击回执，请重新生成更新证书 ID。

回执地址： `https://api.im.qcloud.com/v3/offline_push_report/honor`

回调地址配置	回执 ID 配置 IM 控制台

回执地址：`https://api.im.qcloud.com/v3/offline_push_report/vivo`

打开回执开关

配置回执地址

回执地址：`https://api.im.qcloud.com/v3/offline_push_report/meizu`

注意：

打开回执开关后，请务必确保回执地址正确配置。不配置或者配置地址错误，都会影响推送功能。

iOS 端推送触达统计配置，请参见 **统计推送抵达率**。

其余支持厂商不需要配置，FCM 暂不支持推送统计功能。

恭喜您已经完成了推送插件的接入，需要提醒您：推送插件**试用或购买到期后，将自动停止提供推送服务（包括普通消息离线推送、全员/标签推送等服务）**。为避免影响您业务正常使用，请提前 购买/续费。

数据统计

最近更新时间：2024-06-13 10:39:26

本文旨在介绍推送统计数据的各个统计页面，方便用户快速查询统计数据。

注意：

数据统计功能只会记录最后一个登录设备的推送数据详情，不支持多端登录场景。

普通推送

推送记录

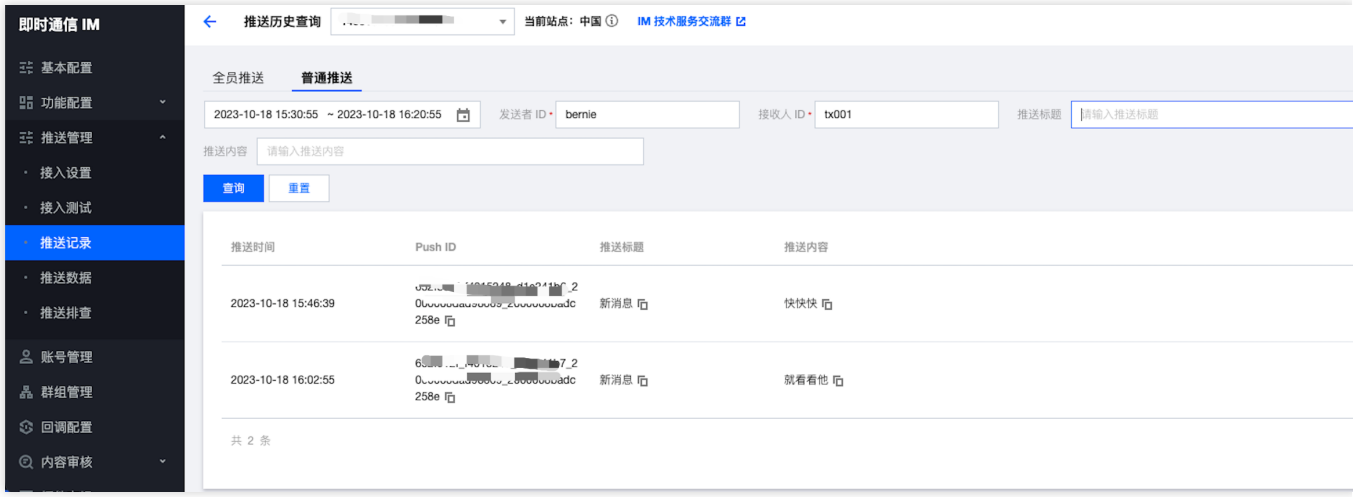
可以查询该应用下用户的所有推送记录数据，需要指定查询的时间窗口、发送者 ID 和接受者 ID，查询用户推送数据包括推送时间、Push ID、推送标题和推送内容；并支持按照推送标题或者推送内容搜索定位具体的推送记录。

查询条件

时间窗口：指定日期内某个时间段，最大一个小时，精确到分钟秒，必选。
发送者 ID：必填。
接受者 ID：必填。
推送标题：选填。
推送内容：选填。

查询结果（近7天内记录）

推送时间：具体推送触达的时间。
Push ID：推送消息的唯一 ID，可用来在排查工具中定位该条推送的全推送链路情况。
推送标题：推送显示标题。
推送内容：推送显示内容。



推送数据

统计展示了应用近日的各类推送指标数据，主要包括昨日推送总览、多类型时间区间的推送数据转化漏斗、支持按照厂商维度分类查看、以及每天的推送指标数据详情。

昨日推送总览指标数据

包括可发送数量、发送数量、触达数量、点击数量、实发率、触达率、点击率。

时间区间

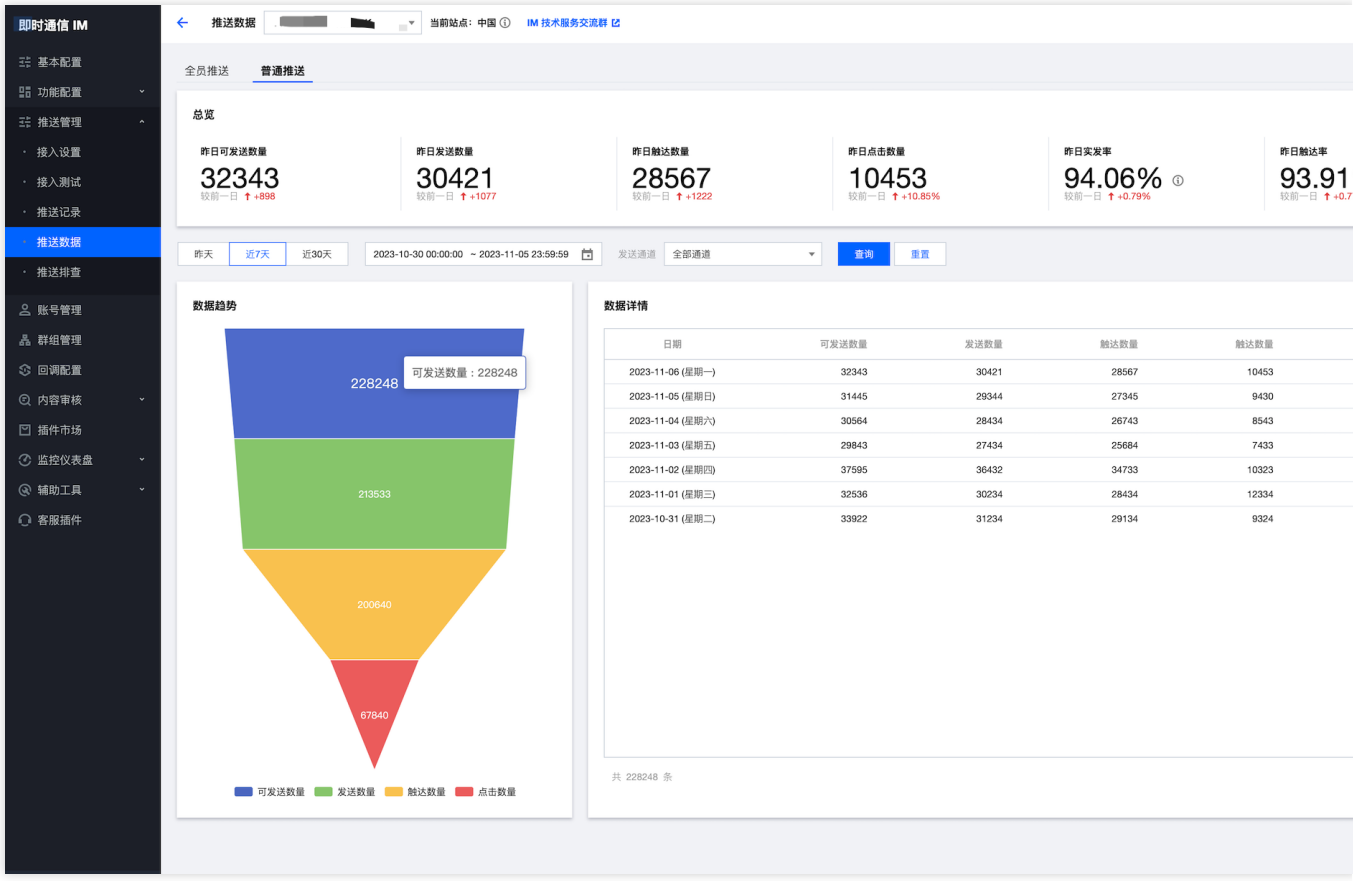
支持昨日、近7天、近30天以及指定时间窗口类型。

推送数据转化漏斗

包括可发送数量、发送数量、触达数量、点击数量维度。

支持按照厂商维度分类查看

支持所有厂商分类查看。



全员/标签推送

推送记录

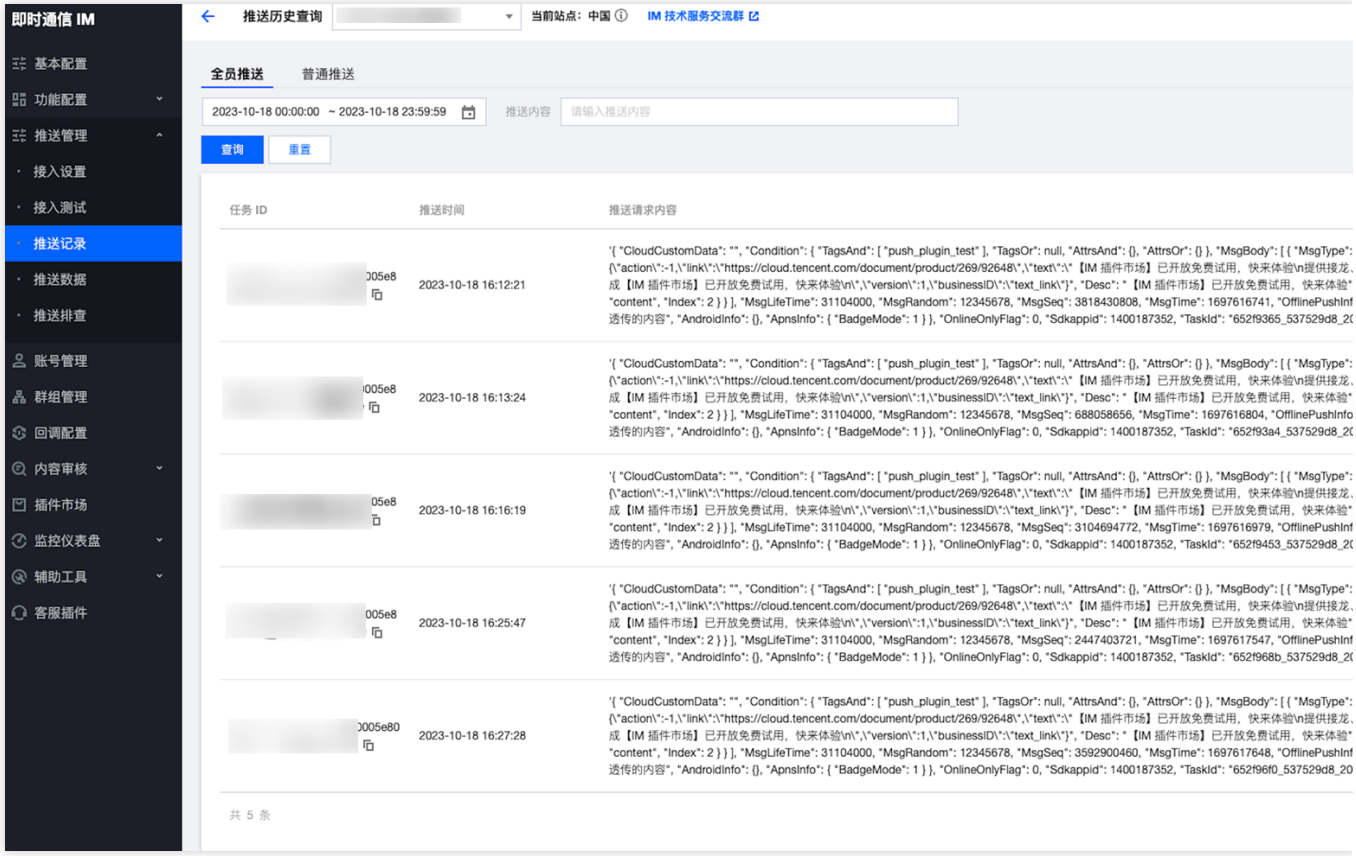
可以查询该应用下向用户发送的所有全员/标签推送记录数据，需要指定查询的时间窗口，查询用户推送数据包括推送时间、任务 ID、推送请求内容；并支持按照推送内容搜索定位具体的推送记录。

查询条件

时间窗口： 指定日期内某个时间段，最大 7 天，精确到分钟秒，必选。
推送内容：选填。

查询结果（近7天内记录）

推送时间：具体推送触达的时间。
任务 ID：推送消息的唯一 ID，可用来在排查工具中定位该条推送的全推送链路情况。
推送内容：推送的 json 格式详细数据。



推送数据

统计展示了应用近日的各类推送指标数据，功能页面和指标数据含义同普通推送。

昨日推送总览指标数据

包括可发送数量、发送数量、触达数量、点击数量、实发率、触达率、点击率。

时间区间

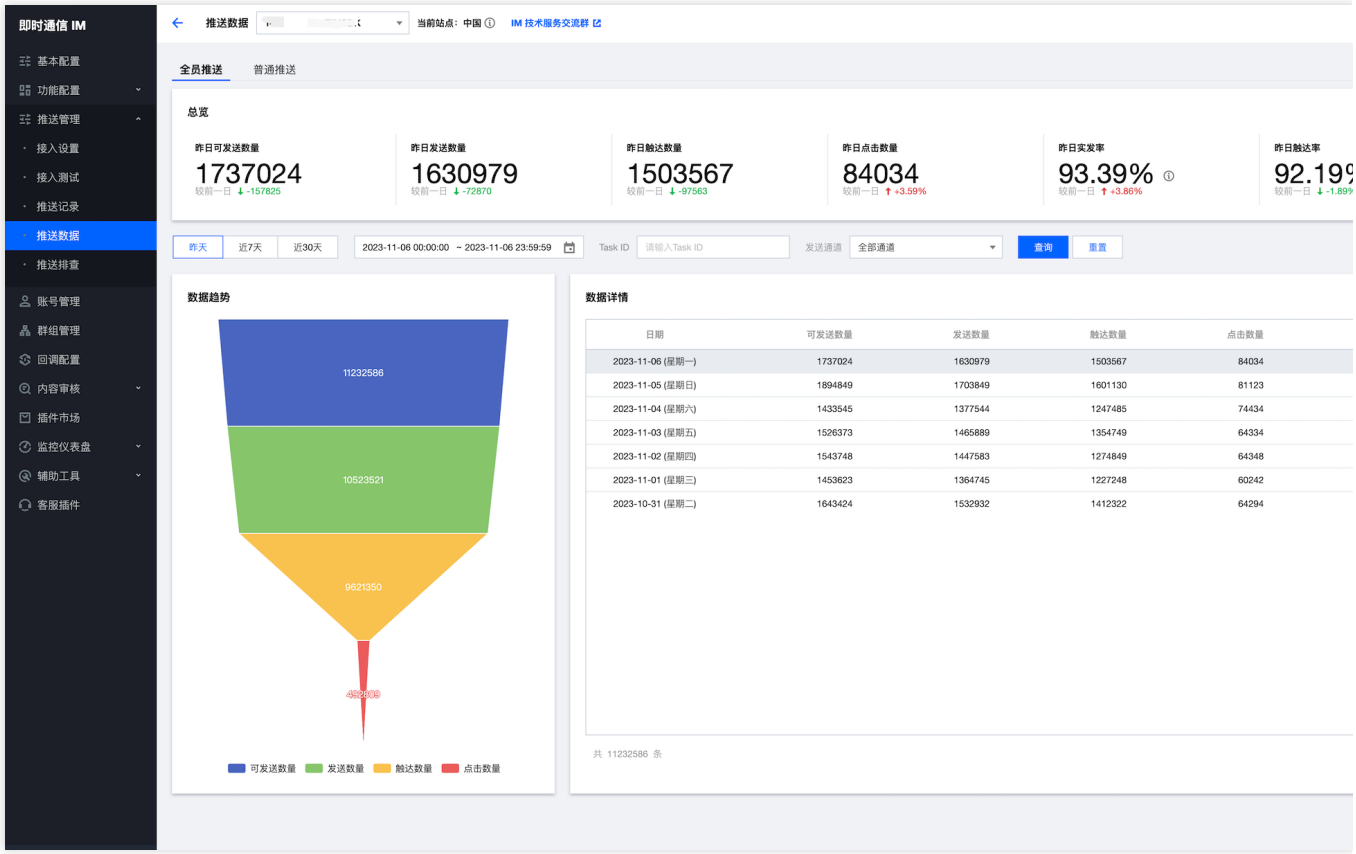
支持昨日、近7天、近30天以及指定时间窗口类型。

推送数据转化漏斗

包括可发送数量、发送数量、触达数量、点击数量维度。

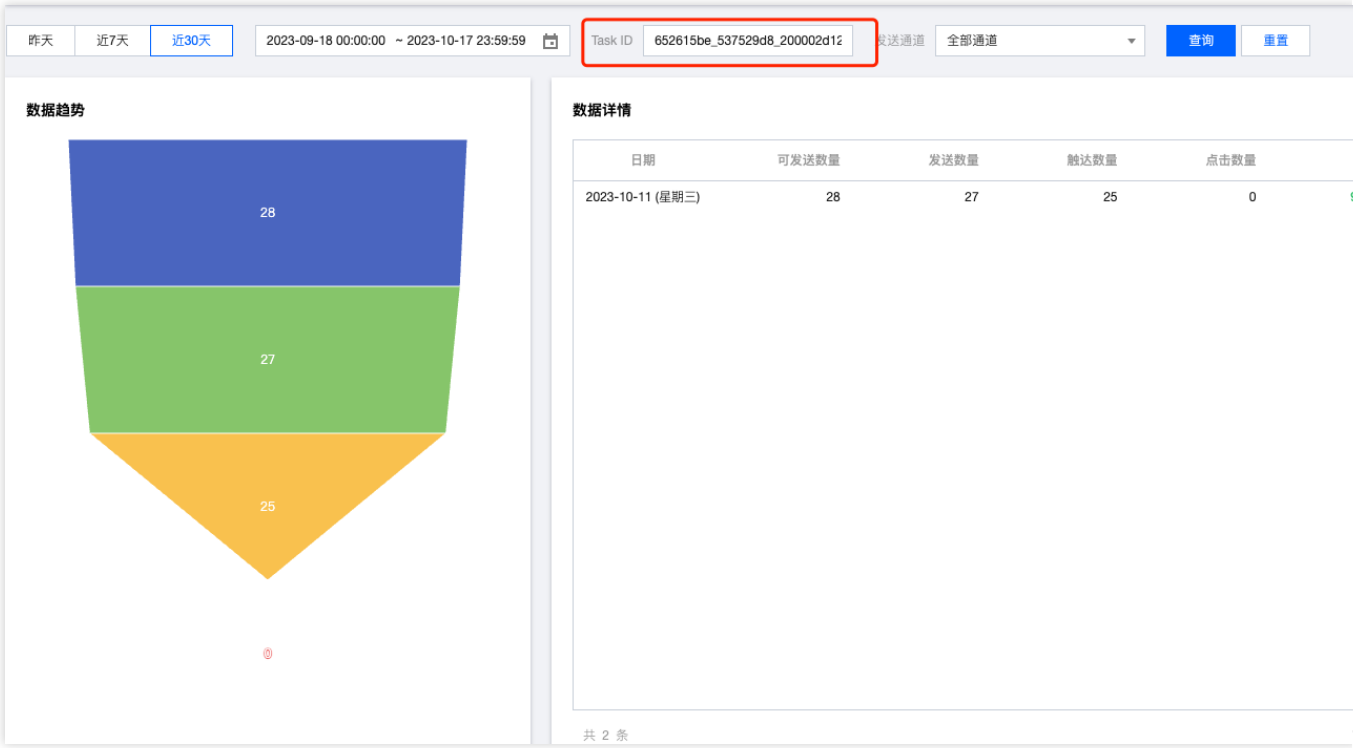
支持按照厂商维度分类查看

支持所有厂商分类查看。



Task ID

全员/标签推送支持按照 任务 ID 查看单个推送的详细指标数据和详情。



指标数据计算方法

可发送数量：对推送任务所选定的目标人群筛选，得到的可下发的有效设备数量之和（当设备在线并且切后台时，将同时发送在线推送和离线推送，此时将被计作两个设备，不去重）。

发送数量：可发送的有效设备中，已经成功通过即时通信IM通道、厂商通道下发的有效设备数量之和（当设备在线并且切后台时，将同时发送在线推送和离线推送，此时将被计作两个设备，不去重）。

触达数量：通过即时通信IM通道、厂商通道下发，设备终端成功收到的回执数量之和（当设备在线并且切后台时，将同时发送在线推送和离线推送，此时将被计作两个设备，不去重）。

点击数量：推送成功展示后，点击推送的回执数量之和。

实发率：发送数量/ 可发送数量 * 100 %。

触达率：触达数量/ 发送数量 * 100 %。

点击率：点击数量 / 触达数量 * 100 %。

推送统计各厂商支持情况

厂商	触达	点击
华为	支持（需配置回执）	支持（需集成 TIMPush）

荣耀	支持（需配置回执）	支持（需集成 TIMPush）
vivo	支持（需配置回执）	支持（需集成 TIMPush）
OPPO	支持	支持（需集成 TIMPush）
小米	支持	支持（需集成 TIMPush）
魅族	支持（需配置回执）	支持（需集成 TIMPush）
FCM	不支持	不支持
Apns	支持（需集成 TIMPush）	支持（需集成 TIMPush）

排查工具

最近更新时间：2024-06-13 10:39:26

本文旨在介绍推送排查工具的各个页面和功能用法，引导用户在离线推送消息没有收到时，可以通过该工具来全链路排查推送详情。

普通推送

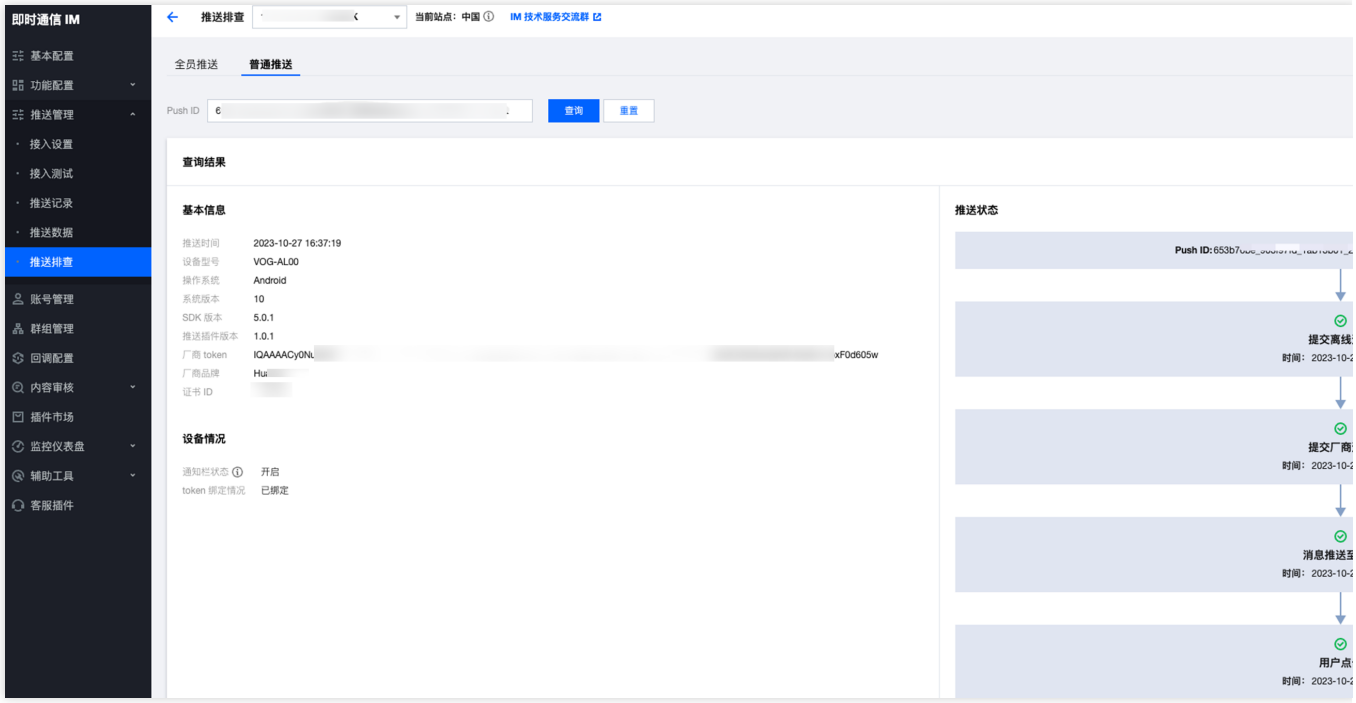
普通推送排查工具主要面向开发用户用来排查具体一条推送收不到问题。通过在推送记录中找到该条推送，复制本条推送的唯一 Push ID，来查询该条推送的详细推送链路详情。

查询字段

Push ID：用来标识普通推送的唯一 ID，可以在推送记录中查询得到，必填。

查询结果

基本信息：该条推送下发时，当前设备的基本信息包括型号、操作系统、SDK和插件版本号等。
设备情况：该条推送下发时，通知栏开关状态，以及设备的 token 绑定状态。
推送状态：该条推送下发的全链路信息，包含 IM 服务器 > 厂商服务器 > 终端设备 > 用户点击的整个链路情况。



全员/标签推送

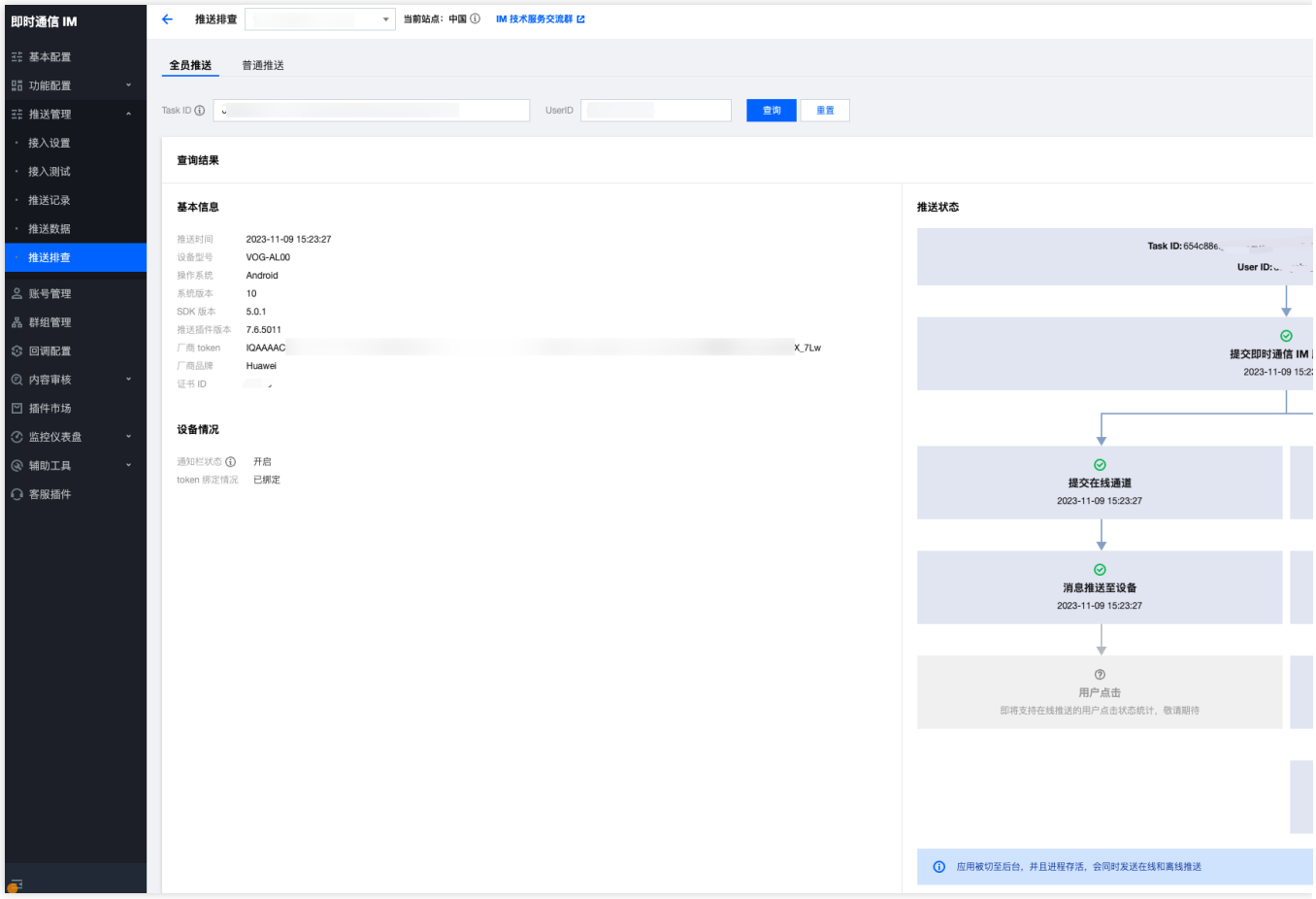
全员/标签推送排查工具主要面向管理员用户用来排查一次全员/标签推送的某个用户的接收情况。通过在推送记录中找到该条推送，复制本条推送的唯一任务 ID，输入接收方的 UserID，来查询该条推送到该 UserID 用户的详细推送链路详情。

查询字段

TaskID：用来标识一条全员/标签推送的唯一 ID，可以在推送记录中查询得到，必填。
UserID：全员/标签推送下发的一个用户 ID，及推送的一个接收方，必填。

查询结果

基本信息：该条推送下发时，指定 UserID 用户当前设备的基本信息包括型号、操作系统、SDK和插件版本号等。
设备情况：该条推送下发时，指定 UserID 用户设备的通知栏开关状态，以及设备的 token 绑定状态。
推送状态：该条推送针对指定 UserID 用户下发的全链路信息，包含 IM 服务器 > 厂商服务器 > 终端设备 > 用户点击的整个链路情况。


即时通信 IM
基本配置
功能配置
推送管理
接入设置
接入测试
推送记录
推送数据
推送排查
账号管理
群组管理
回调配置
内容审核
插件市场
监控仪表盘
辅助工具
客服插件
推送排查
当前站点：中国
IM 技术服务交流群
全员推送
普通推送
Task ID
UserID
查询
重置
查询结果
基本信息
推送时间 2023-11-09 15:23:27
设备型号 VOG-AL00
操作系统 Android
系统版本 10
SDK 版本 5.0.1
推送插件版本 7.6.5011
厂商 token IQAAAAC
厂商品牌 Huawei
证书 ID
设备情况
通知栏状态 开启
token 绑定情况 已绑定
推送状态
提交即时通信 IM
提交在线通道
2023-11-09 15:23:27
消息推送至设备
2023-11-09 15:23:27
用户点击
即将支持在线推送的用户点击状态统计，敬请期待
应用被切至后台，并且进程存活，会同时发送在线和离线推送

客户端 API
Android

最近更新时间：2024-06-13 10:39:26

TIMPushManager

public abstract class TIMPushManager：推送插件接口类。

接口概览

注册/反注册推送服务接口

初始化并成功登录 IM 后，可以注册推送服务。

API	描述
registerPush	注册推送服务，推送信息读取工程中的配置文件 timpush-configs.json。
registerPush	注册推送服务，推送信息来自接口参数 json。
unRegisterPush	反注册关闭离线推送服务，IM 账号登出前调用。
disableAutoRegisterPush	关闭插件在登录后自动注册推送服务，需要在注册推送服务之前调用。

FCM 自定义铃音配置接口

配置打开后，自定义铃音生效，发送方发送消息的离线信息中需要带上该 channelId。

API	描述
configFCMPrivateRing	配置 FCM 的自定义铃音，需要在注册推送服务之前调用。
setCustomTIMPushConfigs	自定义替换插件默认读取的注册推送配置文件 timpush-configs.json，需要在注册推送服务之前调用

推送通道特殊配置接口

API	描述
setPushChannel	指定设备离线推送使用的厂商通道类型，需要在注册推送服务之前调

	用。
getPushChannel	获取设备离线推送正在使用的厂商通道类型。
checkPushStatus	各个厂商接入配置完成后，可使用该接口在对应厂商设备上测试可 push 状态。

接口详情

静态 Public 成员函数

static TIMPushManager getInstance()：获取 TIMPushManager 管理器实例。

成员函数说明

abstract void registerPush(Context context, TIMPushCallback callback)

注册离线推送服务，IM 账号登录成功时调用。（为了方便您尽可能简单地接入推送服务，插件会默认自动读取工程中的配置文件 timpush-configs.json，来获取注册推送服务需要的信息）

注意：

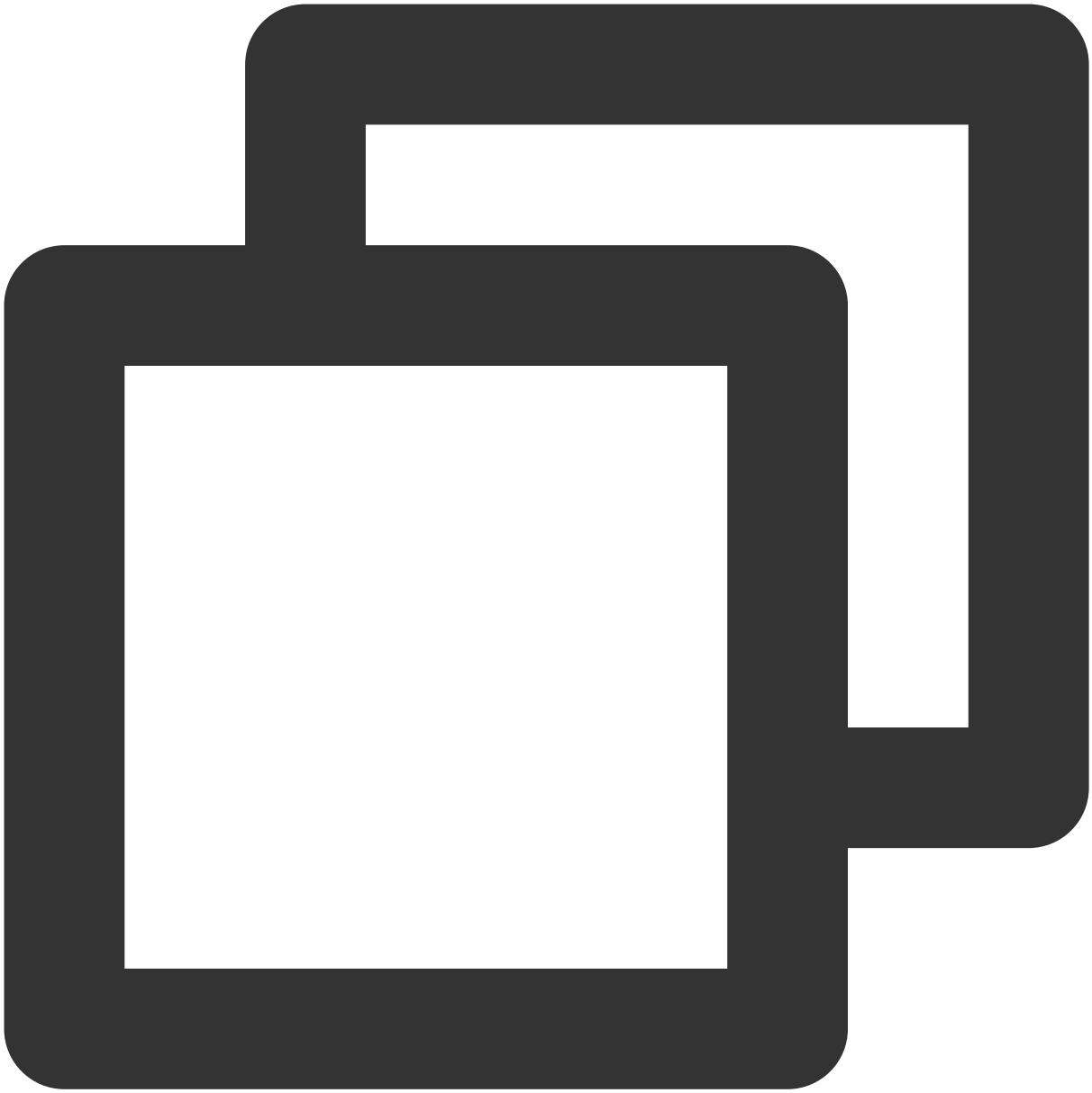
您需要使用 TUICore 组件中的 TUILogin 提供的 login 接口登录，插件会自动感知并注册推送服务。

如果您不想使用 TUILogin 提供的接口，您需要在完成登录操作后，手动调用该接口注册服务。

abstract void registerPush(String json, Context context, TIMPushCallback callback)

注册离线推送服务，IM 账号登录成功时调用。（注册推送服务需要的信息来自接口参数 json）

参数 json：



```
{
    // huawei
    "huaweiPushBussinessId": "",
    // 在腾讯云控制台上传第三方推送证书后分配的证书ID
    "huaweiBadgeClassName": "", // 角标参数，默认为应用的 launcher 界面的类名
    // xiaomi
    "xiaomiPushBussinessId": "",// 在腾讯云控制台上传第三方推送证书后分配的证书ID
    "xiaomiPushAppId": "",// 小米开放平台分配的应用APPID "xiaomiPushAppKey": "",// 小

    // meizu
    "meizuPushBussinessId": "", // 在腾讯云控制台上传第三方推送证书后分配的证书ID
```

```
        "meizuPushAppId": "",// 魅族开放平台分配的应用APPID "meizuPushAppKey": "",// 魅族

        // vivo
        "vivoPushBussinessId": "", // 在腾讯云控制台上传第三方推送证书后分配的证书ID

        // google
        "fcmPushBussinessId": "", // 在腾讯云控制台上传第三方推送证书后分配的证书ID

        // oppo
        "oppoPushBussinessId": "", // 在腾讯云控制台上传第三方推送证书后分配的证书ID
        "oppoPushAppKey": "",// oppo开放平台分配的应用 AppKey
        "oppoPushAppSecret": "",// oppo开放平台分配的应用 AppSecret

        // honor
        "honorPushBussinessId": "", // 在腾讯云控制台上传第三方推送证书后分配的证书ID
    }
```

abstract void unRegisterPush(TIMPushCallback callback)

反注册关闭离线推送服务，IM 账号登出前调用。

注意：

如果您不想使用推送服务，手动调用该接口反注册服务即可。

如果您使用 TUICore 组件中的 TUILogin 提供的 logout 接口登出，插件会自动感知并反注册推送服务。

abstract void disableAutoRegisterPush()

关闭插件自动注册推送服务，需要在登录之前调用。

注意：

如果您使用 TUICore 组件中的 TUILogin 提供的 login 接口登录，插件默认自动注册推送服务，调用该接口可关闭自动注册。

abstract void configFCMPrivateRing(String channelId, String ringName, boolean enable)

配置 FCM 的自定义铃音，需要在注册推送服务之前调用。

注意：

配置打开后，自定义铃音生效，发送方发送消息的离线信息中需要带上该 channelId。

参数说明：

API	描述
channelId	FCM 通道自定义通知栏的 channel ID，应用内唯一。
ringName	FCM 通道自定义通知栏的推送铃音名称，raw 目录下铃音且不需要后缀名。
enable	设置离线推送提示铃音是否使用自定义铃音。

abstract void setCustomTIMPushConfigs(String configs)

自定义替换插件默认读取的注册推送配置文件 timpush-configs.json，需要在注册推送服务之前调用。

说明：

主要用于多环境下动态切换不同配置文件的推送注册，例如：正式环境和测试环境不同配置文件下的推送功能集成和测试；

静态编译期切换方法请参考：buildConfigField("String", "custom_timpush_configs", "\\"自定义文件名称\\"")

参数说明：

参数	描述
configs	自定义配置文件的名称，路径需保持不变："工程根目录/app/src/assets/"

abstract void setPushChannel(int channelId)

指定设备离线推送使用的厂商通道类型，需要在注册推送服务之前调用。

说明：

该接口可以指定使用厂商推送通道类型，例如在国外的小米设备指定使用 FCM 通道推送 setPushChannel(2002)。

一般不指定通道类型，组件会自动识别设备厂商类别来注册使用对应的厂商通道。

参数说明：

参数	描述	
channelId	厂商	设备类型
	XiaoMi	2000
	HuaWei	2001
	FCM	2002
	Meizu	2003
	Oppo	2004
	Vivo	2005
	Honor	2006

static TIMPushManager getPushChannel()

获取设备离线推送正在使用的厂商通道类型。

abstract void checkPushStatus(int brandId, TIMPushCallback<String> callback)

各个厂商接入配置完成后，可使用该接口在对应厂商设备上测试是否可接收推送。

参数说明：

参数	描述
callback	true 表示配置成功可推送。

iOS

最近更新时间：2024-06-13 10:39:26

接口概览

注册/反注册推送服务接口

初始化并成功登录 IM 后，可以注册推送服务。

API	描述
registerPush	注册推送服务, 在登录完成之后调用
unRegisterPush	反注册离线推送服务，IM 账号登出时调用。
disableAutoRegisterPush	关闭插件在登录后自动注册推送服务，需要在注册推送服务之前调用

统计 TIMPush 的推送抵达率

如果您需要统计推送的抵达和点击数据,您需要在 Notification Service Extension 中主动调用本函数。

API	描述
onReceiveNotificationRequest:inAppGroupID:callback:	仅支持在 Notification Service Extension 的 '-didReceiveNotificationRequest:withContentHandler:' 方法中调用； appGroup 标识当前主 App 和 Extension 之间共享的 App Group，需要在主 App 的 Capability 中配置 App Groups 能力。

接口详情

函数说明

+ (void)registerPush

注册离线推送服务, IM 账号登录成功时调用。(注册推送服务需要的信息来自您 AppDelegate 实现的 TIMPushDelegate 协议的 offlinePushCertificateID)

注意：

您需要使用 TUICore 组件中的 TUILogin 提供的 login 接口登录，插件会自动感知并注册推送服务。

如果您不想使用 TUILogin 提供的接口，您需要在完成登录操作后，手动调用该接口注册服务。

用法： `[TIMPush registerPush];`

+ (void)unRegisterPush

反注册离线推送服务，IM 账号登出时调用。

注意：

您需要使用 TUICore 组件中的 TUILogin 提供的 login 接口登录，插件会自动感知并注册推送服务。

如果您不想使用 TUILogin 提供的接口，您需要在完成登录操作后，手动调用该接口注册服务。

用法： `[TIMPush unRegisterPush];`

+ (void)disableAutoRegisterPush

关闭插件自动注册推送服务，需要在登录之前调用。

注意：

如果您使用 TUICore 组件中的 TUILogin 提供的 login 接口登录，插件默认自动注册推送服务，调用该接口可关闭自动注册。

用法： `[TIMPush disableAutoRegisterPush];`

+ (void)onReceiveNotificationRequest:(UNNotificationRequest *)request inAppGroupID:(NSString *)appGroupID callback:(TIMPushNotificationExtensionCallback)callback

统计 TIMPush 的推送抵达率

您需要在 AppDelegate.m 文件中实现 `- applicationGroupID` 方法，返回 App Group ID。

并在 Notification Service Extension 的 '- didReceiveNotificationRequest:withContentHandler:' 方法中调用本函数。

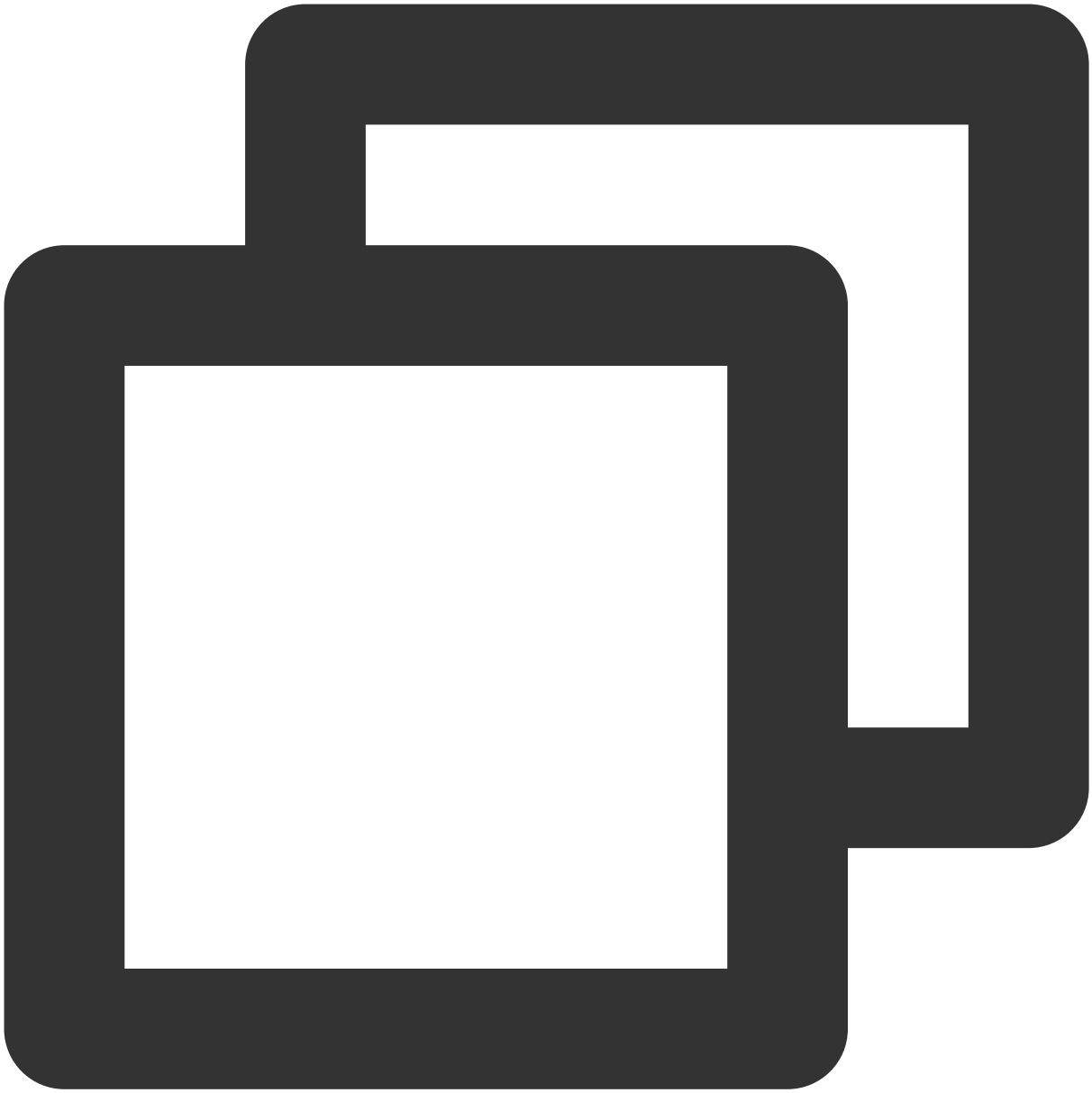
注意：

appGroup 标识当前主 App 和 Extension 之间共享的 App Group，需要在主 App的 Capability 中配置 App Groups 能力。

参数说明：

request	UNNotificationServiceExtension 回调携带的参数
appGroupID	appGroup 标识当前主 App和 Extension 之间共享的 App Group，需要在主 App 的 Capability 中配置 App Groups 能力。
callback	typedef void(^TIMPushNotificationExtensionCallback)(UNNotificationContent *content) 统计函数Callback，携带content信息

用法：



```
- (void)didReceiveNotificationRequest:(UNNotificationRequest *)request withContentH
    self.contentHandler = contentHandler;
    NSString * appGroupID = kTIMPushAppGorupKey;
    __weak typeof(self) weakSelf = self;
    [TIMPush onReceiveNotificationRequest:request inAppGroupID:appGroupID callback:
        weakSelf.bestAttemptContent = [content mutableCopy];
        // Modify the notification content here...
        // self.bestAttemptContent.title = [NSString stringWithFormat:@"%@ [modifie
        weakSelf.contentHandler(weakSelf.bestAttemptContent);
    }];
```

```
}
```

Flutter

最近更新时间：2024-06-13 10:39:26

TencentCloudChatPush

class TencentCloudChatPush：推送插件接口类。

接口概览

注册/反注册推送服务接口

初始化并成功登录 IM 后，可以注册推送服务。

API	描述
registerOnNotificationClickedEvent	提前注册消息点击回调。
registerPush	注册推送服务，可选覆盖推送信息来自接口参数 json。
unRegisterPush	反注册离线推送服务，IM 账号登出时调用。

FCM 自定义铃音配置接口

配置打开后，自定义铃音生效，发送方发送消息的离线信息中需要带上该 channelId。

API	描述
configFCMPrivateRing	配置 FCM 的自定义铃音，需要在注册推送服务之前调用。

推送通道特殊配置接口

API	描述
setPushBrandId	指定设备离线推送使用的厂商通道类型，需要在注册推送服务之前调用。
getPushBrandId	获取设备离线推送正在使用的厂商通道类型。
checkPushStatus	各个厂商接入配置完成后，可使用该接口在对应厂商设备上测试可 push 状态。
setApnsCertificateID	单独配置 APNs 的推送证书 ID。

setApplicationGroupID	配置 iOS 项目的 Application Group ID。
getAndroidPushToken	获取 Android 设备厂商 Token。
setAndroidPushToken	手动指定 Android 设备厂商 Token。
setAndroidCustomTIMPushConfigs	自定义替换插件默认读取的注册推送配置文件 timpush-configs.json，需要在注册推送服务之前调用。

接口详情

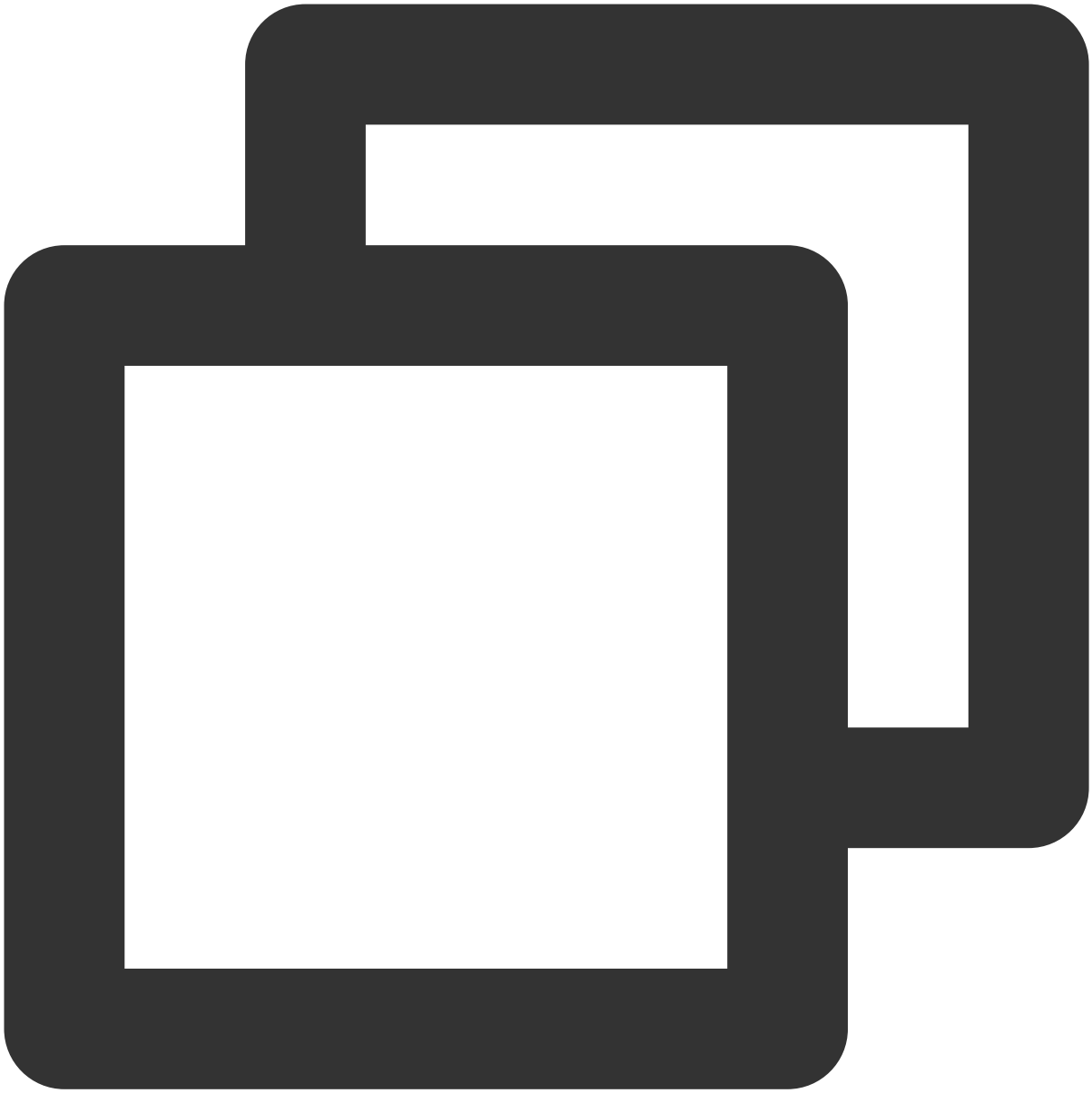
推送插件类

TencentCloudChatPush()：获取 TencentCloudChatPush 推送插件实例，是一个静态单例。后续步骤，均通过此单例实例，进行方法调用。

成员函数说明

registerOnNotificationClickedEvent

配置消息点击回调函数。
您可根据需要，提前调用或动态调整该回调函数，也可以在 `registerPush` 的时候直接传入。
示例代码：



```
void _onNotificationClicked({required String ext, String? userID, String? groupID})
  print("_onNotificationClicked: $ext, userID: $userID, groupID: $groupID");
  /// 自定义处理
}
TencentCloudChatPush().registerOnNotificationClickedEvent(onNotificationClicked: _o
```

参数说明：

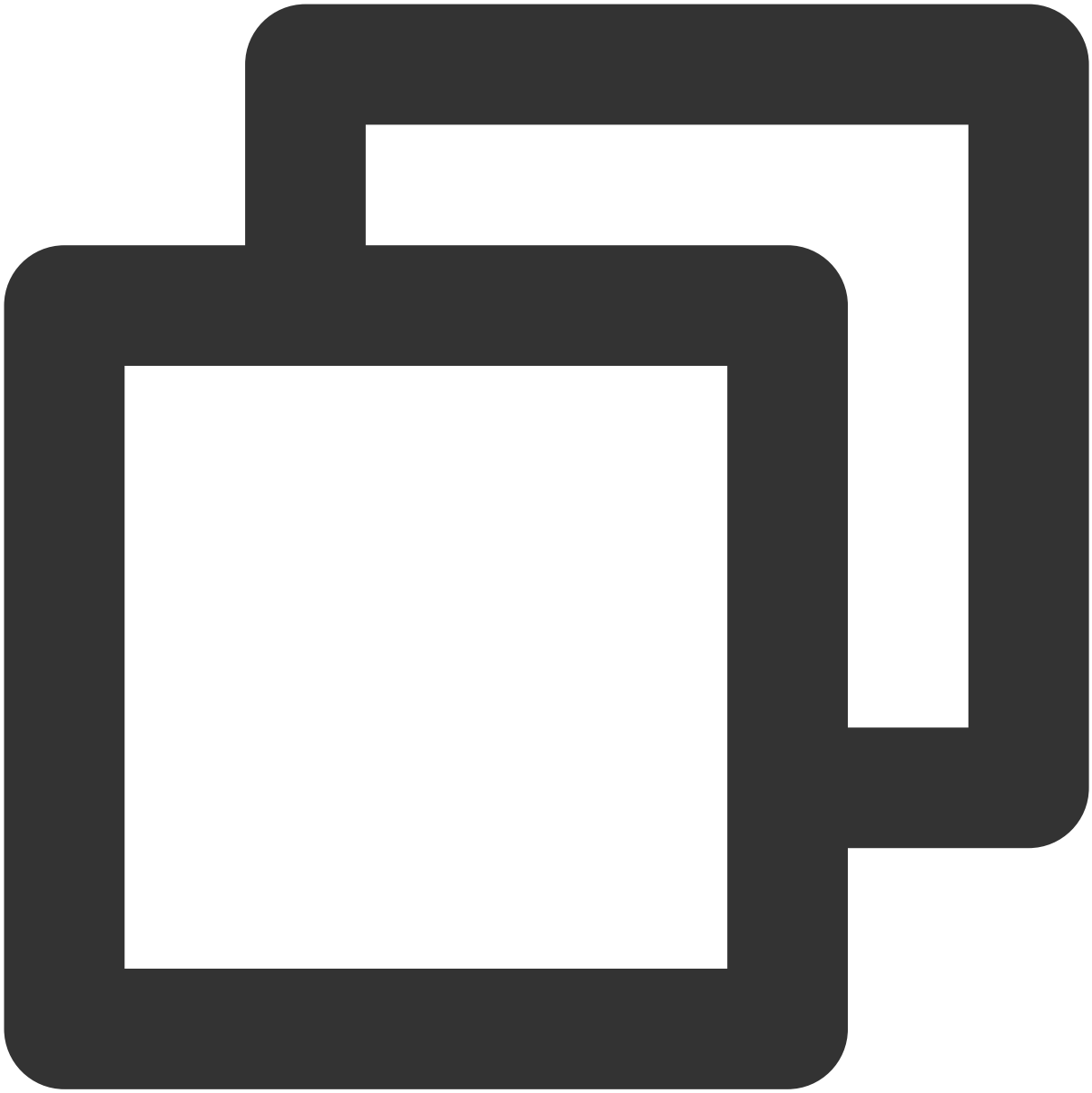
参数		类型	说明
onNotificationClicked	ext	String	为该消息所携带的完整 ext 信息，由发送方指定, 如果未指

			定，则有默认值。您可根据解析该字段，跳转至对应页面。
	userID	String?	本参数对应 userID，自动尝试解析 ext Json String， 获取里面携带的单聊对方 userID 。 **说明：** 如果您未自定义 ext 字段，ext 字段由 SDK 或 UIKit 默认指定，则可使用此处的默认解析。如果尝试解析失败，则为 null 空。
	groupID	String?	本参数对应 groupID，自动尝试解析 ext Json String， 获取里面携带的群聊 groupID 信息。 **说明：** 如果您未自定义 ext 字段，ext 字段由 SDK 或 UIKit 默认指定，则可使用此处的默认解析。如果尝试解析失败，则为 null 空。

registerPush

注册离线推送服务，IM 账号登录成功后调用。

示例代码：



```
TencentCloudChatPush().registerPush(
  onNotificationClicked: _onNotificationClicked,
  androidPushOEMConfig: "可留空 null",
  apnsCertificateID: 0,
);
```

参数说明：

参数		类型	说明
onNotificationClicked	ext	String	为该消息所携带的完整 ext 信息，由发送方指定, 如果未指定，则

	userID	String?	本参数对应 userID，自动尝试解析 ext Json String， 获取里面携 **说明：** 如果您未自定义 ext 字段，ext 字段由 SDK 或 UIKit 默认指定，则
	groupID	String?	本参数对应 groupID，自动尝试解析 ext Json String， 获取里面携 **说明：** 如果您未自定义 ext 字段，ext 字段由 SDK 或 UIKit 默认指定，则
androidPushOEMConfig		String	如果已配置导入 timpush-configs.json 配置文件，则此项留空即可 可选 Android 端参数 json：

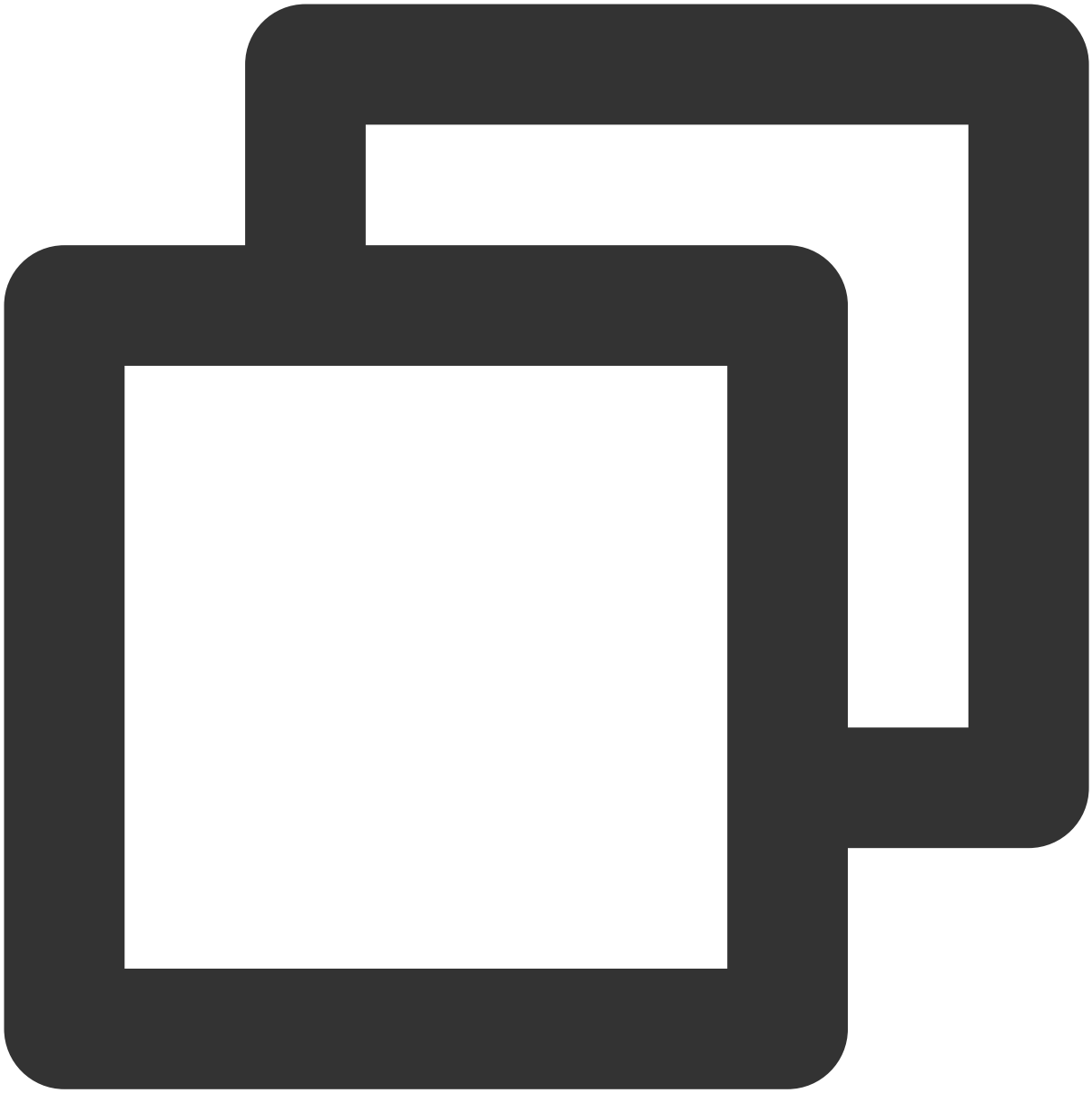
		androidPushOEMConfig: {
		// huawei
		"huaweiPushBussinessId": "",
		// 在腾讯云控制台上传第三方推送证书后分配的证书
		"huaweiBadgeClassName": "", // 角标参数,
		// xiaomi
		"xiaomiPushBussinessId": "",// 在腾讯云控
		"xiaomiPushAppId": "",// 小米开放平台分配的
		
		// meizu
		"meizuPushBussinessId": "", // 在腾讯云控
		"meizuPushAppId": "",// 魅族开放平台分配的
		
		// vivo
		"vivoPushBussinessId": "", // 在腾讯云控制
		
		// google
		"fcmPushBussinessId": "", // 在腾讯云控制
		
		// oppo
		"oppoPushBussinessId": "", // 在腾讯云控制
		"oppoPushAppKey": "",// oppo开放平台分配的
		"oppoPushAppSecret": "",// oppo开放平台分
		
		// honor
		"honorPushBussinessId": "", // 在腾讯云控
		}

apnsCertificateID	int	如单独调用 setApnsCertificateID 方法已配置, 此项可不传。

unRegisterPush

反注册离线推送服务，IM 账号登出后调用。

示例代码：

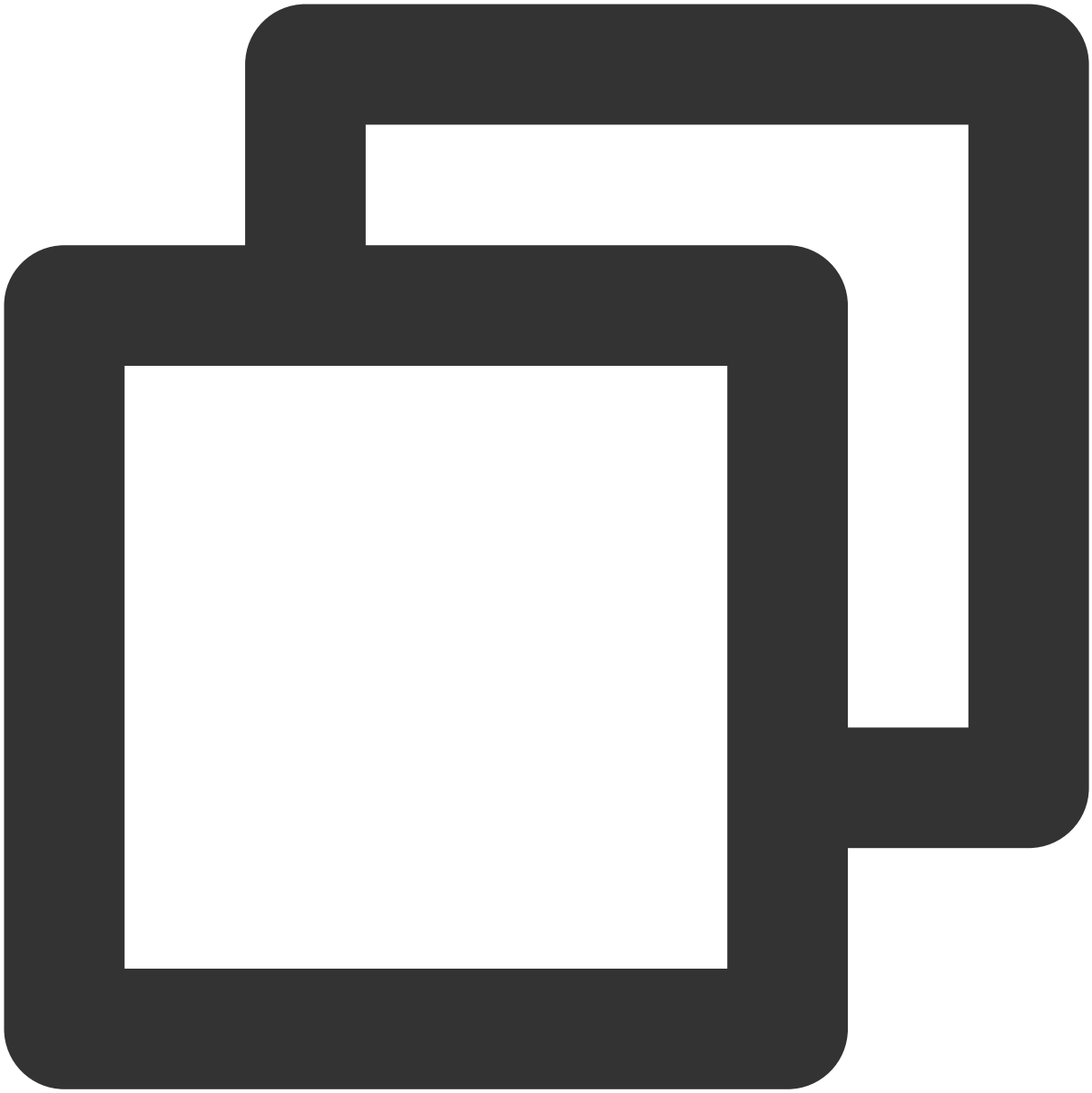


```
TencentCloudChatPush().unRegisterPush();
```

configFCMPrivateRing

配置 FCM 的自定义铃音，需要在注册推送服务之前调用。

示例代码：



```
TencentCloudChatPush().configFCMPrivateRing(channelId: channelId, ringName: ringNam
```

参数说明：

参数名	类型	说明
channelId	String	FCM 通道自定义通知栏的 channel ID，应用内唯一。
ringName	String	FCM 通道自定义通知栏的推送铃音名称，raw 目录下铃音且不需要后缀名。

enable	bool	设置离线推送提示铃音是否使用自定义铃音。

注意：

配置打开后，自定义铃音生效，发送方发送消息的离线信息中需要带上该 channelId。

setPushBrandId

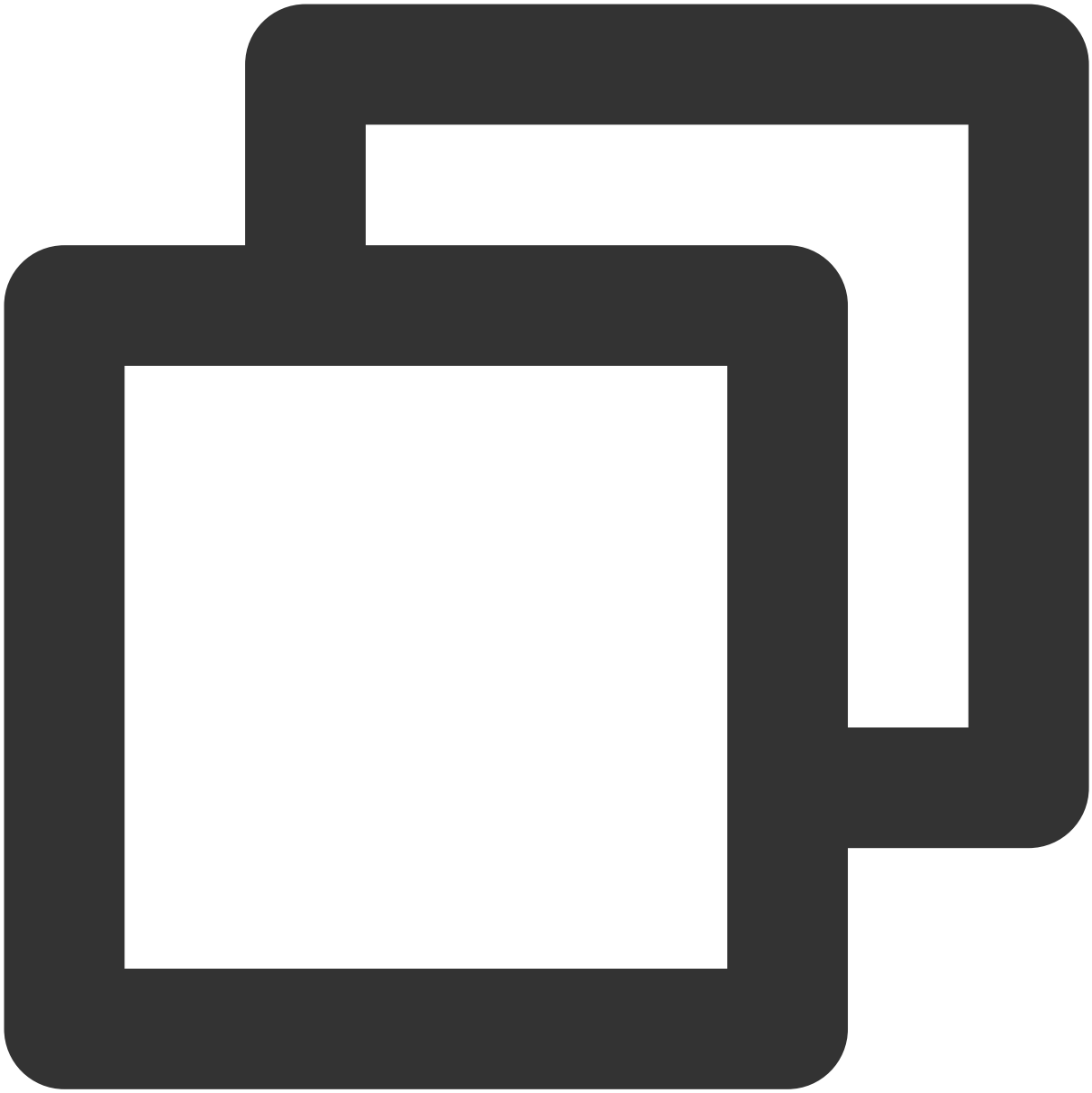
指定设备离线推送使用的厂商通道类型，需要在注册推送服务之前调用。

说明：

该接口可以指定使用厂商推送通道类型，例如在国外的小米设备指定使用 FCM 通道推送 setPushBrandId(TencentCloudChatPushBrandID.FCM)。

一般不指定通道类型，组件会自动识别设备厂商类别来注册使用对应的厂商通道。

示例代码：



```
TencentCloudChatPush().setPushBrandId(brandID: brandID);
```

参数说明

：

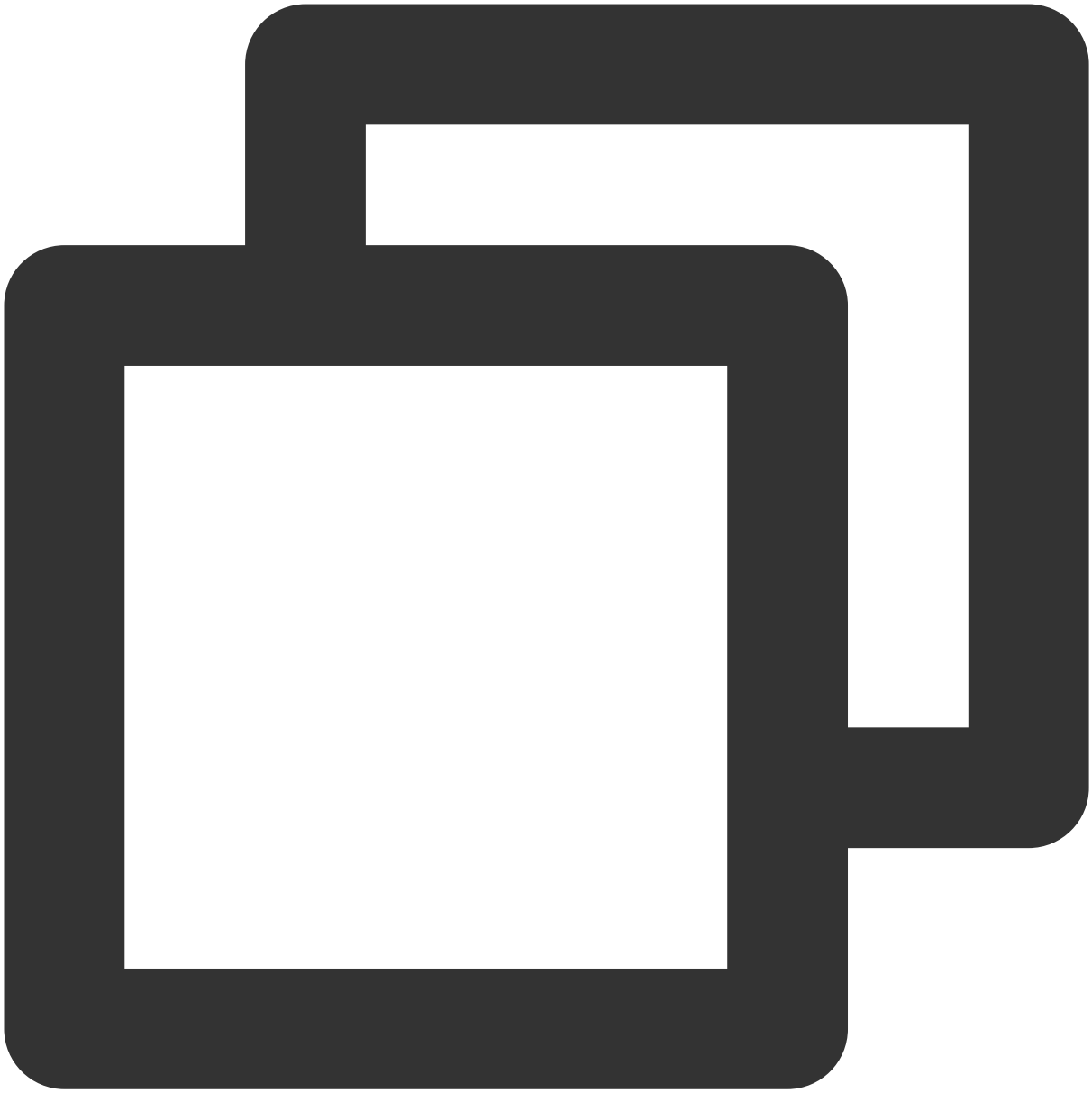
参数	描述	
brandID	厂商	设备类型
	XiaoMi	TencentCloudChatPushBrandID.XiaoMi

HuaWei	TencentCloudChatPushBrandID.HuaWei
FCM	TencentCloudChatPushBrandID.FCM
Meizu	TencentCloudChatPushBrandID.Meizu
Oppo	TencentCloudChatPushBrandID.Oppo
Vivo	TencentCloudChatPushBrandID.Vivo
Honor	TencentCloudChatPushBrandID.Honor

getPushBrandId

获取设备离线推送正在使用的厂商通道类型。

示例代码：



```
final res = await TencentCloudChatPush().getPushBrandId();
if(res.code == 0){
  final TencentCloudChatPushBrandID brandID = res.data;
}
```

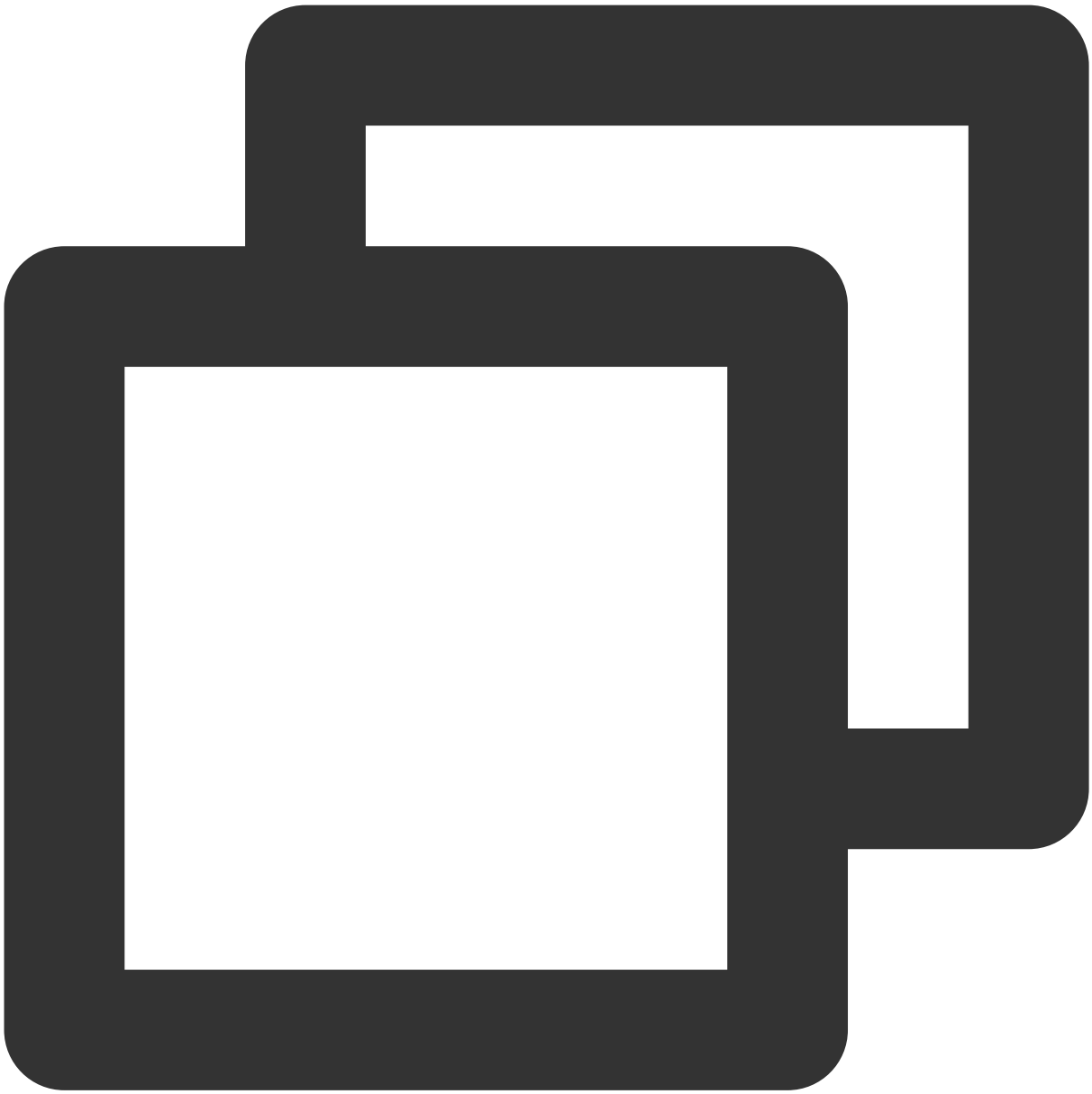
返回类型:

Future<TencentCloudChatPushResult<TencentCloudChatPushBrandID>>

checkPushStatus

各个厂商接入配置完成后，可使用该接口在对应厂商设备上测试是否可接收推送。

示例代码：



```
final res = await TencentCloudChatPush().checkPushStatus(brandID: 2002);
if(res.code == 0){
  final status = res.data;
}
```

参数说明:

参数名	类型	说明
brandID	TencentCloudChatPushBrandID	BrandID 定义如上表所示。

返回类型:

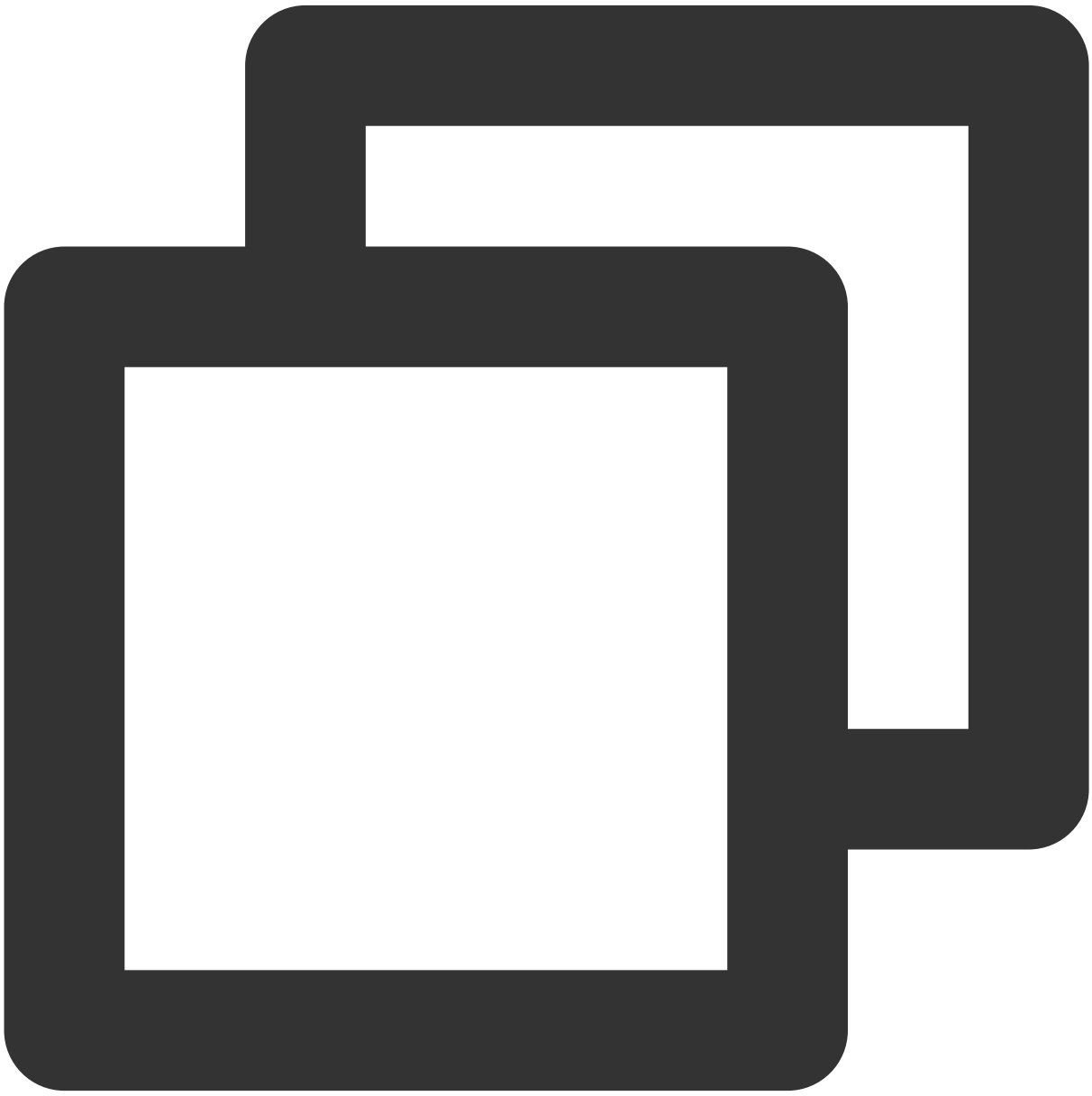
Future<TencentCloudChatPushResult<String>>

若为true, 则可以成功推送。

setApnsCertificateID

单独配置 APNs 的推送证书 ID。 您可根据需要, 提前调用或动态调整证书 ID 用此方法, 也可以在 `registerPush` 的时候直接传入。

示例代码：



```
TencentCloudChatPush().setApnsCertificateID(apnsCertificateID: 0);
```

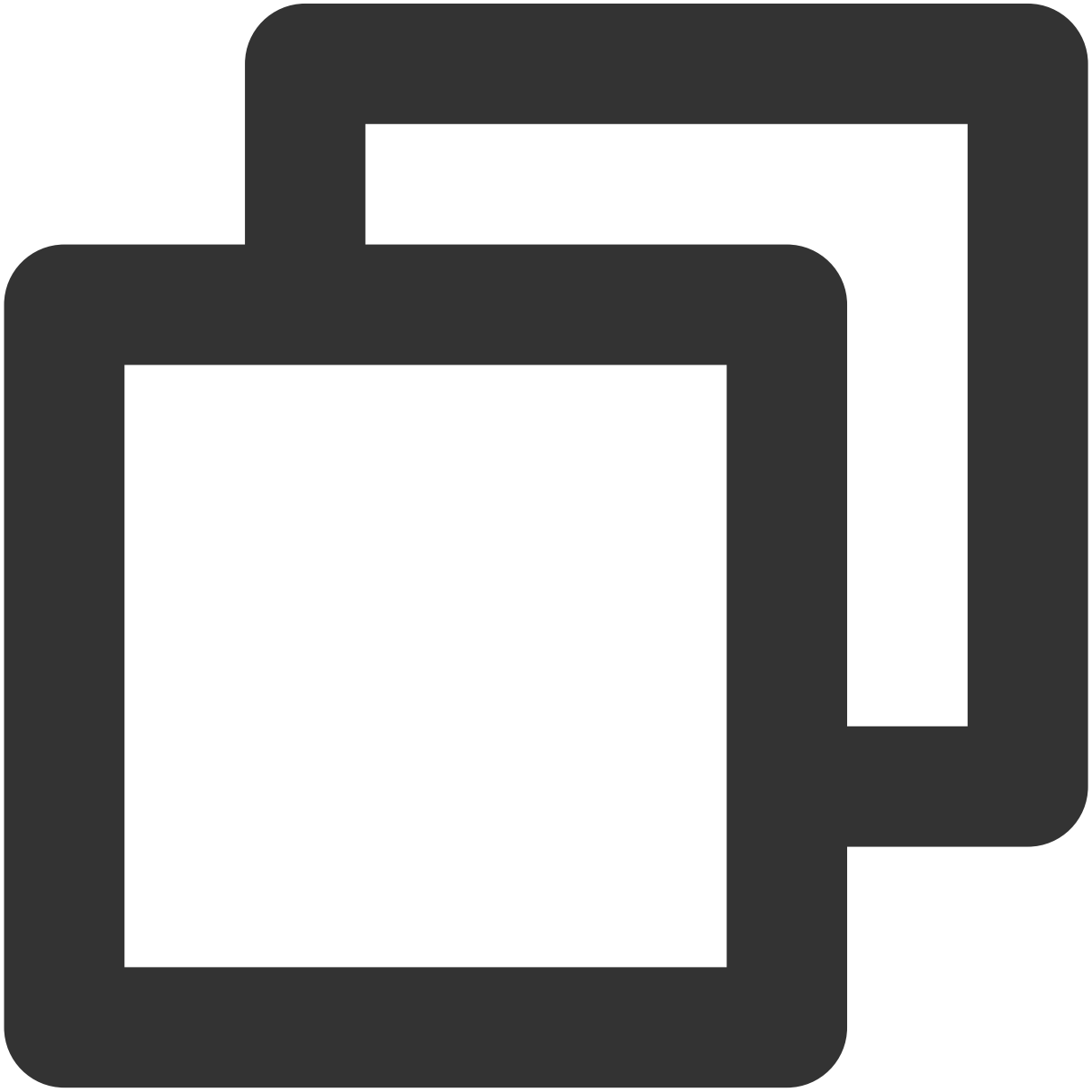
参数说明:

参数名	类型	说明
apnsCertificateID	int	腾讯云 IM 控制台上为 APNs 证书分配到的证书 ID。

setApplicationGroupID

配置 iOS 项目的 Application Group ID。

示例代码：



```
TencentCloudChatPush().setApplicationGroupID(applicationGroupID: "");
```

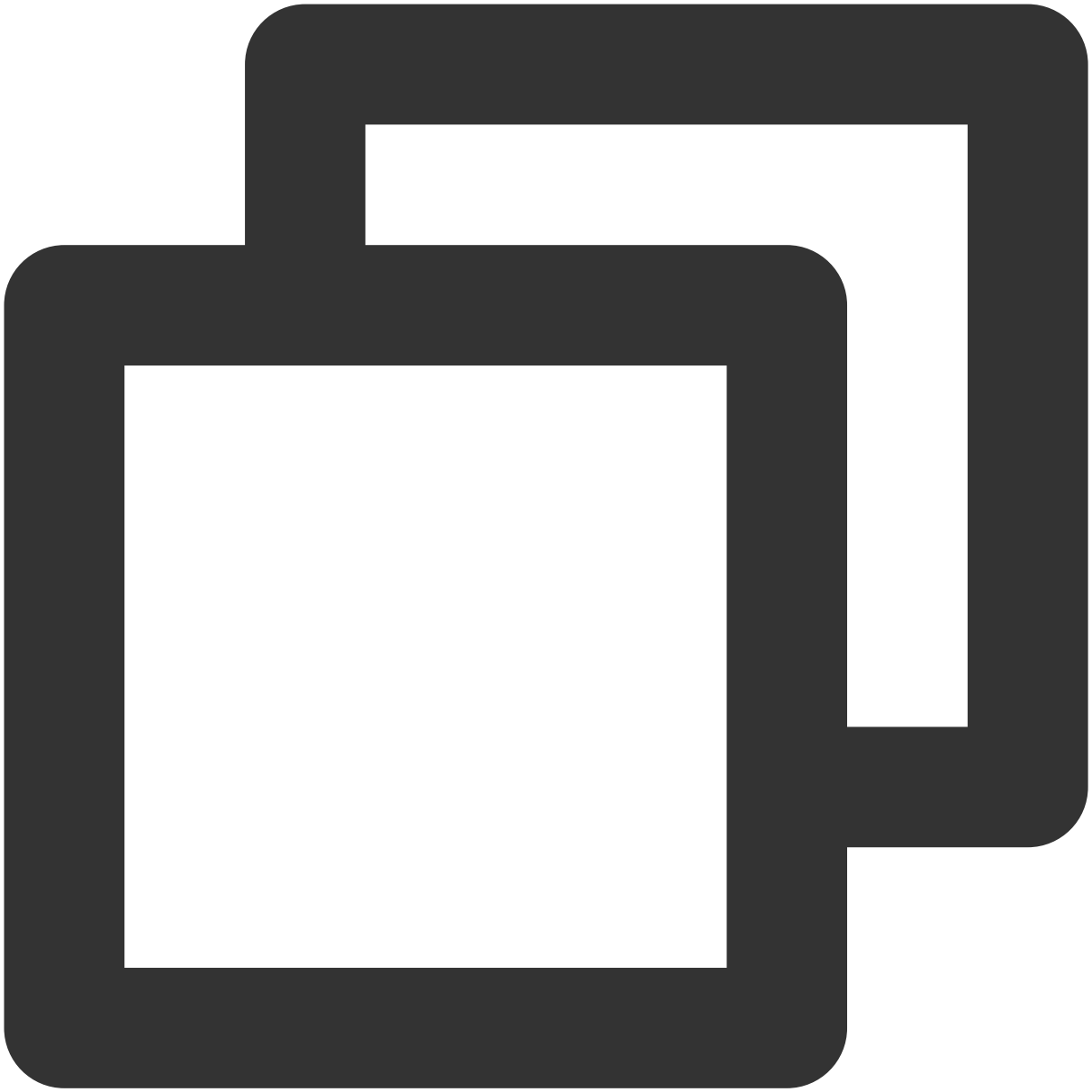
参数说明:

参数名	类型	说明
applicationGroupID	String	格式为: group + [主bundleID] + key。

getAndroidPushToken

获取 Android 设备厂商 Token。

示例代码：



```
TencentCloudChatPush().getAndroidPushToken();
```

返回类型:

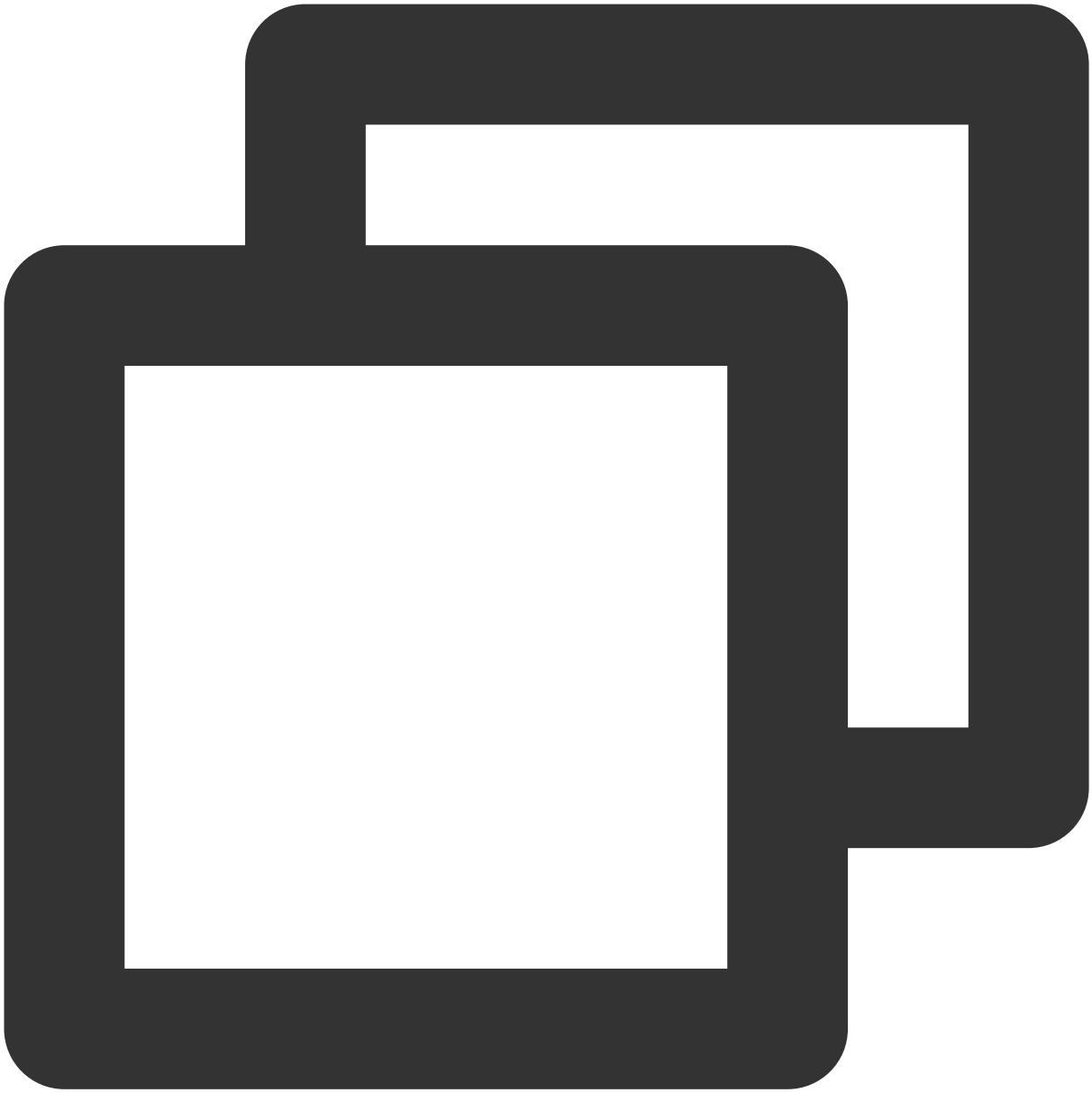
Future<TencentCloudChatPushResult<String>>

返回的 String 类型 data 即为厂商推送 Token.。

setAndroidPushToken

手动指定 Android 设备厂商 Token。

示例代码：



```
TencentCloudChatPush().setAndroidPushToken(businessID: 10000, pushToken: "pushToken
```

参数说明:

参数名	类型	说明
businessID	String	推送证书 ID , 从腾讯云IM控制台, 该推送证书卡片查看获取。
pushToken	String	通过自己的方式, 获取到的厂商推送 Token。

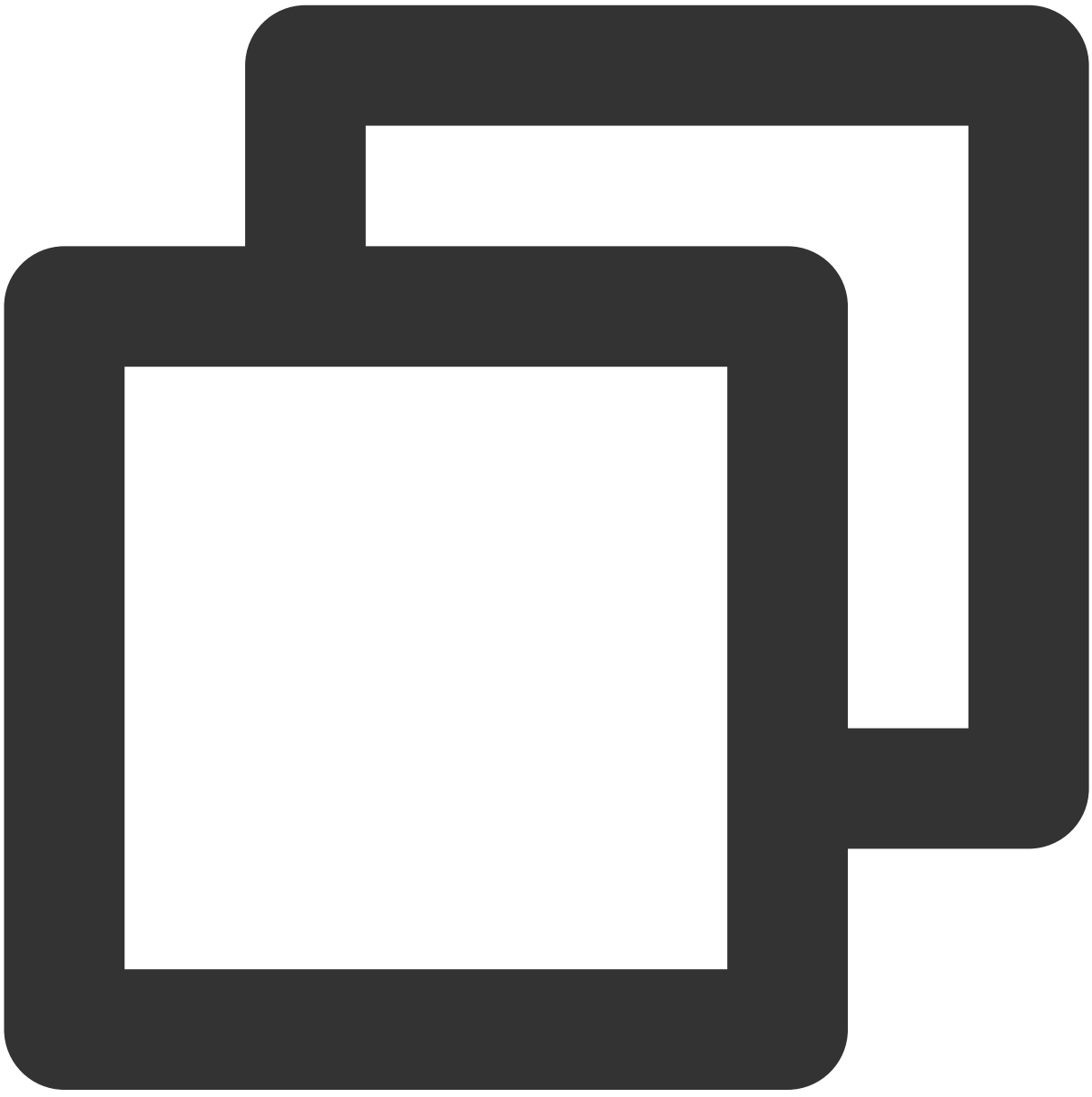
setAndroidCustomTIMPushConfigs

自定义替换插件默认读取的注册推送配置文件 timpush-configs.json，需要在注册推送服务 (registerPush) 之前调用。

说明：

主要用于多环境下动态切换不同配置文件的推送注册，例如：正式环境和测试环境不同配置文件下的推送功能集成和测试。

示例代码：



```
TencentCloudChatPush().setAndroidCustomTIMPushConfigs(configs: "");
```

参数说明:

参数名	类型	说明

configs	String	自定义配置文件的名称，路径需保持不变："工程根目录/android/app/src/assets/"。

全员/标签推送
发起全员/标签推送

最近更新时间：2024-06-13 10:39:26

全员/标签推送支持发送特定内容，还可根据标签、属性，针对特定用户群体发送个性化内容，例如会员活动、区域通知等，助力拉新、转化、促活等各个阶段运营工作的有效进行，同时支持推送送达报表，自助推送故障排查工具，具体效果请参见 效果展示。

注意：

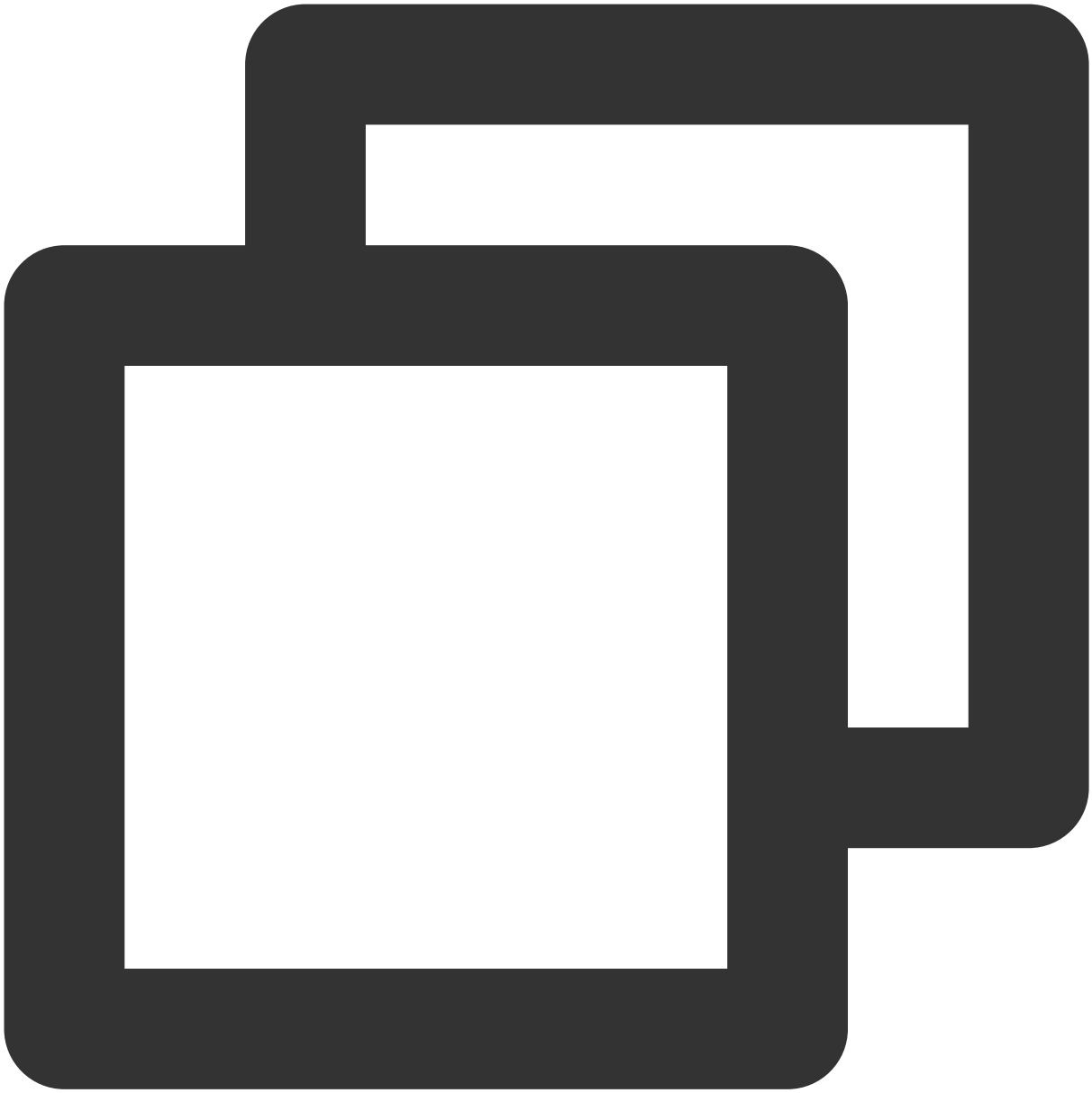
账号必须曾经登录过或者手动导入过，才可以接收到全员/标签推送的消息。

功能说明

支持向全部用户发送推送。
支持按用户属性发送推送。
支持按用户标签发送推送。
管理员推送消息，接收方看到消息发送者是管理员。
管理员指定某一账号向其他账号推送消息，接收方看到发送者不是管理员，而是管理员指定的账号。
支持漫游，漫游存储时长与普通消息的存储时长一致。
由于全员/标签推送需要下发的账号数量巨大，下发完全部账号需要一定时间（根据账号总数而定）。
支持通过 OnlineOnlyFlag 设置为1，则可以只发送推送，不保存会话和漫游以及未读消息。

接口调用说明

请求 URL 示例



```
https://xxxxxx/v4/timpush/push?usersig=xxx&identifier=admin&sdkappid=88888888&rando
```

请求参数说明

参数	说明
https	请求协议：HTTPS 请求方式：POST
xxxxxx	SDKAppID 所在国家/地区对应的专属域名。

	中国：`console.tim.qq.com` 新加坡：`adminapisgp.im.qcloud.com` 首尔：`adminapikr.im.qcloud.com` 法兰克福：`adminapiger.im.qcloud.com` 孟买：`adminapiind.im.qcloud.com` 硅谷：`adminapiusa.im.qcloud.com` **注意：** 目前印度站暂未上架推送插件。
v4/timpush/push	请求接口
usersig	App 管理员账号生成的签名，参见 UserSig 后台 API
identifier	必须为 App 管理员账号
sdkappid	创建应用时，即时通信控制台分配的 SdkAppid
random	32位无符号整数随机数
contenttype	固定值为：json

调用频率

本接口包含全员、属性、标签推送，默认每天最多调用100次，每两次推送间隔必须大于1s。

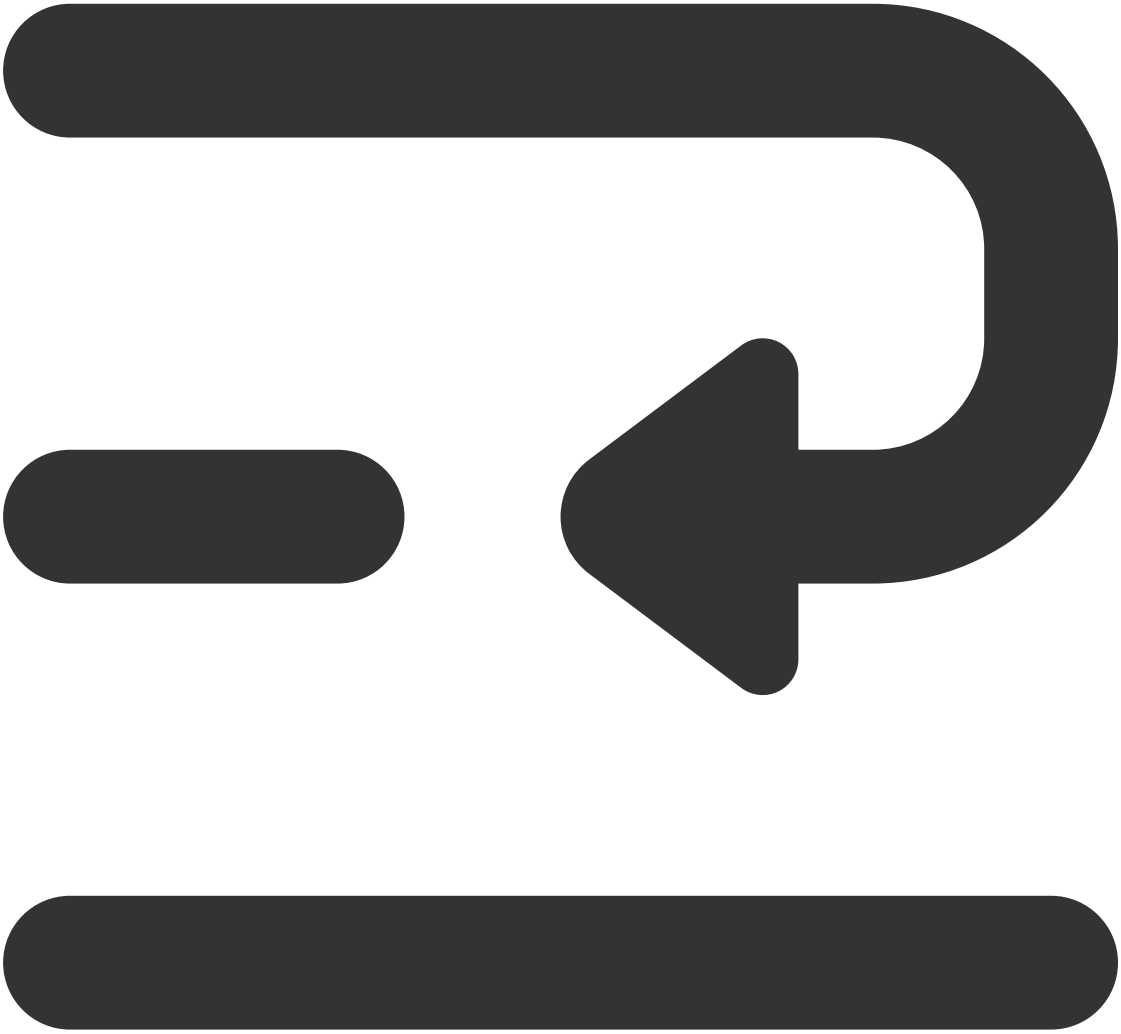
注意：

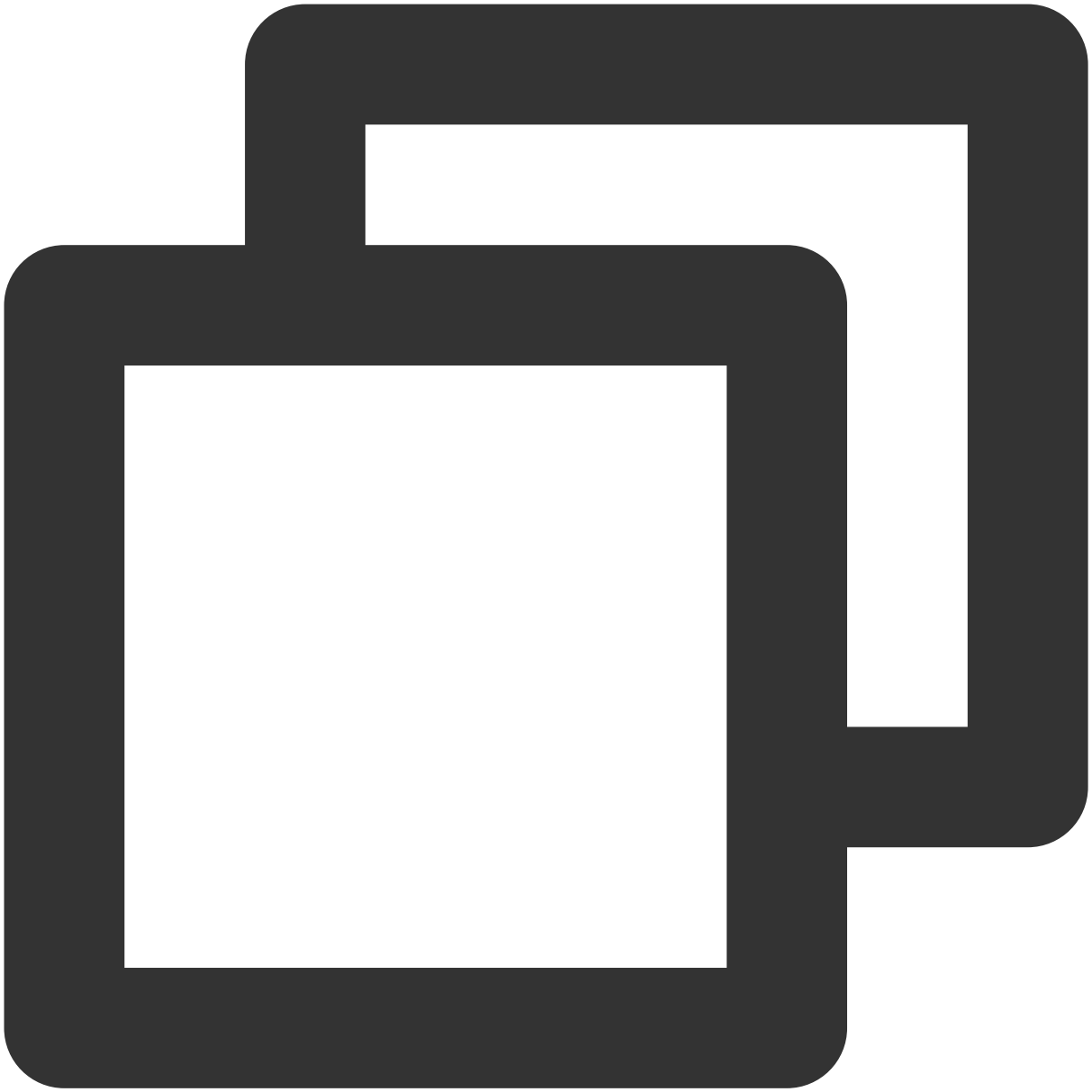
接口调用频率，默认可免费调用100次/日。每提高1次/日，费用将增加10美元/日。如需调整接口频率，请前往 IM 控制台 操作。

请求包示例

全员推送示例

管理员进行全员推送：

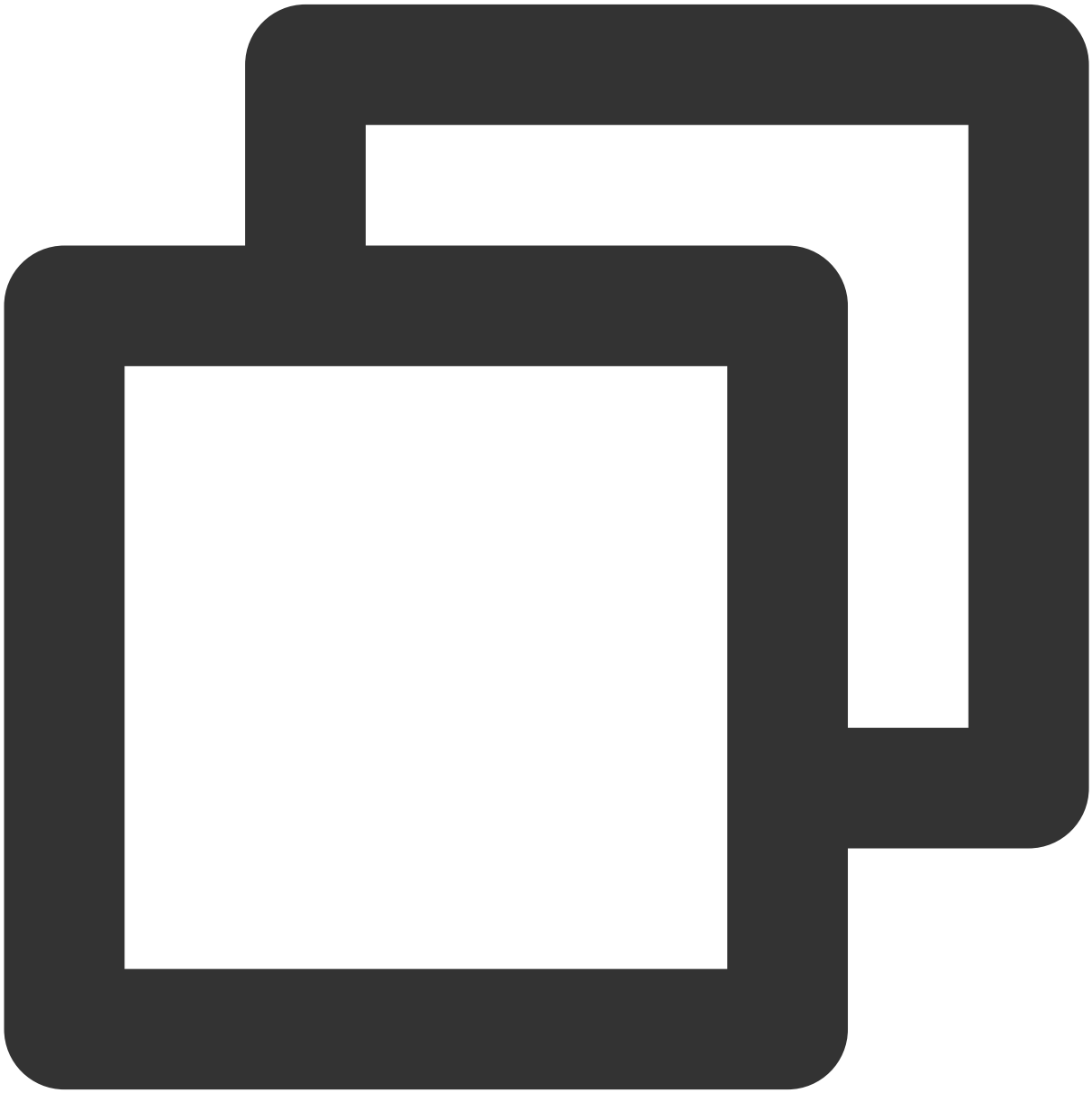




```
{
  "From_Account": "admin",
  "MsgRandom": 3674128,
  "OnlineOnlyFlag": 0, // 0表示存漫游和未读，此时推送并发用户限制200人/秒；1表示不存漫游和未读
  "MsgBody": [
      {
          "MsgType": "TIMTextElem",
          "MsgContent": {
              "Text": "hi, beauty"
          }
      }
```

```
    ],
    "OfflinePushInfo": {
        "PushFlag": 0,  // 0表示进行离线推送，1表示不进行离线推送
        "Title": "离线推送标题",
        "Desc": "离线推送内容"
    }
}
```

管理员指定某一账号进行全员推送（示例中发送方账号为 xiaoming）：



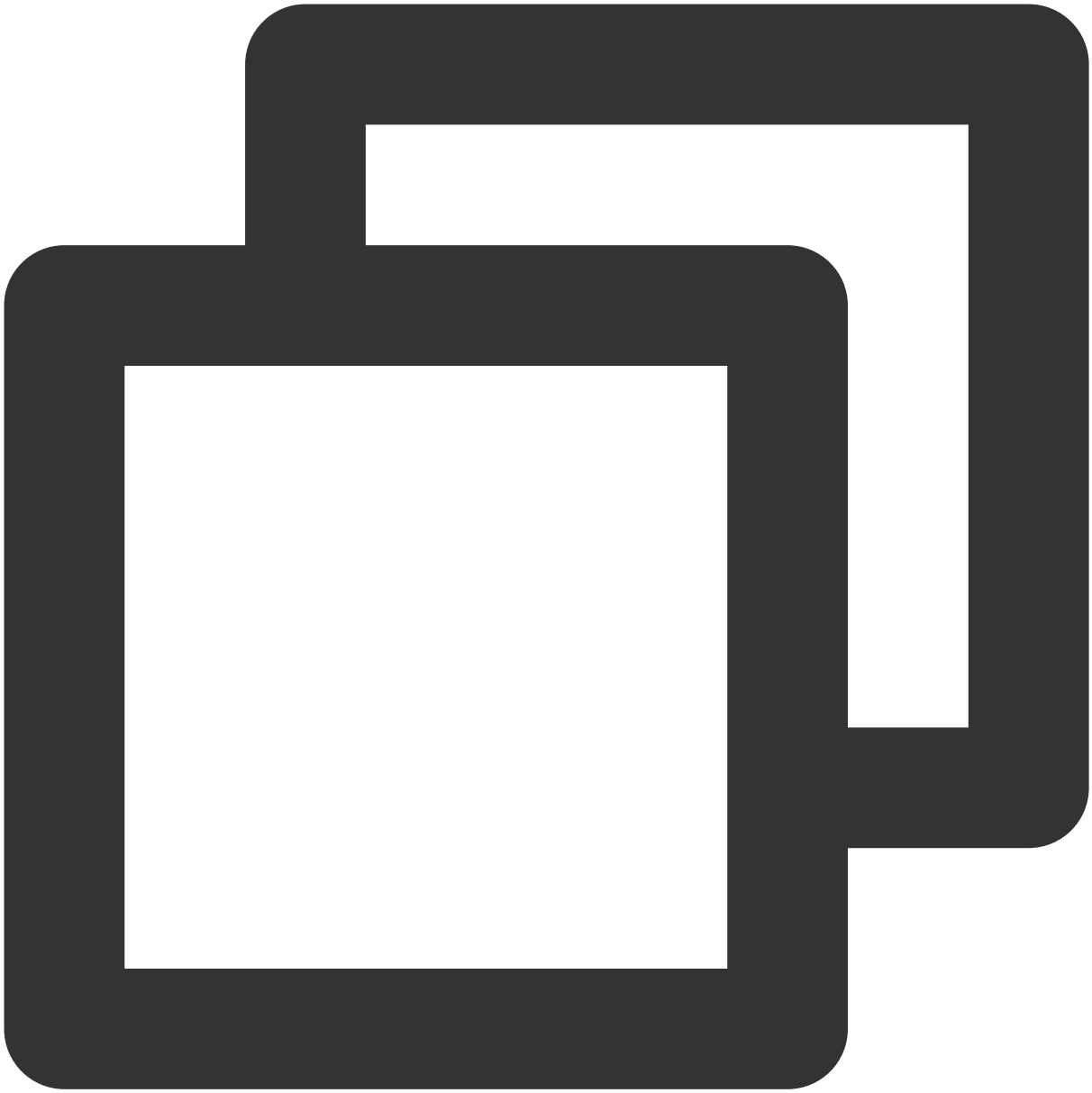
```
{
    "From_Account": "xiaoming",
```

```
    "MsgRandom": 3674128,
    "OnlineOnlyFlag": 0, // 0表示存漫游和未读，此时推送并发用户限制200人/秒；1表示不存漫游和未读
    "MsgBody": [
        {
            "MsgType": "TIMTextElem",
            "MsgContent": {
                "Text": "hi, beauty"
            }
        }
    ],
    "OfflinePushInfo": {
        "PushFlag": 0,  // 0表示进行离线推送，1表示不进行离线推送
        "Title": "离线推送标题",
        "Desc": "离线推送内容"
    }
}
```

注意：

From_Account 为消息推送方账号，支持指定为任意存在的账号。如果未指定发送方或指定的发送方不存在，则默认取接口调用方账号。

只推送在线用户（在线推送）：



```
{
  "From_Account": "xiaoming",
  "MsgRandom": 3674128,
  "OnlineOnlyFlag": 1, // 0表示存漫游和未读，此时推送并发用户限制200人/秒；1表示不存漫游和未读
  "MsgBody": [
      {
          "MsgType": "TIMTextElem",
          "MsgContent": {
              "Text": "hi, beauty"
          }
      }
```

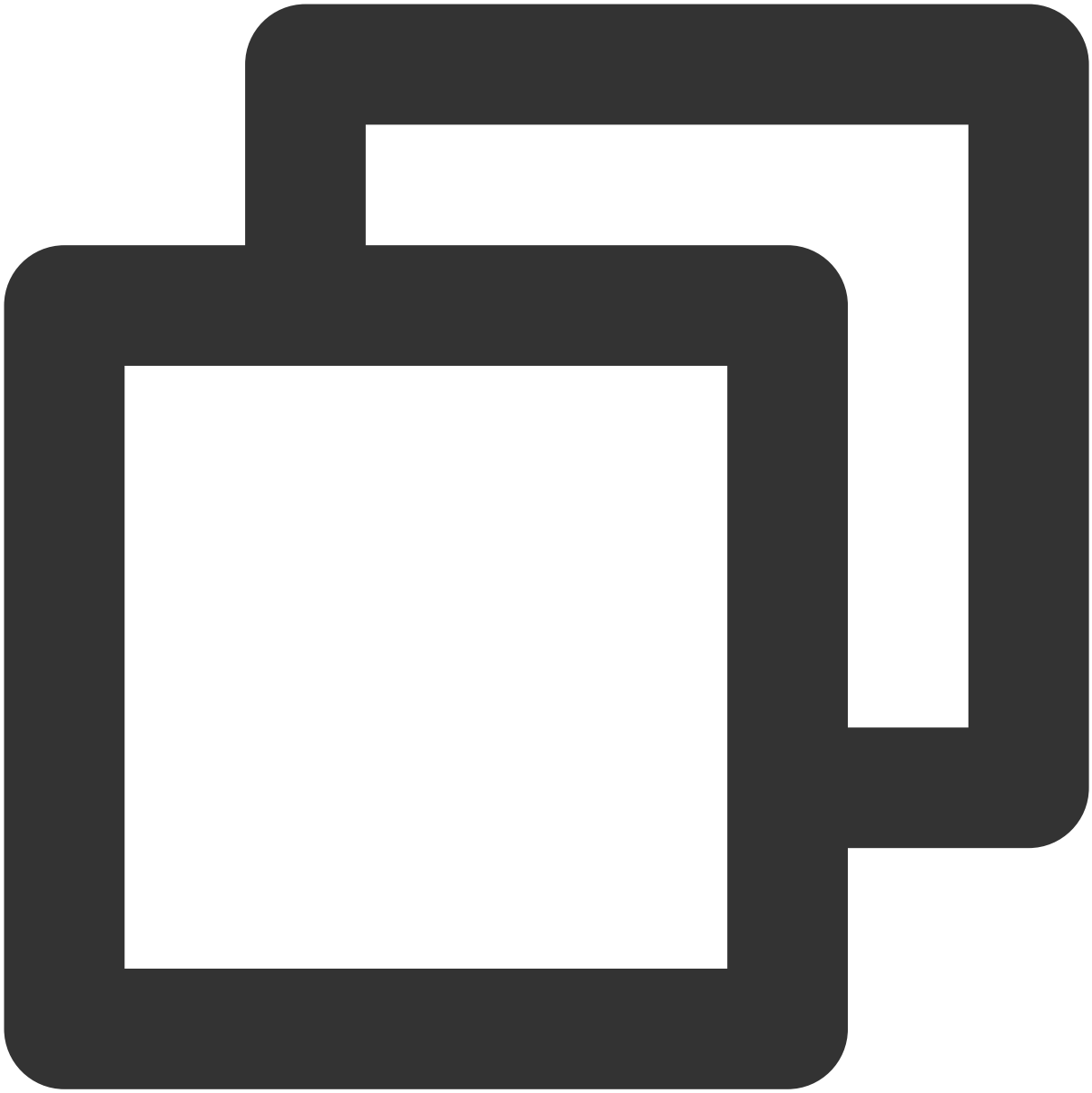
```
    ],
    "OfflinePushInfo": {
        "PushFlag": 1   // 0表示进行离线推送，1表示不进行离线推送
    }
}
```

注意：

OnlineOnlyFlag为1表示只进行推送（在线推送+离线推送），OfflinePushInfo.PushFlag为1表示不进行离线推送。因此上述实例表示只推送在线用户。

按用户标签推送示例

管理员给带有关注“股票A”和“股票B”标签的用户推送消息：



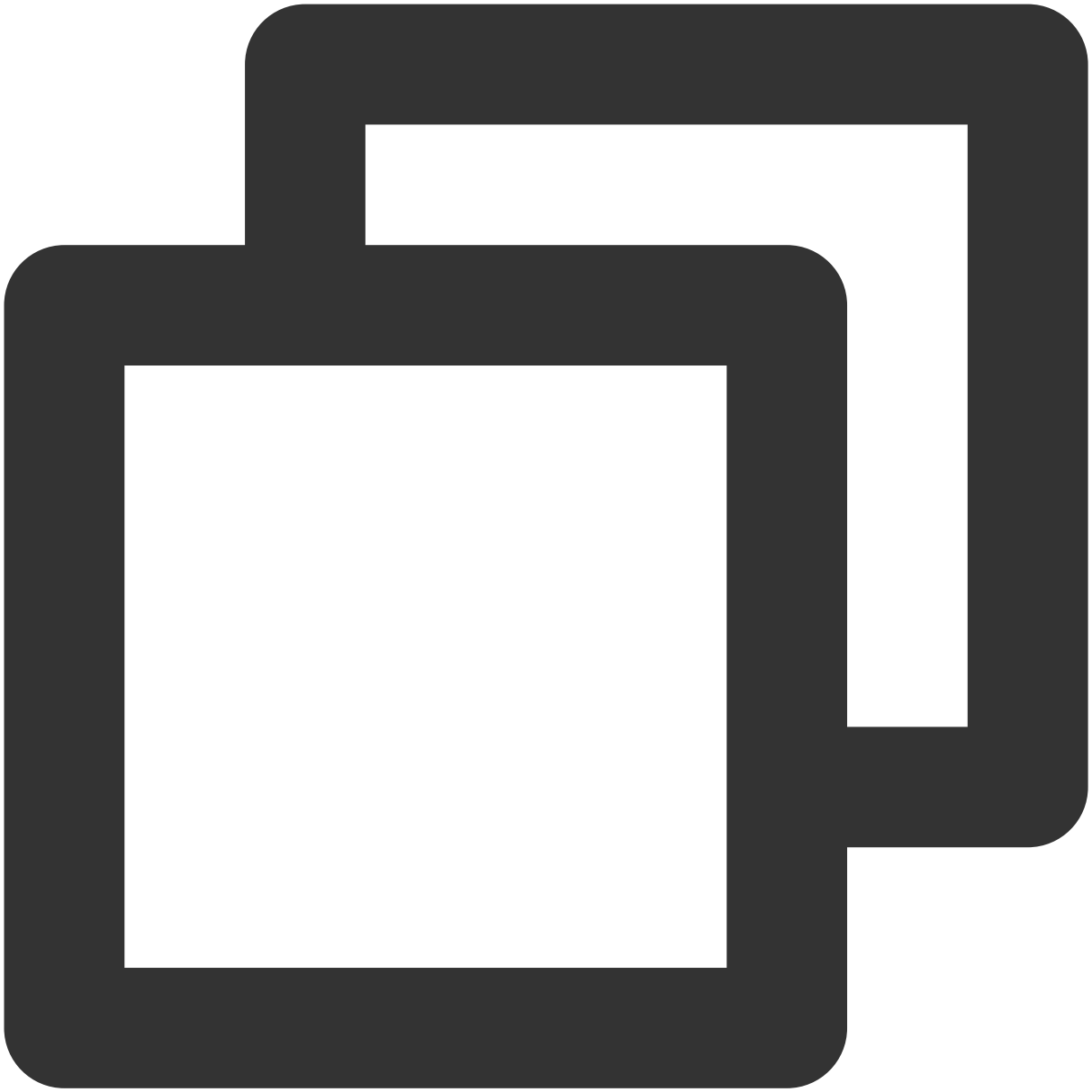
```
{
  "From_Account": "admin",
  "MsgRandom": 124032,
  "OnlineOnlyFlag": 0, // 0表示存漫游和未读，此时推送并发用户限制200人/秒；1表示不存漫游和未读
  "Condition": {
      "TagsAnd": ["股票A","股票B"]
  },
  "MsgBody": [
      {
          "MsgType": "TIMTextElem",
          "MsgContent": {
```

```
                "Text": "hi, beauty"
            }
        }
    ],
    "OfflinePushInfo": {
        "PushFlag": 0, // 0表示进行离线推送，1表示不进行离线推送
        "Title": "离线推送标题",
        "Desc": "离线推送内容"
    }
}
```

注意：

From_Account 为消息推送方账号，支持指定为任意存在的账号。如果未指定发送方或指定的发送方不存在，则默认取接口调用方账号。

管理员给关注“股票A”或“股票B”的用户推送消息：

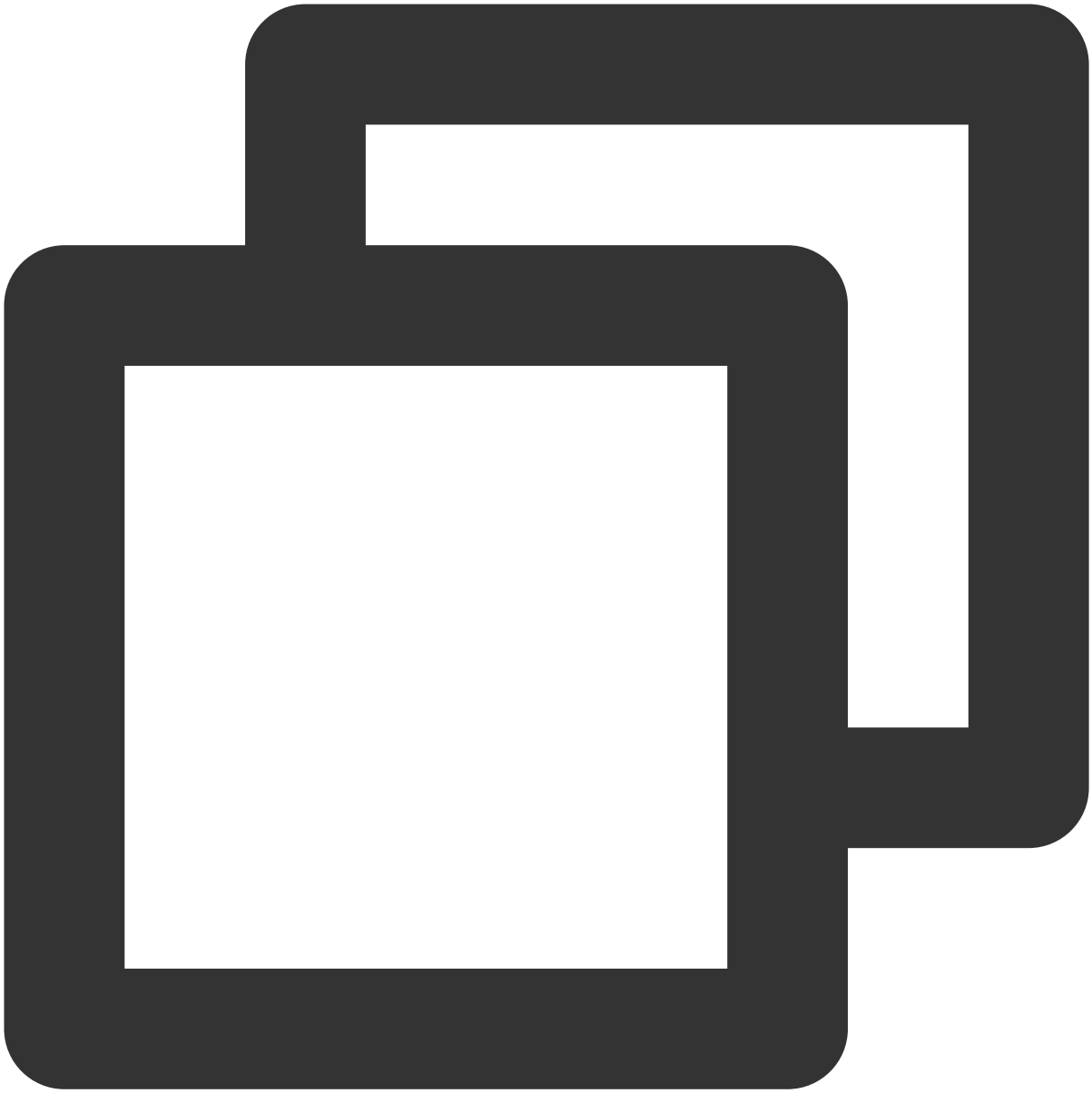


```
{
  "From_Account": "admin",
  "MsgRandom": 124032,
  "OnlineOnlyFlag": 0, // 0表示存漫游和未读，此时推送并发用户限制200人/秒；1表示不存漫游和未读
  "Condition": {
      "TagsOr": ["股票A","股票B"]
  },
  "MsgBody": [
      {
          "MsgType": "TIMTextElem",
          "MsgContent": {
```

```
                "Text": "hi, beauty"
            }
        }
    ],
    "OfflinePushInfo": {
        "PushFlag": 0, // 0表示进行离线推送, 1表示不进行离线推送
        "Title": "离线推送标题",
        "Desc": "离线推送内容"
    }
}
```

按用户属性推送

管理员给在深圳的超白金会员用户推送消息：

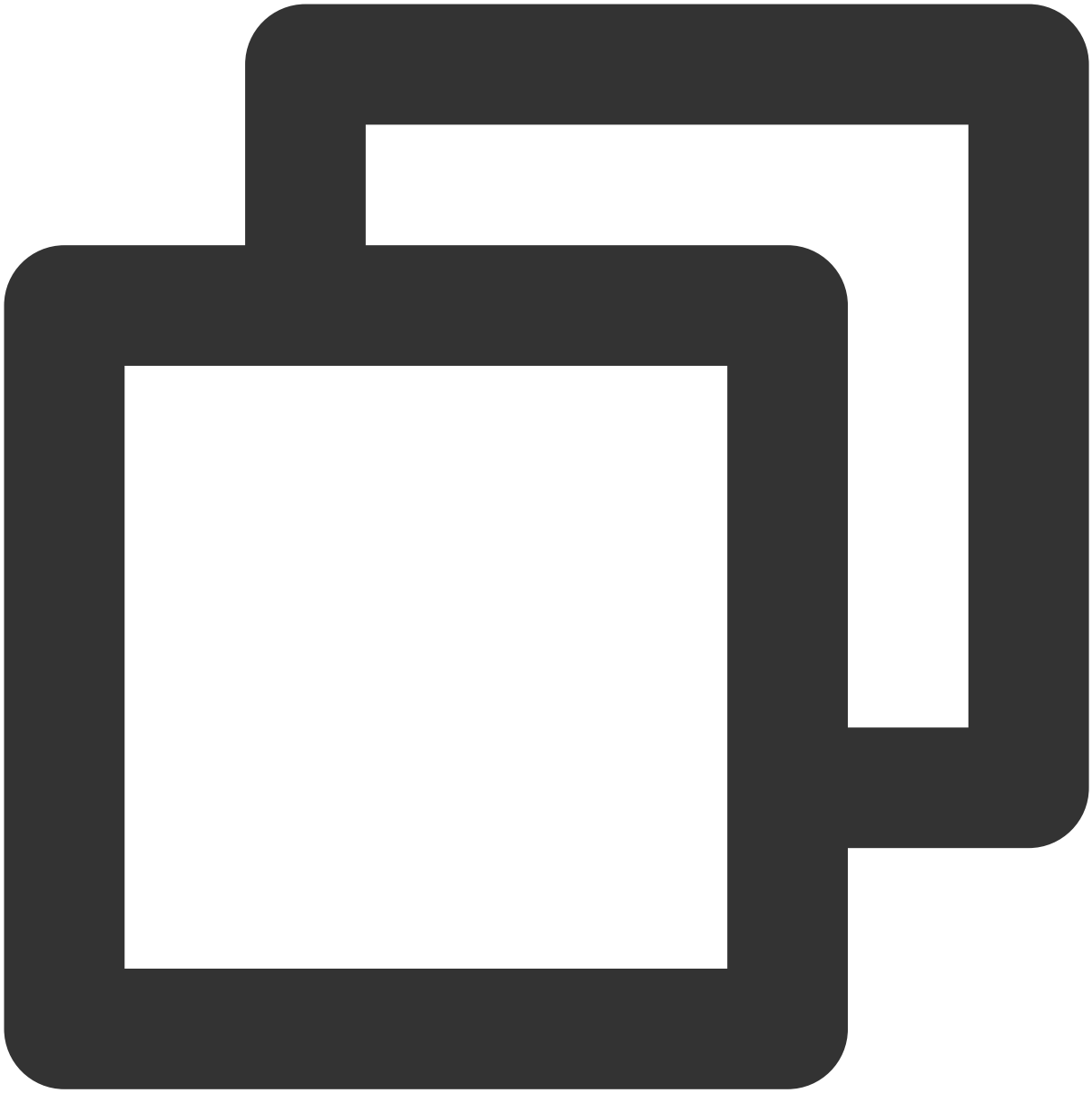


```
{
  "From_Account": "admin",
  "MsgRandom": 389475,
  "OnlineOnlyFlag": 0, // 0表示存漫游和未读，此时推送并发用户限制200人/秒；1表示不存漫游和未读
  "Condition": {
      "AttrsAnd": {
          "会员等级": "超白金会员",
          "city": "深圳"
      }
  },
  "MsgBody": [
```

```
        {
            "MsgType": "TIMTextElem",
            "MsgContent": {
                "Text": "hi, beauty"
            }
        }
    ],
    "OfflinePushInfo": {
        "PushFlag": 0, // 0表示进行离线推送，1表示不进行离线推送
        "Title": "离线推送标题",
        "Desc": "离线推送内容"
    }
}
```

注意：From_Account 为消息推送方账号，支持指定为任意存在的账号。如果未指定发送方或指定的发送方不存在，则默认取接口调用方账号。

管理员给在深圳的超白金用户推送消息：



```
{
  "From_Account": "admin",
  "MsgRandom": 389475,
  "OnlineOnlyFlag": 0, // 0表示存漫游和未读，此时推送并发用户限制200人/秒；1表示不存漫游和未读
  "Condition": {
      "AttrsAnd": {
          "会员等级": "超白金用户",
          "city": "深圳"
      }
  },
  "MsgBody": [
```

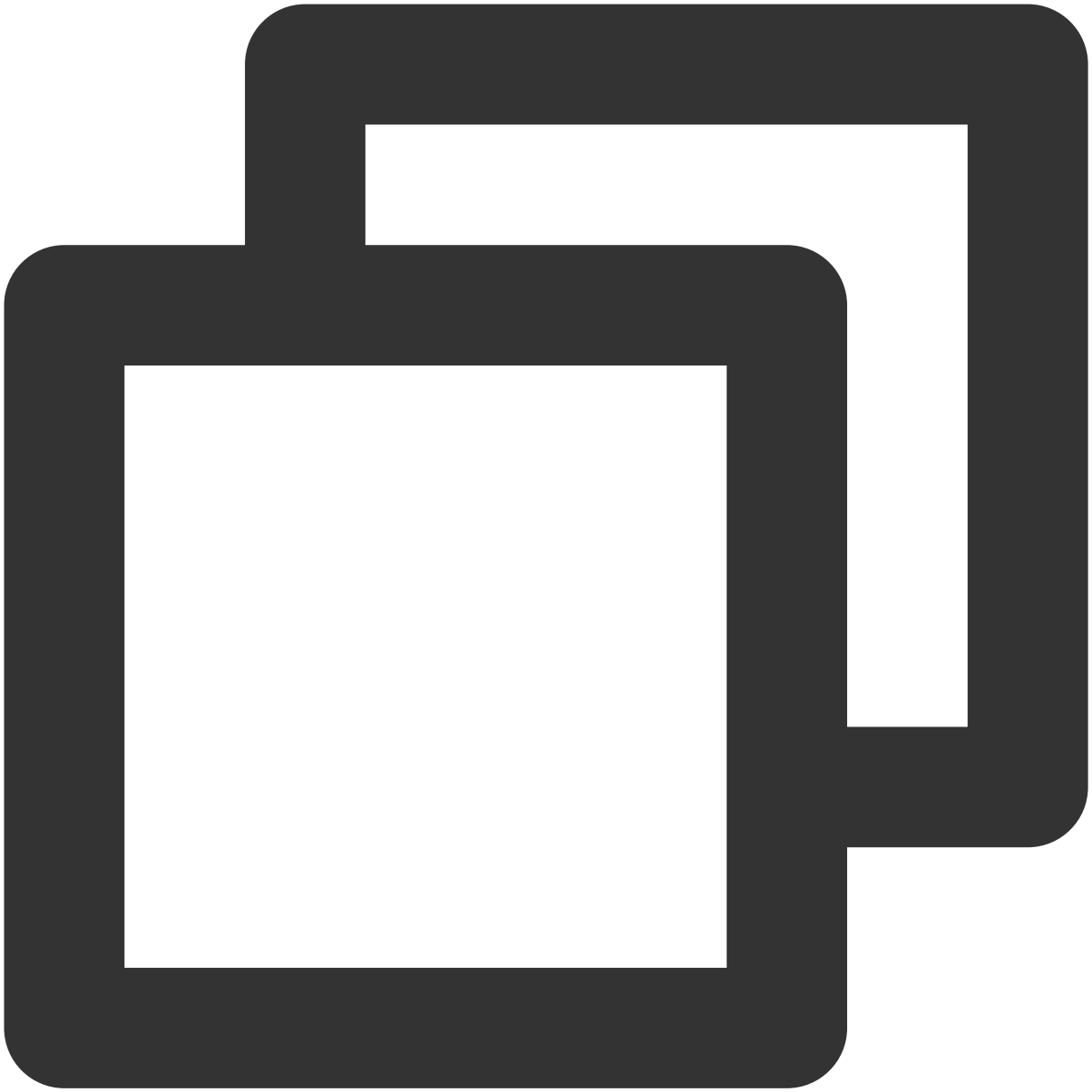
```
        {
            "MsgType": "TIMTextElem",
            "MsgContent": {
                "Text": "hi, beauty"
            }
        }
    ],
    "OfflinePushInfo": {
        "PushFlag": 0, // 0表示进行离线推送，1表示不进行离线推送
        "Title": "离线推送标题",
        "Desc": "离线推送内容"
    }
}
```

请求包字段说明

字段	类型	属性	说明
From_Account	String	选填	消息推送方账号（支持指定为任意存在的账号） **注意：**如果未指定发送方或指定的发送方不存在，则默认取接口调用方账号。
MsgRandom	Integer	必填	消息随机数（32位无符号整数），后台用于同一秒内的消息去重。请确保该字段填的是随机
OnlineOnlyFlag	Integer	选填	默认为 0，表示存漫游和未读，此时推送并发用户限制200人/秒。 1 表示不存历史消息且不计未读，此时无推送并发限制。
Condition	Object	选填	Condition 共有4种条件类型，分别是： 属性的 “或条件” AttrsOr 属性的 “与条件” AttrsAnd 标签的 “或条件” TagsOr 标签的 “与条件” TagsAnd **注意：**AttrsOr 、 AttrsAnd 、TagsOr 、TagsAnd 不能并存。如果没有 Condition，则推送给全部用户。
TagsOr	Array	选填	标签条件的并集。标签是一个不超过50字节的字符串。 **注意：**属性推送和标签推送不可同时作为推送条件。TagsOr 条件中的标签个数不超过10个。
TagsAnd	Array	选填	标签条件的交集。标签是一个不超过50字节的字符串。 **注意：**属性推送和标签推送不可同时作为推送条件。TagsAnd 条件中的标签个数不超过10个。
AttrsOr	Object	选填	属性条件的并集。 **注意：**属性推送和标签推送不可同时作为推送条件。

AttrsAnd	Object	选填	属性条件的交集。 **注意：**属性推送和标签推送不可同时作为推送条件。
MsgBody	Array	必填	消息内容，具体格式请参见 MsgBody 消息内容说明（一条消息可包括多种消息元素，所以 MsgBody 为 Array 类型）
MsgType	String	必填	TIM 消息对象类型，目前支持的消息对象包括： TIMTextElem（文本消息） TIMLocationElem（位置消息） TIMFaceElem（表情消息） TIMCustomElem（自定义消息） TIMSoundElem（语音消息） TIMImageElem（图像消息） TIMFileElem（文件消息） TIMVideoFileElem（视频消息）
MsgContent	Object	必填	对于每种 MsgType，用不同的 MsgContent 格式，具体请参见 TIMMsgElement 对象 的定义
OfflinePushInfo	Object	选填	离线推送信息配置，具体请参见 消息格式描述

应答包体示例



```
{
    "ActionStatus": "OK",
    "ErrorInfo": "",
    "ErrorCode": 0,
    "TaskId": "53743040_144115212910570789_4155518400_15723514"
}
```

应答包字段说明

字段	类型	说明

ActionStatus	String	请求处理的结果： OK：表示处理成功 FAIL：表示失败
ErrorCode	Integer	错误码
ErrorInfo	String	错误信息
TaskId	String	推送任务 ID

错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。**真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的**。公共错误码（60000到79999）参见 错误码 文档。
本 API 私有错误码如下：

错误码	含义说明
90001	JSON 格式解析失败，请检查请求包是否符合 JSON 规范。
90002	JSON 格式请求包中 MsgBody 不符合消息格式描述，或者 MsgBody 不是 Array 类型，请参见 TIMMsgElement 对象 的定义。
90005	JSON 格式请求包体中缺少 MsgRandom 字段或者 MsgRandom 字段不是 Integer 类型。
90007	JSON 格式请求包体中 MsgBody 类型不是 Array 类型，请将其修改为 Array 类型。
90009	请求需要 App 管理员权限。
90010	JSON 格式请求包不符合消息格式描述，请参见 TIMMsgElement 对象 的定义。
90020	标签长度超过限制（不能超过50字节）。
90022	推送条件中的 TagsOr 和 TagsAnd 有重复标签。
90024	推送过于频繁，每两次推送间隔必须大于1秒。
90026	消息离线存储时间错误（最多不能超过7天）。
90032	推送条件中的 tag 数量大于10，或添加标签请求中的标签数量大于10。
90033	属性无效。
90039	按属性推送和按标签推送不可同时存在。

90040	推送条件中其中1个 tag 为空。
90045	未开通全员/标签推送功能。
90047	推送次数超过当天限额（默认为100次）。
91000	服务内部错误，请重试。

接口调试工具

通过 REST API 在线测试 工具调试本接口。

参考

发起全员/标签推送
设置应用属性名称
获取应用属性名称
设置用户属性
删除用户属性
获取用户属性
添加用户标签
获取用户标签
删除用户标签
清空用户标签
推送撤回

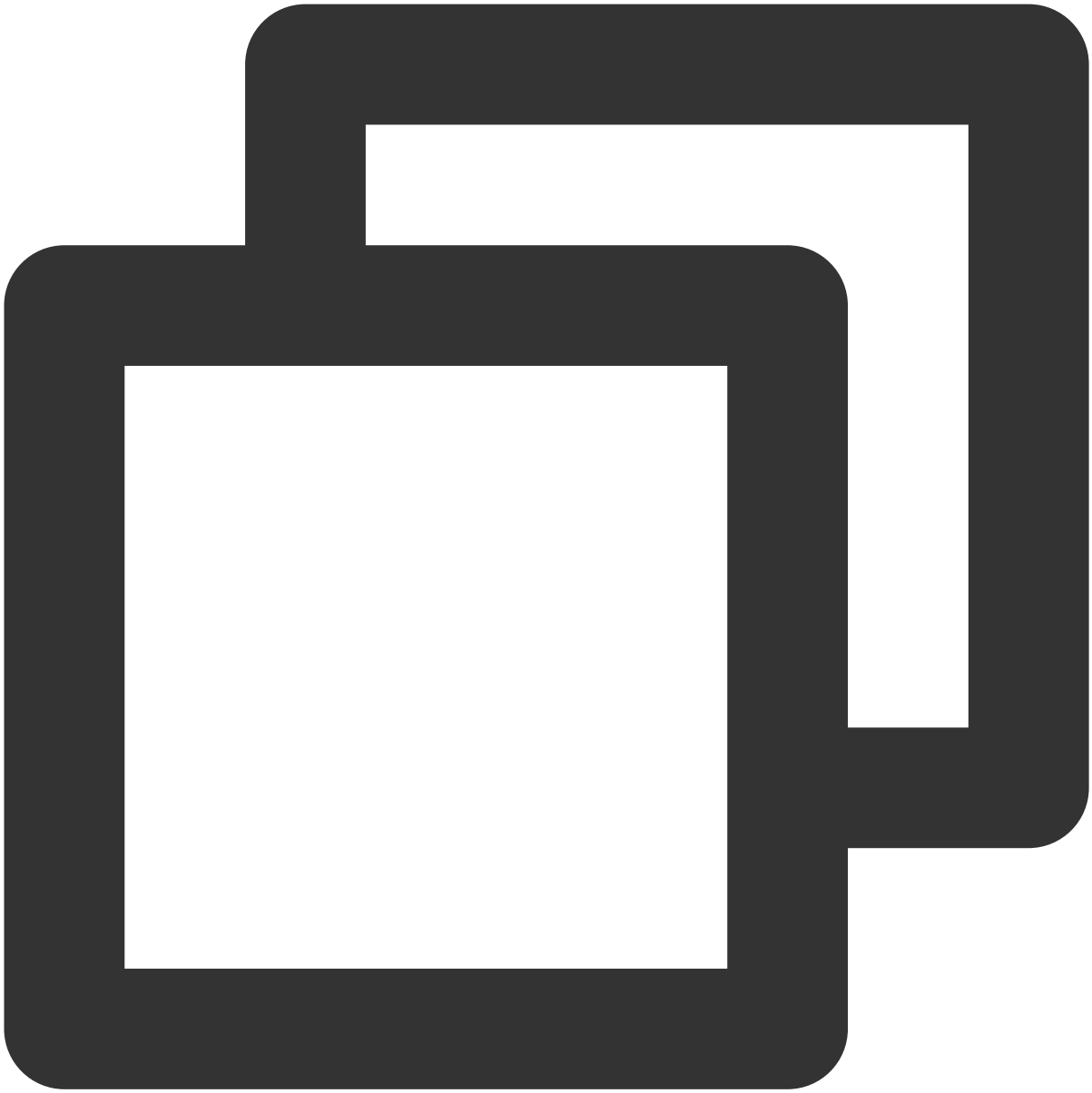
获取应用属性名称

最近更新时间：2024-06-13 10:39:26

功能说明

管理员获取应用属性名称。使用前请先 设置应用属性名称 。

请求 URL 示例



```
https://xxxxxx/v4/timpush/get_attr_name?usersig=xxx&identifier=admin&sdkappid=88888
```

请求参数说明

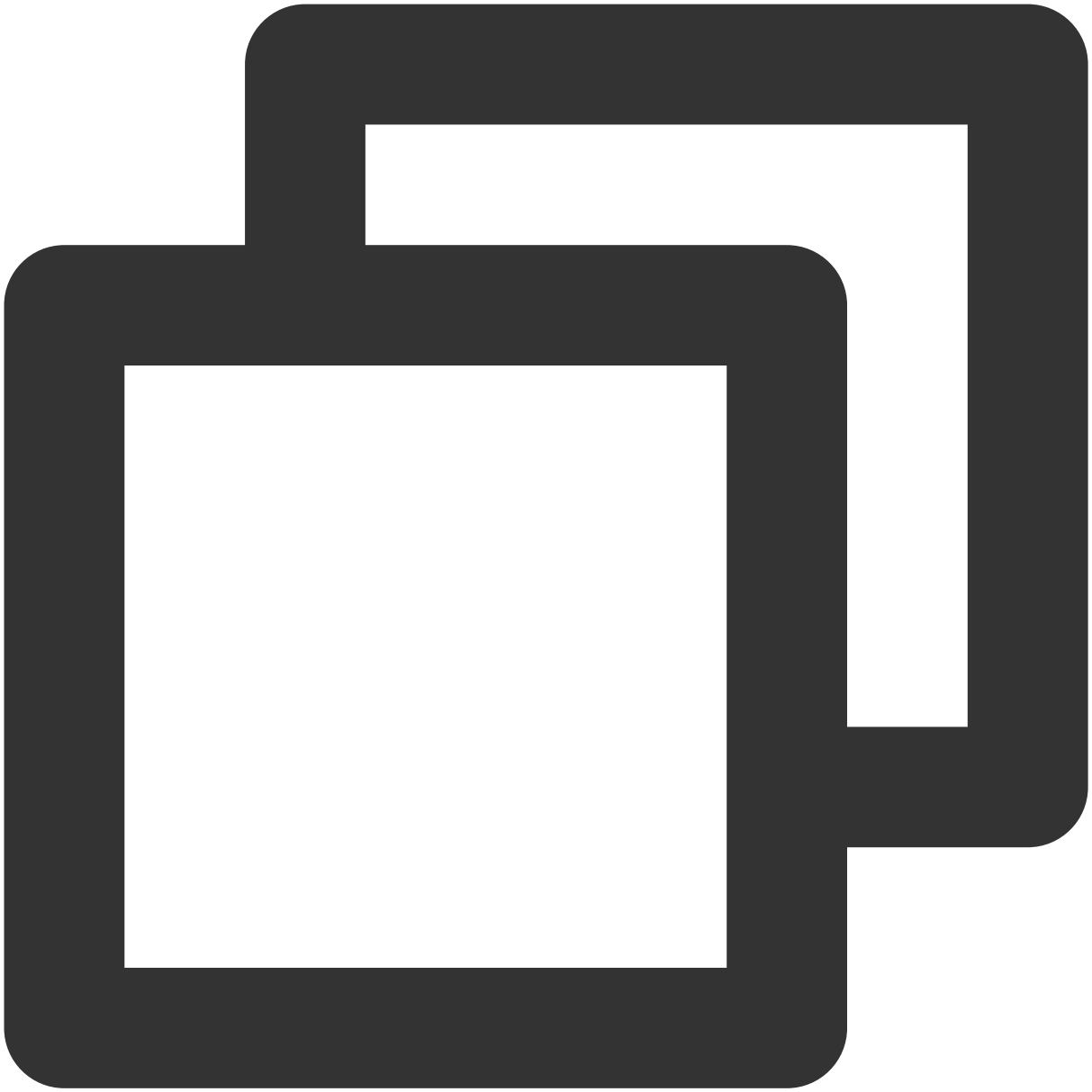
参数	说明
https	请求协议为 HTTPS 请求方式为 POST
xxxxxx	SDKAppID 所在国家/地区对应的专属域名。

	中国：`console.tim.qq.com` 新加坡：`adminapisgp.im.qcloud.com` 首尔：`adminapikr.im.qcloud.com` 法兰克福：`adminapiger.im.qcloud.com` 孟买：`adminapiind.im.qcloud.com` 硅谷：`adminapiusa.im.qcloud.com` **注意：** 目前印度站暂未上架推送插件。
v4/timpush/get_attr_name	请求接口
usersig	App 管理员账号生成的签名，参见 UserSig 后台 API
identifier	必须为 App 管理员账号
sdkappid	创建应用时即时通信控制台分配的 SdkAppid
random	32位无符号整数随机数
contenttype	固定值为：json

调用频率限制

每秒100次。

请求包示例

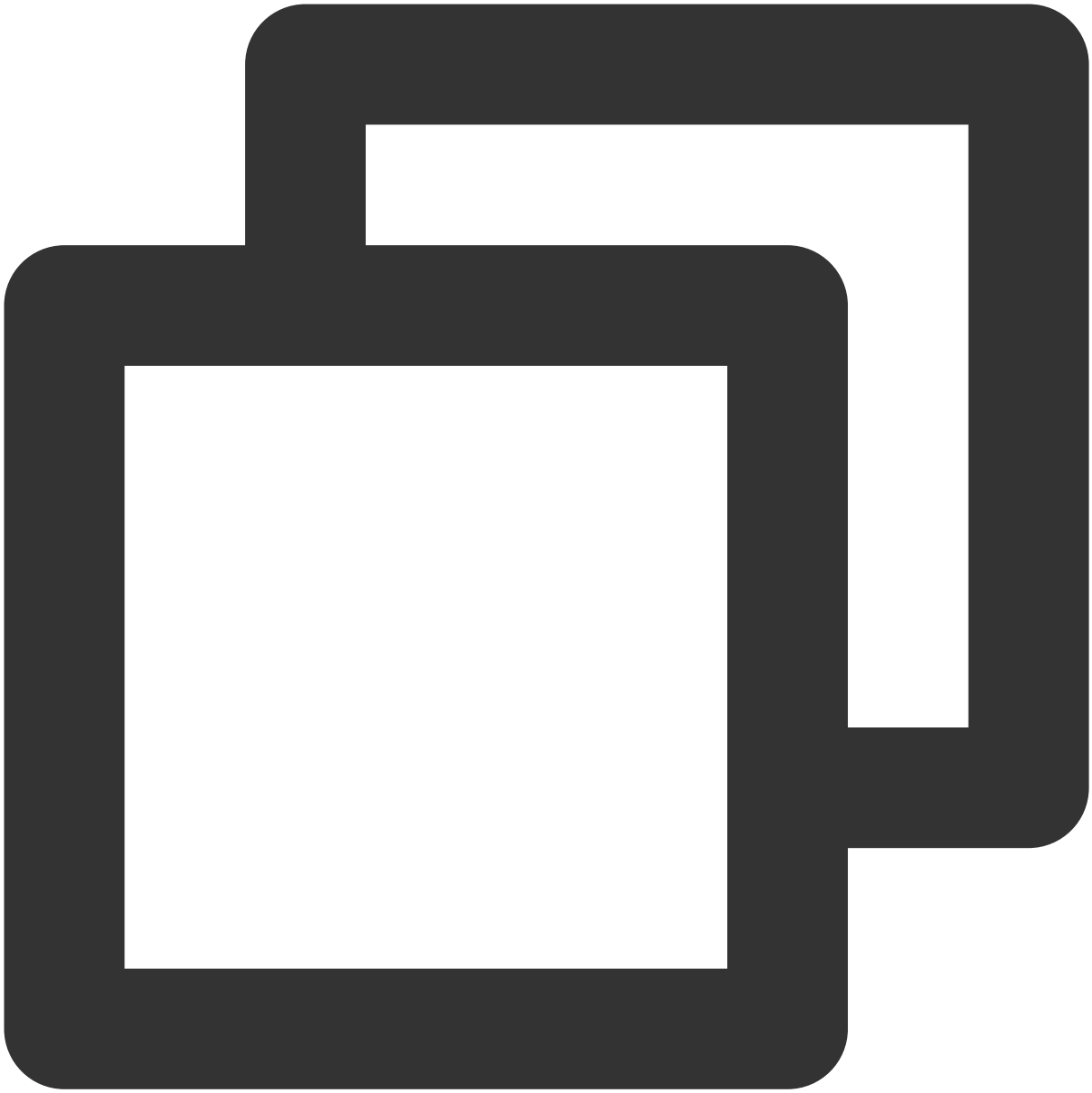


```
{}
```

请求包字段说明

无。

应答包体示例



```
{
    "ActionStatus": "OK",
    "ErrorInfo": "",
    "ErrorCode": 0,
    "AttrNames": {
        "0": "sex",
        "1": "city",
        "2": "会员等级"
    }
}
```

应答包字段说明

字段	类型	说明
ActionStatus	String	请求处理的结果： OK：表示处理成功 FAIL：表示失败
ErrorCode	Integer	错误码
ErrorInfo	String	错误信息
AttrNames	Object	包含多个键对。每对键值对，表示第几个属性对应的名称。例如"0":"xxx"表示第0号属性的名称是 xxx

错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。**真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的**。公共错误码（60000到79999）参见 错误码 文档。
本 API 私有错误码如下：

错误码	含义说明
90001	JSON 格式解析失败，请检查请求包是否符合 JSON 规范。
90009	请求需要 App 管理员权限。
90018	请求的账号数量超过限制。
91000	服务内部错误，请重试。

接口调试工具

通过 REST API 在线测试 工具调试本接口。

参考

发起全员/标签推送
设置应用属性名称
获取应用属性名称
设置用户属性

删除用户属性
获取用户属性
添加用户标签
获取用户标签
删除用户标签
清空用户标签
推送撤回

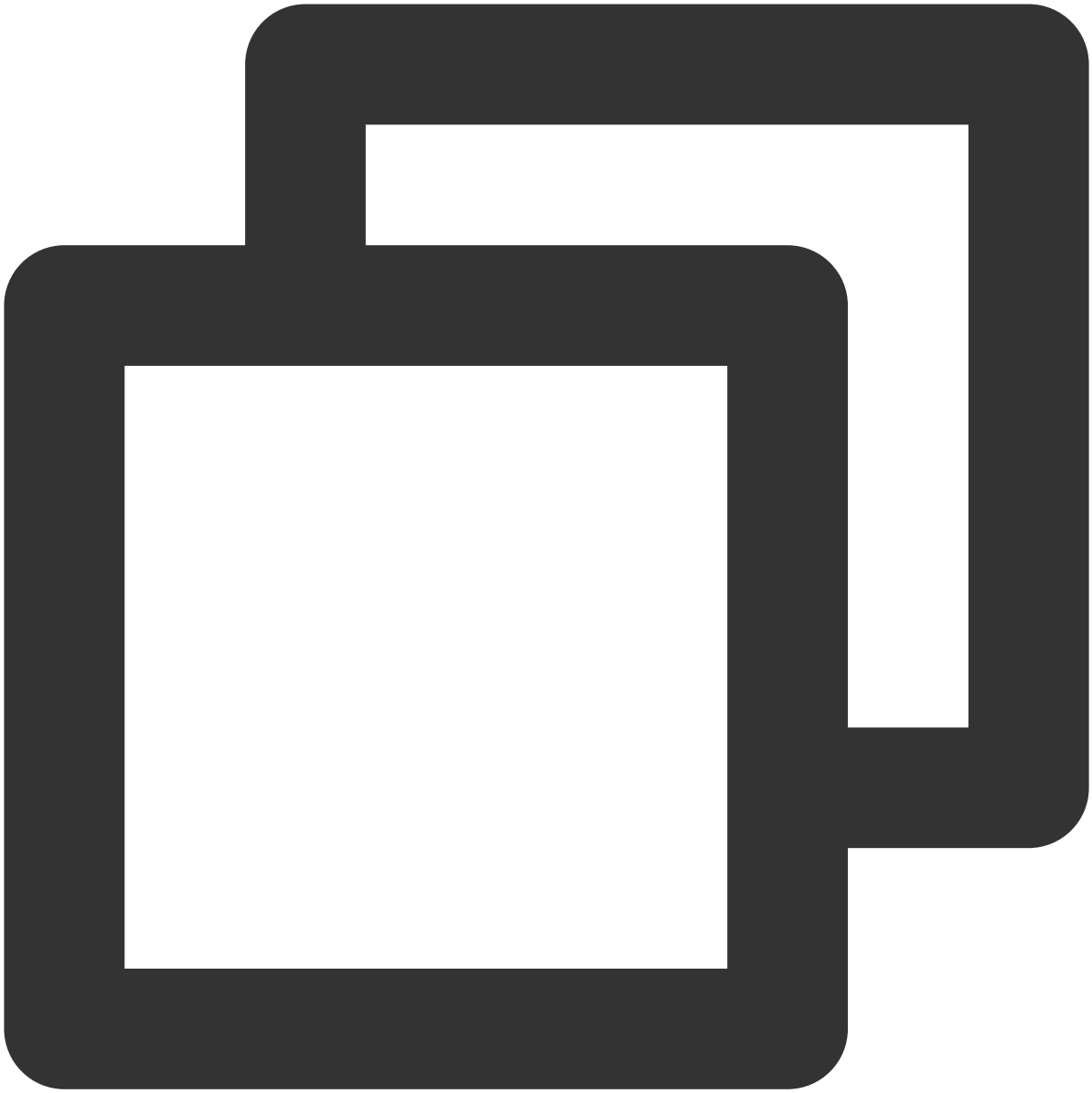
设置应用属性名称

最近更新时间：2024-06-13 10:39:26

功能说明

每个应用可以设置自定义的用户属性，最多可以有10个。通过本接口可以设置每个属性的名称，设置完成后，即可用于按用户属性推送等。

请求 URL 示例



```
https://xxxxxx/v4/timpush/set_attr_name?usersig=xxx&identifier=admin&sdkappid=88888
```

请求参数说明

参数	说明
https	请求协议：HTTPS 请求方式：POST
xxxxxx	SDKAppID 所在国家/地区对应的专属域名。

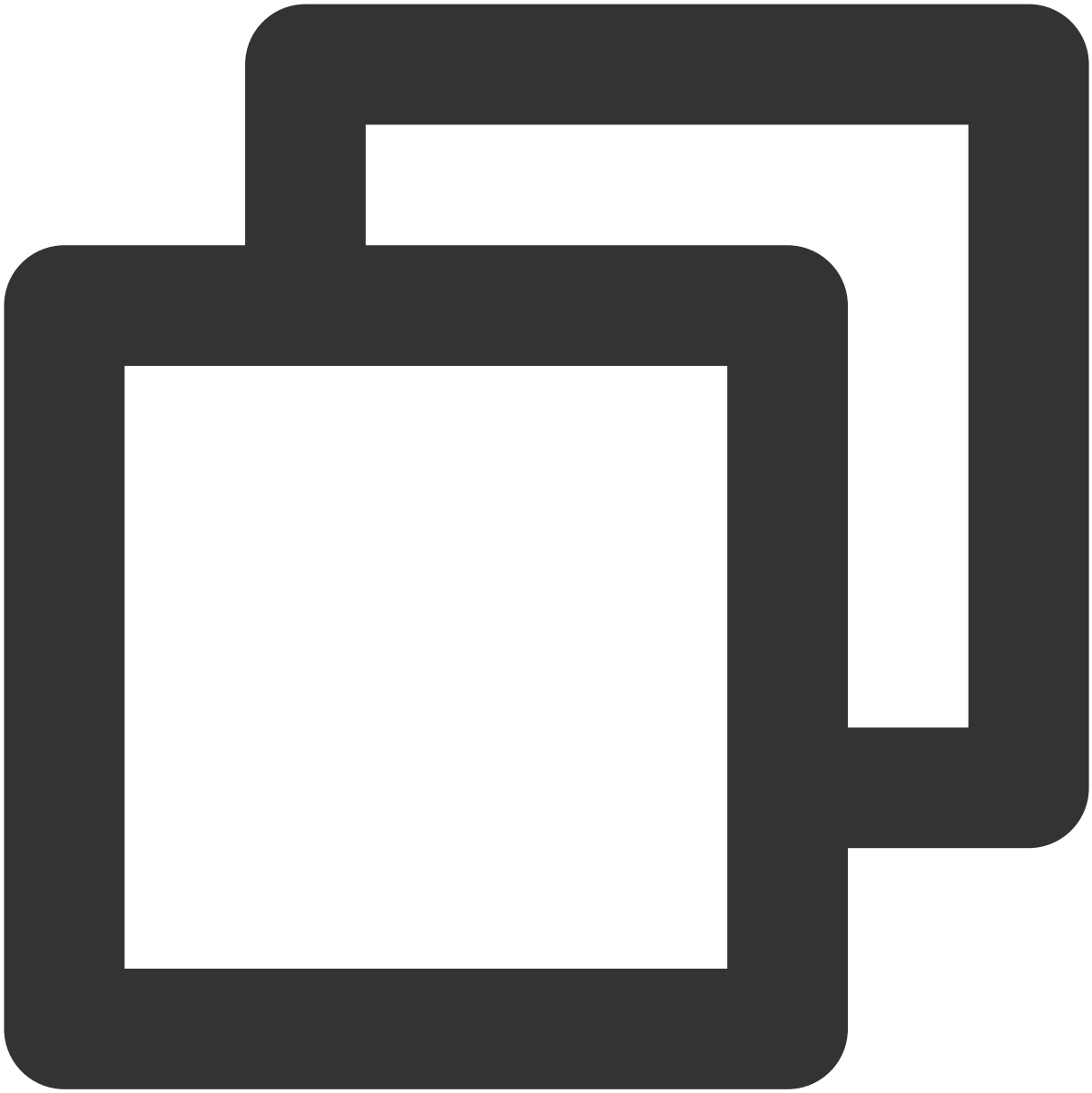
	中国：`console.tim.qq.com` 新加坡：`adminapisgp.im.qcloud.com` 首尔：`adminapikr.im.qcloud.com` 法兰克福：`adminapiger.im.qcloud.com` 孟买：`adminapiind.im.qcloud.com` 硅谷：`adminapiusa.im.qcloud.com` **注意：** 目前印度站暂未上架推送插件。
v4/timpush/set_attr_name	请求接口
usersig	App 管理员账号生成的签名，参见 UserSig 后台 API
identifier	必须为 App 管理员账号
sdkappid	创建应用时即时通信控制台分配的 SdkAppid
random	32位无符号整数随机数
contenttype	固定值为：json

调用频率限制

每秒100次。

请求包示例

设置应用第0号属性表示性别，第1号属性表示城市，第2号属性表示国家。

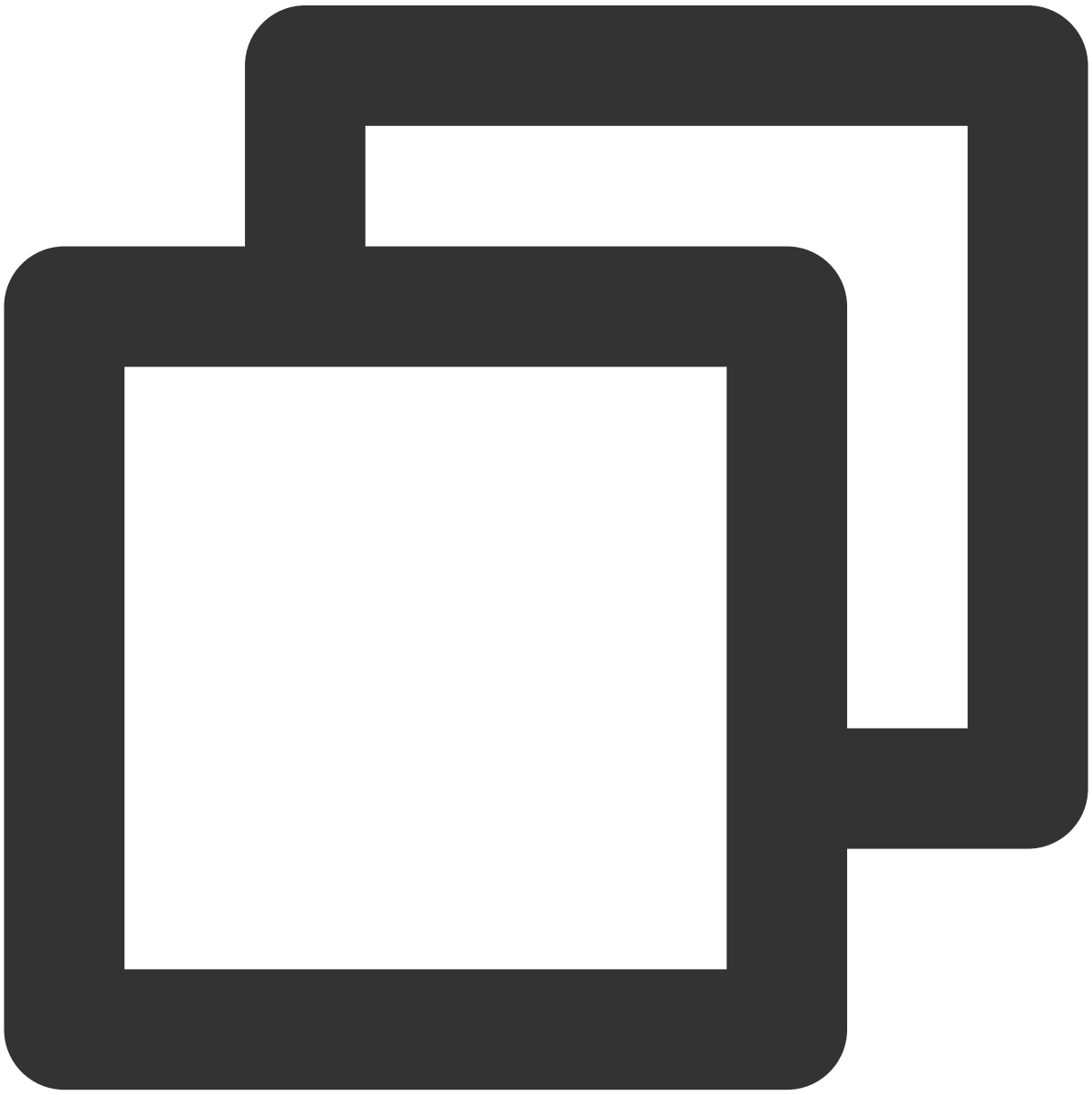


```
{
    "AttrNames": {
        "0": "sex",
        "1": "city",
        "2": "country"
    }
}
```

请求包字段说明

字段	类型	属性	说明
数字键	String	必填	表示第几个属性（“0”到“9”之间）
属性名	String	必填	属性名最长不超过50字节。应用最多可以有10个推送属性（编号从0到9），用户自定义每个属性的含义

应答包体示例



```
{
    "ActionStatus": "OK",
    "ErrorInfo": "",
    "ErrorCode": 0
}
```

应答包字段说明

字段	类型	说明
ActionStatus	String	请求处理的结果： OK：表示处理成功 FAIL：表示失败
ErrorCode	Integer	错误码
ErrorInfo	String	错误信息

错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。**真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的**。公共错误码（60000到79999）参见 错误码 文档。
本 API 私有错误码如下：

错误码	含义说明
90001	JSON 格式解析失败，请检查请求包是否符合 JSON 规范。
90009	请求需要 App 管理员权限。
91000	服务内部错误，请重试。

接口调试工具

通过 REST API 在线测试 工具调试本接口。

参考

发起全员/标签推送
设置应用属性名称

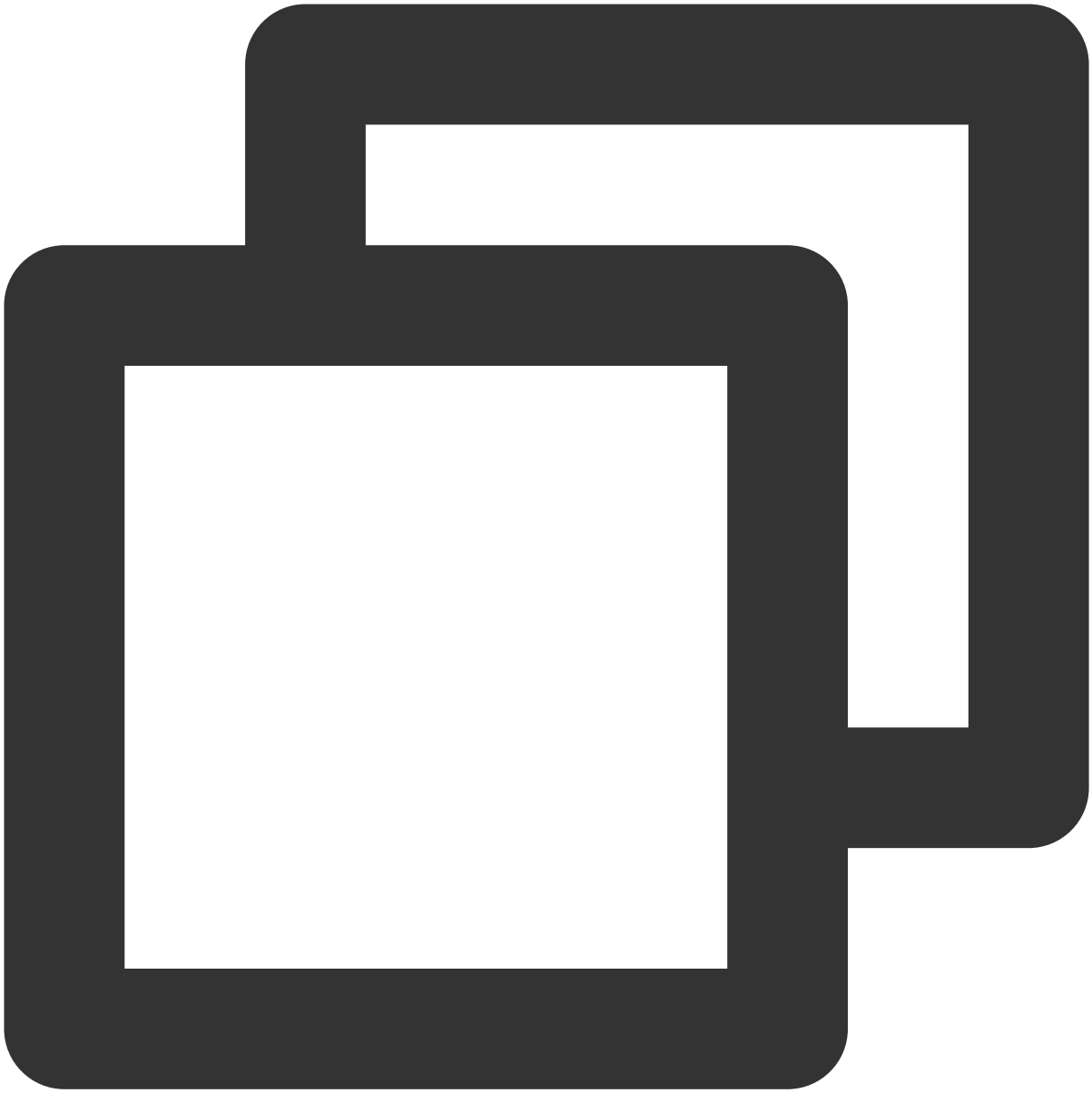
获取用户属性

最近更新时间：2024-06-13 10:39:26

功能说明

获取用户属性（必须以管理员账号调用）；每次最多只能获取100个用户的属性。使用前请先 设置应用属性名称 。

请求 URL 示例



```
https://xxxxxx/v4/timpush/get_attr?usersig=xxx&identifier=admin&sdkappid=88888888&r
```

请求参数说明

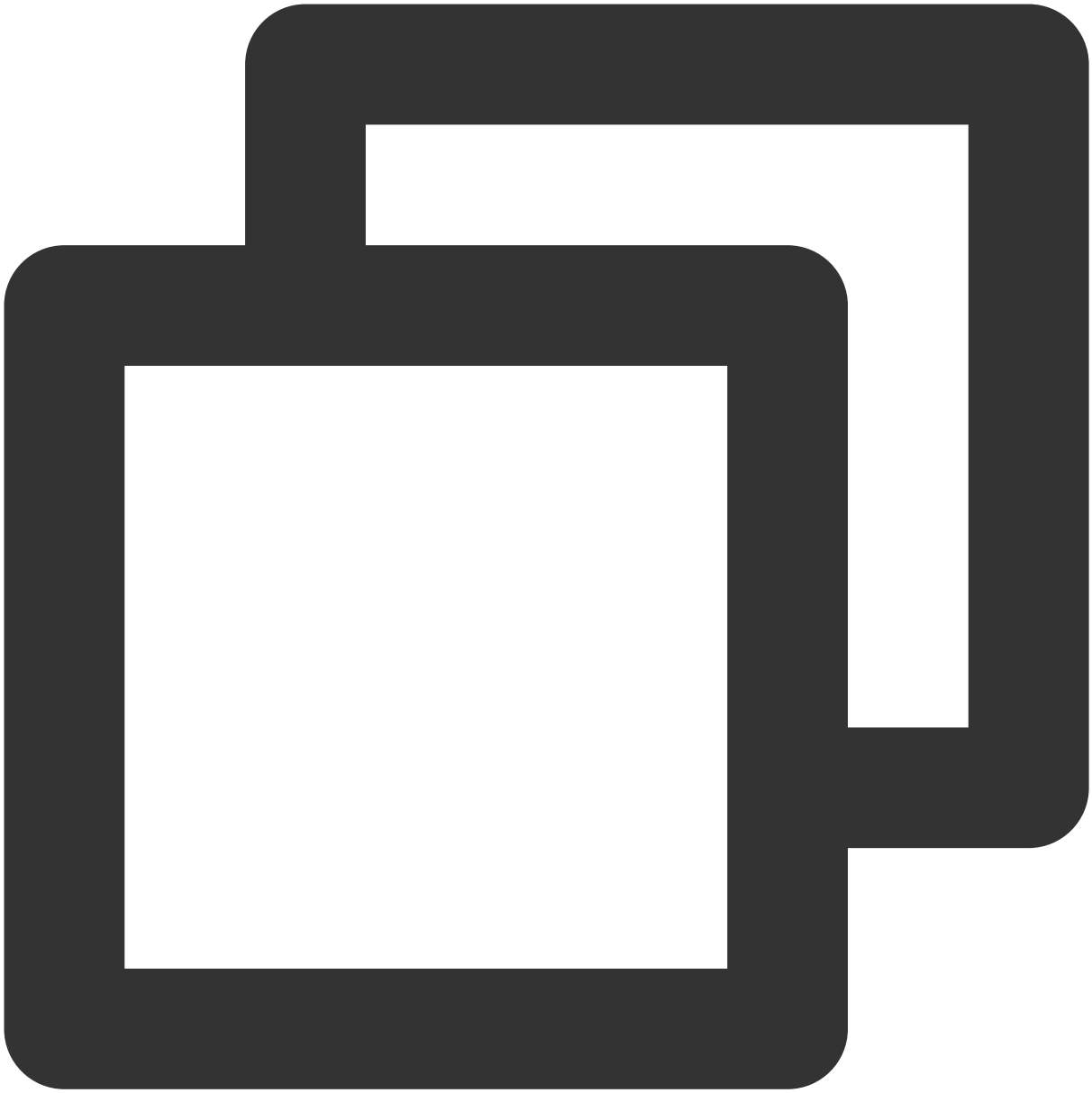
参数	说明
https	请求协议：HTTPS 请求方式：POST
xxxxxx	SDKAppID 所在国家/地区对应的专属域名。

	中国：`console.tim.qq.com` 新加坡：`adminapisgp.im.qcloud.com` 首尔：`adminapikr.im.qcloud.com` 法兰克福：`adminapiger.im.qcloud.com` 孟买：`adminapiind.im.qcloud.com` 硅谷：`adminapiusa.im.qcloud.com` **注意：** 目前印度站暂未上架推送插件。
v4/timpush/get_attr	请求接口
usersig	App 管理员账号生成的签名，参见 UserSig 后台 API
identifier	必须为 App 管理员账号
sdkappid	创建应用时即时通信控制台分配的 SdkAppid
random	32位无符号整数随机数
contenttype	固定值为：json

调用频率限制

每秒100次。

请求包示例

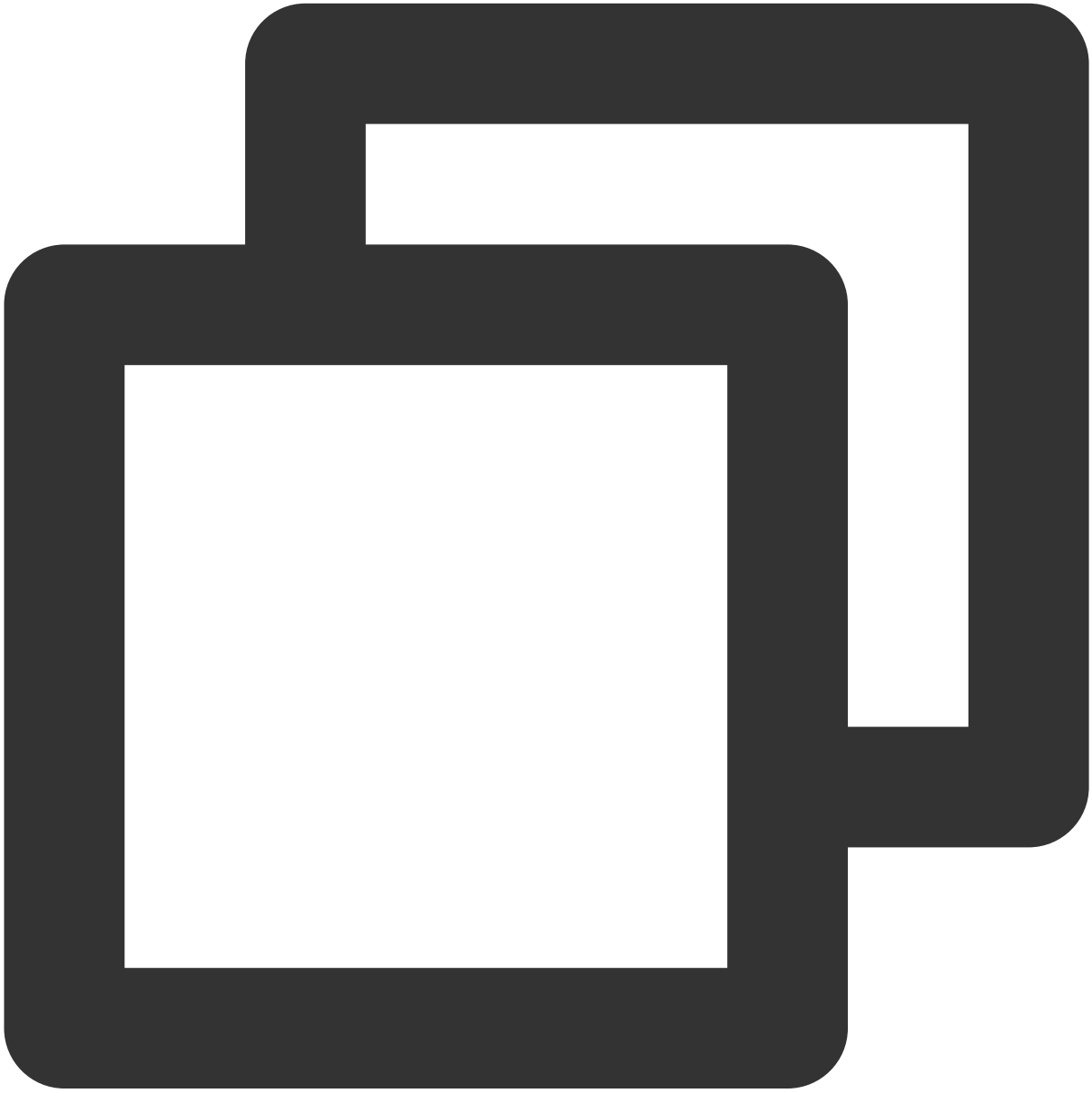


```
{
    "To_Account": [
        "张小红",
        "陈小明",
        "abc"
    ]
}
```

请求包字段说明

字段	类型	属性	说明
To_Account	Array	必填	目标用户账号列表

应答包体示例



```
{
    "ActionStatus": "OK",
    "ErrorInfo": "",
    "ErrorCode": 0,
```

```
    "UserAttrs": [
        {
            "To_Account": "张小红",
            "Attrs": {
                "sex": "女",
                "city": "纽约"
            }
        },
        {
            "To_Account": "abc",
            "Attrs": {}
        },
        {
            "To_Account": "陈小明",
            "Attrs": {
                "sex": "男",
                "city": "深圳"
            }
        }
    ]
}
```

应答包字段说明

字段	类型	说明
ActionStatus	String	请求处理的结果： OK：表示处理成功 FAIL：表示失败
ErrorCode	Integer	错误码
ErrorInfo	String	错误信息
UserAttrs	Array	用户标签内容列表
To_Account	String	用户账号
Attrs	Object	属性内容

错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。**真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的**。公共错误码（60000到79999）参见 错误码 文档。

本 API 私有错误码如下：

错误码	含义说明
90001	JSON 格式解析失败，请检查请求包是否符合 JSON 规范。
90018	请求的账号数量超过限制。
91000	服务内部错误，请重试。

接口调试工具

通过 REST API 在线测试 工具调试本接口。

参考

发起全员/标签推送
设置应用属性名称
获取应用属性名称
设置用户属性
删除用户属性
获取用户属性
添加用户标签
获取用户标签
删除用户标签
清空用户标签
推送撤回

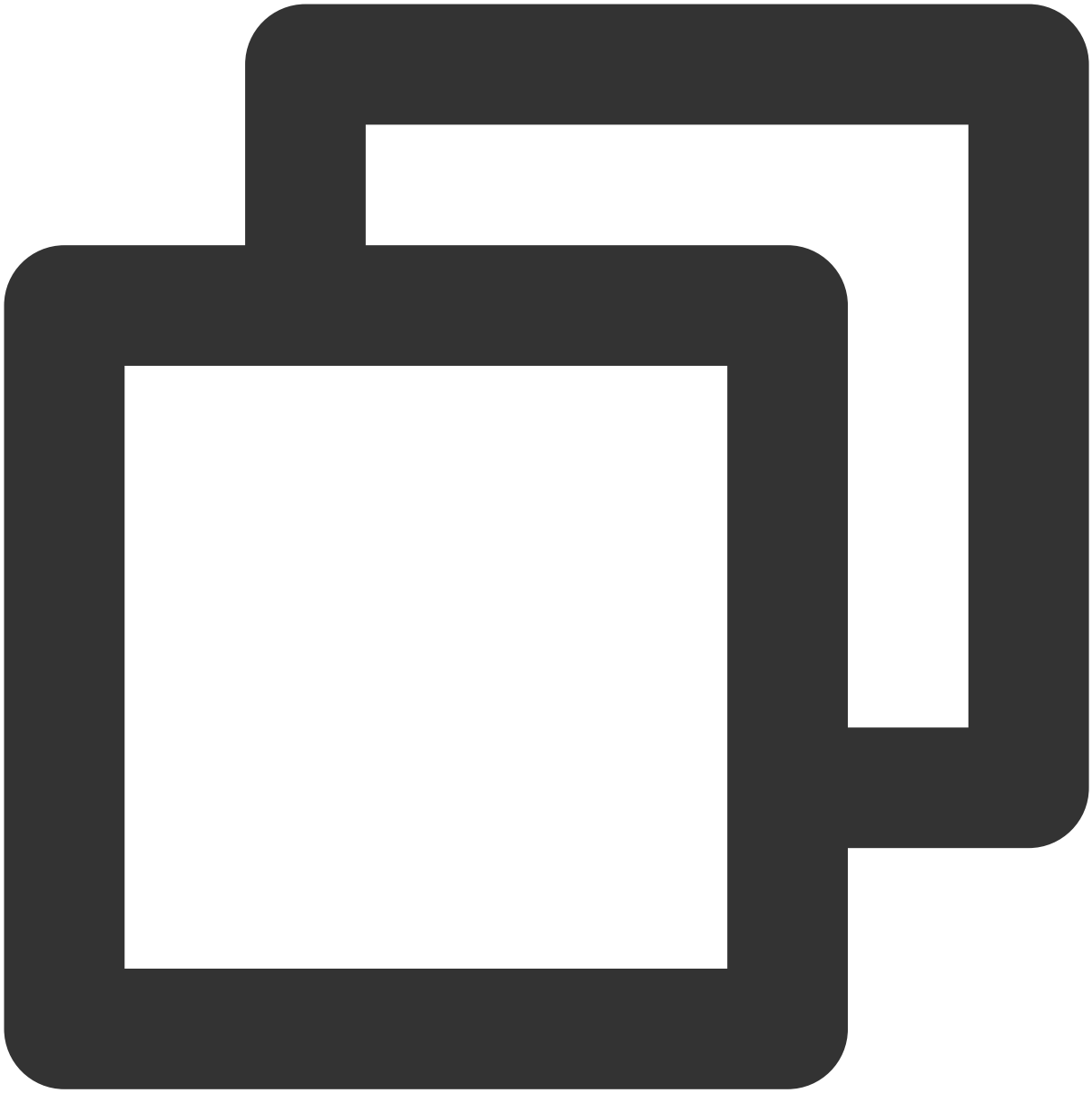
设置用户属性

最近更新时间：2024-06-13 10:39:26

功能说明

管理员给用户设置属性。每次最多只能给100个用户设置属性。使用前请先 设置应用属性名称 。

请求 URL 示例



```
https://xxxxxx/v4/timpush/set_attr?usersig=xxx&identifier=admin&sdkappid=88888888&r
```

请求参数说明

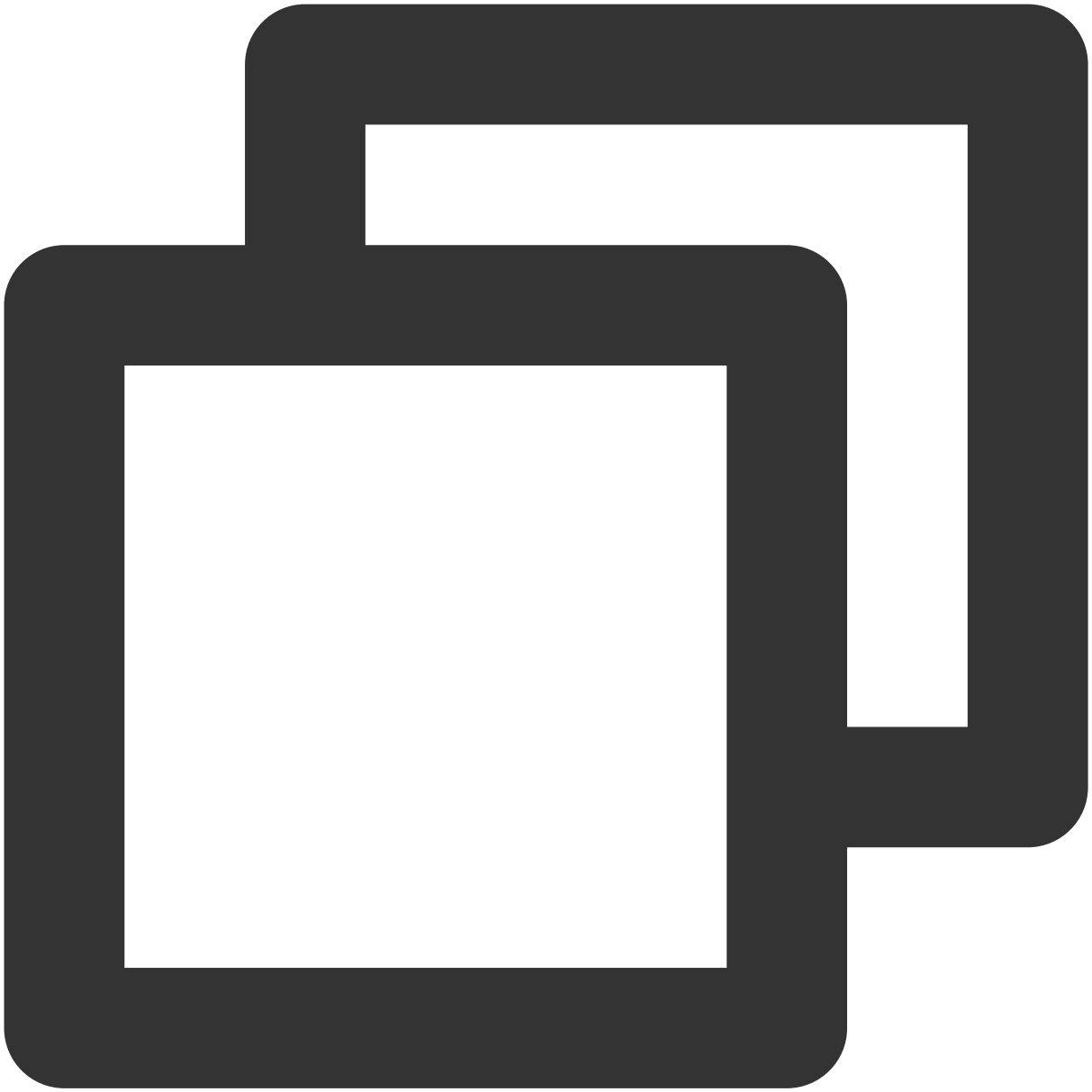
参数	说明
https	请求协议为：HTTPS 请求方式为：POST
xxxxxx	SDKAppID 所在国家/地区对应的专属域名。

	中国：`console.tim.qq.com` 新加坡：`adminapisgp.im.qcloud.com` 首尔：`adminapikr.im.qcloud.com` 法兰克福：`adminapiger.im.qcloud.com` 孟买：`adminapiind.im.qcloud.com` 硅谷：`adminapiusa.im.qcloud.com` **注意：** 目前印度站暂未上架推送插件。
v4/timpush/set_attr	请求接口
usersig	App 管理员账号生成的签名，参见 UserSig 后台 API
identifier	必须为 App 管理员账号
sdkappid	创建应用时即时通信控制台分配的 SdkAppid
random	32位无符号整数随机数
contenttype	固定值为：json

调用频率限制

每秒100次。

请求包示例



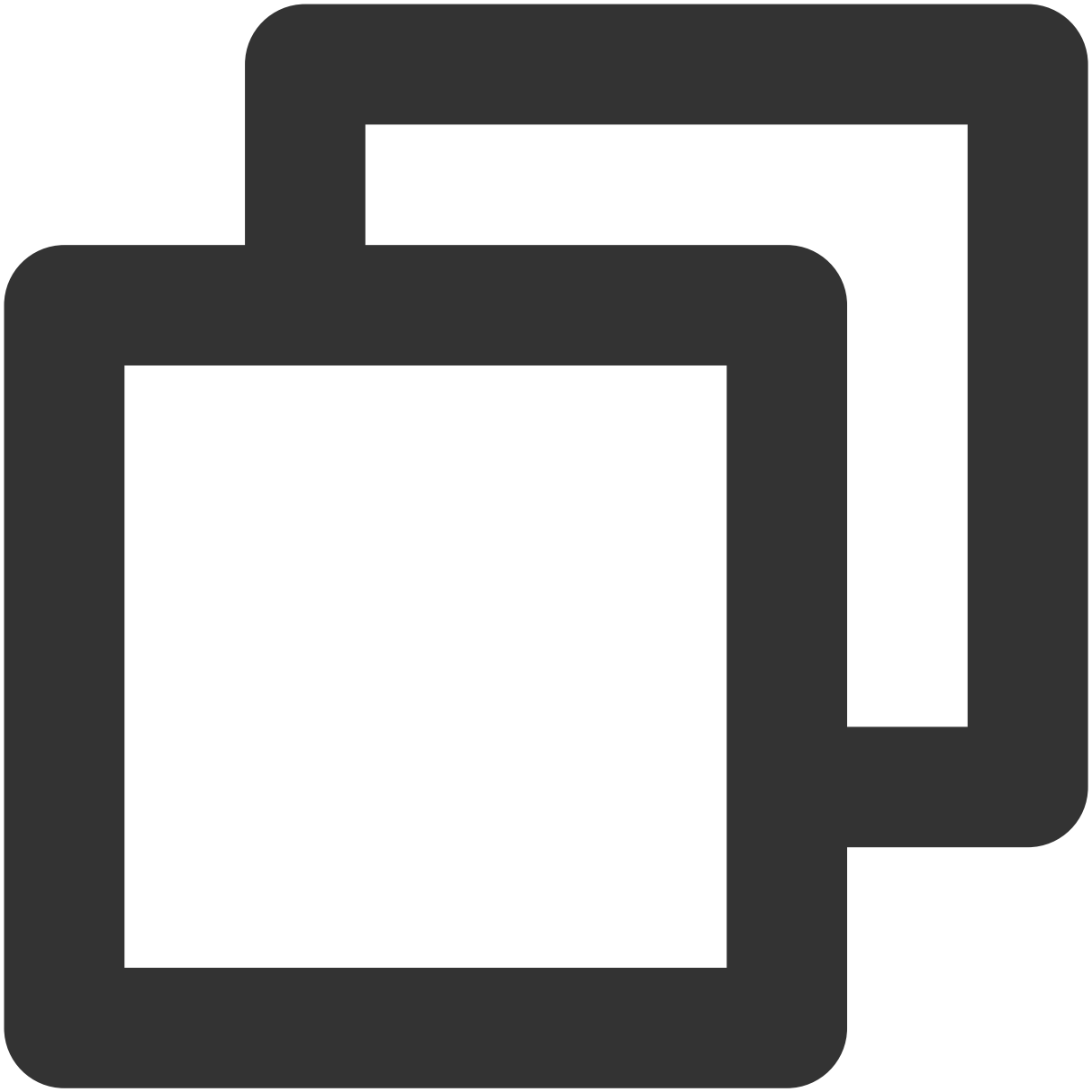
```
{
    "UserAttrs":
    [
        {
            "To_Account": "xiaojun012",
            "Attrs": {
                "sex": "attr1",
                "city": "attr2"
            }
        },
        {
```

```
            "To_Account": "xiaojun013",
            "Attrs": {
                "city": "attr3",
                "sex": "attr4"
            }
        }
    ]
}
```

请求包字段说明

字段	类型	属性	说明
To_Account	String	必填	目标用户账号
Attrs	Object	必填	属性集合。每个属性是一个键值对，键为属性名，值为该用户对应的属性值。用户属性值不能超过50字节

应答包体示例



```
{
    "ActionStatus": "OK",
    "ErrorInfo": "",
    "ErrorCode": 0
}
```

应答包字段说明

字段	类型	说明

ActionStatus	String	请求处理的结果： OK：表示处理成功 FAIL：表示失败
ErrorCode	Integer	错误码
ErrorInfo	String	错误信息

错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。**真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的**。公共错误码（60000到79999）参见 错误码 文档。
本 API 私有错误码如下：

错误码	含义说明
90001	JSON 格式解析失败，请检查请求包是否符合 JSON 规范。
90009	请求需要 App 管理员权限。
90018	请求的账号数量超过限制。
90033	属性无效。
91000	服务内部错误，请重试。

接口调试工具

通过 REST API 在线测试 工具调试本接口。

参考

发起全员/标签推送
设置应用属性名称
获取应用属性名称
设置用户属性
删除用户属性
获取用户属性
添加用户标签
获取用户标签

删除用户标签

清空用户标签

推送撤回

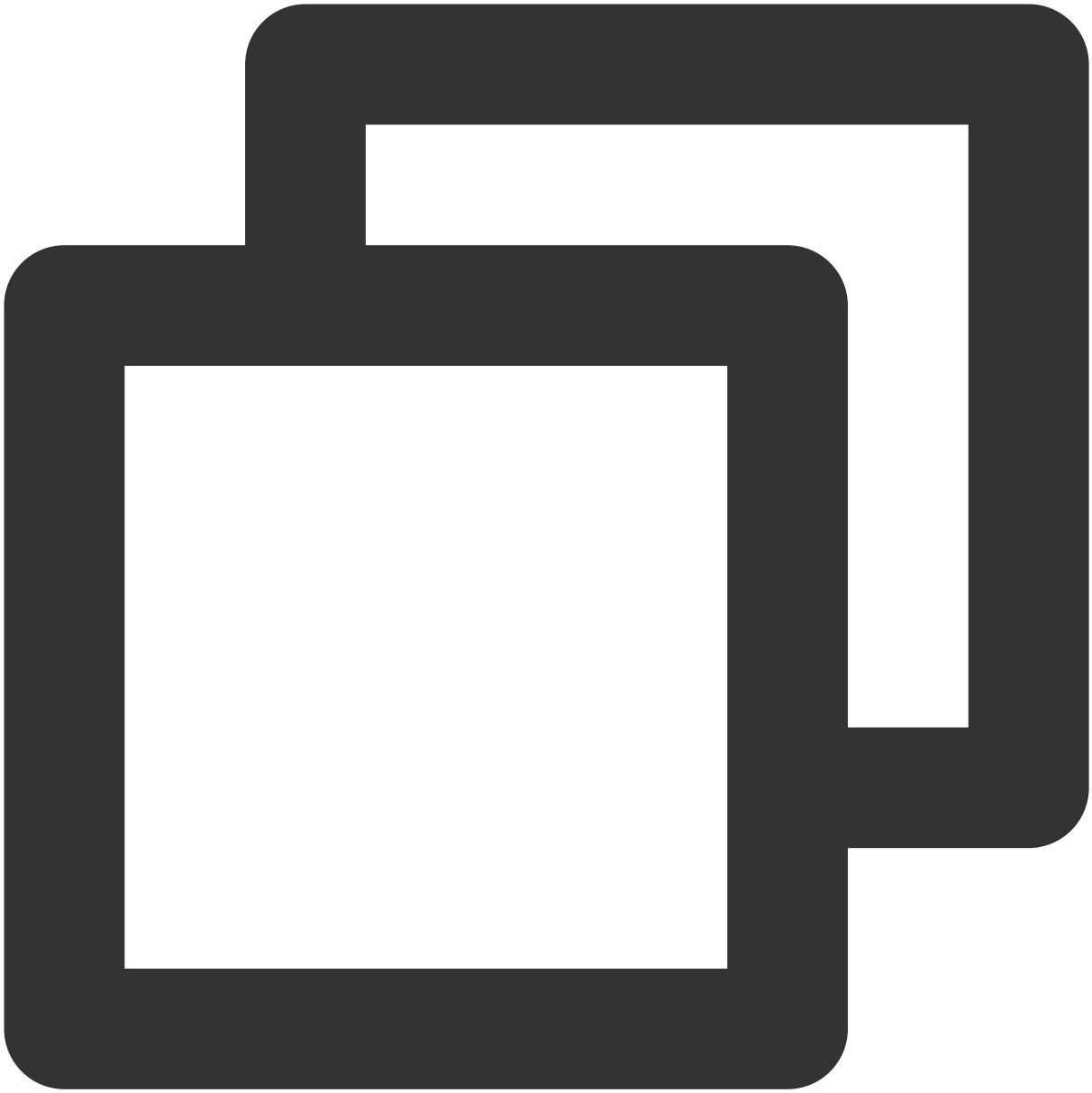
删除用户属性

最近更新时间：2024-06-13 10:39:26

功能说明

管理员给用户删除属性。注意每次最多只能给100个用户删除属性。使用前请先 设置应用属性名称。

请求 URL 示例



```
https://xxxxxx/v4/timpush/remove_attr?usersig=xxx&identifier=admin&sdkappid=8888888
```

请求参数说明

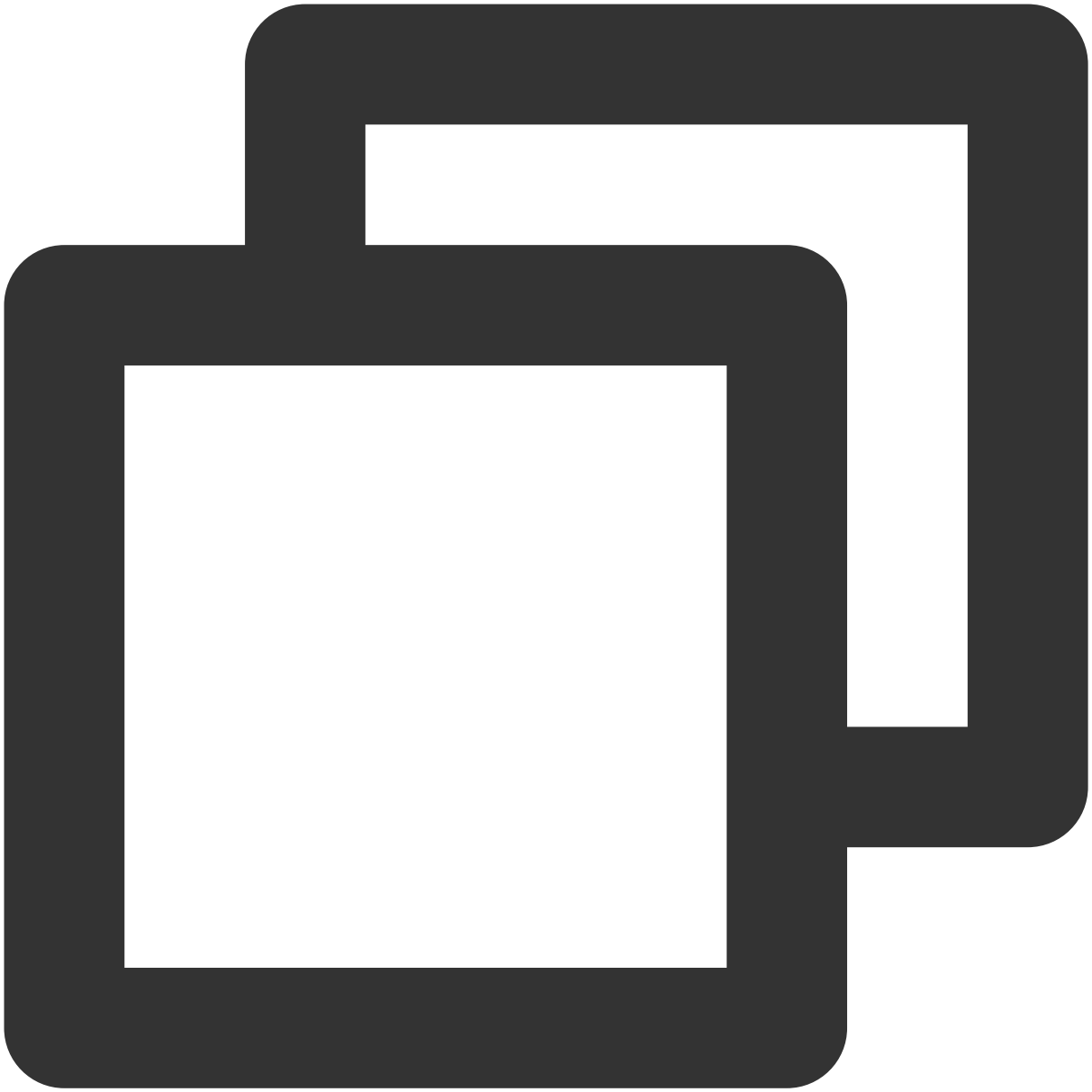
参数	说明
https	请求协议：HTTPS 请求方式：POST
xxxxxx	SDKAppID 所在国家/地区对应的专属域名。

	中国：`console.tim.qq.com` 新加坡：`adminapisgp.im.qcloud.com` 首尔：`adminapikr.im.qcloud.com` 法兰克福：`adminapiger.im.qcloud.com` 孟买：`adminapiind.im.qcloud.com` 硅谷：`adminapiusa.im.qcloud.com` **注意：** 目前印度站暂未上架推送插件。
v4/timpush/remove_attr	请求接口
usersig	App 管理员账号生成的签名，参见 UserSig 后台 API
identifier	必须为 App 管理员账号
sdkappid	创建应用时即时通信控制台分配的 SdkAppid
random	32位无符号整数随机数
contenttype	固定值为：json

调用频率限制

每秒100次。

请求包示例



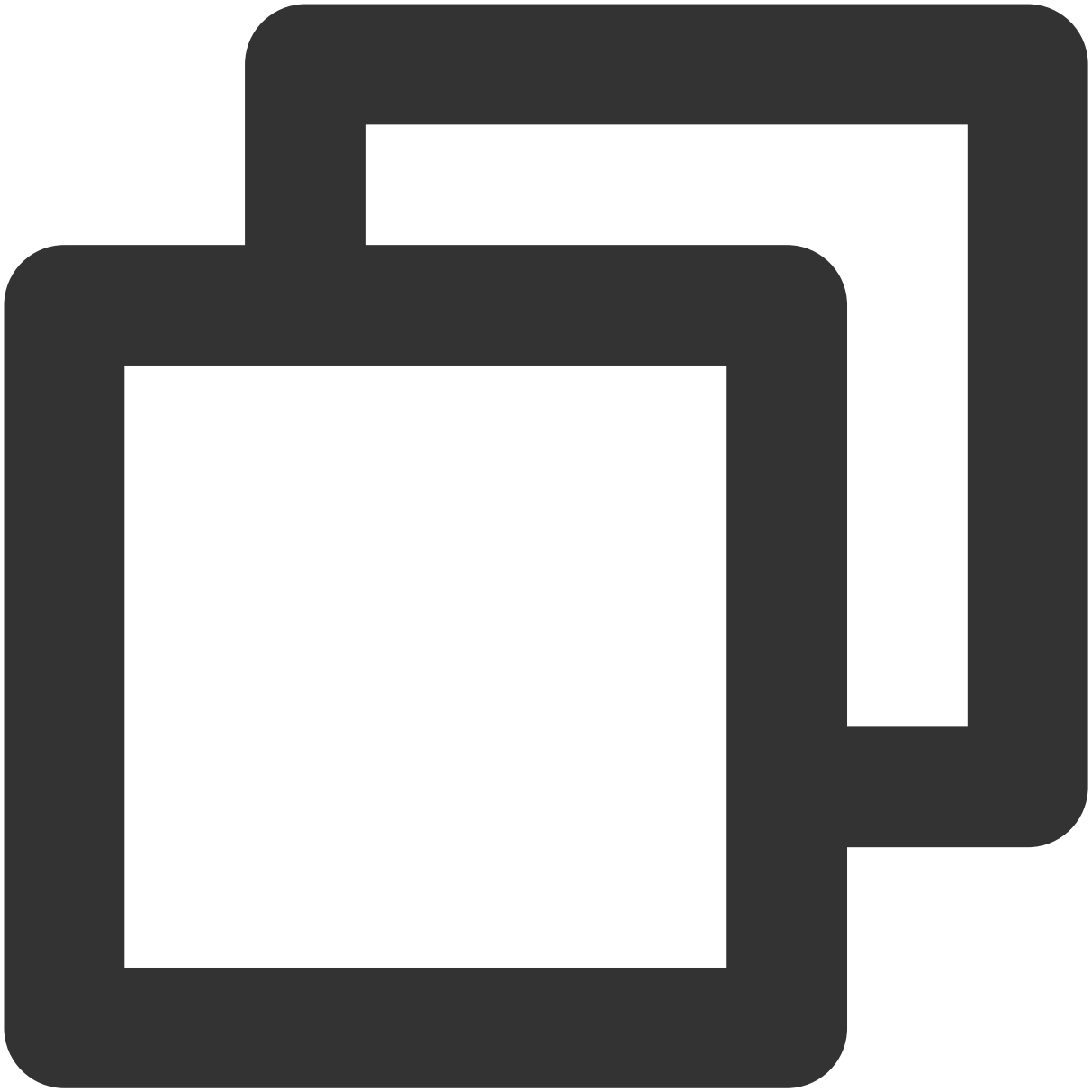
```
{
    "UserAttrs": [
        {
            "To_Account": "xiaojun013",
            "Attrs": [
                "sex",
                "city"
            ]
        },
        {
            "To_Account": "xiaojun012",
```

```
            "Attrs": [
                "sex",
                "city"
            ]
        }
    ]
}
```

请求包字段说明

字段	类型	属性	说明
To_Account	String	必填	目标用户账号
Attrs	Array	必填	属性集合，注意这里只需要给出属性名即可；Attrs 形式及含义参见 设置应用属性名称 。

应答包体示例



```
{
    "ActionStatus": "OK",
    "ErrorInfo": "",
    "ErrorCode": 0
}
```

应答包字段说明

字段	类型	说明

ActionStatus	String	请求处理的结果： OK：表示处理成功 FAIL：表示失败
ErrorCode	Integer	错误码
ErrorInfo	String	错误信息

错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。**真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的**。公共错误码（60000到79999）参见 错误码 文档。
本 API 私有错误码如下：

错误码	含义说明
90001	JSON 格式解析失败，请检查请求包是否符合 JSON 规范
90009	请求需要 App 管理员权限
90018	请求的账号数量超过限制
90033	属性无效
91000	服务内部错误，请重试

接口调试工具

通过 REST API 在线测试 工具调试本接口。

参考

发起全员/标签推送
设置应用属性名称
获取应用属性名称
设置用户属性
删除用户属性
获取用户属性
添加用户标签
获取用户标签

删除用户标签

清空用户标签

推送撤回

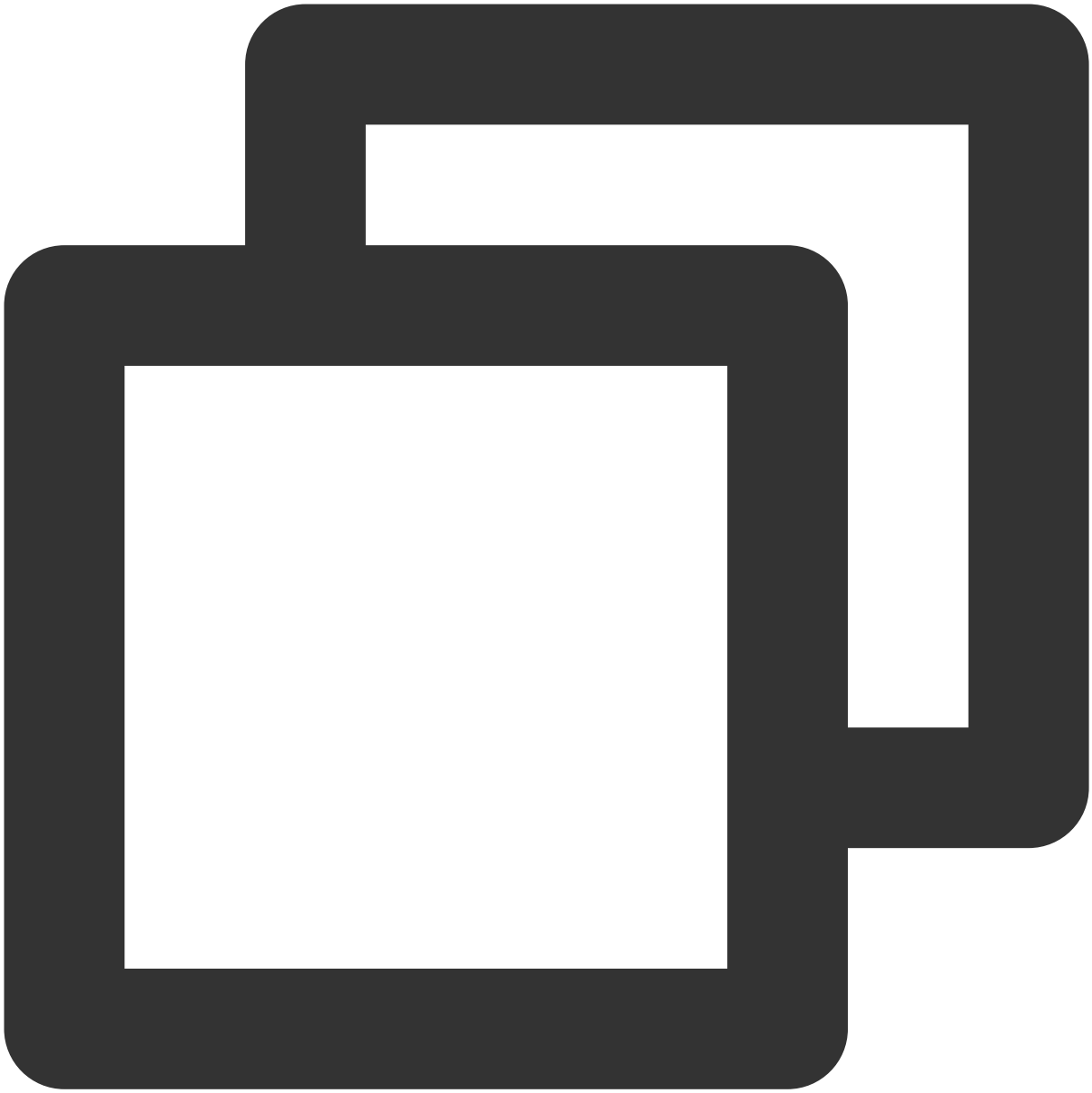
获取用户标签

最近更新时间：2024-06-13 10:39:26

功能说明

获取用户标签（必须以管理员账号调用）。每次最多只能获取100个用户的标签。

请求 URL 示例



```
https://xxxxxx/v4/timpush/get_tag?usersig=xxx&identifier=admin&sdkappid=88888888&ra
```

请求参数说明

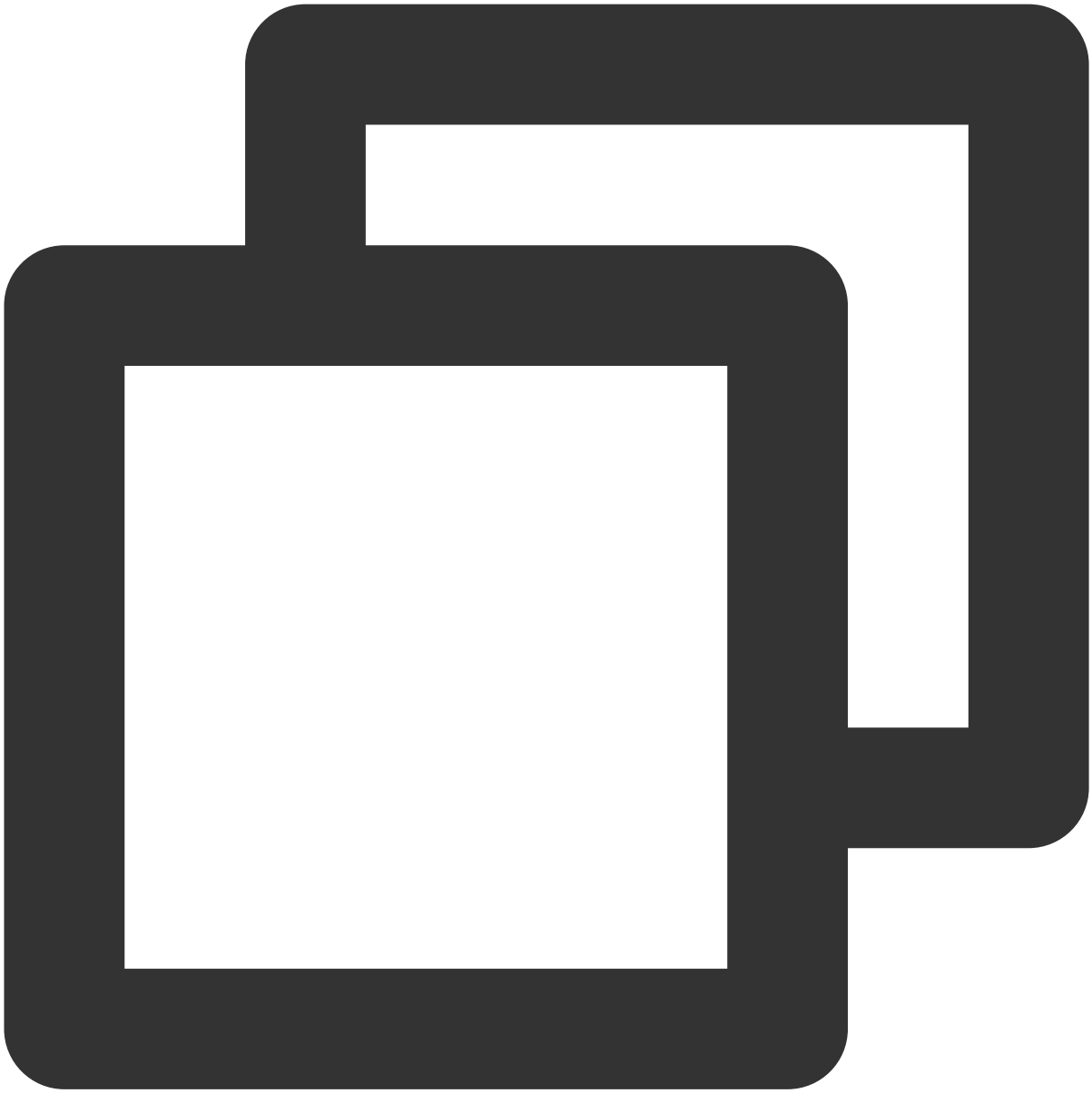
参数	说明
https	请求协议为：HTTPS 请求方式为：POST
xxxxxx	SDKAppID 所在国家/地区对应的专属域名。

	中国：`console.tim.qq.com` 新加坡：`adminapisgp.im.qcloud.com` 首尔：`adminapikr.im.qcloud.com` 法兰克福：`adminapiger.im.qcloud.com` 孟买：`adminapiind.im.qcloud.com` 硅谷：`adminapiusa.im.qcloud.com` **注意：** 目前印度站暂未上架推送插件。
v4/timpush/get_tag	请求接口
usersig	App 管理员账号生成的签名，参见 UserSig 后台 API
identifier	必须为 App 管理员账号
sdkappid	创建应用时即时通信控制台分配的 SdkAppid
random	32位无符号整数随机数
contenttype	固定值为：json

调用频率限制

每秒100次。

请求包示例



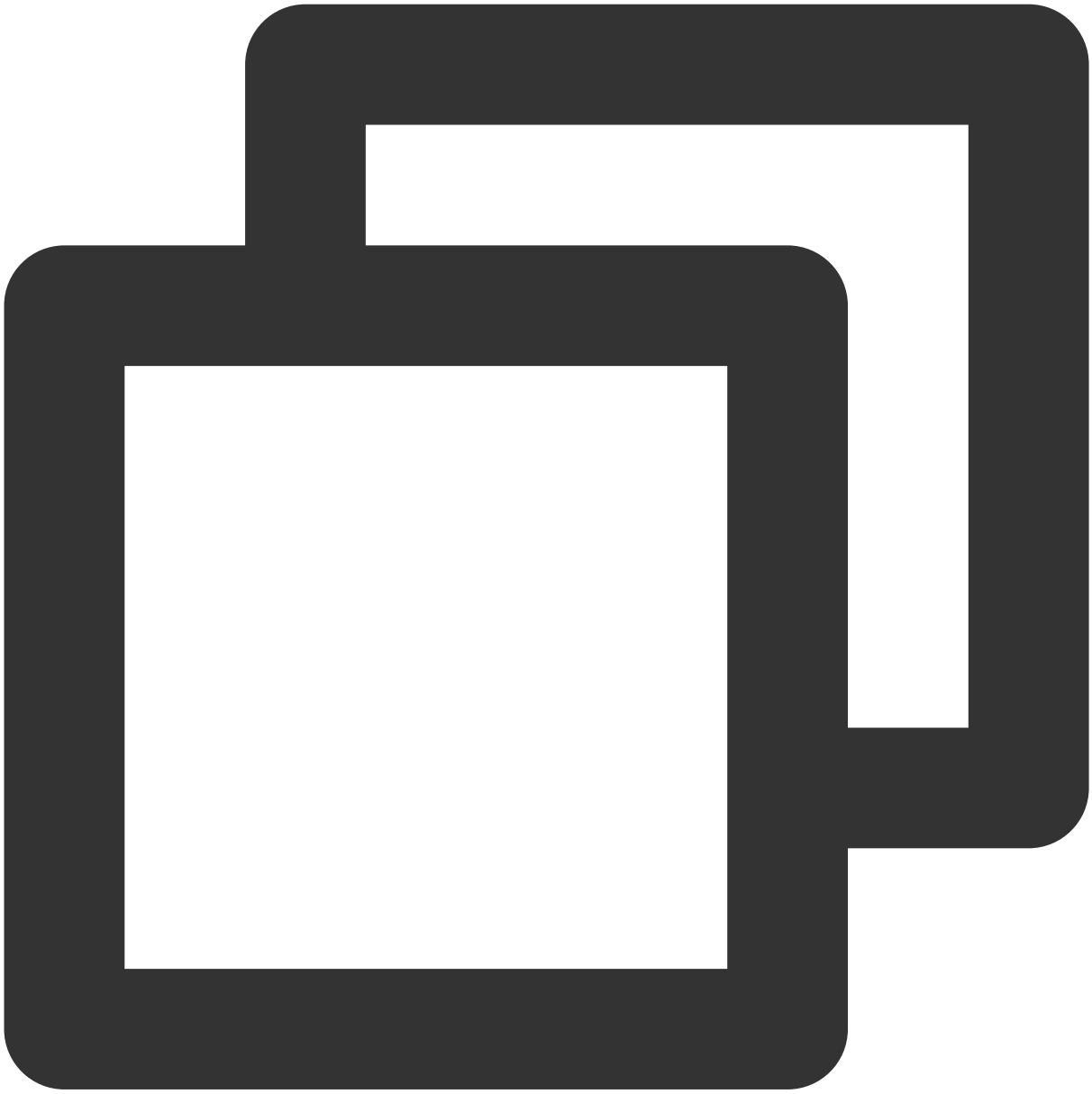
```
{
    "To_Account": [
        "xiaojun012",
        "xiaojun013"
    ]
}
```

请求包字段说明

字段	类型	属性	说明

To_Account	Array	必填	目标用户账号列表

应答包体示例



```
{
    "ActionStatus": "OK",
    "ErrorInfo": "",
    "ErrorCode": 0,
    "UserTags": [
        {
```

```
            "To_Account": "xiaojun012",
            "Tags": ["a", "b"]
        },
        {
            "To_Account": "xiaojun013",
            "Tags": ["a", "c"]
        }
    ]
}
```

应答包字段说明

字段	类型	说明
ActionStatus	String	请求处理的结果： OK：表示处理成功 FAIL：表示失败
ErrorCode	Integer	错误码
ErrorInfo	String	错误信息
UserTags	Array	用户标签内容列表
To_Account	String	用户账号
Tags	Array	Tags 内容

错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。**真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的**。公共错误码（60000到79999）参见 错误码 文档。

本 API 私有错误码如下：

错误码	含义说明
90001	JSON 格式解析失败，请检查请求包是否符合 JSON 规范。
90009	请求需要 App 管理员权限。
90018	请求的账号数量超过限制。
91000	服务内部错误，请重试。

接口调试工具

通过 REST API 在线测试 工具调试本接口。

参考

发起全员/标签推送
设置应用属性名称
获取应用属性名称
设置用户属性
删除用户属性
获取用户属性
添加用户标签
获取用户标签
删除用户标签
清空用户标签
推送撤回

添加用户标签

最近更新时间：2024-06-13 10:39:26

功能说明

管理员给用户添加标签。

注意

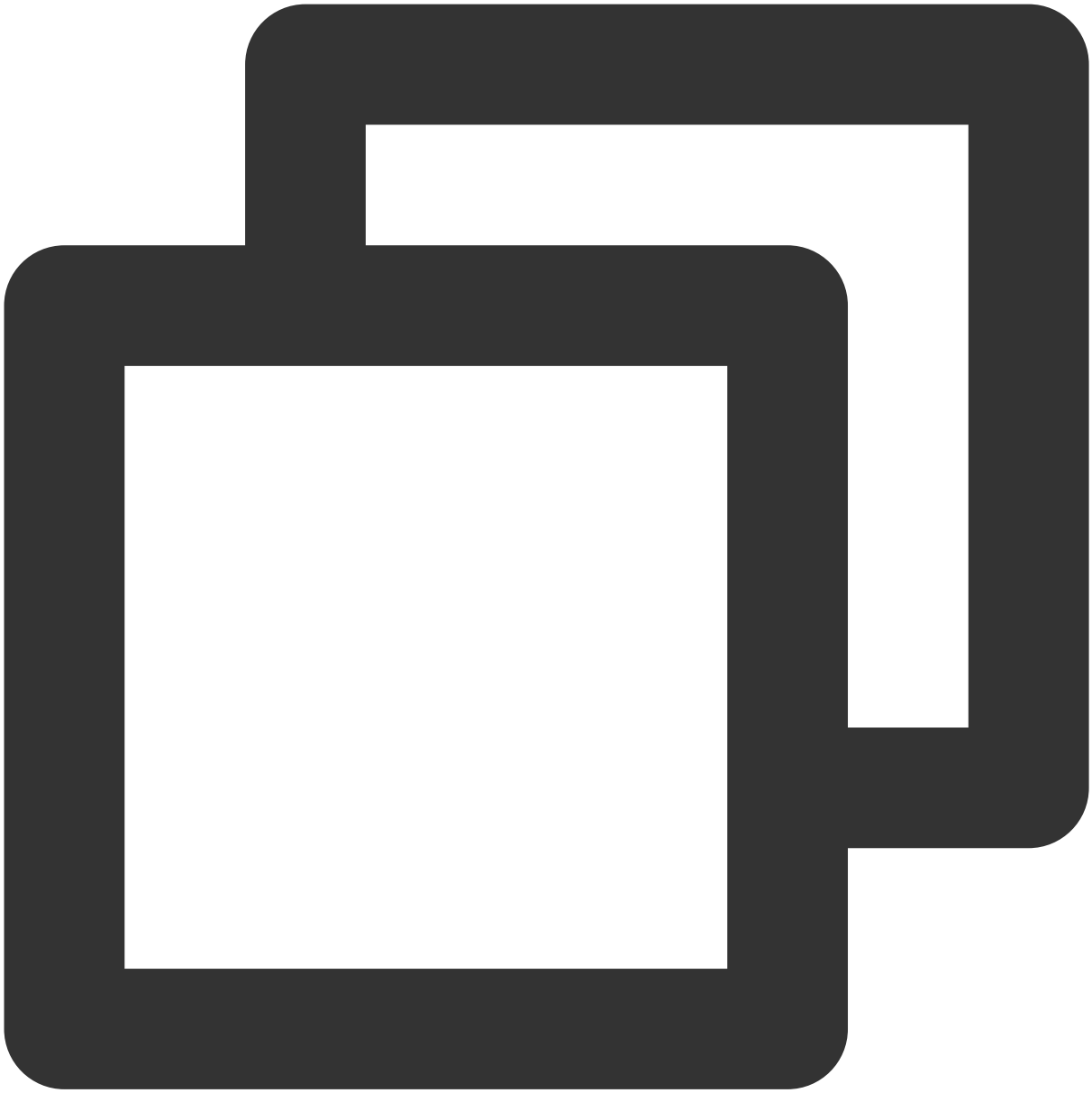
每次请求最多只能给100个用户添加标签，请求体中单个用户添加标签数最多为10个。

单个用户可设置最大标签数为100个，若用户当前标签超过100，则添加新标签之前请先删除旧标签。

应用最大可以设置的标签数为1000个，即所有用户的标签加在一起去重复后的数量为最多1000个。

单个标签最大长度为50字节。

请求 URL 示例



```
https://xxxxxx/v4/timpush/add_tag?usersig=xxx&identifier=admin&sdkappid=88888888&ra
```

请求参数说明

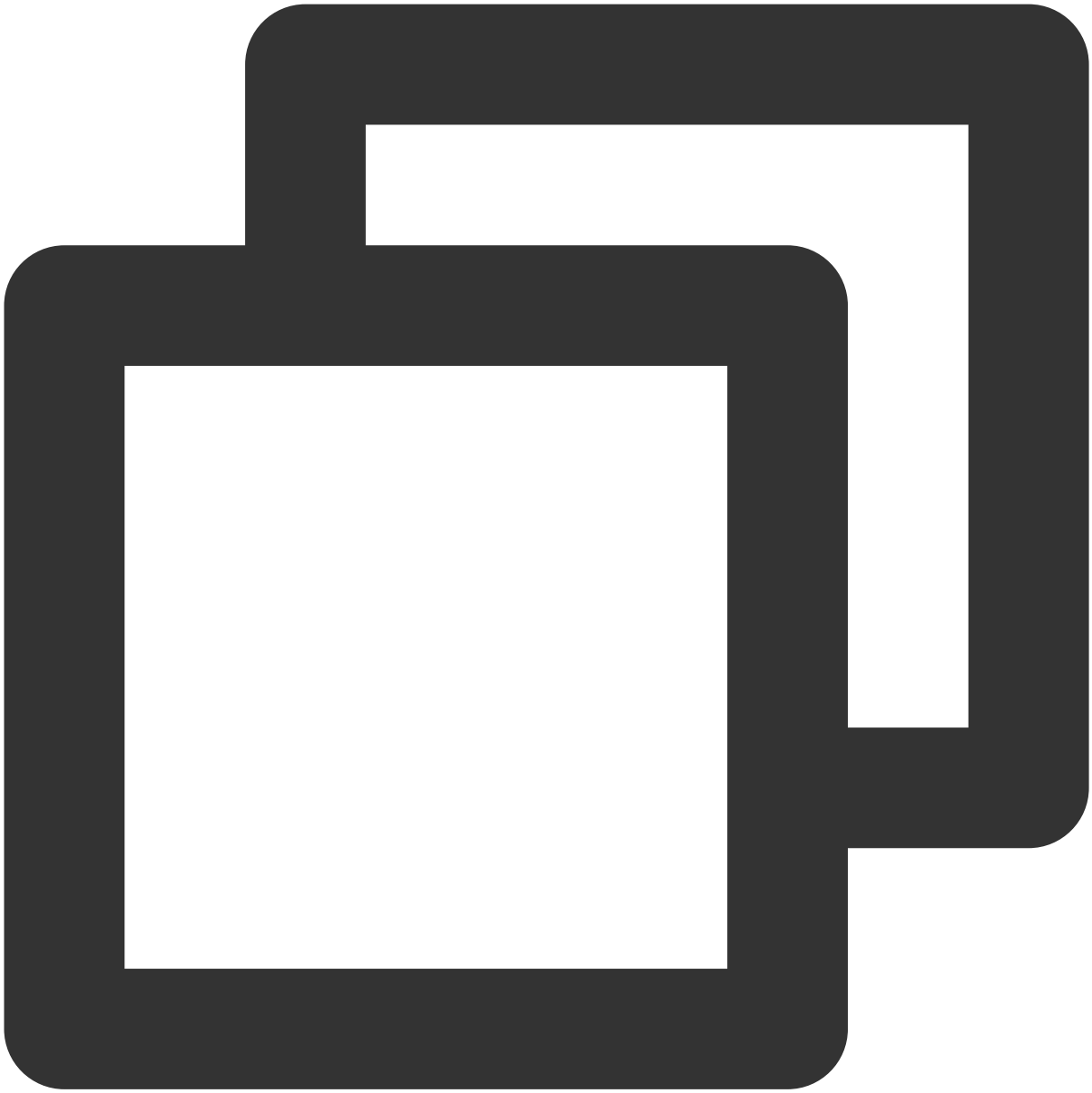
参数	说明
https	请求协议为：HTTPS 请求方式为：POST
xxxxxx	SDKAppID 所在国家/地区对应的专属域名。

	中国：`console.tim.qq.com` 新加坡：`adminapisgp.im.qcloud.com` 首尔：`adminapikr.im.qcloud.com` 法兰克福：`adminapiger.im.qcloud.com` 孟买：`adminapiind.im.qcloud.com` 硅谷：`adminapiusa.im.qcloud.com` **注意：** 目前印度站暂未上架推送插件。
v4/timpush/add_tag	请求接口
usersig	App 管理员账号生成的签名，参见 UserSig 后台 API
identifier	必须为 App 管理员账号
sdkappid	创建应用时即时通信控制台分配的 SdkAppid
random	32位无符号整数随机数
contenttype	固定值为：json

调用频率限制

每秒100次。

请求包示例



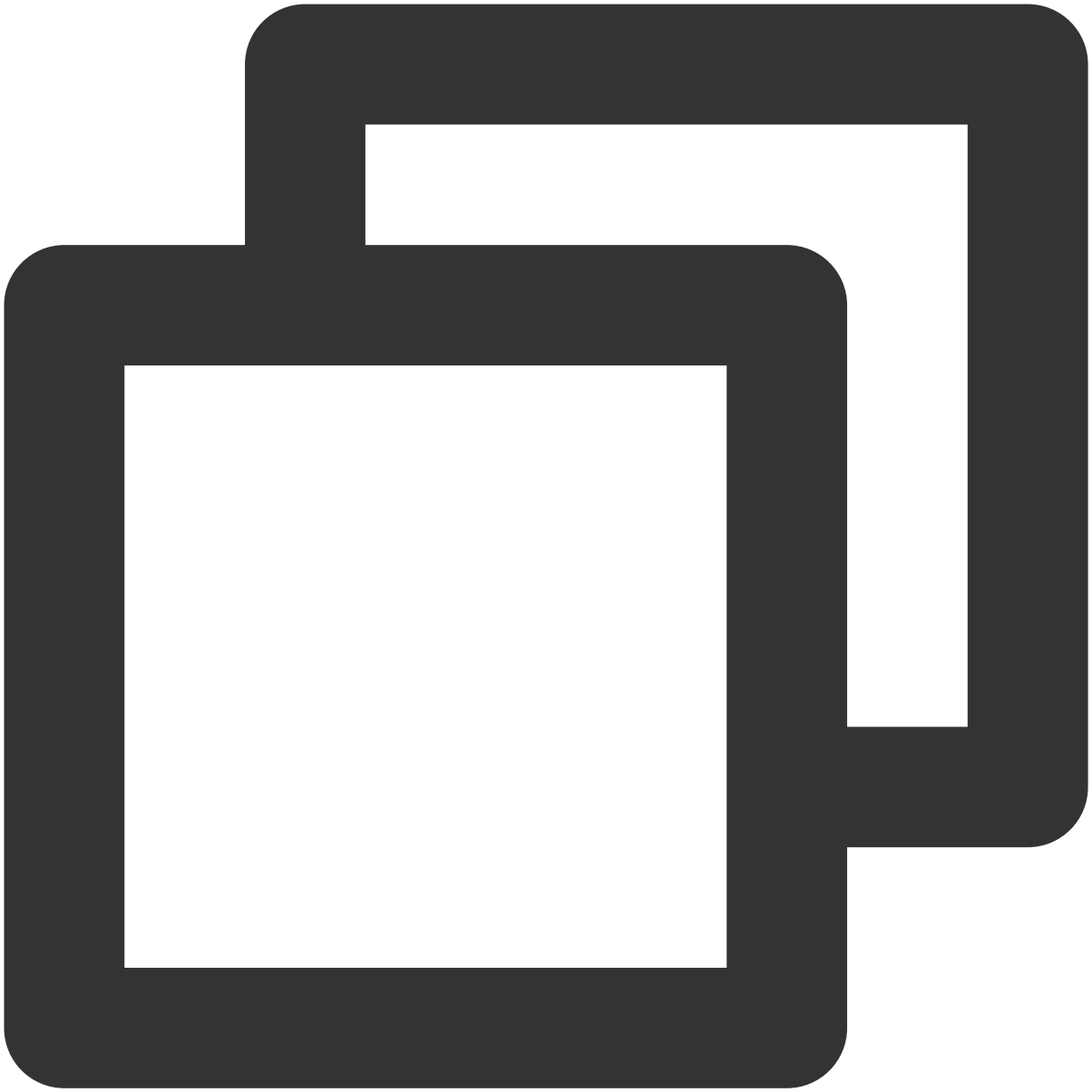
```
{
    "UserTags": [
        {
            "To_Account": "xiaojun012",
            "Tags": ["a", "b"]
        },
        {
            "To_Account": "xiaojun013",
            "Tags": ["a", "b"]
        }
    ]
```

```
}
```

请求包字段说明

字段	类型	属性	说明
To_Account	String	必填	目标用户账号
Tags	Array	必填	标签集合

应答包体示例



```
{
    "ActionStatus": "OK",
    "ErrorInfo": "",
    "ErrorCode": 0
}
```

应答包字段说明

字段	类型	说明

ActionStatus	String	请求处理的结果： OK：表示处理成功 FAIL：表示失败
ErrorCode	Integer	错误码
ErrorInfo	String	错误信息

错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。**真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的**。公共错误码（60000到79999）参见 错误码 文档。
本 API 私有错误码如下：

错误码	含义说明
90001	JSON 格式解析失败，请检查请求包是否符合 JSON 规范。
90009	请求需要 App 管理员权限。
90018	请求的账号数量超过限制。
91000	服务内部错误，请重试。

接口调试工具

通过 REST API 在线测试 工具调试本接口。

参考

全员/标签推送
设置应用属性名称
获取应用属性名称
设置用户属性
删除用户属性
获取用户属性
添加用户标签
获取用户标签
删除用户标签

清空用户标签

推送撤回

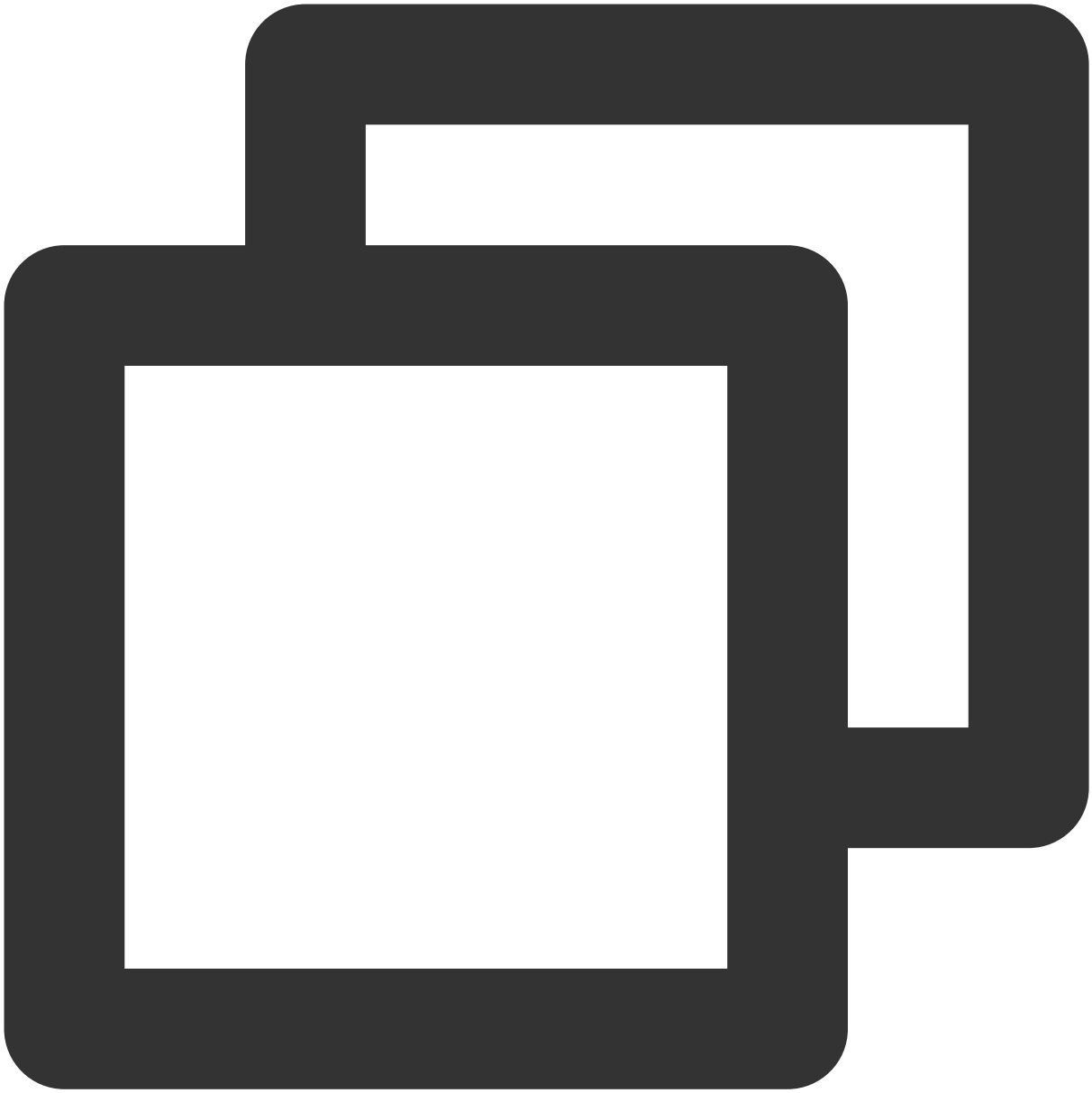
删除用户标签

最近更新时间：2024-06-13 10:39:26

功能说明

管理员给用户删除标签。注意每次最多只能给100个用户删除标签。

请求 URL 示例



```
https://xxxxxx/v4/timpush/remove_tag?usersig=xxx&identifier=admin&sdkappid=88888888
```

请求参数说明

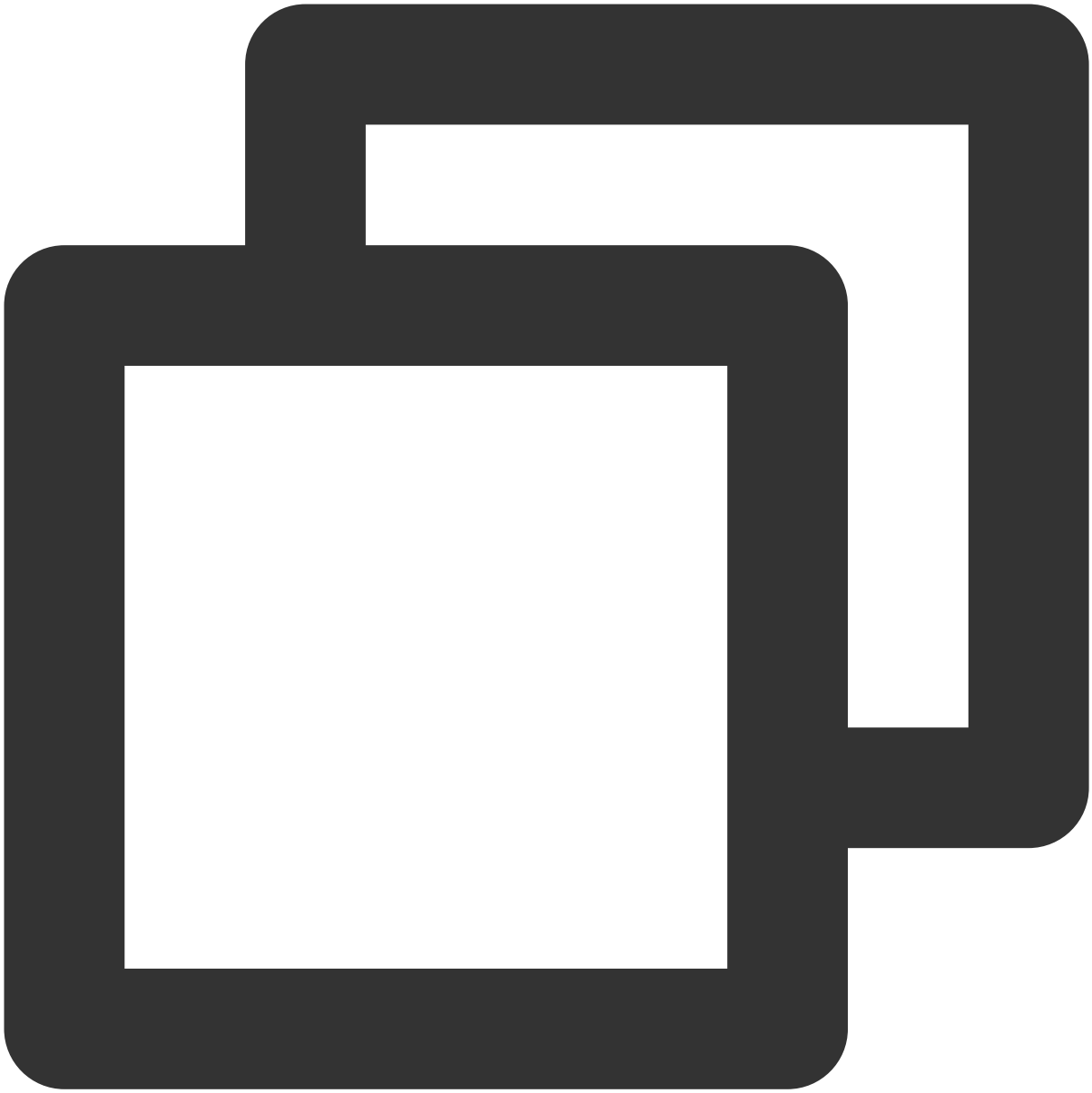
参数	说明
https	请求协议为：HTTPS 请求方式为：POST
xxxxxx	SDKAppID 所在国家/地区对应的专属域名。

	注意： 中国：`console.tim.qq.com` 新加坡：`adminapisgp.im.qcloud.com` 首尔：`adminapikr.im.qcloud.com` 法兰克福：`adminapiger.im.qcloud.com` 孟买：`adminapiind.im.qcloud.com` 硅谷：`adminapiusa.im.qcloud.com` **注意：** 目前印度站暂未上架推送插件。
v4/timpush/remove_tag	请求接口
usersig	App 管理员账号生成的签名，参见 UserSig 后台 API
identifier	必须为 App 管理员账号
sdkappid	创建应用时即时通信控制台分配的 SdkAppid
random	32位无符号整数随机数
contenttype	固定值为：json

调用频率限制

每秒100次。

请求包示例



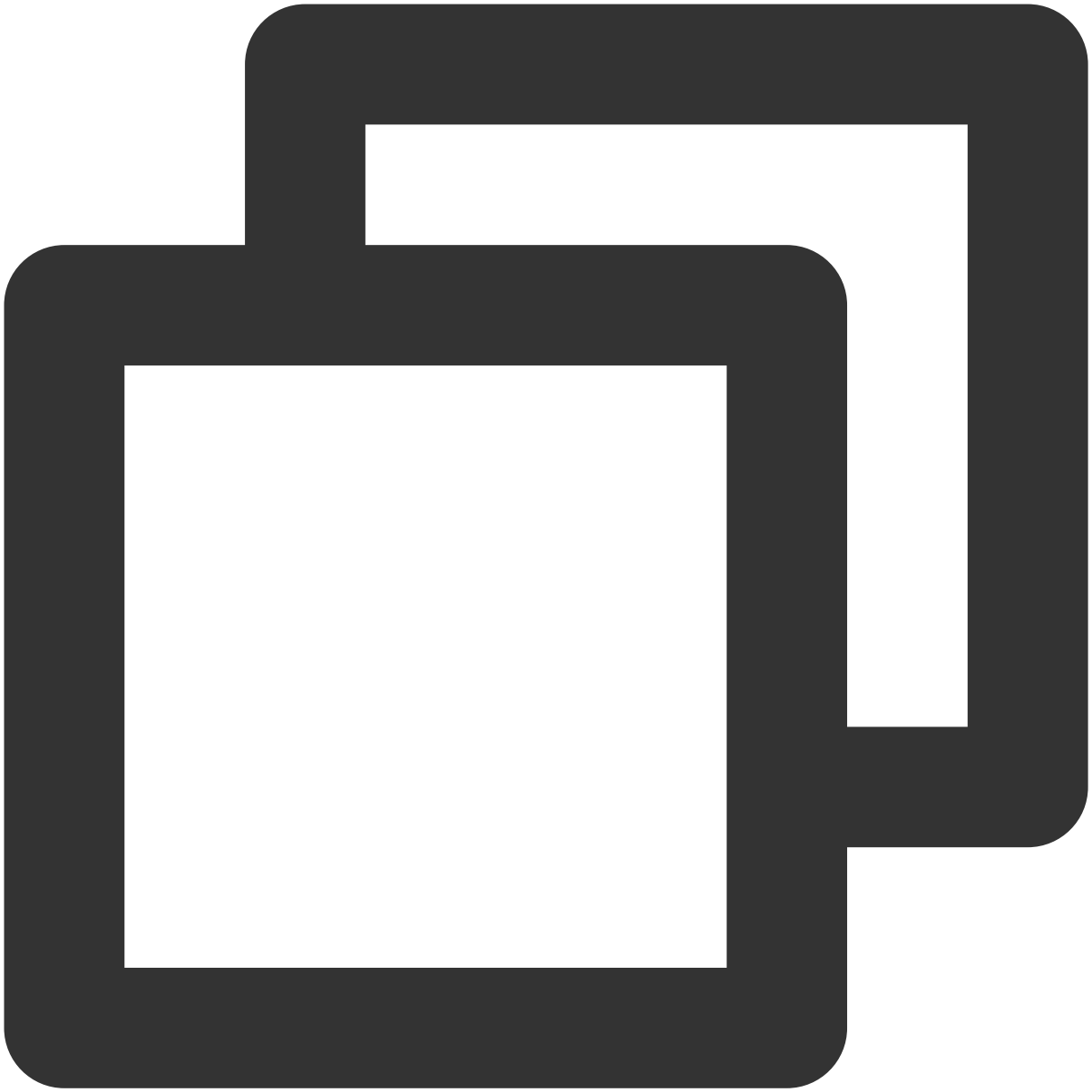
```
{
    "UserTags": [
        {
            "To_Account": "xiaojun012",
            "Tags": ["a", "b"]
        },
        {
            "To_Account": "xiaojun013",
            "Tags": ["a", "b"]
        }
    ]
```

```
}
```

请求包字段说明

字段	类型	属性	说明
To_Account	String	必填	目标用户账号
Tags	Array	必填	标签集合

应答包体示例



```
{
    "ActionStatus": "OK",
    "ErrorInfo": "",
    "ErrorCode": 0
}
```

应答包字段说明

字段	类型	说明

ActionStatus	String	请求处理的结果： OK：表示处理成功 FAIL：表示失败
ErrorCode	Integer	错误码
ErrorInfo	String	错误信息

错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。**真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的**。公共错误码（60000到79999）参见 错误码 文档。
本 API 私有错误码如下：

错误码	含义说明
90001	JSON 格式解析失败，请检查请求包是否符合 JSON 规范。
90009	请求需要 App 管理员权限。
90018	请求的账号数量超过限制。
91000	服务内部错误，请重试。

接口调试工具

通过 REST API 在线测试 工具调试本接口。

参考

发起全员/标签推送
设置应用属性名称
获取应用属性名称
设置用户属性
删除用户属性
获取用户属性
添加用户标签
获取用户标签
删除用户标签

清空用户标签

推送撤回

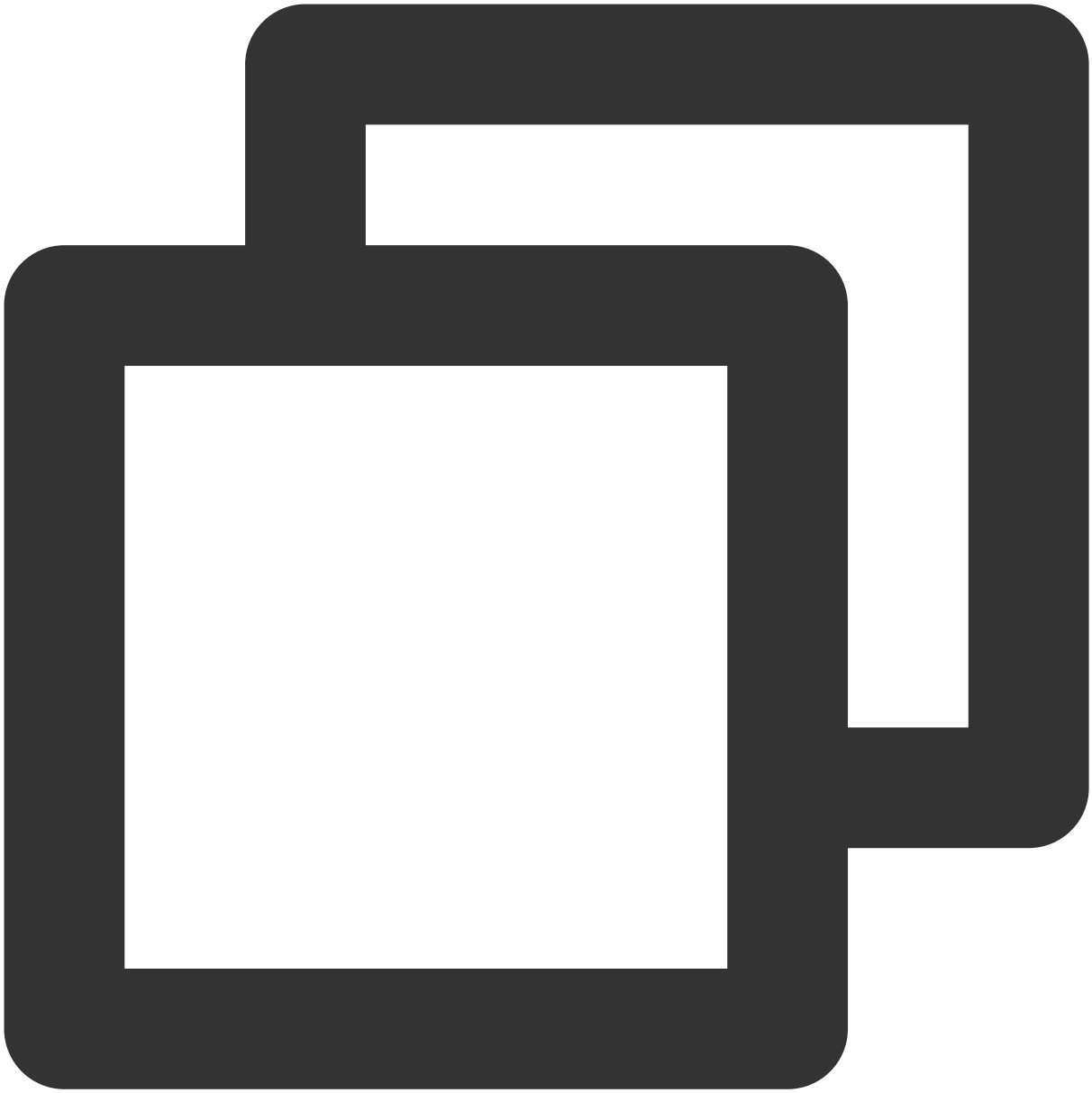
清空用户标签

最近更新时间：2024-06-13 10:39:26

功能说明

管理员为用户删除所有标签。注意每次最多只能给100个用户删除所有标签。

请求 URL 示例



```
https://xxxxxx/v4/timpush/clear_all_tags?usersig=xxx&identifier=admin&sdkappid=8888
```

请求参数说明

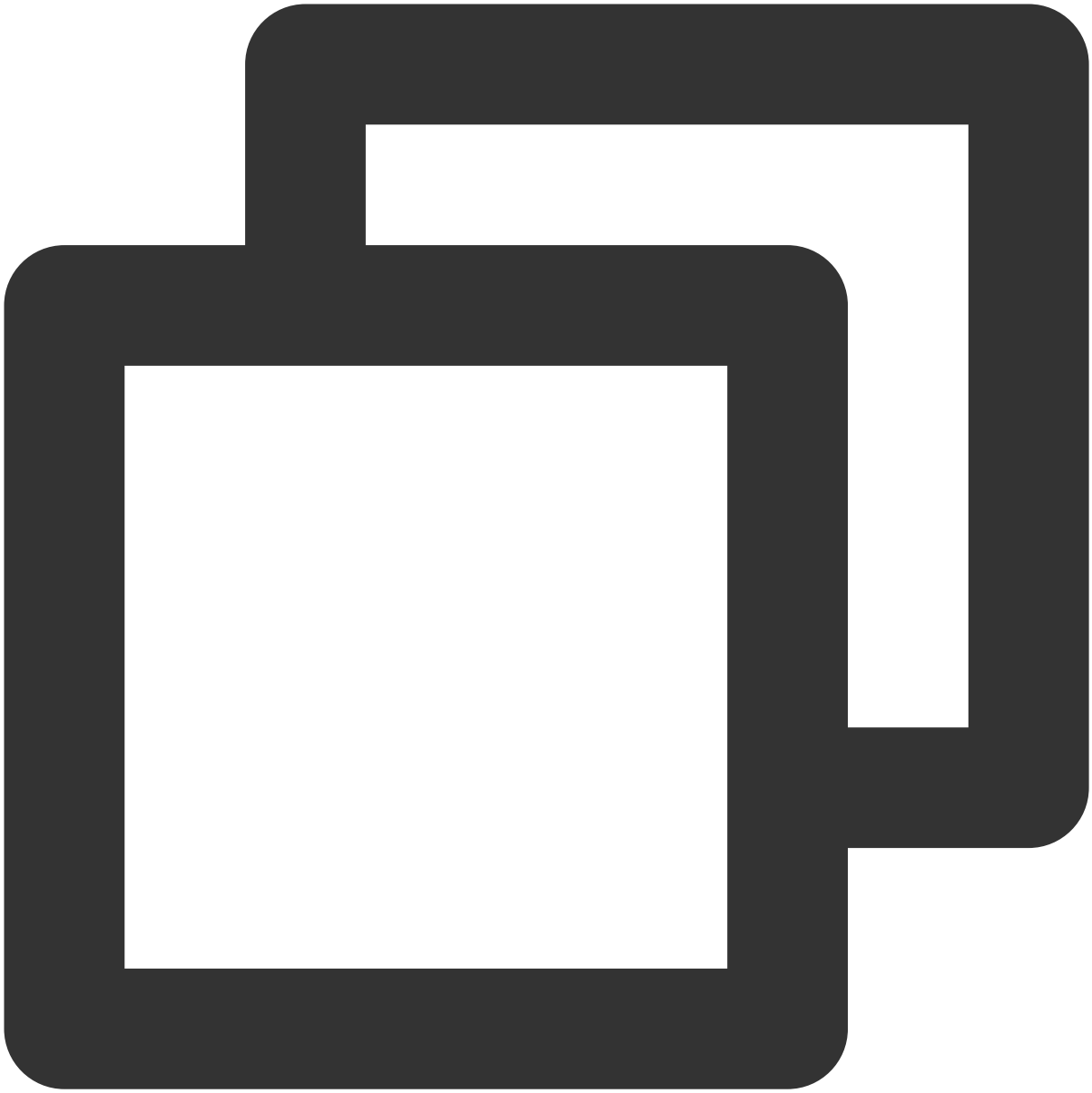
参数	说明
https	请求协议为：HTTPS 请求方式为：POST
xxxxxx	SDKAppID 所在国家/地区对应的专属域名。

	中国：`console.tim.qq.com` 新加坡：`adminapisgp.im.qcloud.com` 首尔：`adminapikr.im.qcloud.com` 法兰克福：`adminapiger.im.qcloud.com` 孟买：`adminapiind.im.qcloud.com` 硅谷：`adminapiusa.im.qcloud.com` **注意：** 目前印度站暂未上架推送插件。
v4/timpush/clear_all_tags	请求接口
usersig	App 管理员账号生成的签名，参见 UserSig 后台 API
identifier	必须为 App 管理员账号
sdkappid	创建应用时即时通信控制台分配的 SdkAppid
random	32位无符号整数随机数
contenttype	固定值为：json

调用频率限制

每秒100次。

请求包示例



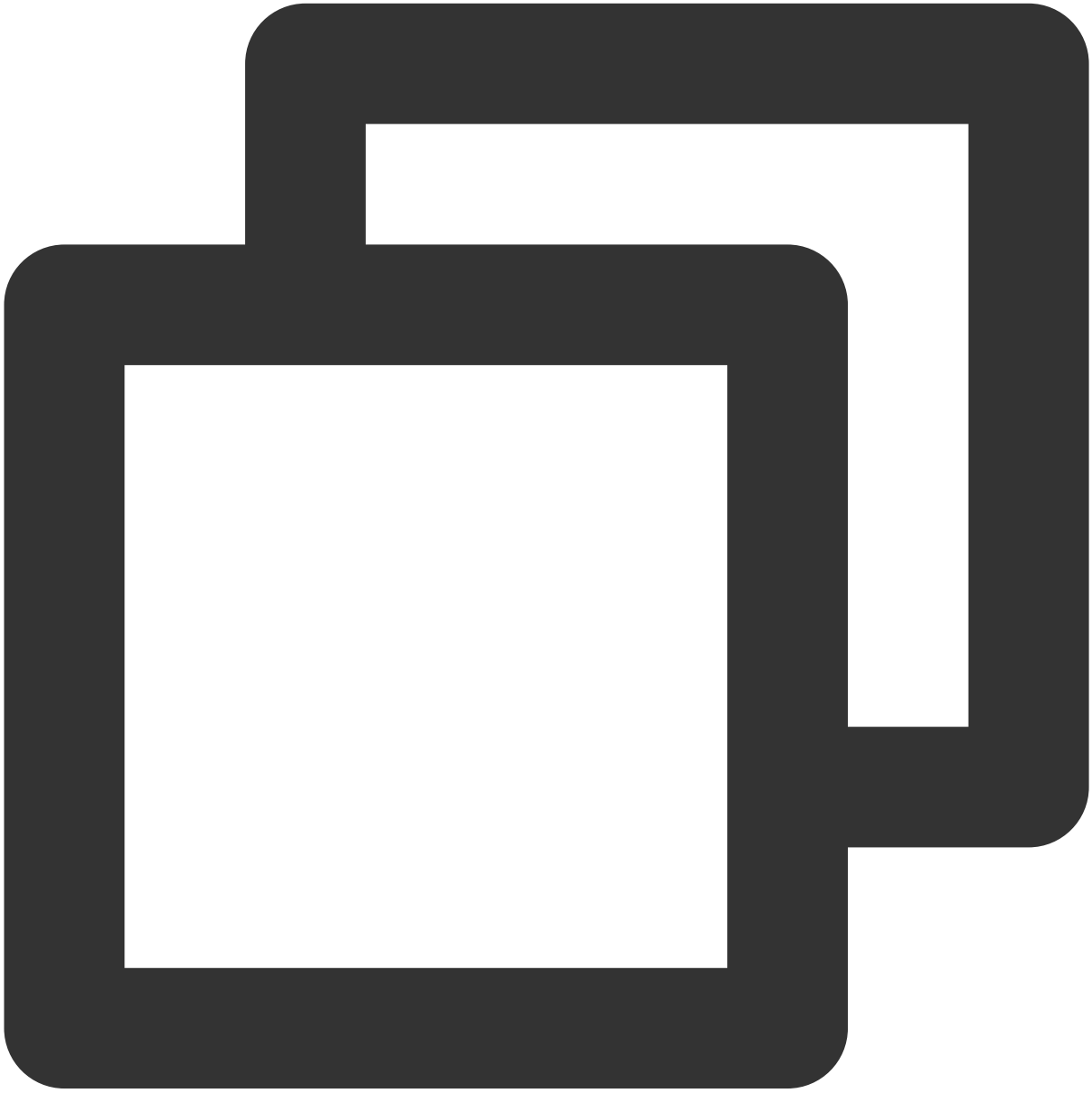
```
{
    "To_Account": [
        "xiaojun012",
        "xiaojun013"
    ]
}
```

请求包字段说明

字段	类型	属性	说明

To_Account	Array	必填	目标用户账号

应答包体示例



```
{
    "ActionStatus": "OK",
    "ErrorInfo": "",
    "ErrorCode": 0
}
```

应答包字段说明

字段	类型	说明
ActionStatus	String	请求处理的结果： OK：表示处理成功 FAIL：表示失败
ErrorCode	Integer	错误码
ErrorInfo	String	错误信息

错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。**真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的**。公共错误码（60000到79999）参见 错误码 文档。

本 API 私有错误码如下：

错误码	含义说明
90001	JSON 格式解析失败，请检查请求包是否符合 JSON 规范。
90009	请求需要 App 管理员权限。
90018	请求的账号数量超过限制。
91000	服务内部错误，请重试。

接口调试工具

通过 REST API 在线调试接口调试本接口。

参考

全员/标签推送
设置应用属性名称
获取应用属性名称
设置用户属性
删除用户属性
获取用户属性

推送撤回

最近更新时间：2024-06-13 10:39:26

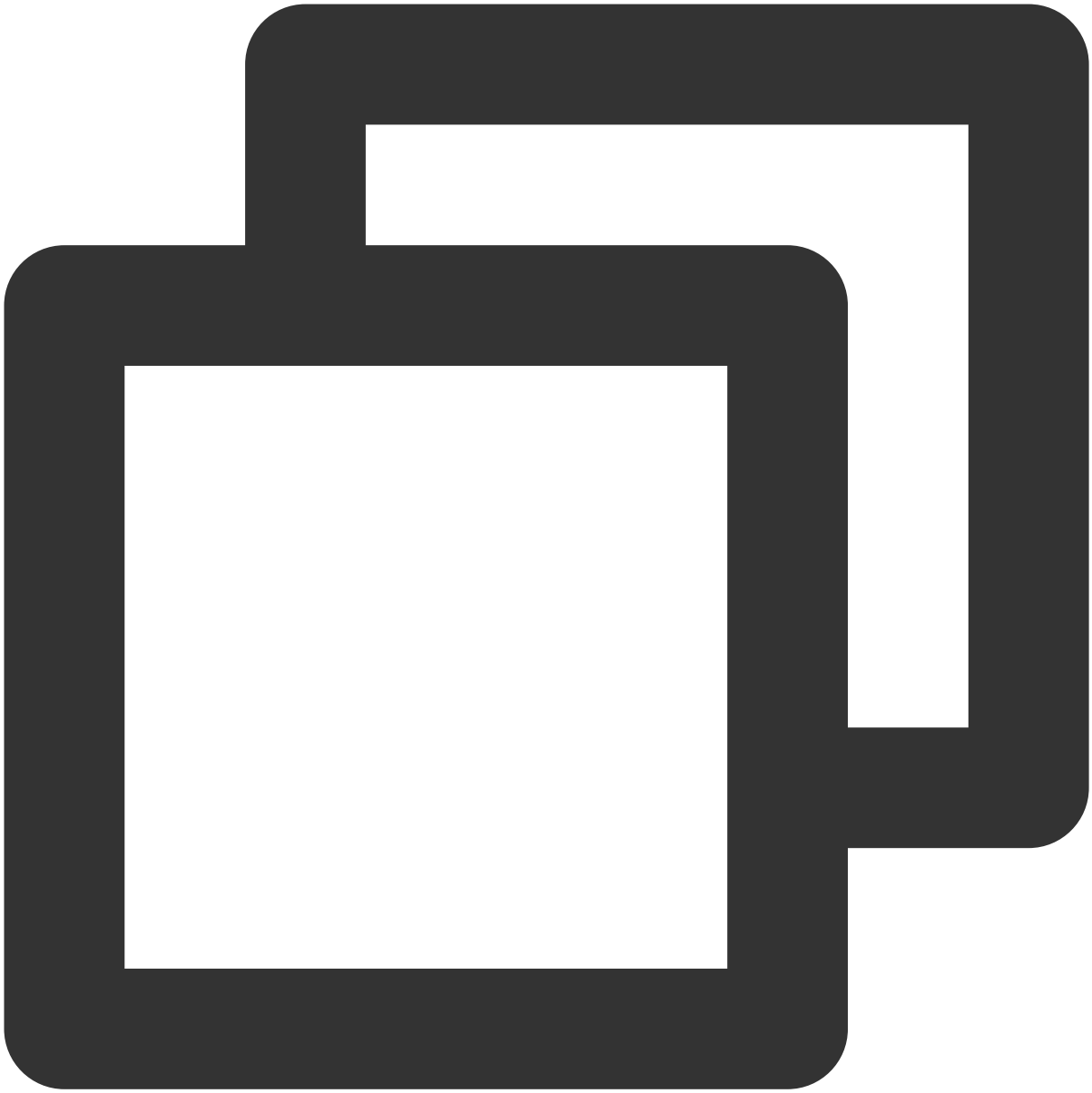
若全员/标签推送内容有误，终端用户查看或点击后会对产品有负面影响，此时需要及时处理。您可以选择撤回该推送。

功能说明

终止：推送任务下发需要一定时间，未下发的账号会终止下发。
撤回：已下发的账号，支持撤回未读/漫游。
覆盖：若已下发的账号收到了离线推送，支持覆盖该条推送。
本接口支持全员/标签推送任务的终止/撤回/覆盖。本文后续默认将终止/撤回/覆盖简称为撤回。
撤回有效期为24小时，从任务发起时间开始计算，超过24小时的推送任务无法撤回。

接口调用说明

请求 URL 示例



```
https://xxxxxx/v4/timpush/revoke?usersig=xxx&identifier=admin&sdkappid=88888888&ran
```

请求参数说明

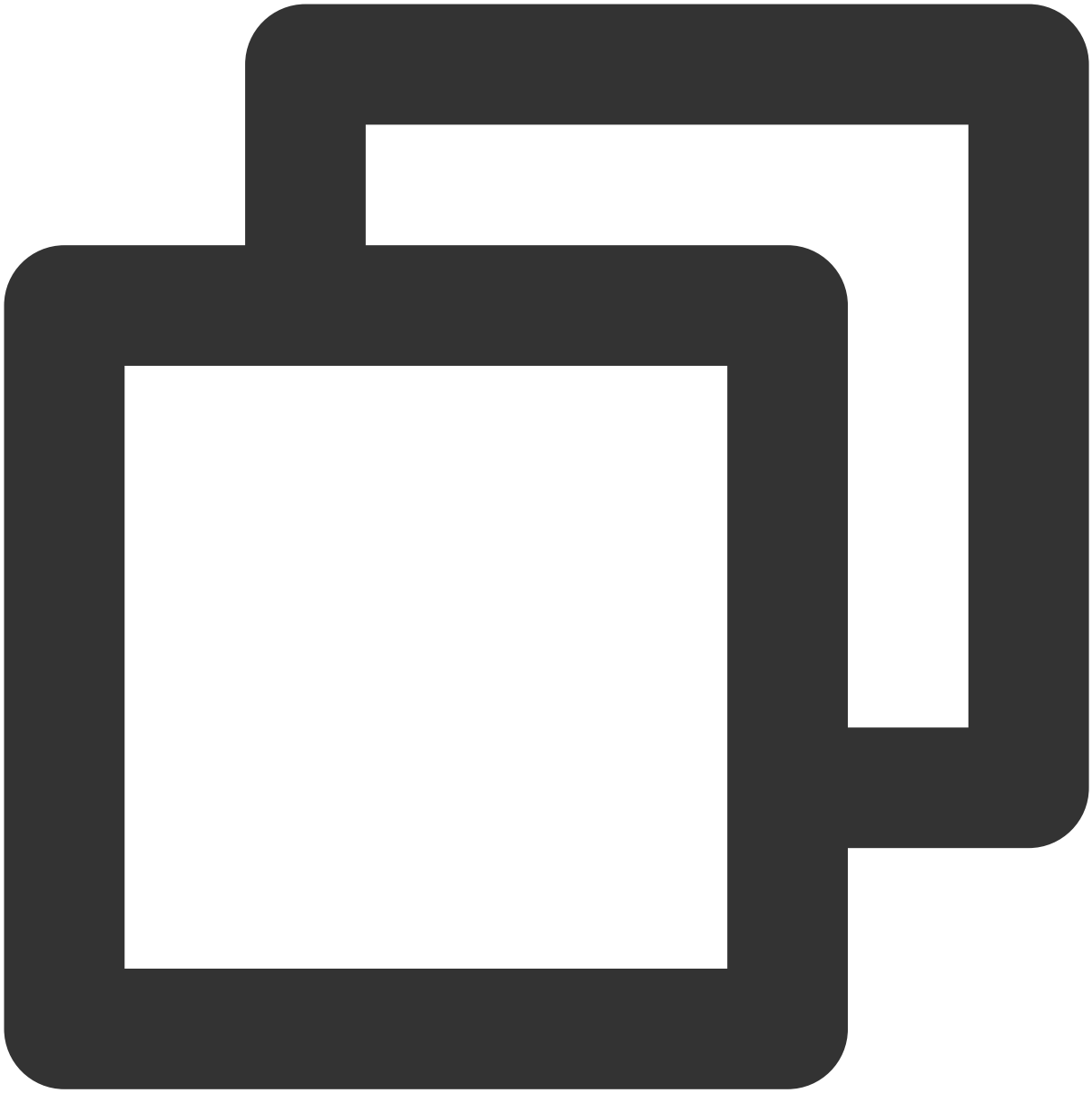
参数	说明
https	请求协议：HTTPS 请求方式：POST
xxxxxx	SDKAppID 所在国家/地区对应的专属域名。

	中国：console.tim.qq.com 新加坡：adminapisgp.im.qcloud.com 首尔： adminapikr.im.qcloud.com 法兰克福：adminapiger.im.qcloud.com 孟买：adminapiind.im.qcloud.com 硅谷：adminapiusa.im.qcloud.com **注意：** 目前印度站暂未上架推送插件。
v4/timpush/revoke	请求接口
usersig	App 管理员账号生成的签名，参见 UserSig 后台 API
identifier	必须为 App 管理员账号
sdkappid	创建应用时，即时通信控制台分配的 SdkAppid
random	32位无符号整数随机数
contenttype	固定值为：json

调用频率

本接口接口调用限制1次/s。

请求包示例



```
{
    "TaskId": "660cc447_537ed82a_200000cd7ee17f5_84035729_bc614e",  // 24小时内发送的
     "OfflinePushInfo": {    // 若推送时指定不存漫游/未读（OnlineOnlyFlag=0），则撤回时必须
        "Desc": "对方撤回了一条消息",
        "Ext": "这是透传的内容"
    }
}
```

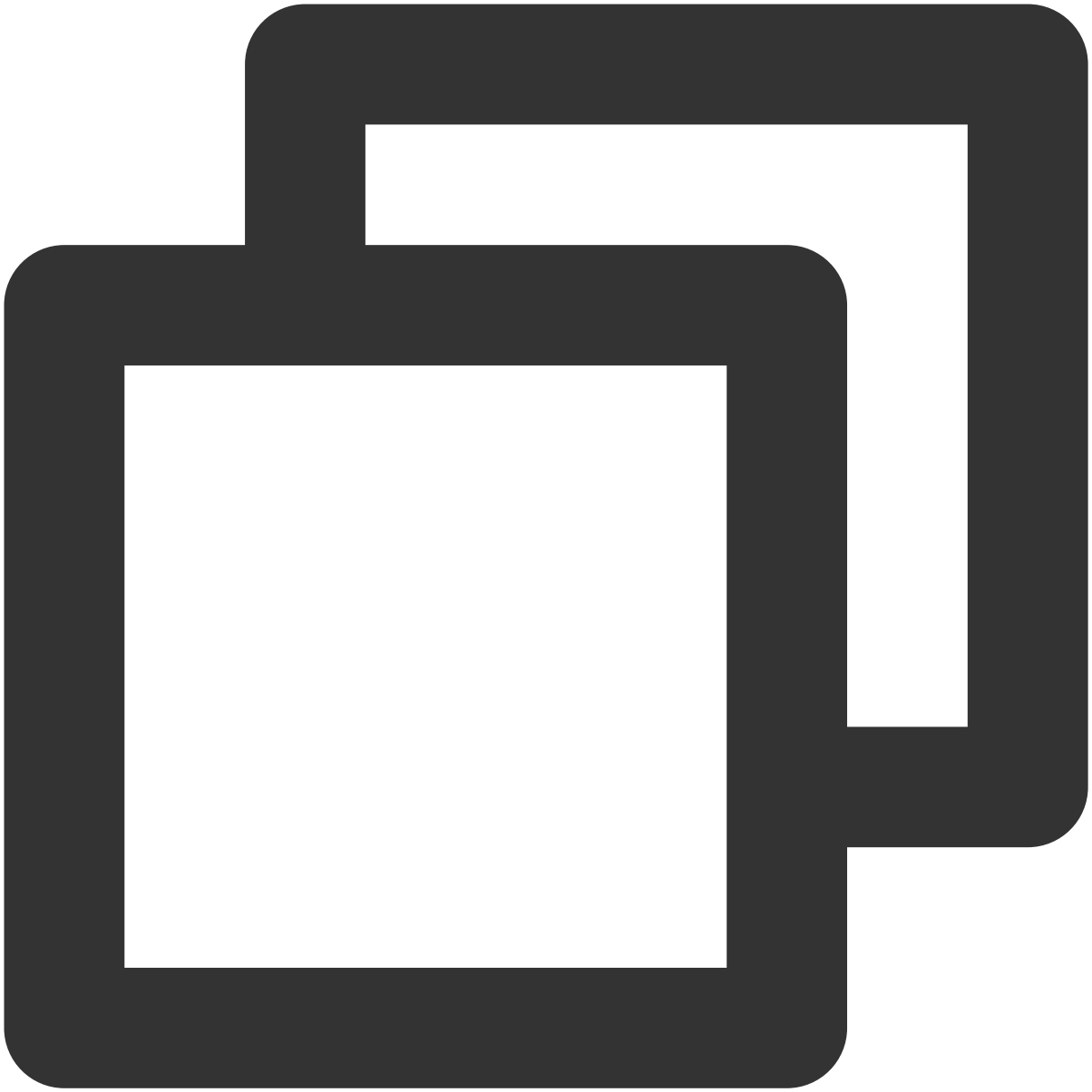
注意：

1. 支持离线推送覆盖的厂商：APNS/Google FCM/华为/荣耀。其他厂商的离线推送不支持覆盖。（Google FCM的notification模式支持覆盖，data模式暂不支持覆盖）
2. 撤回时若接受方在前台，默认离线推送（通知栏消息）已读，则不会覆盖该条离线推送。

请求包字段说明

字段	类型	属性	说明
TaskId	String	必填	全员/标签推送任务 ID。
OfflinePushInfo	Object	选填	离线推送信息配置，具体请参见 消息格式描述。 **注意：** 若设置OfflinePushinfo.PushFlag=1或者不设置OfflinePushInfo，不会覆盖离线推送。

应答包体示例



```
{
    "ActionStatus": "OK",
    "ErrorInfo": "",
    "ErrorCode": 0
}
```

应答包字段说明

字段	类型	说明

ActionStatus	String	请求处理的结果： OK：表示处理成功 FAIL：表示失败
ErrorCode	Integer	错误码： 0 表示成功 非0 表示失败
ErrorInfo	String	错误信息

错误码说明

除非发生网络错误（例如502错误），否则该接口的 HTTP 返回码均为200。真正的错误码，错误信息是通过应答包体中的 ErrorCode、ErrorInfo 来表示的。公共错误码（60000到79999）参见 错误码 文档。 本 API 私有错误码如下：

错误码	含义说明
90001	JSON 格式解析失败，请检查请求包是否符合 JSON 规范。
90009	请求需要 App 管理员权限。
90049	撤回 TaskId 不合法，无推送记录。通过 timpush/push 接口进行推送，返回的 TaskId 才能用于撤回。
90050	重复撤回，已经撤回的推送任务不能重复调用。
90051	撤回过于频繁，撤回限频1次/s。
90052	超过撤回有效期，撤回要求在24小时内，超过24小时的推送任务无法撤回。
90053	撤回无效。推送任务指定不存漫游/未读（OnlineOnlyFlag=0），但是撤回时没有带上 OfflinePushInfo。
91000	服务内部错误，请重试。

接口调试工具

通过 REST API 在线调试接口调试本接口。

参考

高级功能
自定义角标

最近更新时间：2024-06-13 10:39:26

Flutter

请参照下方 iOS 和 Android 章节进行配置即可。调用的方法均在 Flutter 版本的 IM SDK 中有同名方法。

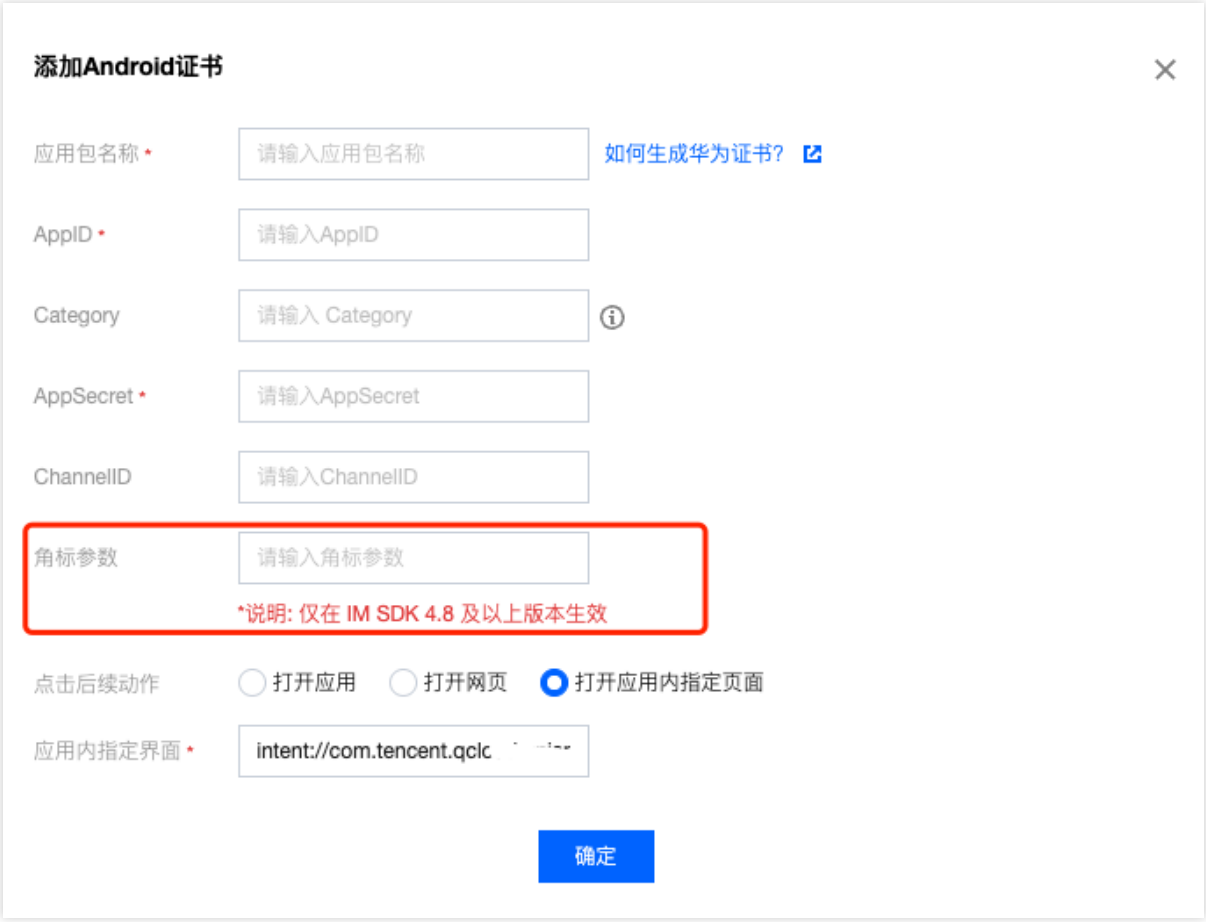
Android

支持厂商

华为。

配置方法

控制台配置华为角标参数为应用的启动类，例如 “com.tencent.qcloud.tim.demo.SplashActivity”。组件会自动解析和更新角标；反之不会更新角标。



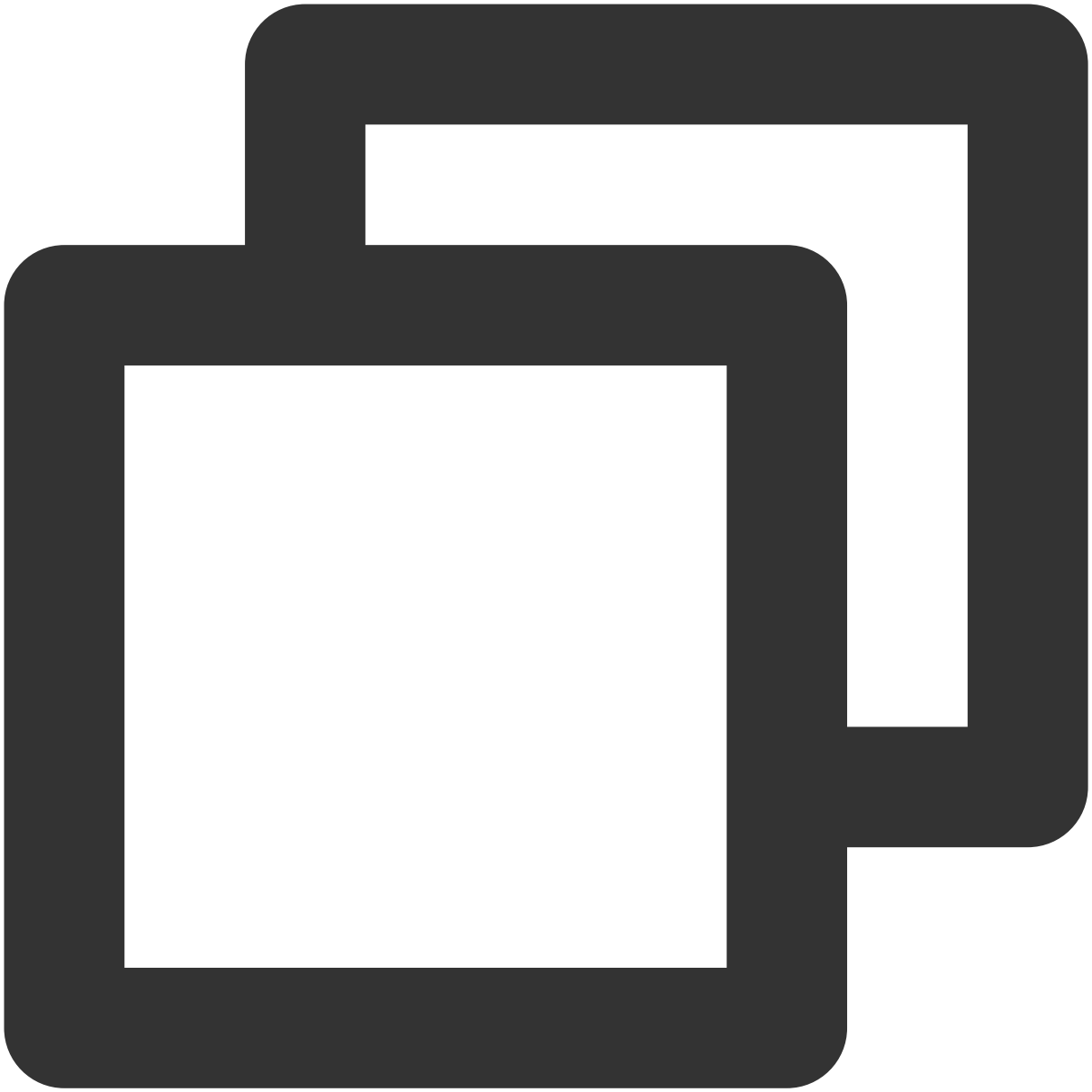
iOS

默认情况下，当 App 进入后台后，IMSDK 会将当前 IM 未读消息总数设置为角标。如果 App 接入了离线推送，当接收到新的离线推送时，App 角标会在基准角标（默认是 IM 未读消息总数，如果自定义了角标，则以自定义角标为准）的基础上加 1 逐条递增。

配置方法

如果想自定义角标，可按照如下步骤设置：

1. App 调用 - (void)setAPNSListener:(id<V2TIMAPNSListener>)apnsListener 接口设置监听。
2. App 实现 - (uint32_t)onSetAPPUnreadCount 接口，并在内部返回需要自定义的角标。



```
// 1. 设置监听
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSD
    // 监听推送
    [V2TIMManager.sharedInstance setAPNSListener:self];
    // 监听会话的未读数
    [[V2TIMManager sharedInstance] setConversationListener:self];
    return YES;
}

// 2. 未读数发生变化后保存未读数
- (void)onTotalUnreadMessageCountChanged:(UInt64)totalUnreadCount {
```

```
    self.unreadNumber = totalUnreadCount;
}

// 3. App 推到后台后上报自定义未读数
/** 程序进后台后，自定义 App 的未读数，如果不处理，App 未读数默认为所有会话未读数之和
 *  <pre>
 *
 *   - (uint32_t)onSetAPPUnreadCount {
 *       return 100;  // 自定义未读数
 *   }
 *
 *  </pre>
 */
- (uint32_t)onSetAPPUnreadCount {
    // 1. 获取自定义的角标
    uint32_t customBadgeNumber = ...

    // 2. 加上 IM 的消息未读数
    customBadgeNumber += self.unreadNumber;

    // 3. 通过 IMSDK 上报给 IM 服务器
    return customBadgeNumber;
}
```

自定义铃音

最近更新时间：2024-06-13 10:39:26

uniapp

注意：

接收端需集成TencentCloud-TIMPush。

发送端 **@tencentcloud/chat ≥ 3.3.2** 。

支持华为、小米、FCM 和 APNS。

接收端

Android

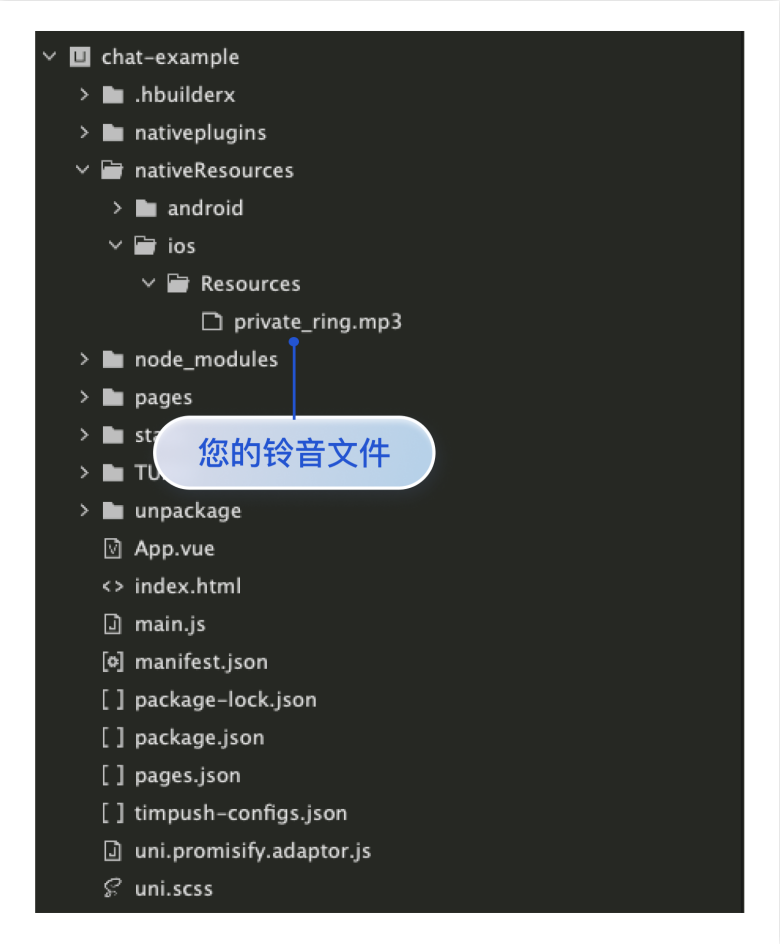
iOS

定制的铃音资源文件，添加到项目 **nativeResources/android/res/raw** 目录下，如图所示：

注意：

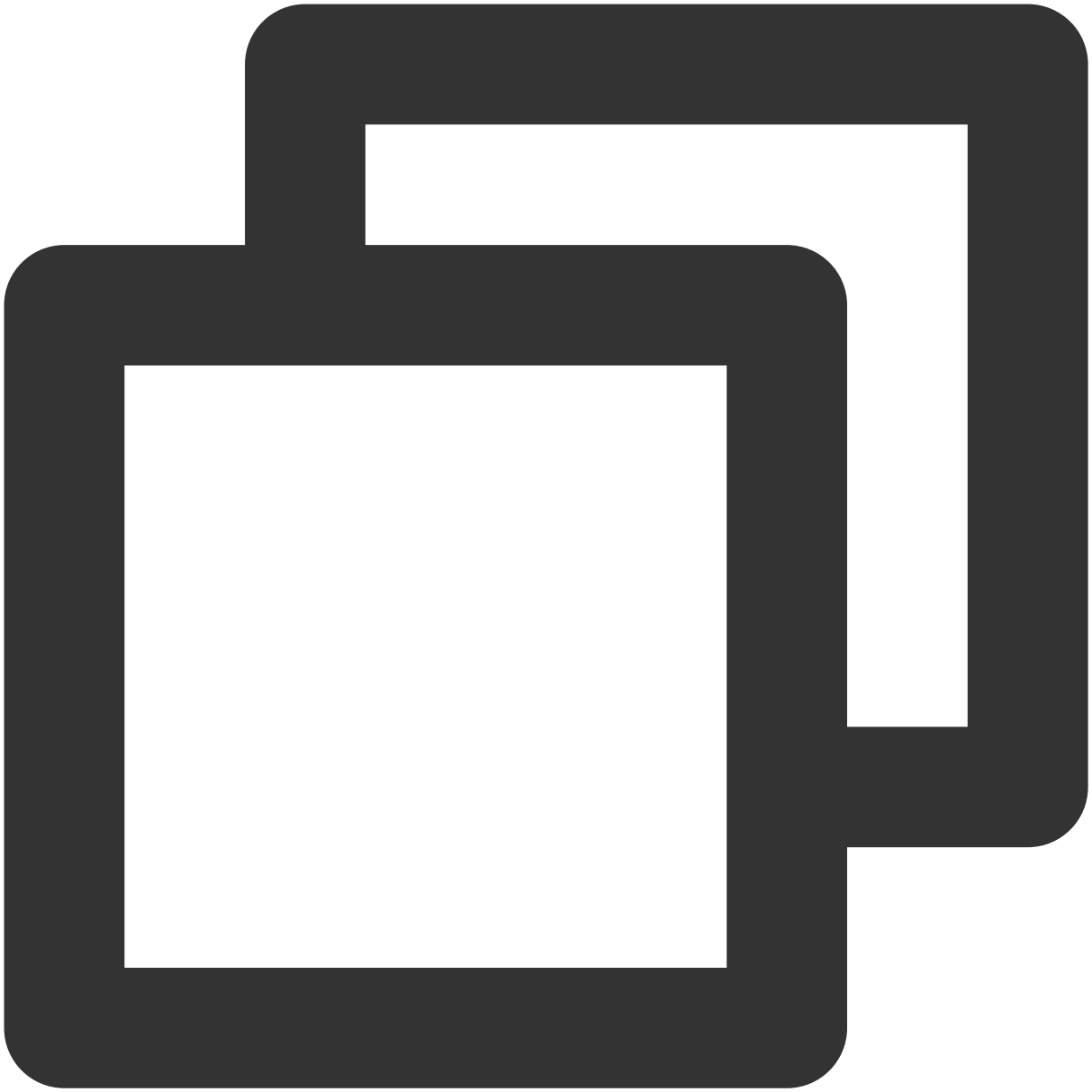
iOS 自定义铃音，uniapp 必须为正式包。

定制的铃音资源文件，添加到项目nativeResources/ios/Resources 目录下，如图所示：



发送端

1. 升级 @tencentcloud/chat 到最新版本。



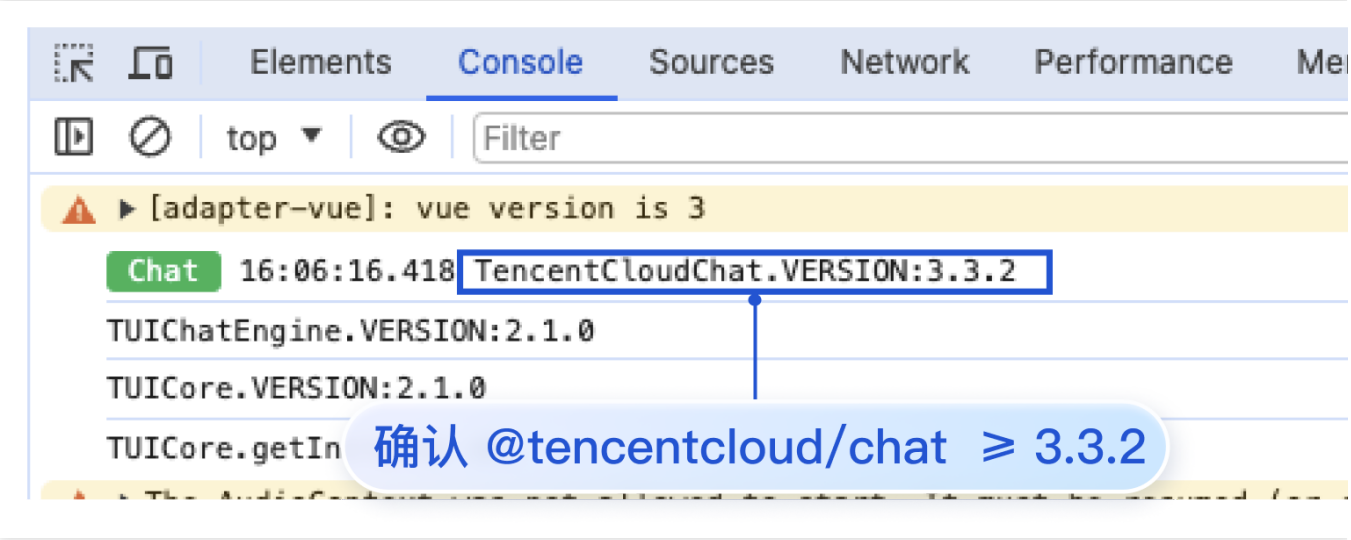
```
npm install @tencentcloud/chat@latest
```

web

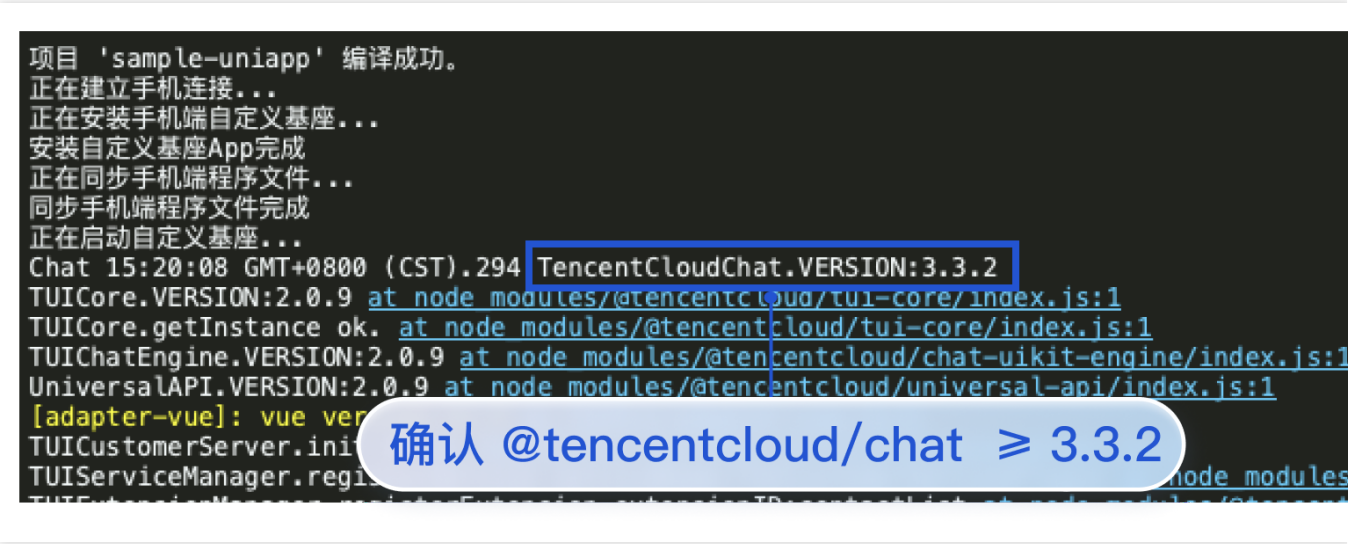
uniapp

小程序

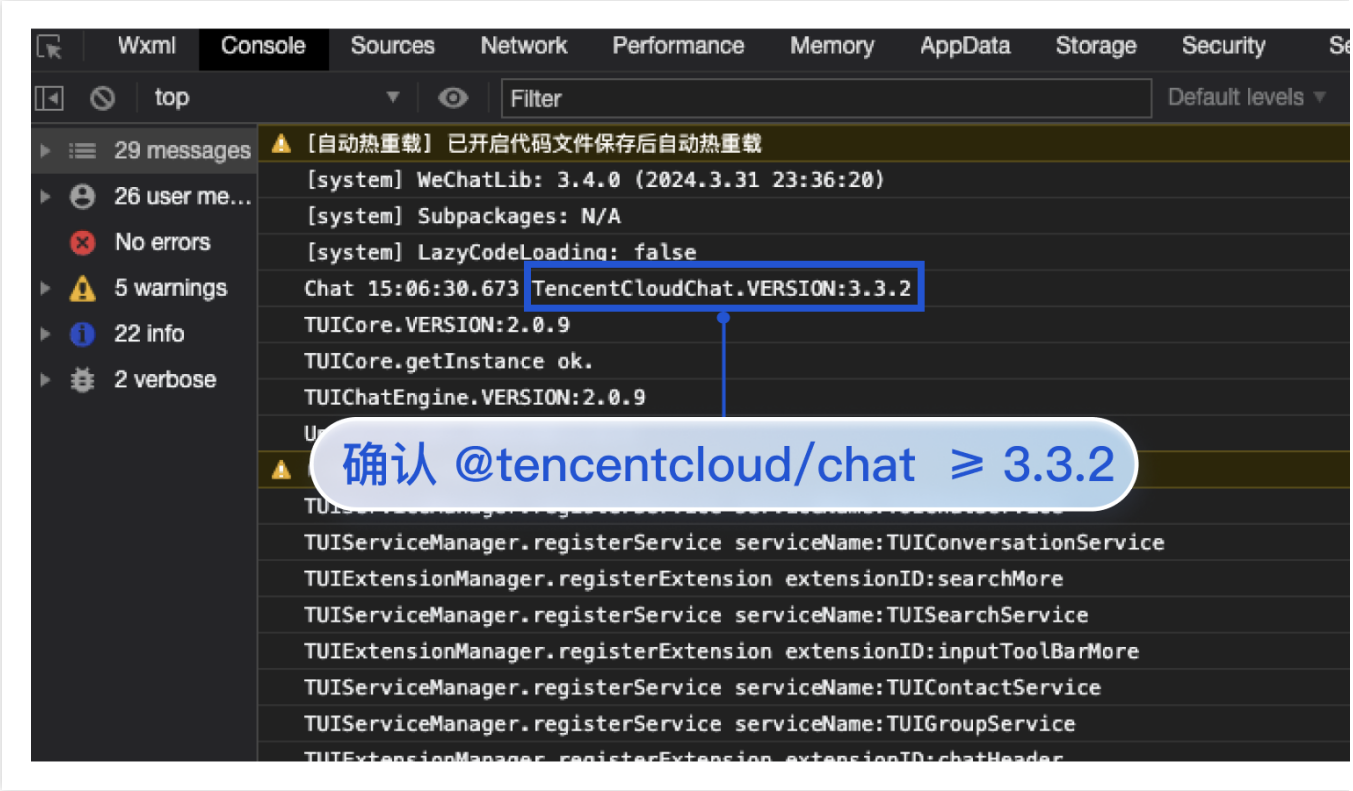
在浏览器控制台查看 TencentCloudChat.VERSION 版本号，来确认 **@tencentcloud/chat ≥ 3.3.2** 如图所示：



在 HBuilder 日志中查看 TencentCloudChat.VERSION 版本号，来确认 **@tencentcloud/chat ≥ 3.3.2** 如图所示：



在小程序开发者工具控制台查看 TencentCloudChat.VERSION 版本号，来确认 **@tencentcloud/chat ≥ 3.3.2** 如图所示：



2. 发送消息，设置 offlinePushInfo 自定义铃音的相关参数。

注意：

小米手机在 Android 8.0 及以上版本必须设置 androidInfo.XiaoMiChannelID，请您参见：小米自定义铃声。

谷歌手机 FCM 推送在 Android 8.0 及以上系统设置声音提示，必须设置 androidInfo.FCMChannelID。

含 UI 集成

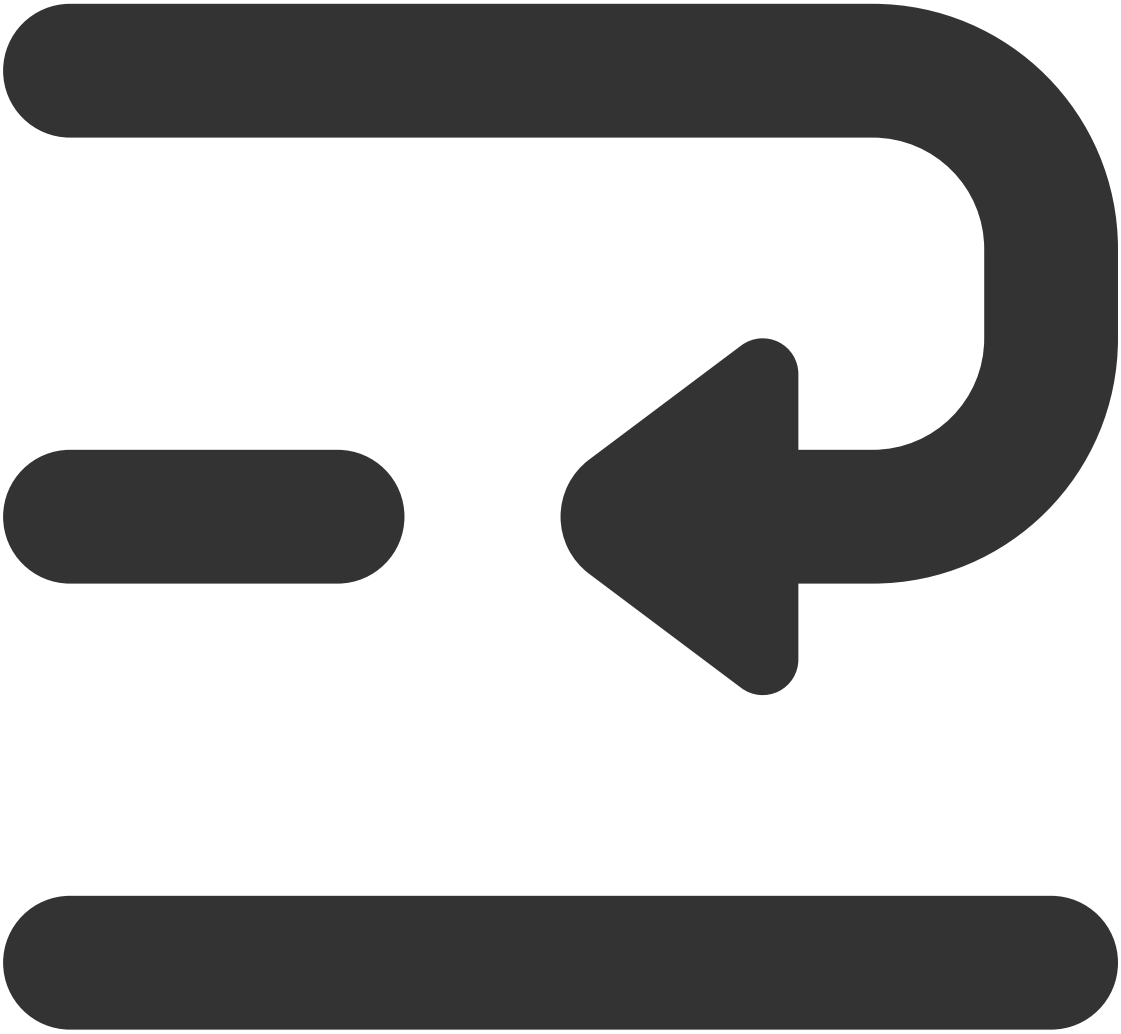
无 UI 集成

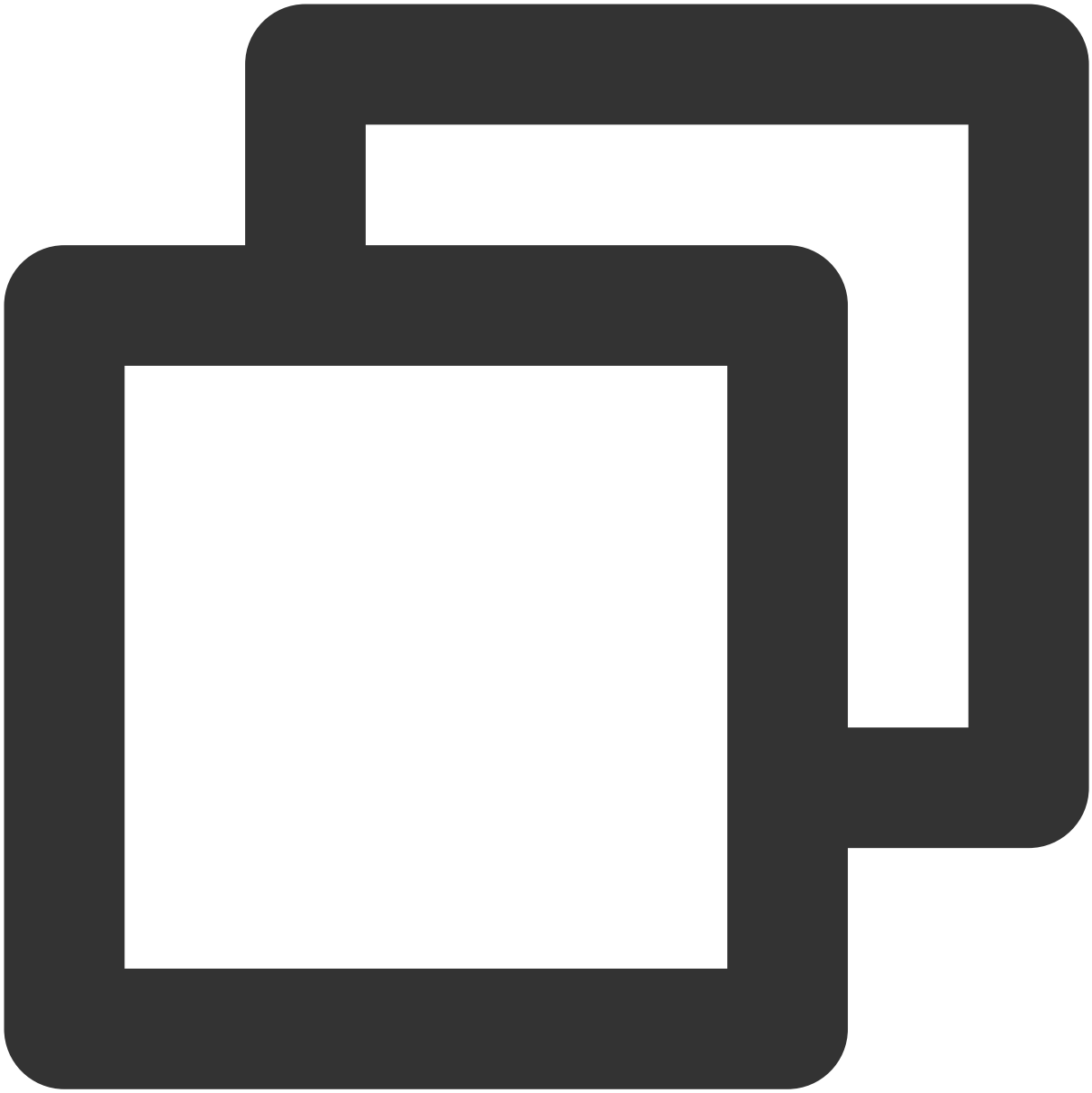
注意：

当 apnsInfo.sound = TUIChatEngine.TYPES.IOS_OFFLINE_PUSH_NO_SOUND，表示接收时不会播放声音。

当 apnsInfo.sound = TUIChatEngine.TYPES.IOS_OFFLINE_PUSH_DEFAULT_SOUND，表示接收时播放系统声音。

在 UIKit 中使用 TUIChatService 发送消息时，设置 offlinePushInfo 相关参数，如发送普通文本消息，代码如下：





```
// 发送普通文本消息
let promise = TUIChatService.sendTextMessage(
{
   payload: {   text: 'Hello world!' }
},
{
  // 如果接收方不在线，则消息将存入漫游，且进行离线推送（在接收方 App 退后台或者进程被 kill 的情
   offlinePushInfo: {
     androidInfo: { // Android 推送配置
        sound: 'private_ring.mp3', // Andorid 自定义铃音
        XiaoMiChannelID: '', // 小米手机在 Android 8.0 及以上版本必须设置 XiaoMiChannelI
```

```
          FCMChannelID: '', // 谷歌手机 FCM 在 Android 8.0 及以上系统设置声音提示，必须设置
        },
        apnsInfo: { // APNs 推送配置
      // apnsInfo.sound = TUIChatEngine.TYPES.IOS_OFFLINE_PUSH_NO_SOUND，表示接收时不会
      // apnsInfo.sound = TUIChatEngine.TYPES.IOS_OFFLINE_PUSH_DEFAULT_SOUND，表示接收
          sound: 'private_ring.mp3', // iOS 自定义铃音
        }
    }
}
);
promise.catch((error) => {
    // 调用异常时业务侧可以通过 promise.catch 捕获异常进行错误处理
});
```

参考文档：

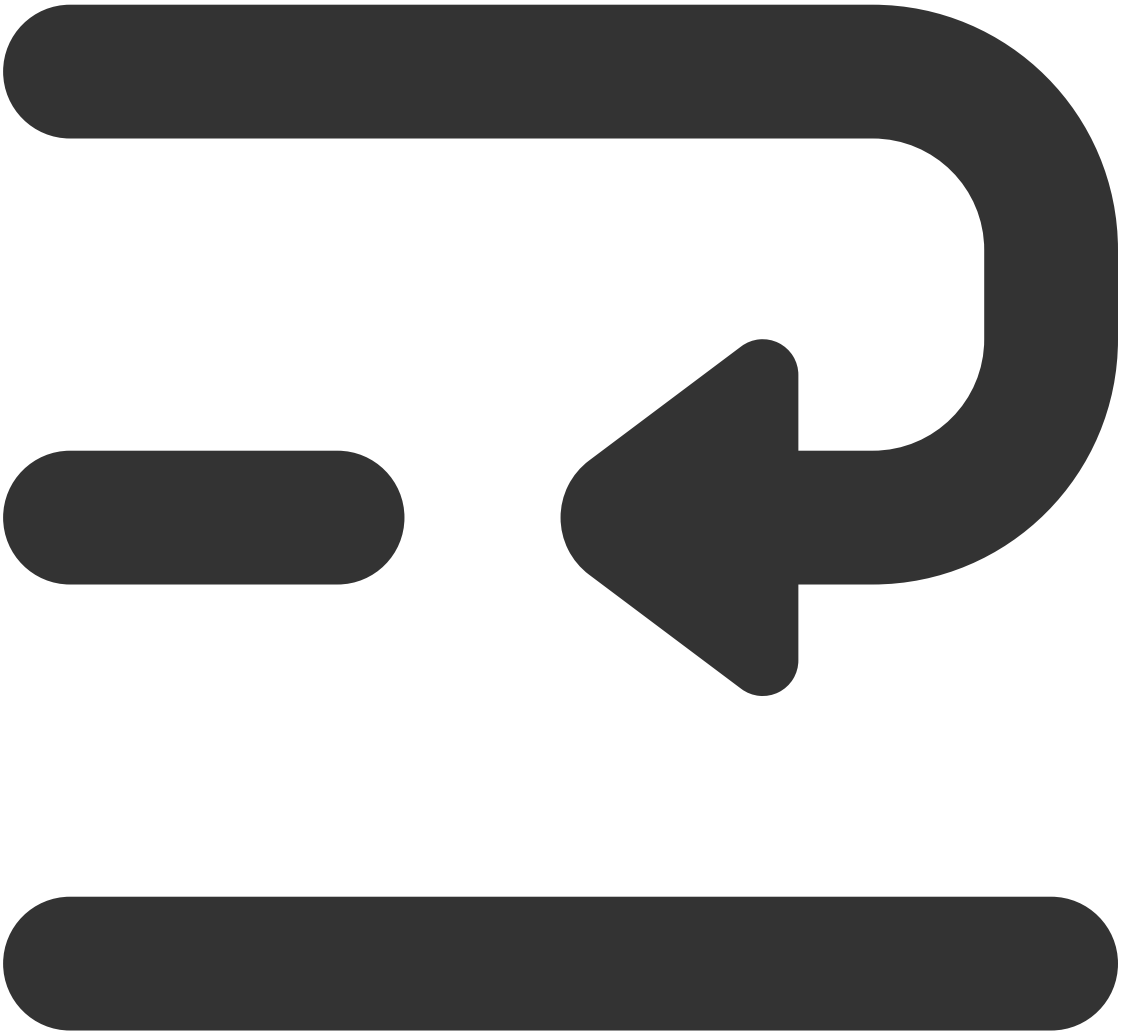
uikit - TUIChatService 发送消息相关文档

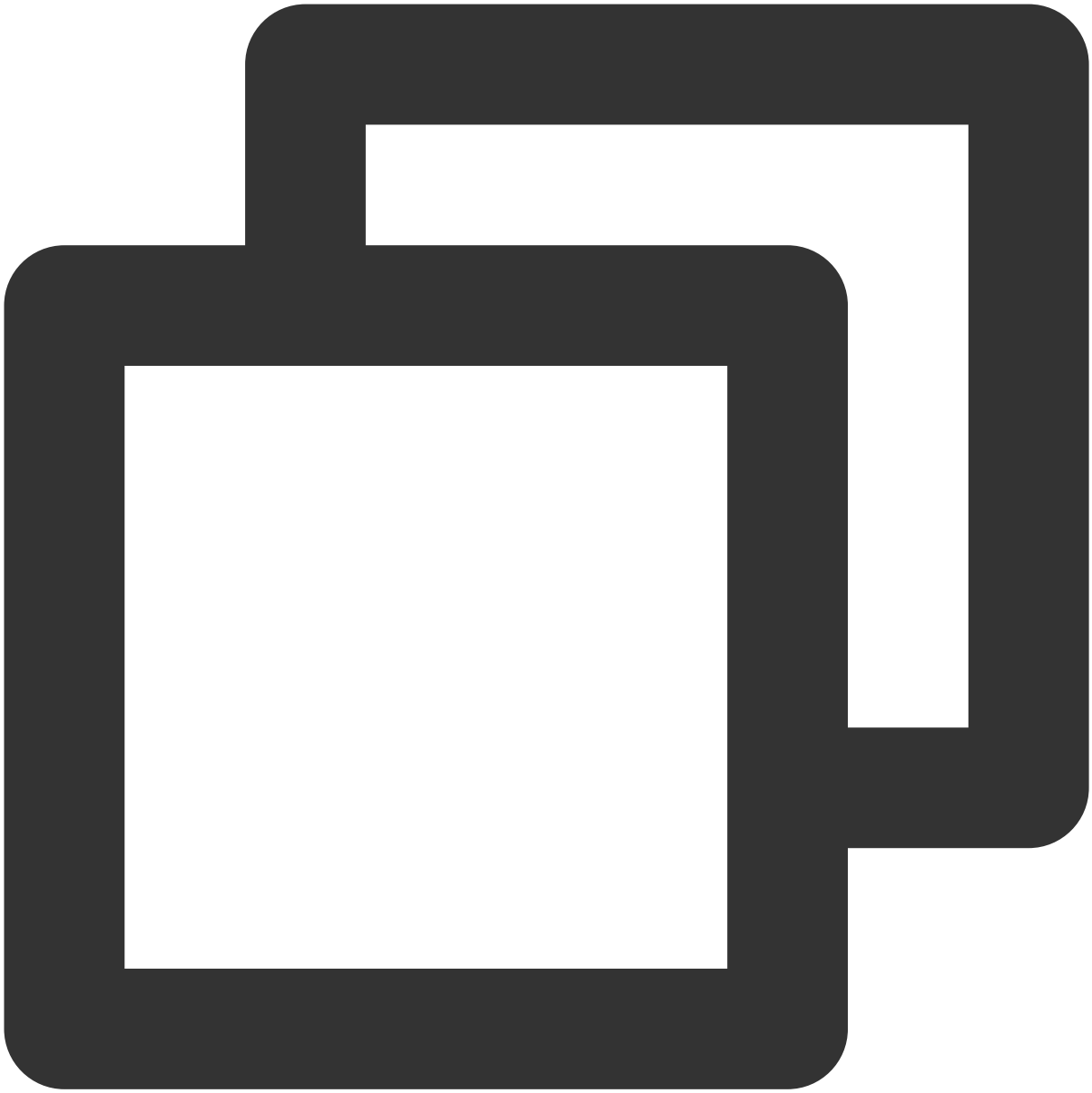
注意：

当 apnsInfo.sound = TencentCloudChat.TYPES.IOS_OFFLINE_PUSH_NO_SOUND，表示接收时不会播放声音。

当 apnsInfo.sound = TencentCloudChat.TYPES.IOS_OFFLINE_PUSH_DEFAULT_SOUND，表示接收时播放系统声音。

在 chat 发送消息时，设置 offlinePushInfo 相关字段，代码如下:





```
// 消息发送选项
chat.sendMessage(message, {
  // 如果接收方不在线，则消息将存入漫游，且进行离线推送（在接收方 App 退后台或者进程被 kill 的情
   offlinePushInfo: {
     androidInfo: { // Android 推送配置
        sound: 'private_ring.mp3', // Andorid 自定义铃音
        XiaoMiChannelID: '', // 小米手机在 Android 8.0 及以上版本必须设置 XiaoMiChannelI
        FCMChannelID: '', // 谷歌手机 FCM 在 Android 8.0 及以上系统设置声音提示，必须设置
      },
      apnsInfo: { // APNs 推送配置
     // apnsInfo.sound = TencentCloudChat.TYPES.IOS_OFFLINE_PUSH_NO_SOUND, 表示接收时
```

```
        // apnsInfo.sound = TencentCloudChat.TYPES.IOS_OFFLINE_PUSH_DEFAULT_SOUND, 表示
           sound: 'private_ring.mp3', // iOS 自定义铃音
         }
      }
  });
```

参考文档：Chat SDK - sendMessage 文档

Flutter

注意：

接口支持华为、小米、FCM 和 APNS。

定制的铃音资源文件，Android 添加到工程 raw 目录下，iOS 链接进 Xcode 工程。

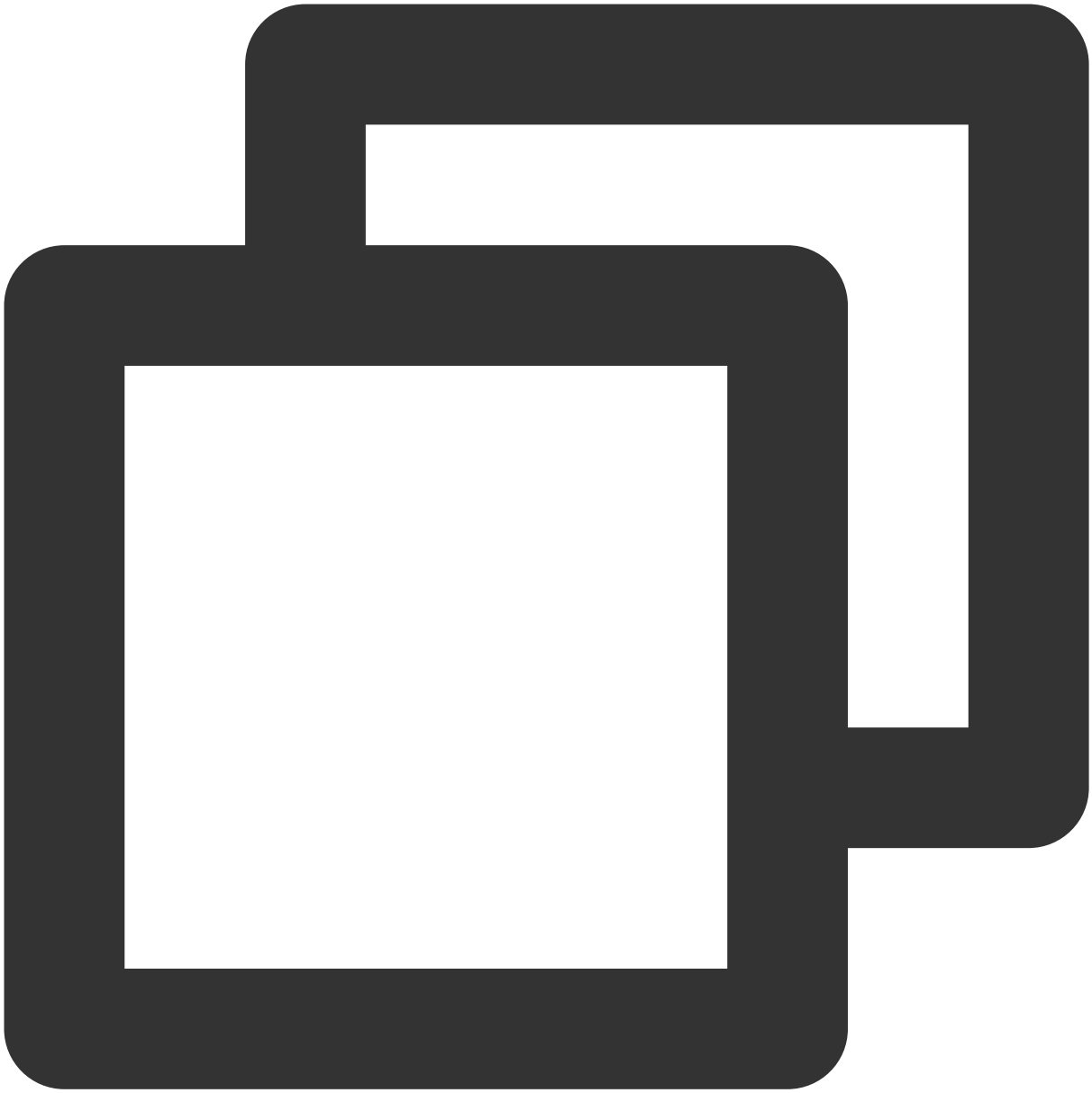
请在调用 sendMessage 发送消息的时候设置 offlinePushInfo 的 `iOSSound` 及 `androidSound` 字段。

具体各厂商配置，请参阅下方 Android 及 iOS 模块的内容。调用的方法均在 Flutter 版本的 IM SDK 中有同名方法。

Android

8.0 以前系统

1. 定制的铃音资源文件，Android 添加到工程 raw 目录下，iOS 链接进 Xcode 工程。
2. 请您发送消息调用接口 setAndroidSound() 和 setIOSSound()。



```
V2TIMOfflinePushInfo v2TIMOfflinePushInfo = new V2TIMOfflinePushInfo();
v2TIMOfflinePushInfo.setAndroidSound("铃音名称");
v2TIMOfflinePushInfo.setIOSSound("铃音名称.mp3");

String msgID = V2TIMManager.getMessageManager().sendMessage(v2TIMMessage, isGroup ?
 V2TIMMessage.V2TIM_PRIORITY_DEFAULT, false, v2TIMOfflinePushInfo, new V2TIMSendCal
@Override
 public void onProgress(int progress) {

 }
```

```
@Override
 public void onError(int code, String desc) {

 }

@Override
 public void onSuccess(V2TIMMessage v2TIMMessage) {

 }
});
```

注意：

IMSDK 6.1.2155 及以上版本支持。

接口支持华为、小米、FCM 和 APNS。

8.0 及以后系统

华为 和 ANPS

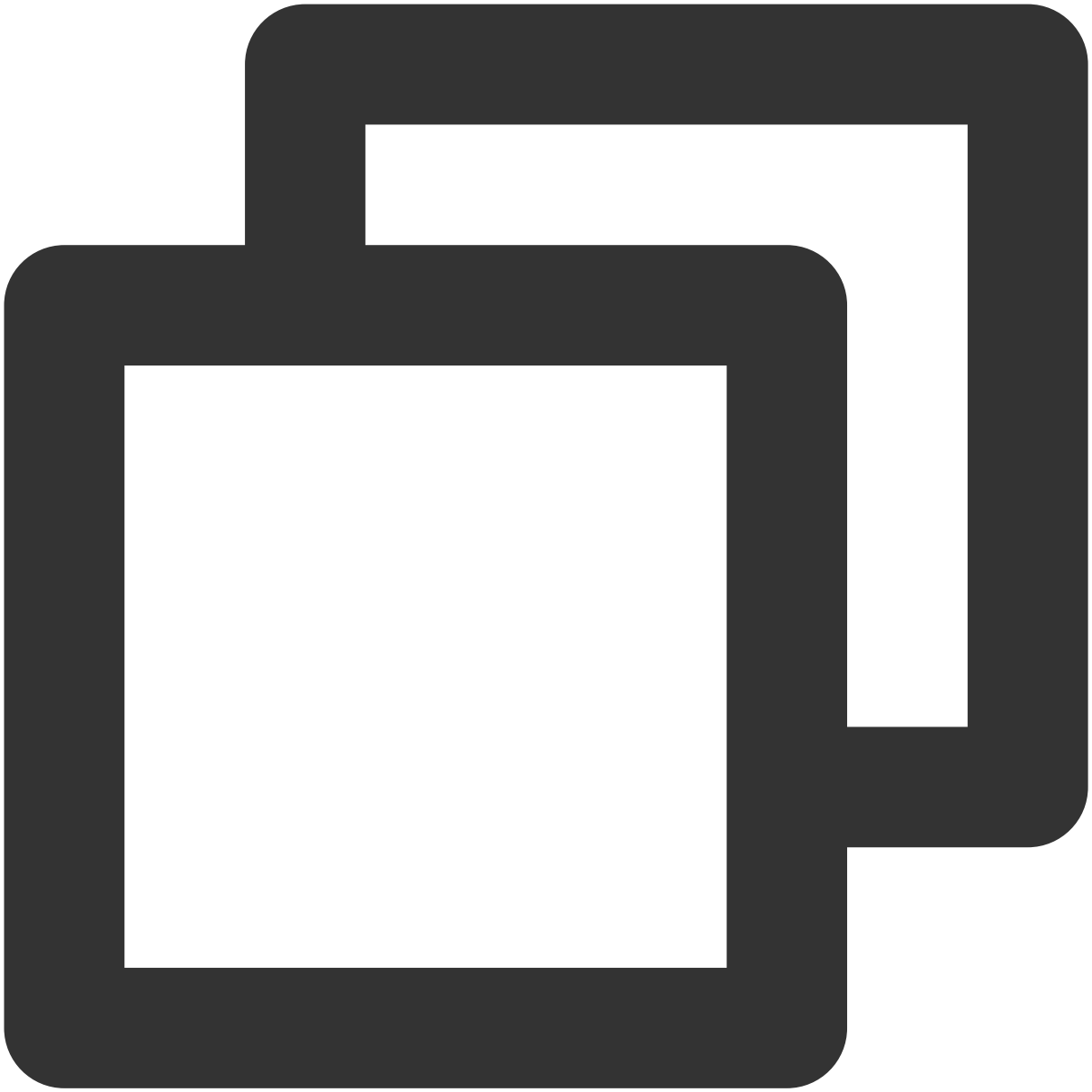
华为、APNS 调用 setAndroidSound() 和 setIOSSound() 来设置离线推送铃音提示。

小米

1. 登录厂商控制台 创建 channel 和配置，其中铃音文件需要添加到您本地 Android Studio 工程的 raw 目录下。

3. 在“创建channel”页面下填写信息并提交申请，如下图所示：

2. 发送消息指定自定义铃音的 channel ID，详情请参见 setAndroidXiaoMiChannelID。

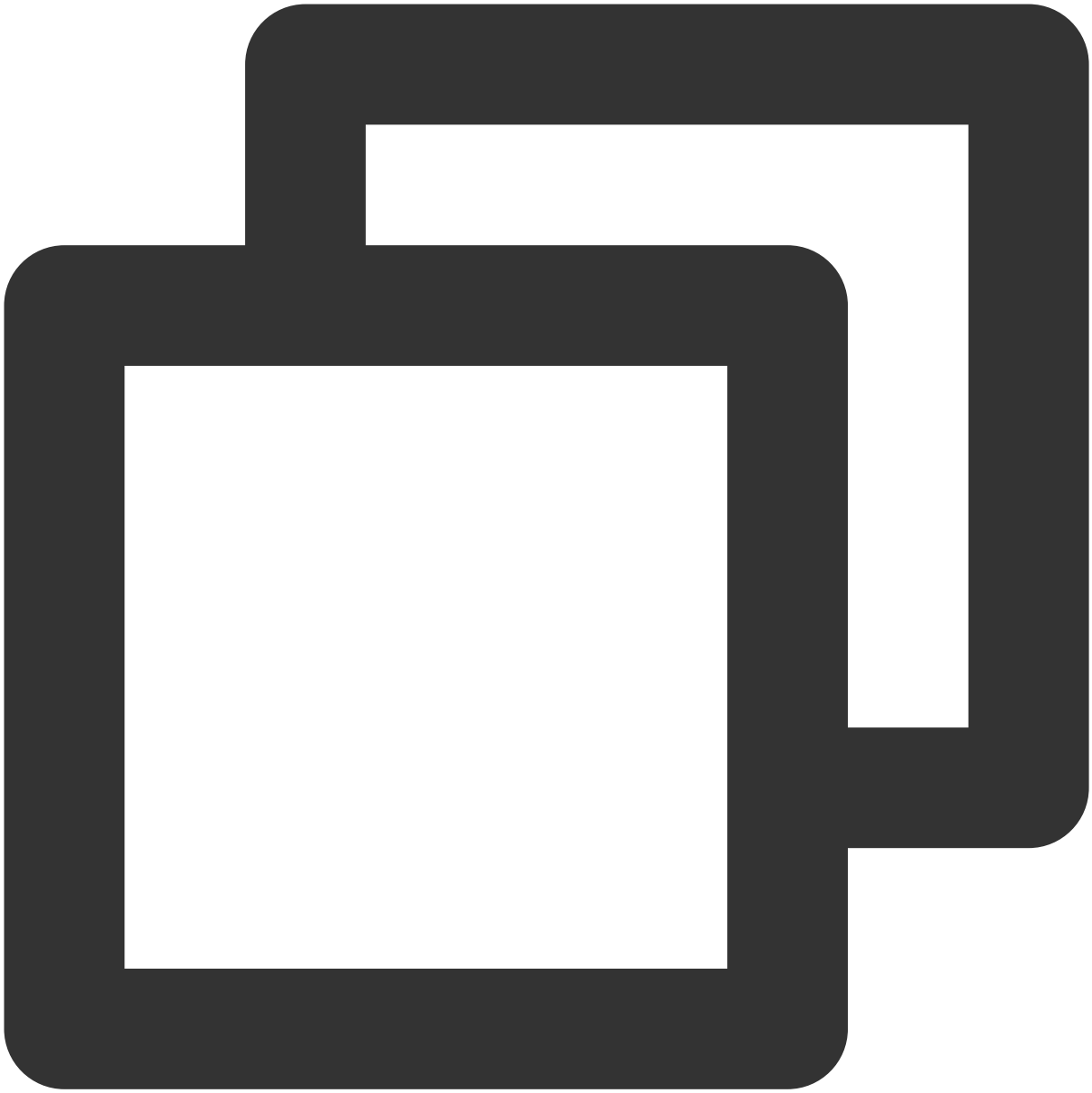


```
V2TIMOfflinePushInfo v2TIMOfflinePushInfo = new V2TIMOfflinePushInfo();
v2TIMOfflinePushInfo.setAndroidXiaoMiChannelID("厂商申请的 channel ID");

String msgID = V2TIMManager.getMessageManager().sendMessage(v2TIMMessage, isGroup ?
 V2TIMMessage.V2TIM_PRIORITY_DEFAULT, false, v2TIMOfflinePushInfo, new V2TIMSendCal
@Override
 public void onProgress(int progress) {
    TUIChatUtils.callbackOnProgress(callBack, progress);
 }

@Override
```

```
    public void onError(int code, String desc) {
        TUIChatUtils.callbackOnError(callBack, TAG, code, desc);
    }

@Override
    public void onSuccess(V2TIMMessage v2TIMMessage) {

    }
});
```

FCM

1. 定制的铃音资源文件添加到工程 raw 目录下，按照接口配置 FCM 自定义铃音参数, 需要在注册推送服务之前调用。详情请参见 configFCMPrivateRing。
2. 发送消息指定自定义铃音的 channel ID，详情请参见 setAndroidFCMChannelID。



```
V2TIMOfflinePushInfo v2TIMOfflinePushInfo = new V2TIMOfflinePushInfo();
v2TIMOfflinePushInfo.setAndroidFCMChannelID(PrivateConstants.fcmPushChannelId);

String msgID = V2TIMManager.getMessageManager().sendMessage(v2TIMMessage, isGroup ?
 V2TIMMessage.V2TIM_PRIORITY_DEFAULT, false, v2TIMOfflinePushInfo, new V2TIMSendCal
@Override
 public void onProgress(int progress) {
    TUIChatUtils.callbackOnProgress(callBack, progress);
 }

@Override
```

```
    public void onError(int code, String desc) {
        TUIChatUtils.callbackOnError(callBack, TAG, code, desc);
    }

@Override
    public void onSuccess(V2TIMMessage v2TIMMessage) {

    }
});
```

注意：

IMSDK 7.0.3754 及以上版本支持。

FCM 自定义铃声或者设置 channnel id 仅支持证书模式。

iOS

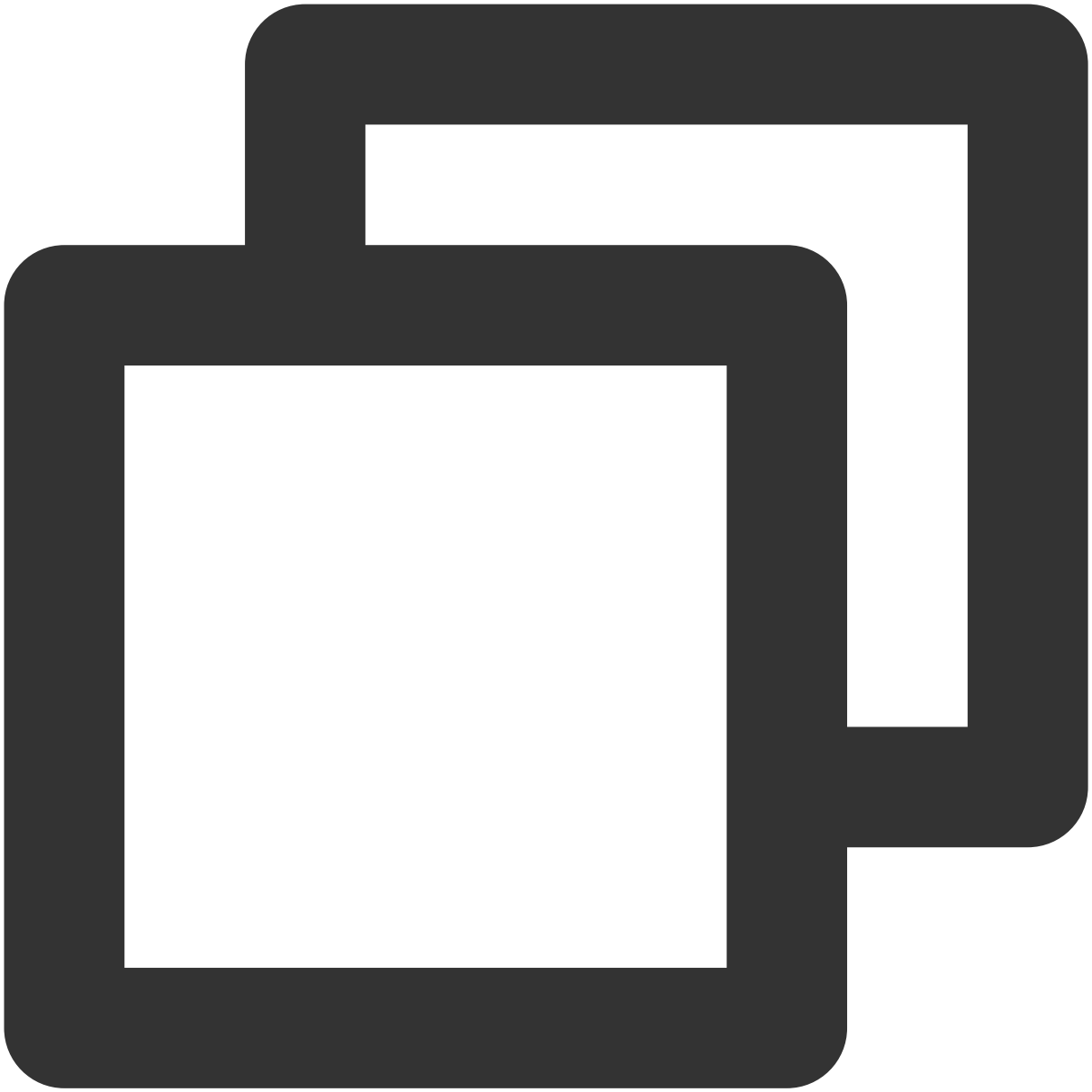
1. 请在调用 sendMessage 发送消息的时候设置 V2TIMOfflinePushInfo 的 `iOSSound` 字段， `iOSSound` 传语音文件名。

说明：

离线推送声音设置（仅对 iOS 生效）， 当 iOSSound = kIOSOfflinePushNoSound，表示接收时不会播放声音。

当 iOSSound = kIOSOfflinePushDefaultSound，表示接收时播放系统声音。

如果要自定义 iOSSound，需要先把语音文件链接进 Xcode 工程，然后把语音文件名（带后缀名）设置给 iOSSound。



```
V2TIMOfflinePushInfo *pushInfo = [[V2TIMOfflinePushInfo alloc] init];
pushInfo.title = @"push title";
pushInfo.iOSSound = @"phone_ringing.mp3"; // your voice file's name
[[V2TIMManager sharedInstance] sendMessage:msg receiver:receiver groupID:groupID pr

} fail:^(int code, NSString *msg) {

}];
```

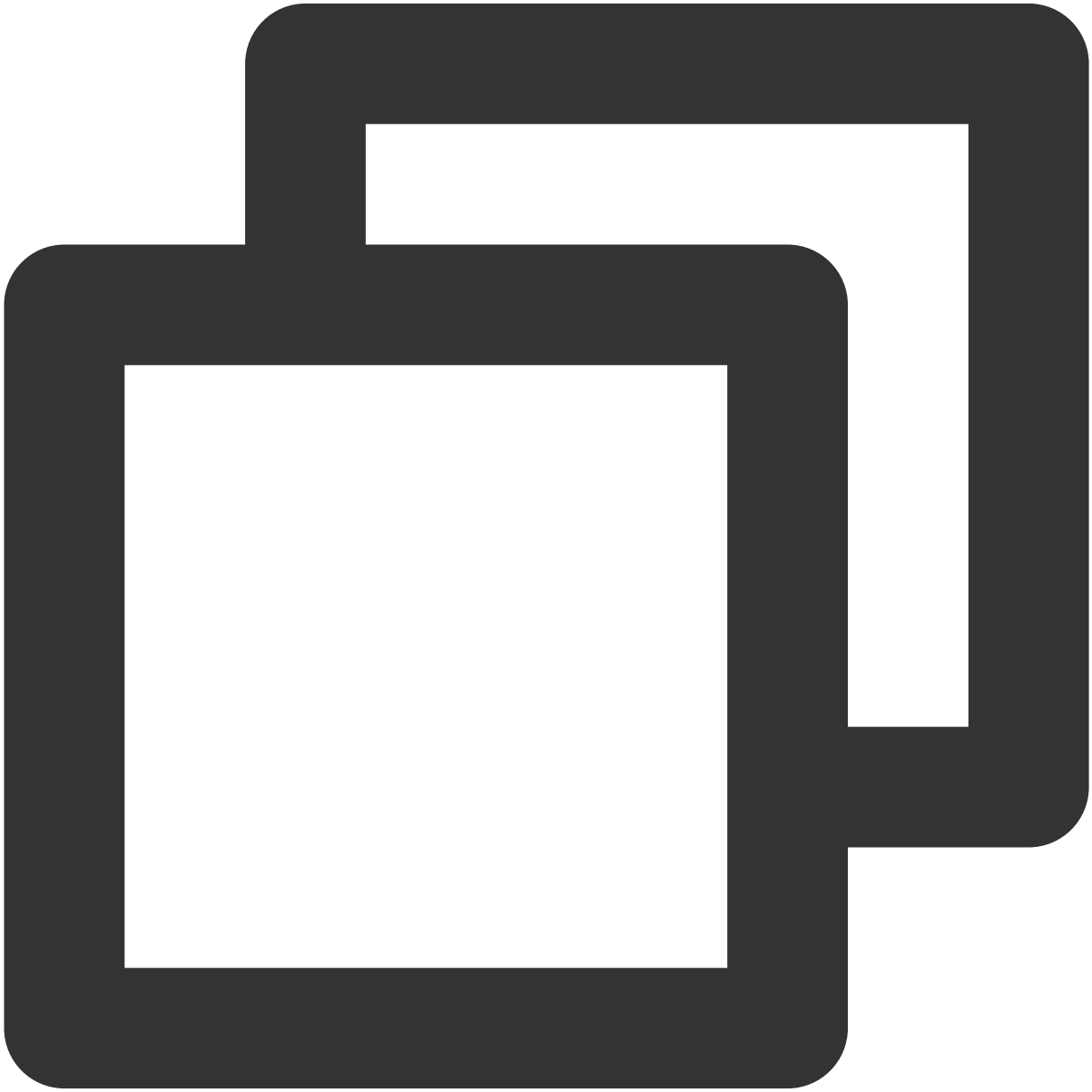
2. 请在调用 sendMessage 发送消息的时候设置 V2TIMOfflinePushInfo 的 `AndroidSound` 字段，`AndroidSound` 传语音文件名。

说明：

离线推送声音设置（仅对 Android 生效, 仅 imsdk 6.1 及以上版本支持） 只有华为和谷歌手机支持设置铃音提示。

小米铃音设置请您参见：服务端 Java SDK 文档。

如果要自定义 AndroidSound，需要先把语音文件放到 Android 工程的 raw 目录中，然后把语音文件名（不需要后缀名）设置给 AndroidSound。



```
V2TIMOfflinePushInfo *pushInfo = [[V2TIMOfflinePushInfo alloc] init];
pushInfo.title = @"push title";
```

```
pushInfo.AndroidSound = @"phone_ringing"; // your voice file's name
[[V2TIMManager sharedInstance] sendMessage:msg receiver:receiver groupID:groupID pr

} fail:^(int code, NSString *msg) {

}];
```

自定义小图标

最近更新时间：2024-06-13 10:39:26

Android

支持厂商

华为和 Google FCM 支持自定义，其他厂商不支持自定义，默认使用 App 图标。

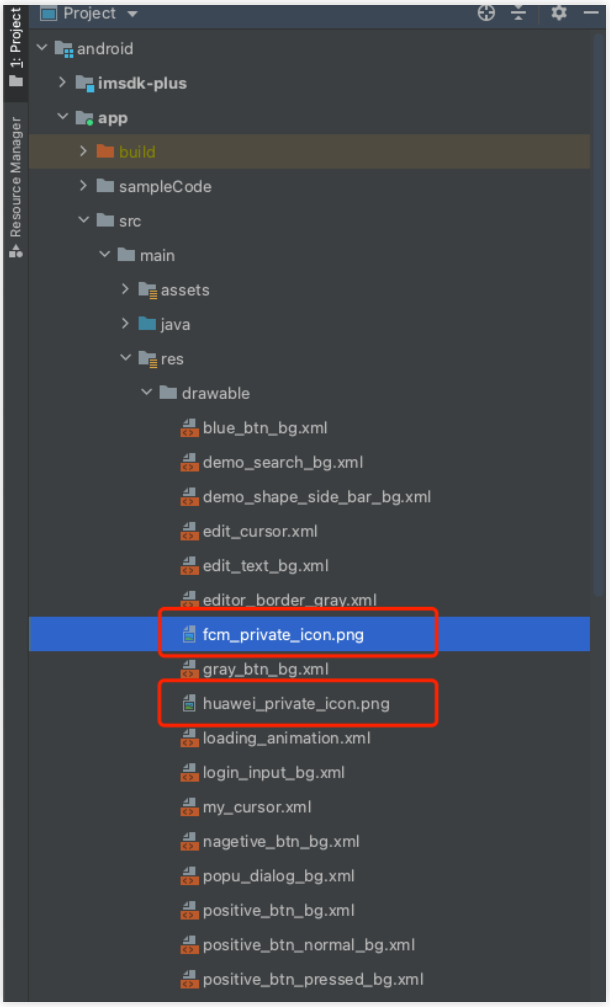
配置方法

方法一

应用工程配置对应的自定义小图标，组件会自动解析和更新；反之使用应用的默认 App 图标。

华为：huawei_private_icon.png

FCM：fcm_private_icon.png

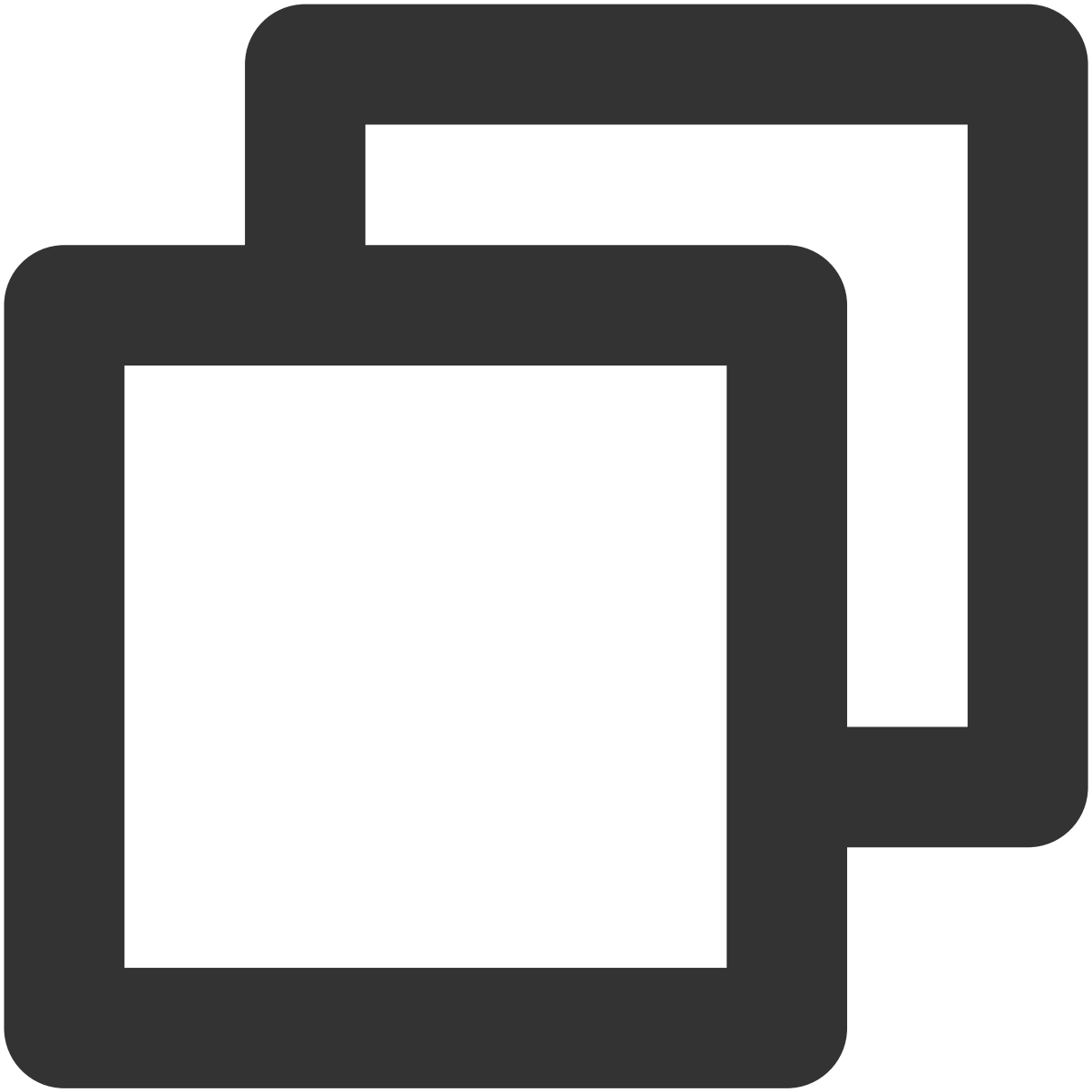


方法二

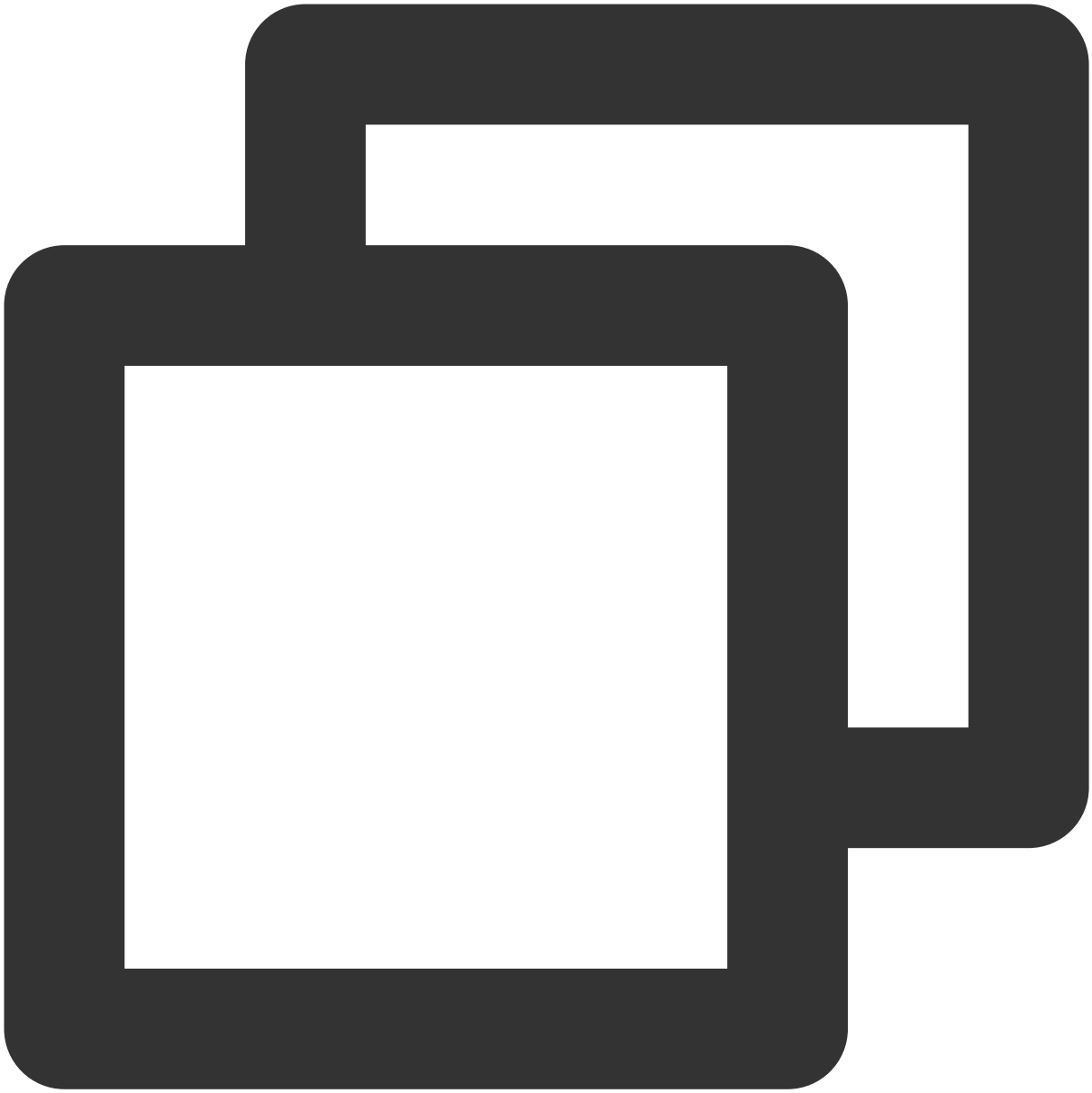
在应用主工程的清单文件配置生效：

华为

Google FCM



```
<meta-data
    android:name="com.huawei.messaging.default_notification_icon"
    android:resource="@drawable/图标资源名称" />
```



```
<!-- [START fcm_default_icon] -->
<!-- Set custom default icon. This is used when no icon is set for incoming notific
     See README(https://goo.gl/l4GJaQ) for more. -->
<meta-data
    android:name="com.google.firebase.messaging.default_notification_icon"
    android:resource="@drawable/图标资源名称" />

<!-- Set color used with incoming notification messages. This is used when no color
     notification message. See README(https://goo.gl/6BKBk7) for more. -->
<meta-data
    android:name="com.google.firebase.messaging.default_notification_color"
```

```
    android:resource="@android:color/white" />
<!-- [END fcm_default_icon] -->
```

注意：

FCM 图标要求：

small icon 必须是带 Alpha 透明通道的 PNG 图片。

背景必须是透明。

周围不宜留过多 padding。

建议统一使用46 x 46px，过小图片会模糊，过大系统会自动缩小。

iOS

不支持自定义，默认使用 App 图标。

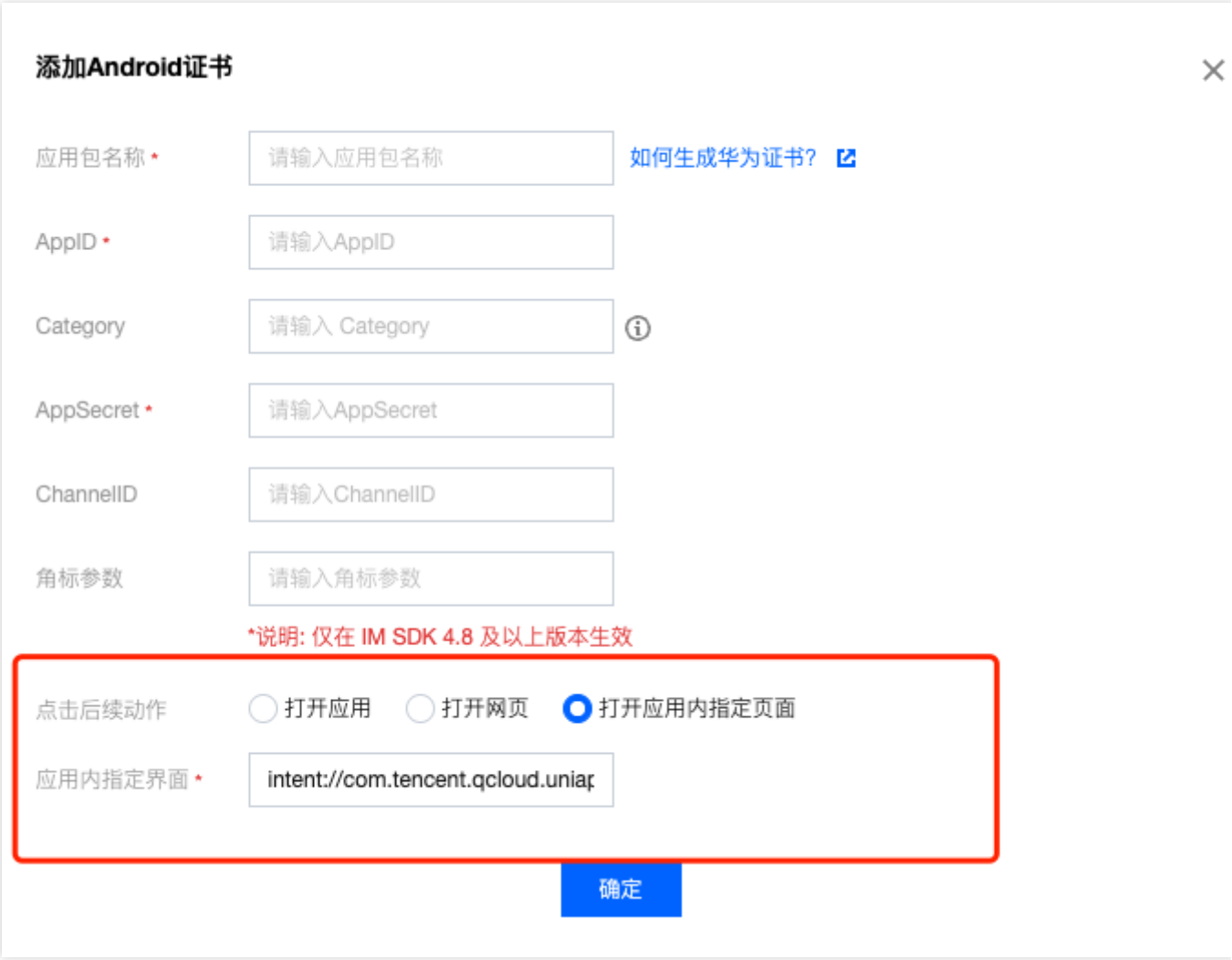
自定义点击跳转

最近更新时间：2024-06-13 10:39:26

Flutter

步骤1: 厂商配置

参见 厂商配置 > Flutter，完成厂商配置。请注意，点击后续动作，请使用默认配置。

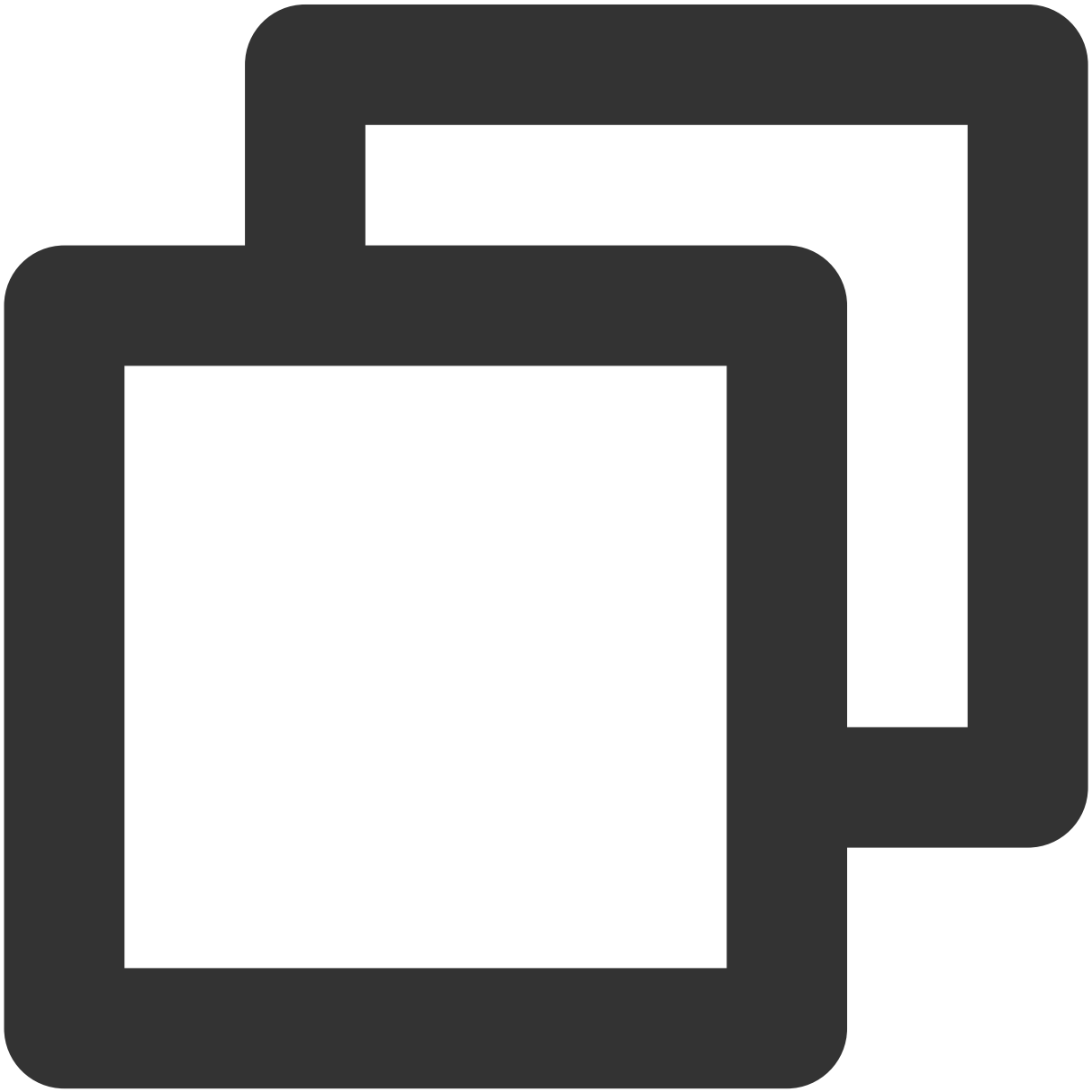


步骤2: 客户端代码配置

参见 客户端代码配置，完成配置。

步骤3: 处理消息点击回调, 并解析参数

请定义一个函数, 用于接受推送消息点击回调事件.
该函数请定义成 `{required String ext, String? userID, String? groupID}` 的入参形式。
其中, ext字段, 为该消息所携带的完整 ext 信息, 由发送方指定, 如果未指定, 则有默认值. 您可根据解析该字段, 跳转至对应页面。
userID 和 groupID 字段，为本插件，自动尝试解析 ext Json String， 获取里面携带的单聊对方 userID 和 群聊 groupID 信息。如果您未自定义 ext 字段，ext 字段由 SDK 或 UIKit 默认指定，则可使用此处的默认解析。如果尝试解析失败, 则为 null 空。
您可定义一个函数来接收该回调，并据此跳转至对应会话页面或您的业务页面。
示例如下:



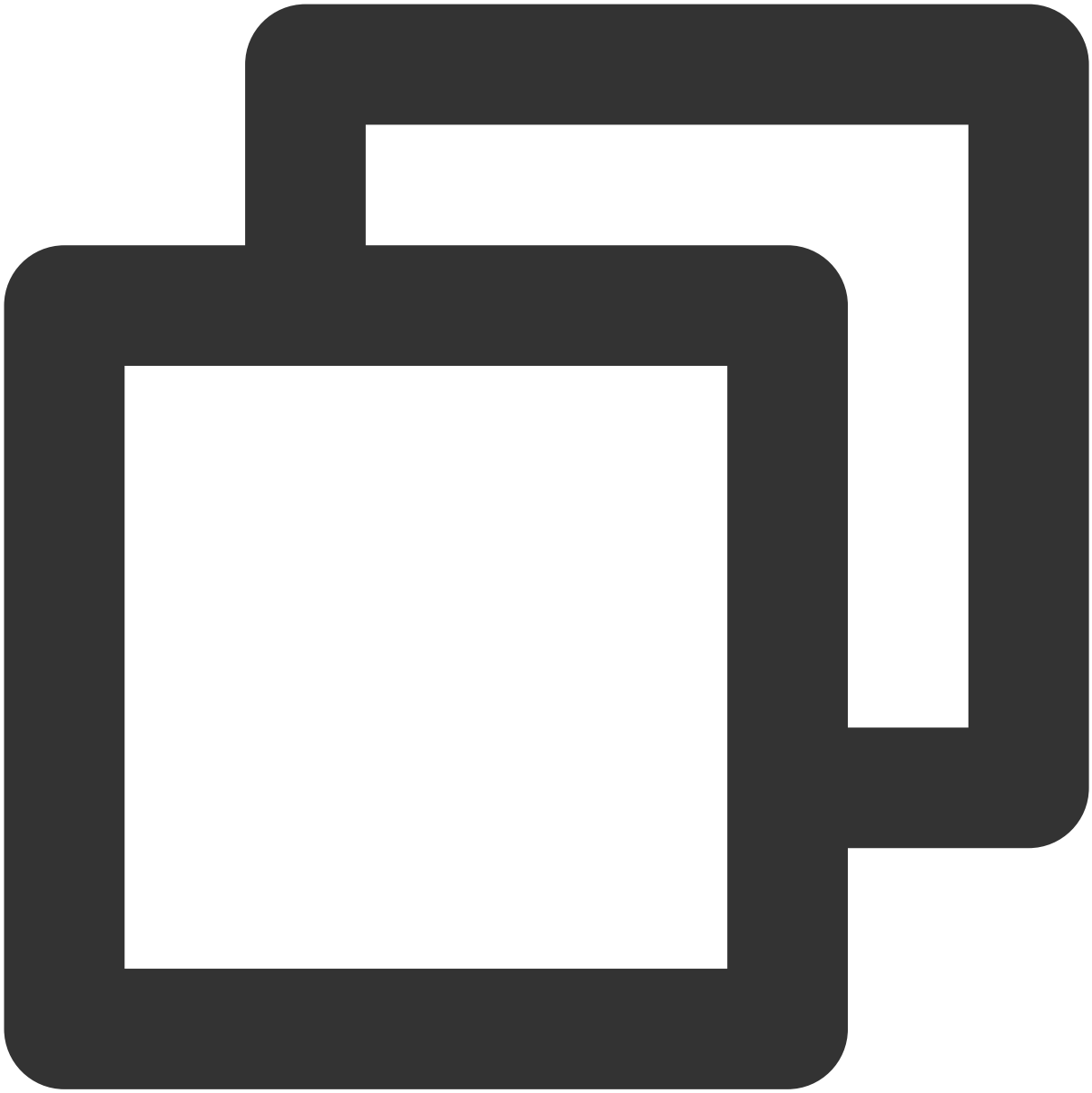
```
void _onNotificationClicked({required String ext, String? userID, String? groupID})
  print("_onNotificationClicked: $ext, userID: $userID, groupID: $groupID");
  if (userID != null || groupID != null) {
    // 根据 userID 或 groupID 跳转至对应 Message 页面.
  } else {
    // 根据 ext 字段, 自己写解析方式, 跳转至对应页面.
  }
}
```

步骤4: 挂载监听回调

请在 IM 登录完成后, 且在其他插件 (如 CallKit) 使用前, 立即注册推送插件。

调用 `TencentCloudChatPush().registerPush` 方法, 需传入上一步定义的点击回调函数。

此外，您还可选传入 `apnsCertificateID` iOS 推送证书 ID 及 `androidPushOEMConfig` Android 推送厂商配置。此二项配置已在前序步骤指定，若无修改必要，可不再传入。

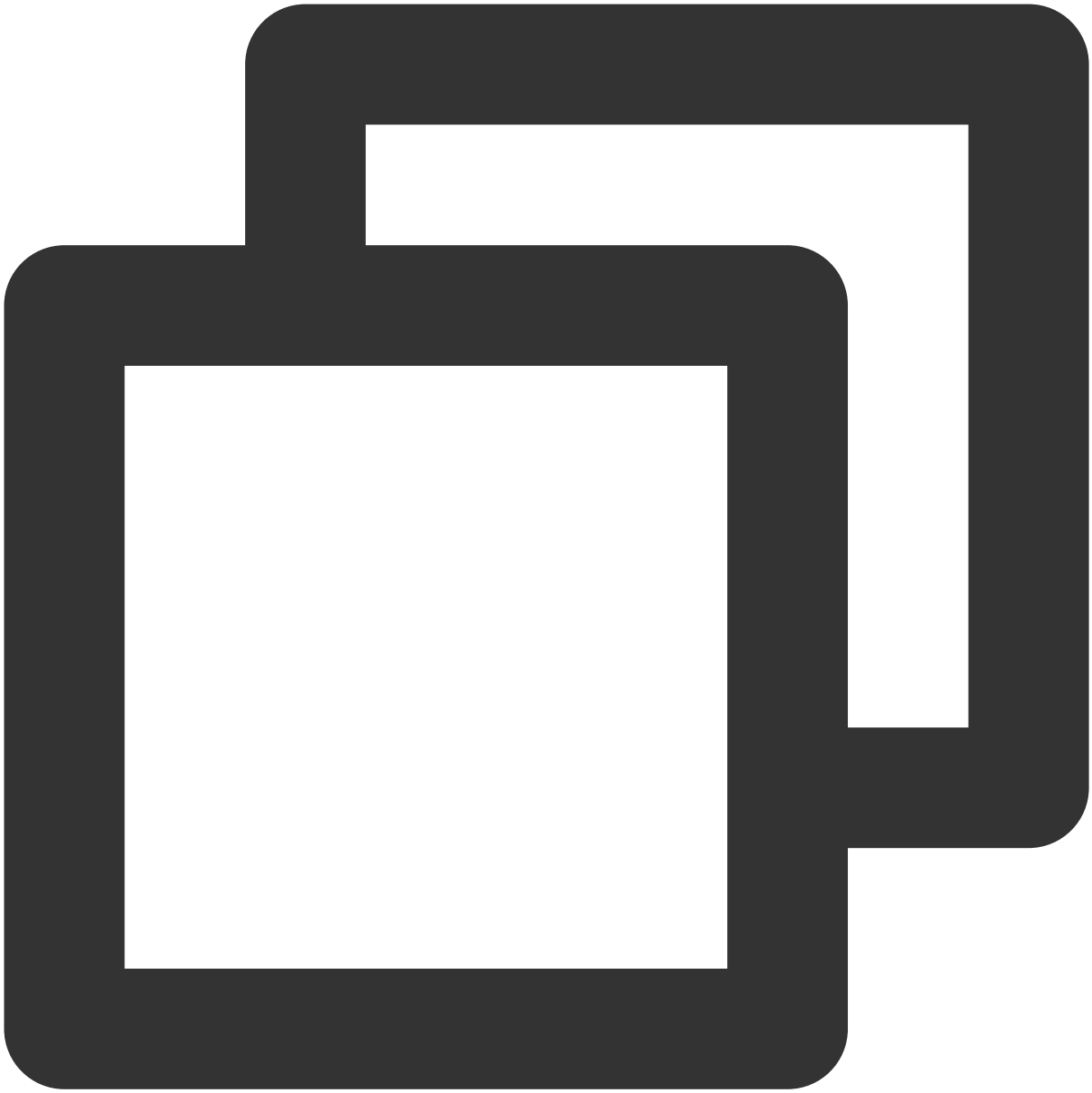


```
TencentCloudChatPush().registerPush(onNotificationClicked: _onNotificationClicked);
```

说明：

如果您的应用需要使用**推送插件进行业务消息通知**，并且在**启动后不会立即启动并登录 IM 模块**，或者在**登录 IM 模块之前需要通过获取消息点击回调来处理业务导航**，建议您尽早调用 `TencentCloudChatPush().registerOnNotificationClickedEvent` 方法，手动挂载消息单击回调，以便及时获取消息参数。

在这种场景下，您可以在调用 `TencentCloudChatPush().registerPush` 之前执行此函数，并尽可能提前将其放置在代码中。

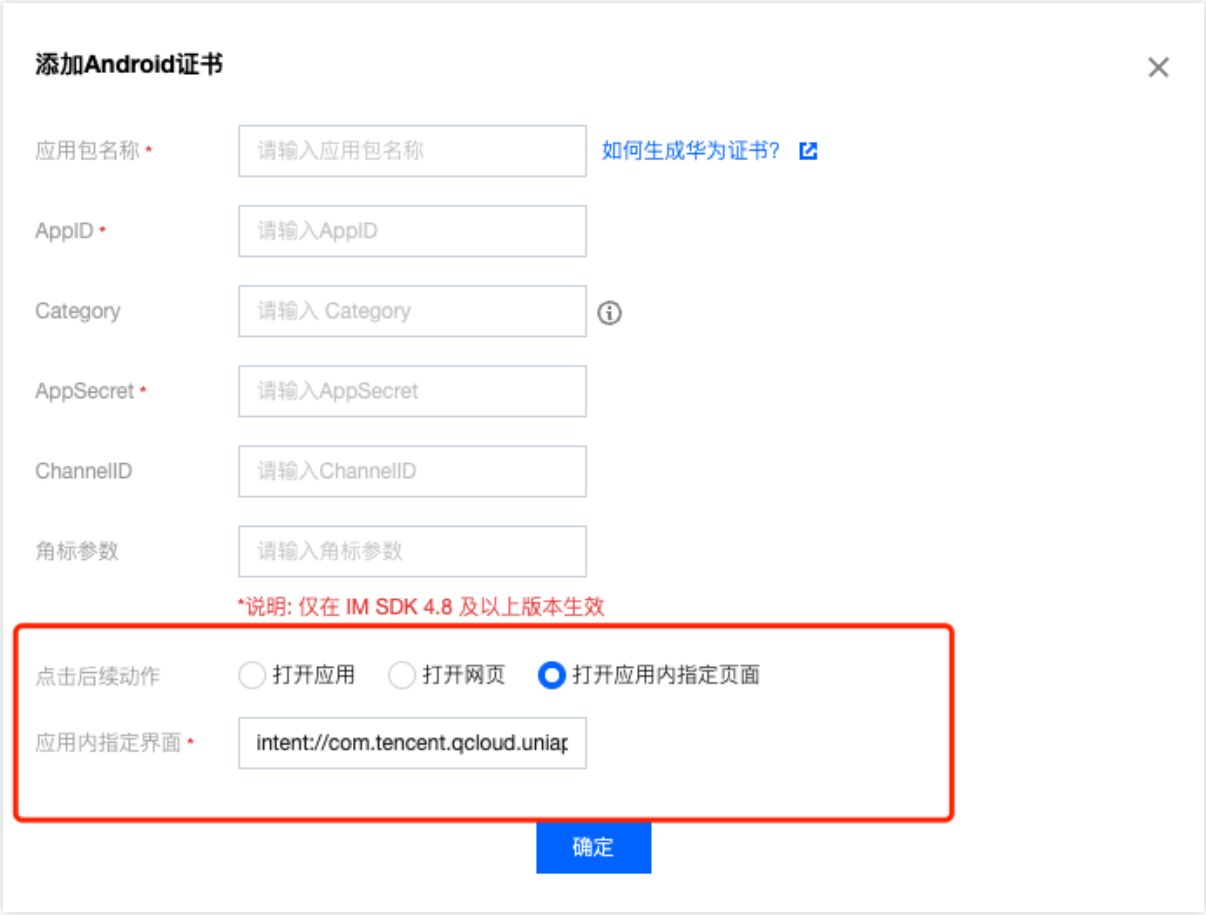


```
TencentCloudChatPush().registerOnNotificationClickedEvent(onNotificationClicked: _o
```

Android

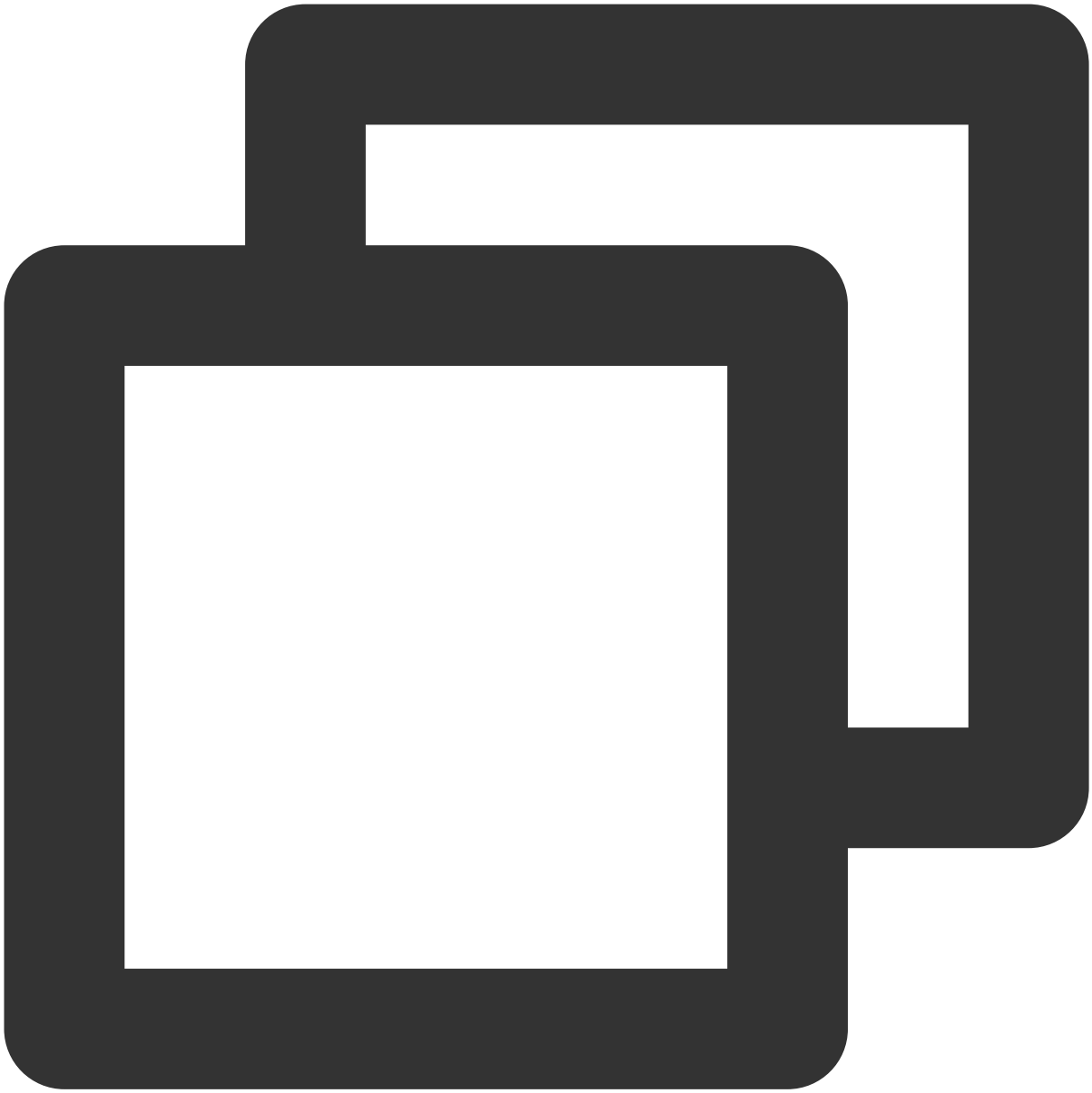
收到离线推送后，通知栏会显示推送信息如图所示，单击通知栏可以自定义打开应用的界面。

1. 控制台配置点击后续动作按如下配置，选择**打开应用内指定界面**，插件用户会默认填写跳转参数。



2. 注册和回调处理点击事件，注册时机建议放在应用 Application 的 oncreate() 函数中。组件会以回调或者广播形式通知应用，应用在回调中配置自定义的跳转页面即可。

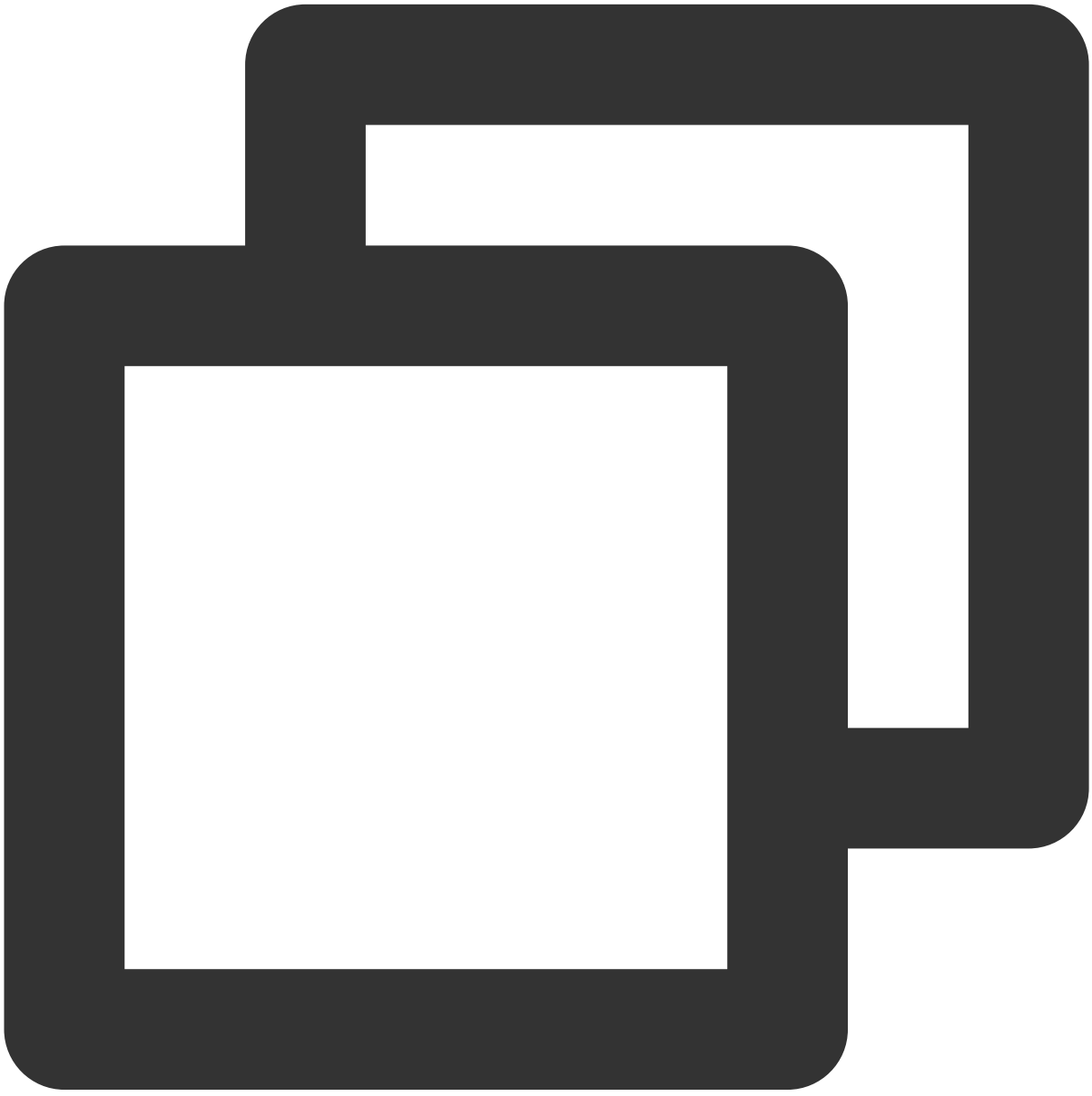
回调方式如下：



```
TUICore.registerEvent(TUIConstants.TIMPush.EVENT_NOTIFY, TUIConstants.TIMPush.EVENT
        @Override
        public void onNotifyEvent(String key, String subKey, Map<String, Object> pa
            Log.d(TAG, "onNotifyEvent key = " + key + "subKey = " + subKey);
            if (TUIConstants.TIMPush.EVENT_NOTIFY.equals(key)) {
                if (TUIConstants.TIMPush.EVENT_NOTIFY_NOTIFICATION.equals(subKey))
                    if (param != null) {
                        String extString = (String)param.get(TUIConstants.TIMPush.N
                        TUIUtils.handleOfflinePush(extString, null);
                    }
                }
```

```
                }
            }
        });
```

广播方式如下：

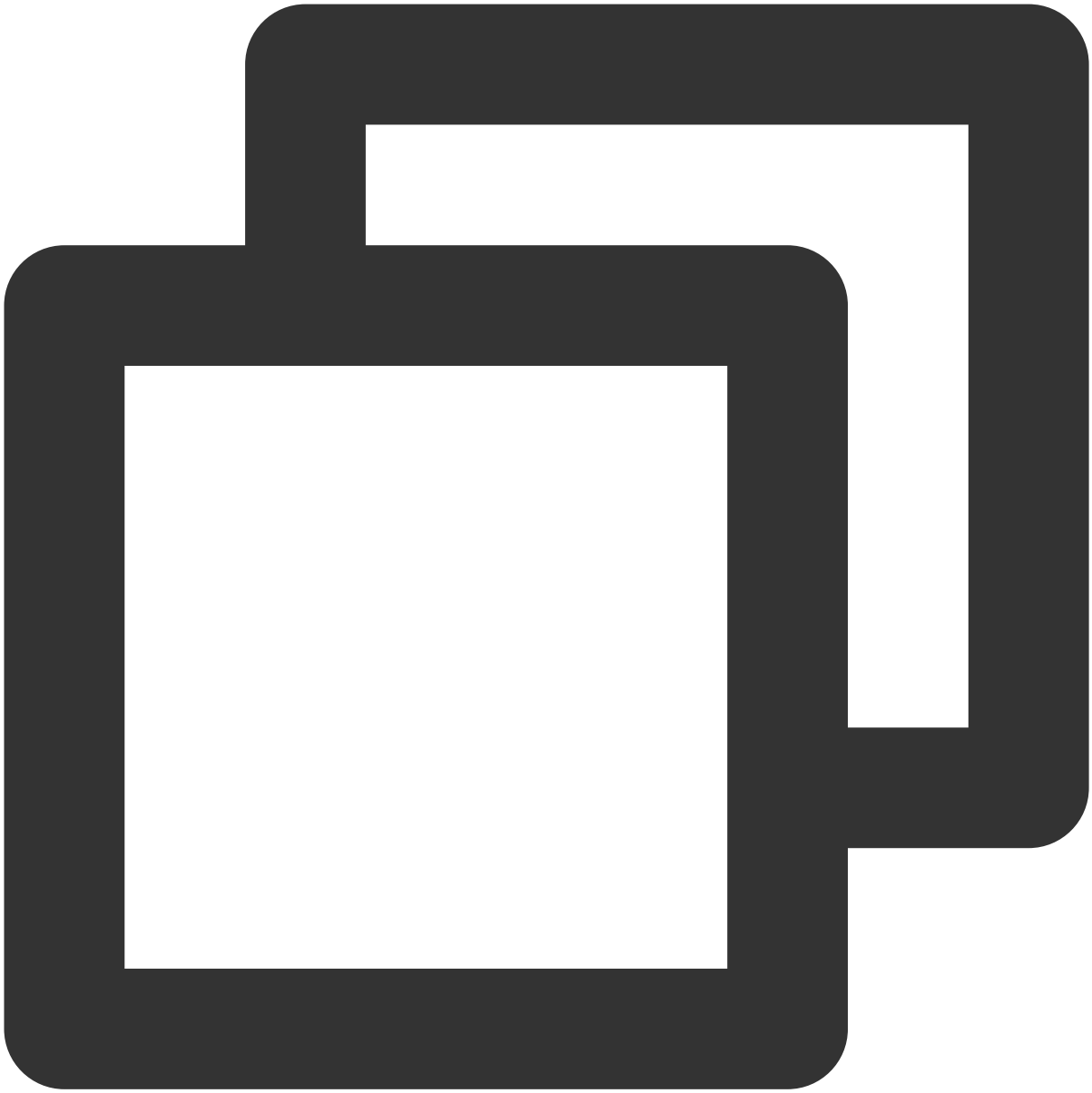


```
// 动态注册广播
IntentFilter intentFilter = new IntentFilter();
intentFilter.addAction(TUIConstants.TIMPush.NOTIFICATION_BROADCAST_ACTION);
LocalBroadcastManager.getInstance(context).registerReceiver(localReceiver, intentFi

//广播接收者
```

```
public class OfflinePushLocalReceiver extends BroadcastReceiver {
    public static final String TAG = OfflinePushLocalReceiver.class.getSimpleName()

    @Override
    public void onReceive(Context context, Intent intent) {
        DemoLog.d(TAG, "BROADCAST_PUSH_RECEIVER intent = " + intent);
        if (intent != null) {
            String ext = intent.getStringExtra(TUIConstants.TIMPush.NOTIFICATION_EX
            TUIUtils.handleOfflinePush(ext, null);
        } else {
            DemoLog.e(TAG, "onReceive ext is null");
        }
    }
}
```

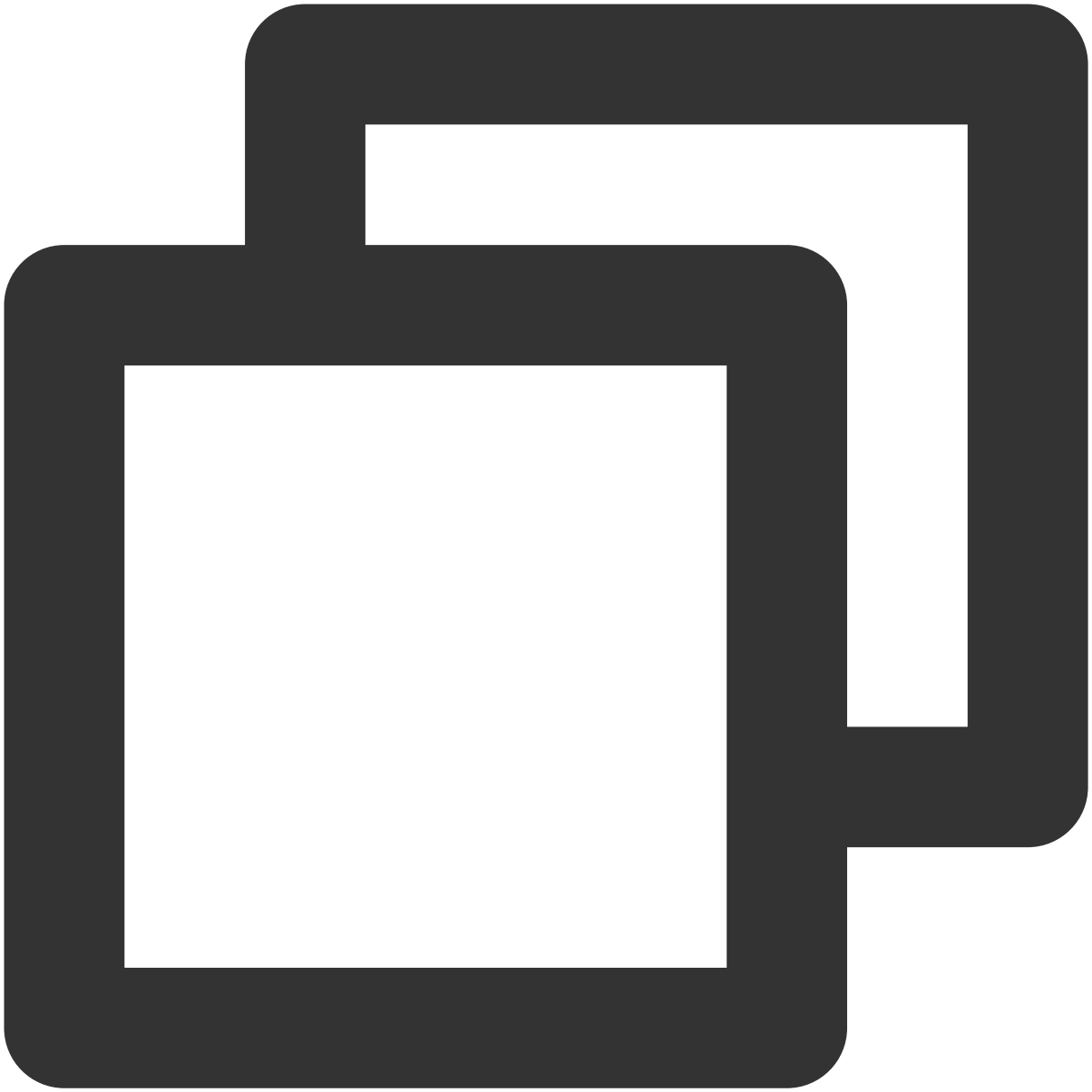
iOS

请在调用 sendMessage 发送消息的时候设置 V2TIMOfflinePushInfo 的 `ext` 字段，当用户收到离线推送启动 App 的时候，可以在 `AppDelegate -> didReceiveRemoteNotification` 系统回调获取到 `ext` 字段，然后根据 `ext` 字段内容跳转到指定的 UI 界面。
以 **denny 给 vinson 发送消息**的场景为例。
发送方：denny 需在发送消息时设置推送扩展字段 `ext` ：



```
// denny在发送消息时设置 offlinePushInfo，并指定 ext 字段
V2TIMMessage *msg = [[V2TIMManager sharedInstance] createTextMessage:@"文本消息"];
V2TIMOfflinePushInfo *info = [[V2TIMOfflinePushInfo alloc] init];
info.ext = @"jump to denny";
[[V2TIMManager sharedInstance] sendMessage:msg receiver:@"vinson" groupID:nil prior
onlineUserOnly:NO offlinePushInfo:info progress:^(uint32_t progress) {
} succ:^{
} fail:^(int code, NSString *msg) {
}];
```

接收方：vinson 的 App 虽然不在线，但可以接收到 APNS 离线推送，当 vinson 点击推送消息时会启动 App：



```
// vinson 启动 APP 后会收到以下回调
- (void)application:(UIApplication *)application didReceiveRemoteNotification:(NSDi
fetchCompletionHandler:(void (^)(UIBackgroundFetchResult result))completionHandler
    // 解析推送扩展字段 desc
    if ([userInfo[@"ext"] isEqualToString:@"jump to denny"]) {
        //跳转到和 denny 的聊天界面
    }
}
```

推送消息分类

最近更新时间：2024-06-13 10:39:26

厂商推送都有消息分类机制，不同类型也会有不同的推送策略。如果推送需求属于 IM 类型推送，想要推送及时触达，需要按照厂商规则设置自己应用为对应的推送类型，会归类为高优先级的系统消息类型或者重要消息类型。反之，离线推送不会及时推送到设备。

注意：

IM 类型消息才有必要配置为系统消息类型或者重要消息类型，进行及时推送。一些营销、广告等类型推送，没有及时推送需求，一定时期内抵达设备即可，不需要配置为系统消息类型。

厂商对于应用每天的推送数量以及推送频率是有限制的，可以在厂商控制台查看应用每日限制的推送数量和限制。

消息类型请勿随意配置，配置不符合标准，可能会被厂商冻结账号。

华为

华为推送从 EMUI 10.0版本开始将通知消息智能分成两个级别：**服务与通讯**和**资讯营销**。EMUI 10.0之前的版本没有对通知消息进行分类，只有一个级别，消息全部通过**默认通知**渠道展示，等价于 EMUI 10.0的服务与通讯。资讯营销类消息的每日推送数量自2023年01月05日起根据应用类型对推送数量进行上限管理，服务与通讯类消息每日推送数量不受限。

自分类推送定制方法

申请自分类权益。

推送消息携带 category 字段，详情请参见 setAndroidHuaWeiCategory，控制台设置见证书编辑 Category 字段，两者设置一个即可。

具体请参见 消息分类标准 或 推送数量管理细则。

荣耀

荣耀手机推送和系统版本有关。

当前荣耀通道仅支持国内 Magic UI 4.0 及以上和境外 Magic UI 4.2 及以上荣耀设备使用。

低于上述版本的荣耀设备可以按照华为厂商接入推送。

具体请参见 产品说明。

vivo

将推送消息分为系统消息类和运营消息类，推送效果和策略不同。系统消息类型还会进行厂商的智能分类二次修正，若智能分类识别出不是系统消息，会自动修正为运营消息，如果误判可邮件申请反馈。另外，消息推送也受日推总数量限制，日推送量由应用在厂商订阅数统计决定。

自分类推送定制方法

推送消息携带 category 字段，详情请参见 setAndroidVIVOCategory，控制台设置参见证书编辑 Category 字段，两者设置一个即可。
具体请参见 推送消息分类说明 或 推送消息限制说明。

OPPO

将推送消息分为私信消息类和公信消息类，推送效果和策略不同。其中私信消息是针对用户有一定关注度，且希望能及时接收的信息，私信通道权益需要邮件申请。公信通道推送数量有限制。

自分类推送定制方法

创建私信通道。
推送消息携带 channel ID 字段，详情请参见 setAndroidOPPOChannelID，控制台设置见证书编辑 ChannelID 字段，两者设置一个即可。
具体请参见 消息分类说明 或 推送服务受限说明。

小米

将推送消息分为“私信消息”和“公信消息”两个类别，默认通道为公信消息。公信消息的单日推送数量将进行上限管理，公信消息适用于推送热点新闻、新品推广、平台公告、社区话题、有奖活动等，多为用户普适性的内容。私信消息适用于推送聊天消息、个人订单变化、快递通知、交易提醒、IOT系统通知等与私人通知相关的内容，通知消息的推送数量不受限制。消息分类管理实现需要在厂商控制台进行 channel 申请及接入。

自分类推送定制方法：

channel申请及接入。
推送消息携带 channel ID 字段，详情请参见 setAndroidXiaoMiChannelID，控制台设置见证书编辑 ChannelID 字段，两者设置一个即可。
具体请参见 推送消息分类新规 或 推送消息限制说明。

魅族

推送消息数量有限制。
具体请参见 推送接入指南。

FCM

推送上行消息频率有限制。
具体请参见 消息限频。
说明：
推送消息 Channel ID 和 分类 category 字段的设置有 API 接口和控制台证书设置两种方式，作用范围会有差异，API 设置优先级高于控制台设置。

更新日志

最近更新时间：2024-06-13 10:39:26

TIMPush 7.9.5668 @2024.04.09

增加 OfflinePushExtInfo 支持透传推送特性。
升级 vivo 推送包版本。
解决 OPPO 和 vivo 机型偶现崩溃问题。
解决 FCM Data 模式多个推送通知，透传数据偶现对应不正确问题。
优化 WorkManager 编译问题。
UniApp 支持触达上报，支持杀进程回调点击事件。

TIMPush - Android 7.8.5484 @2024.02.02

发布 TIMPush-UniApp。
FCM 推送支持透传消息。
推送注册及上报逻辑优化。

TIMPush - iOS 7.8.5483 @2024.02.02

iOS 支持最低版本更新为 10.0及以上。

TIMPush 7.7.5283 @2023.12.28

新增自定义推送配置文件功能。
升级荣耀推送包。
优化 vivo 占位符配置。
解决 iOS 托管 UNUserNotificationCenterDelegate 后，部分函数没有回调的问题。

TIMPush 7.7.5282 @2023.12.18

升级小米和魅族推送包。
优化 context 偶现为空问题。
优化 registerPush 手动调用接口逻辑。

TIMPush 7.6.5011 @2023.11.13

发布推送插件。

常见问题

最近更新时间：2024-06-13 10:39:26

Android

收不到离线推送怎么排查？

特殊情况排查

OPPO 手机

OPPO 手机收不到推送一般有以下几种情况：
按照 OPPO 推送官网要求，在 Android 8.0 及以上系统版本的 OPPO 手机上必须配置 ChannelID，否则推送消息无法展示。配置方法可以参见 setAndroidOPPOChannelID。
OPPO 安装应用通知栏显示默认关闭，需要确认下开关状态。

Google FCM

收不到推送需要确认下 IM 控制台是否正确上传证书。排查路径参见文档 IM 控制台配置 > Google FCM，对照示意图看下是否添加正确。

小米和 vivo

小米和 vivo：需要上架应用市场后，才可以通过厂商通道进行下发。一般会提示：应用在黑名单中，禁止发送消息。或者，该 App 已关闭 push 通道。

发送消息为自定义消息

自定义消息的离线推送和普通消息不太一样，自定义消息的内容我们无法解析，不能确定推送的内容，所以默认不推送，如果您有推送需求，需要您在 sendMessage 的时候设置 offlinePushInfo 的 desc 字段，推送的时候会默认展示 desc 信息。

设备通知栏设置影响

离线推送的直观表现就是通知栏提示，所以同其他通知一样受设备通知相关设置的影响，以华为为例：
手机设置 > 通知 > 锁屏通知 > 隐藏或者不显示通知，会影响锁屏状态下离线推送通知显示。
手机设置 > 通知 > 更多通知设置 > 状态栏显示通知图标，会影响状态栏下离线推送通知的图标显示。
手机设置 > 通知 > 应用的通知管理 > 允许通知，打开关闭会直接影响离线推送通知显示。
手机设置 > 通知 > 应用的通知管理 > 通知铃声和**手机设置 > 通知 > 应用的通知管理 > 静默通知**，会影响离线推送通知铃音的效果。

消息分类

厂商推送都有消息分类机制，不同类型也会有不同的推送策略，详情请参见 推送消息分类。

厂商特性

离线推送依赖厂商能力，一些简单的字符可能会被厂商过滤不能透传推送，建议使用有意义的内容进行推送。

自助推送排查工具

设备推送插件接入是否正常

在 IM 控制台通过 离线测试工具 自测下是否可以正常推送。

推送链路排查

使用 排查工具 查看推送全链路推送详情。

跳转界面不成功怎么排查？

点击跳转配置检查

详见 自定义点击跳转，检查：
控制台配置点击后续动作按如下配置，选择**打开应用内指定界面**，插件用户会默认填写跳转参数，参数是否被修改。
注册和回调处理点击事件，注册时机建议放在应用 Application 的 oncreate() 函数中，需要放在应用生命周期的靠前位置。
点击回调处是否正确处理跳转逻辑。

设备系统权限限制

应用进程不存在，单击通知栏跳转到应用界面，需要将应用从后台拉取到前台，部分厂商系统会去检查 App 是否开启了**后台自启动**或**悬浮窗**权限，不开启系统侧会拦截处理导致失败。
不同厂商、同一厂商不同 Android 版本，其对于应用开放的权限以及权限名称会存在不一致。经测试，小米6只需要开启后台弹出界面权限，而红米需要同时打开后台弹出界面和显示悬浮窗权限，需要针对不同厂商不同处理。

推送链路排查

使用 排查工具 查看推送全链路推送详情。

其他问题

华为

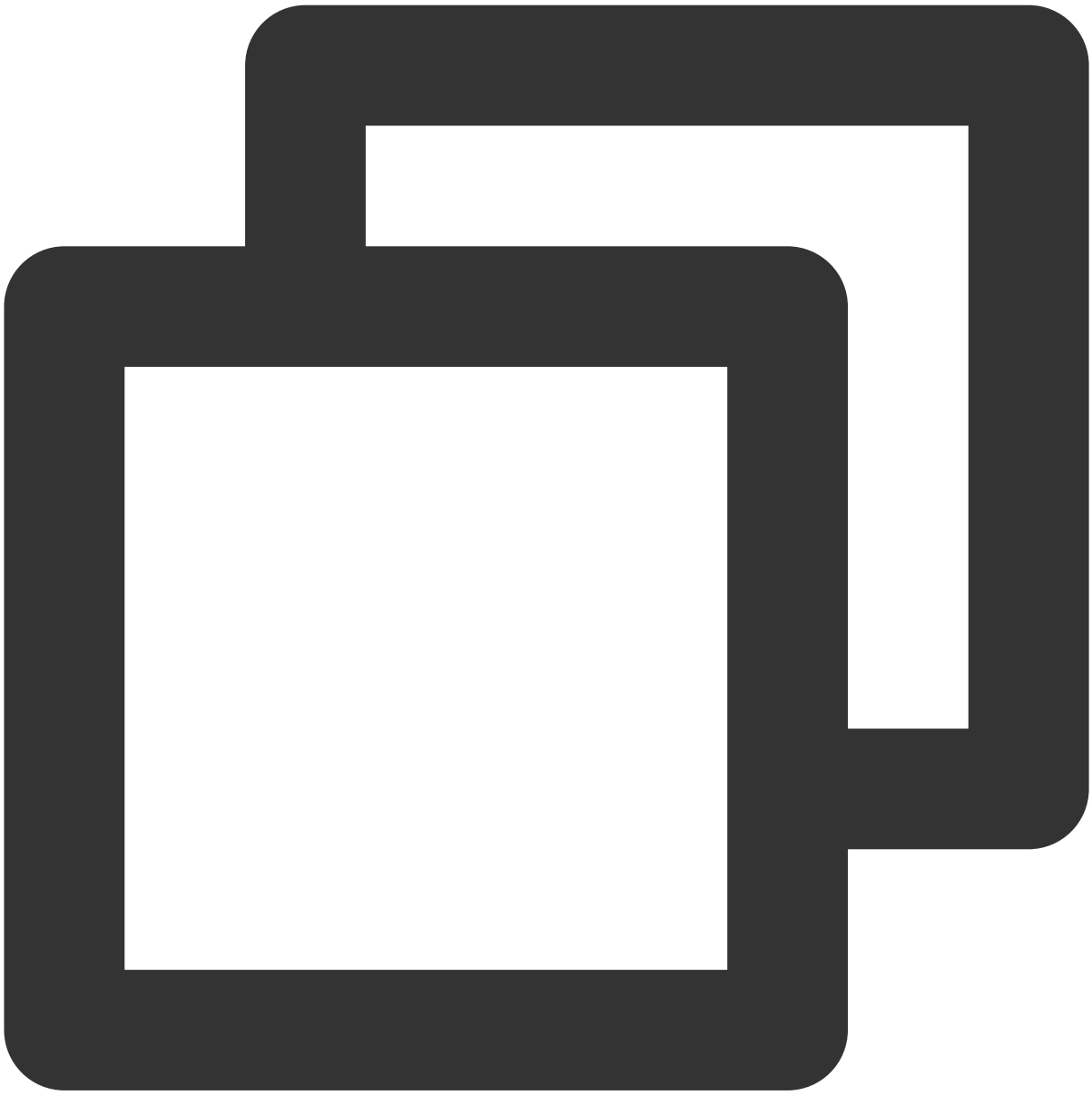
华为推送证书 ID <= 11344 ，使用华为推送 v2 版本接口，不支持触达和单击回执，如需要统计数据功能，请重新生成更新证书 ID 。
AndroidInfo.ExtAsHuaweiIntentParam，传“1”表示将透传内容 Ext 作为 Intent 的参数，“0”表示将透传内容 Ext 作为 Action 参数。restapi 使用“1”暂时不支持点击事件统计。

数据统计功能

只会记录最后一个登录设备的推送数据详情，不支持多端登录场景。

Debug 版本的 App 功能正常，Release 版本的 App 功能出现异常

出现此问题很大概率是混淆导致的，可以添加如下混淆规则：



```
-keep class com.tencent.qcloud.** { *; }
-keep class com.tencent.timpush.** { *; }
```

解决接入 TIMPush 和其他友商产生冲突问题

原因是应用程序自身集成或者依赖的第三方推送客户端，与 TIMPush 中的第三方客户端产生冲突，需要仅保留一个使用。具体方法请参见：TIMPush 集成冲突解决。

iOS

普通消息为什么收不到离线推送？

首先，请检查下 App 的运行环境和证书的环境是否一致，如果不一致，收不到离线推送。
其次，检查下 App 和证书的环境是否为生产环境。如果是开发环境，向苹果申请 `deviceToken` 可能会失败，生产环境暂时没有发现这个问题，请切换到生产环境测试。

自定义消息为什么收不到离线推送？

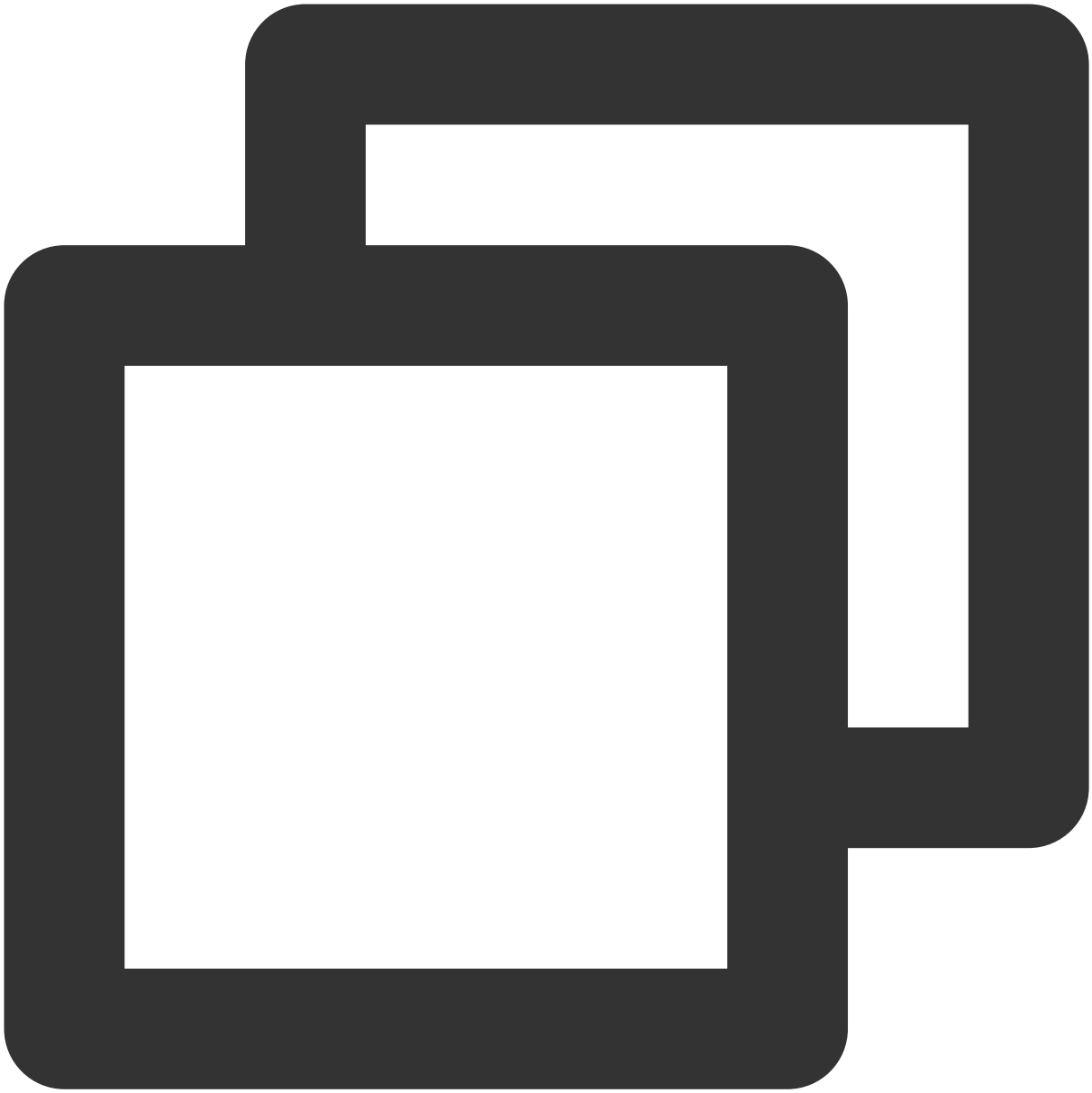
自定义消息的离线推送和普通消息不太一样，自定义消息的内容我们无法解析，不能确定推送的内容，所以默认不推送，如果您有推送需求，需要您在 sendMessage 的时候设置 offlinePushInfo 的 `desc` 字段，推送的时候会默认展示 `desc` 信息。

如何关闭离线推送消息的接收？

如果您想关闭离线推送消息的接收，可以通过设置 setAPNS 接口的 `config` 参数为 `nil` 来实现。该功能从 5.6.1200 版本开始支持。

收不到推送，且后台报错 bad devicetoken。

Apple 的 deviceToken 与当前编译环境有关。如果 登录 IMSDK 后上传 deviceToken 到腾讯云 所使用的证书 ID 和 token 不一致，就会报错。
如果使用的是 Release 环境编译，则 `-application:didRegisterForRemoteNotificationsWithDeviceToken:` 回调返回的是发布环境的 token，此时 businessID 需要设置生产环境的 证书 ID。
如果使用的是 Debug 环境编译，则 `-application:didRegisterForRemoteNotificationsWithDeviceToken:` 回调返回的是开发环境的 token，此时 businessID 需要设置开发环境的 证书 ID。



```
V2TIMAPNSConfig *confg = [[V2TIMAPNSConfig alloc] init];
/* 用户自己到苹果注册开发者证书，在开发者帐号中下载并生成证书(p12 文件)，将生成的 p12 文件传到腾讯
//推送证书 ID
confg.businessID = sdkBusiId;
confg.token = self.deviceToken;
[[V2TIMManager sharedInstance] setAPNS:confg succ:^{
   NSLog(@"%s, succ, %@", __func__, supportTPNS ? @"TPNS": @"APNS");
} fail:^(int code, NSString *msg) {
   NSLog(@"%s, fail, %d, %@", __func__, code, msg);
}];
```

iOS 开发环境下，注册偶现不返回 deviceToken 或提示 APNs 请求 token 失败？

此问题现象是由于 APNs 服务不稳定导致的，可尝试通过以下方式解决：

1. 给手机插入 SIM 卡后使用4G网络测试。
2. 卸载重装、重启 App、关机重启后测试。
3. 打生产环境的包测试。
4. 更换其它 iOS 系统的手机测试。

iOS 没有收到触达回执

1. 上报推送触达数据，需要开启控制台开关来支持 iOS 10 的 Extension 功能，详情请参见 统计推送抵达率。
2. 为保证上报数据在一些异常情况下不丢失，iOS 推送数据的上报会在下次登录时候才上报，可能会有滞后，请尝试重新登录。

iOS 证书过期

在 IM 控制台单击编辑对应证书，更新下 p12 证书即可。需要注意的是：请勿删除证书重新创建，重新创建证书 ID 会变化，需要发版本支持更新。

iOS 统计上报功能

上报推送触达数据，需要开启此开关来支持 iOS 10 的 Extension 功能，详情请参见 统计推送抵达率。

其他问题

收不到推送，插件费用到期

推送插件**试用或购买到期后，将自动停止提供推送服务（包括普通消息离线推送、全员/标签推送等服务）**。为避免影响您业务正常使用，请提前 购买/续费。

提供推送接入方法对比

接入方法	集成方式对比	功能对比
推送插件	插件快速集成： 不再需要逐个厂商填写配置，控制台下载引入 json 配置文件，即可完成所有手机厂商的推送信息配置； 支持按需集成一个或者多个对应厂商的推送渠道包，更加满足合规要求； 不需要客户处理推送注册、token 上报、前后台状态上报等，接入后插件自动闭环； 不再需要关注和添加打点和上报逻辑，开启功能后插件自行上报和汇总，支持链路排查和指标统计等；	普通消息推送 全员标签推送 自定义界面跳转 推送自定义样式 设备推送状态查询 全链路排查工具 推送记录和指标统计分析

	插件封装界面跳转、图标自定义等方法，直接使用即可；	
TUIOfflinePush	源码自行集成（不再支持维护，可能会存在推送失败、组件不适配等问题，待下架）	普通消息推送
自行集成	文档自行集成	普通消息推送

交流与反馈

欢迎加入腾讯云即时通信 IM 社群进行技术交流和反馈。