即时通信 IM

推送服务（Push）

腾讯云

文档目录

推送服务（Push）产品介绍

最近更新时间：2025-05-14 11:59:32

产品概述

推送服务（Push）是腾讯云 IM 推出的一站式 App 消息推送解决方案。全面覆盖多通道多平台、最快3分钟即可实现一键式集成，实现毫秒级触达。以丰富多样的推送方式和消息全链路统计分析能力，保障推送精准高效、服务稳定安全，助您轻松提升用户留存和互动活跃度。

核心优势

3分钟一键式快速集成

- **在线推送支持所有机型**（包括三星、中兴、传音、坚果、海信、索尼等），**离线推送厂商支持小米、华为、荣耀、OPPO、vivo、魅族、APNs**，包含各厂商子品牌例如一加、realme、iQOO 等，**境外支持 Google FCM**。
- 支持 **Android、iOS、Flutter、uniapp、React-Native、微信小程序多端框架** 开发平台。
- 不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。
- 不需要集成不涉及的厂商，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，Push 服务自闭环。
- 不需要自行添加打点和处理上报逻辑，Push 服务自行上报和汇总，还支持链路排查和指标统计等。
- 不需要自行适配厂商规则，Push 服务已封装界面跳转、图标自定义等方法，直接使用即可。

毫秒级推送，下发率有保障

全球近3000个加速节点，对于多地域多节点的接入，通过自研多重最优寻址算法，具有全网调度能力，同时接入节点全球覆盖和最优调度策略，解决了境外“第一公里”的传输问题。支持**毫秒级下发**，保障音视频通话、直播开播提醒、IOT 设备通知等业务体验，**服务可靠性高于99.99%**。

长链接保持，有效提升触达率

App 推送，是移动端用户拿起手机后看到的第一项内容，也是提升用户留存的有效工具。推送服务 Push 支持自建在线通道和厂商离线通道下发，**可触达30天内有活跃的用户**，您可以尽情利用 Push 营造品牌形象、创造营销机会、促进用户活跃，**助您轻松实现用户留存率10%-60%的增长**。

精准推送，高效提升点击率

提供全员推送、标签推送、批量指定 UserID 等推送方式，帮助您**向全员、标签用户、指定用户**进行系统通知、内容订阅、社交互动、广告营销等场景的消息推送。保证您在合适的时机、用合适的方式、将特定内容推送给最需要的用户。

音视频产品矩阵深度打通

推送服务（Push）**支持与腾讯云即时通信 IM SDK、实时音视频 TRTC SDK、音视频通话 SDK（TUICallKit）等音视频终端产品协同集成**，并支持不同场景下的消息推送。接下来介绍两种音视频产品联动的推送场景：

1. **IM 类消息推送场景（需同时集成 IM SDK）**

   支持将IM消息以通知栏通知的形式推送到用户移动设备上，通过点击通知即可跳转至指定聊天会话（IM 消息系统）中，并触发对应会话、消息和未读等模块更新。用户在线时候可以收到，用户不在线但下次登录时可自动拉取到该消息的推送通知。

2. **音视频通话呼叫场景（需同时集成 音视频通话 SDK ）**

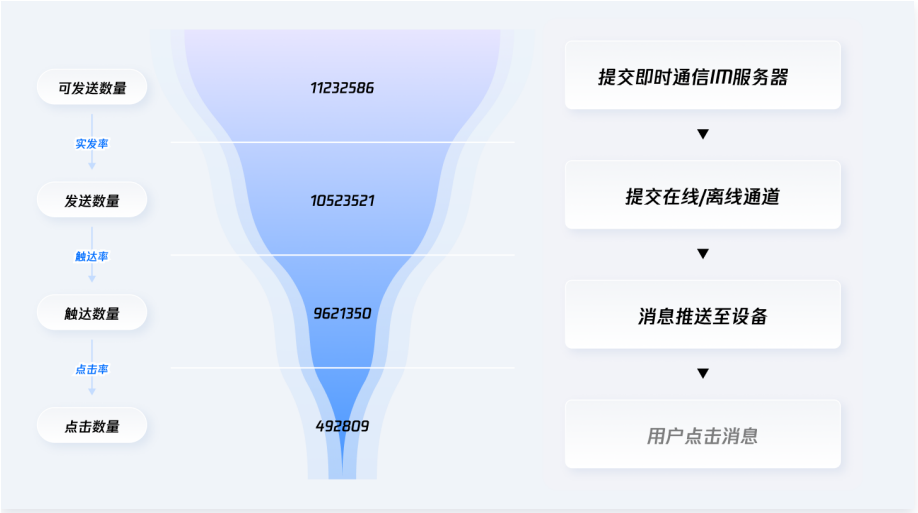
支持通知栏推送和VOIP两种形式实现用户间音视频通话的呼叫（在线或离线均可接收）。

自定义样式，提升推送吸引力

支持**小图标、右侧图标、长文本、大图片、角标和铃音**等自定义样式，同时支持**自定义跳转界面**，为不同场景下的不同推送策略提供丰富的功能体验，帮助推送内容提升吸引力，进一步提升点击率。

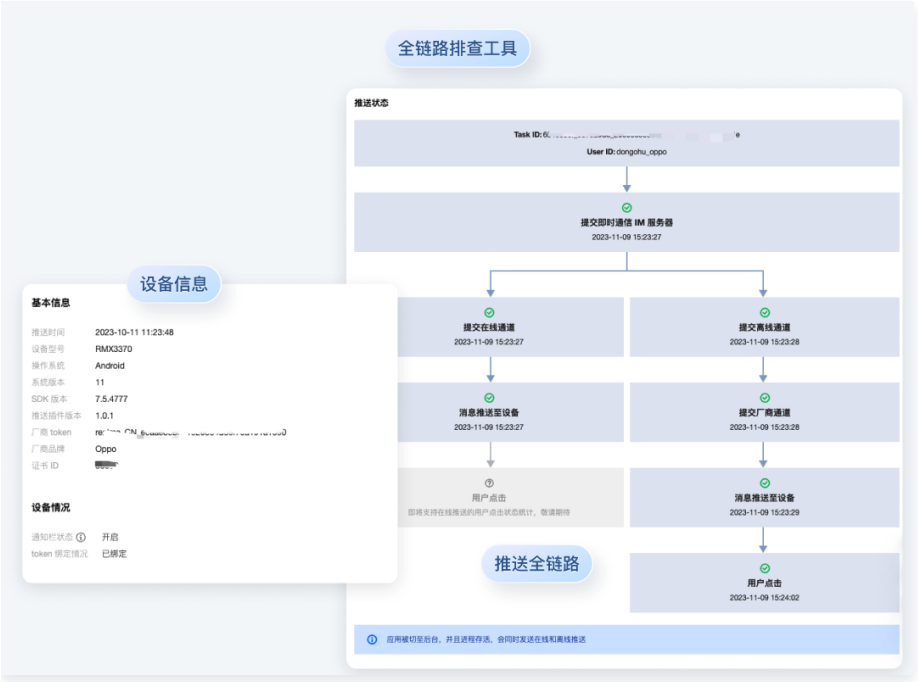
数据可视化，辅助运营策略

提供包含 可发送量 > 发送量 > 触达量 > 点击量 的推送全链路数据，并生成近日**发送率 > 触达率 > 点击率** 的可视化漏斗图表，支持区分厂商通道查看推送效果数据，每日推送转化一目了然。同时提供推送折损原因分析，协助 App 开发者提升推送效果。



支持推送消息全链路问题排查

提供自助式排查工具，支持查看整个推送链路详情、分析推送失败和点击失败原因，提升转化。涵盖了推送下发时的**基本信息**（型号、操作系统、SDK 和版本号等）、**设备情况**（通知栏开关状态、设备的 token 绑定状态等）、**推送状态**（Push 服务器 > 厂商服务器 > 终端设备 > 用户点击的整个链路情况）。



六地服务部署，严守数据安全

提供了**中国、东南亚（新加坡、印尼雅加达）、东北亚（韩国首尔）、欧洲（德国法兰克福）以及北美（美国硅谷）数据存储中心**供选择，每个数据中心均支持全球接入。如果您的应用在境外上线且用户主要在境外，您可以根据消息传输需求及合规要求，选择适合您业务的境外数据中心，保障您的数据安全。

开通使用

请参见 开通服务 文档领取体验版或者购买付费版本。

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。

开通服务

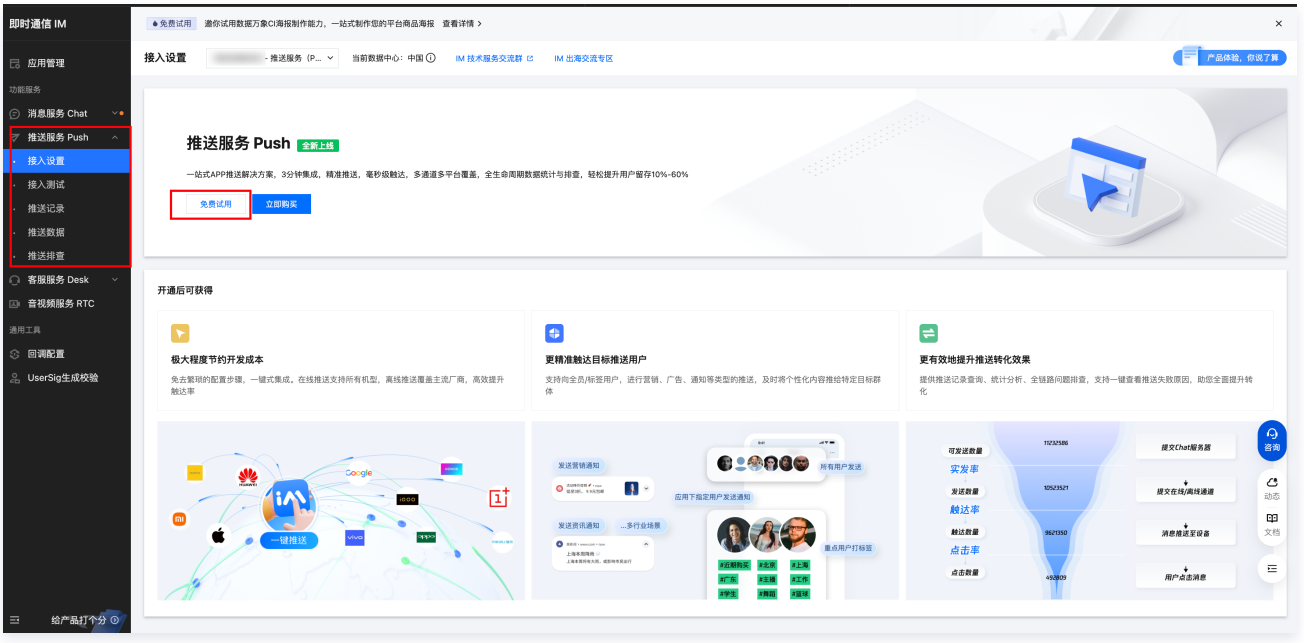
最近更新时间：2025-06-25 16:10:43

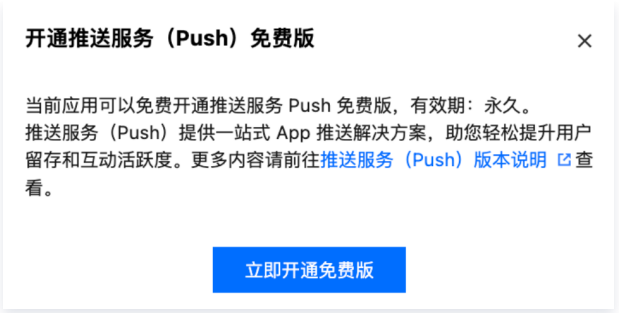
本文介绍如何 开通长期免费版 、开通正式版 和 续费正式版 服务，您可以根据 推送服务（Push）版本计费说明 按需选取，并参考下述指引免费开通和购买开通正式版推送服务能力，实现用户留存和互动活跃度的提升。

开通长期免费版

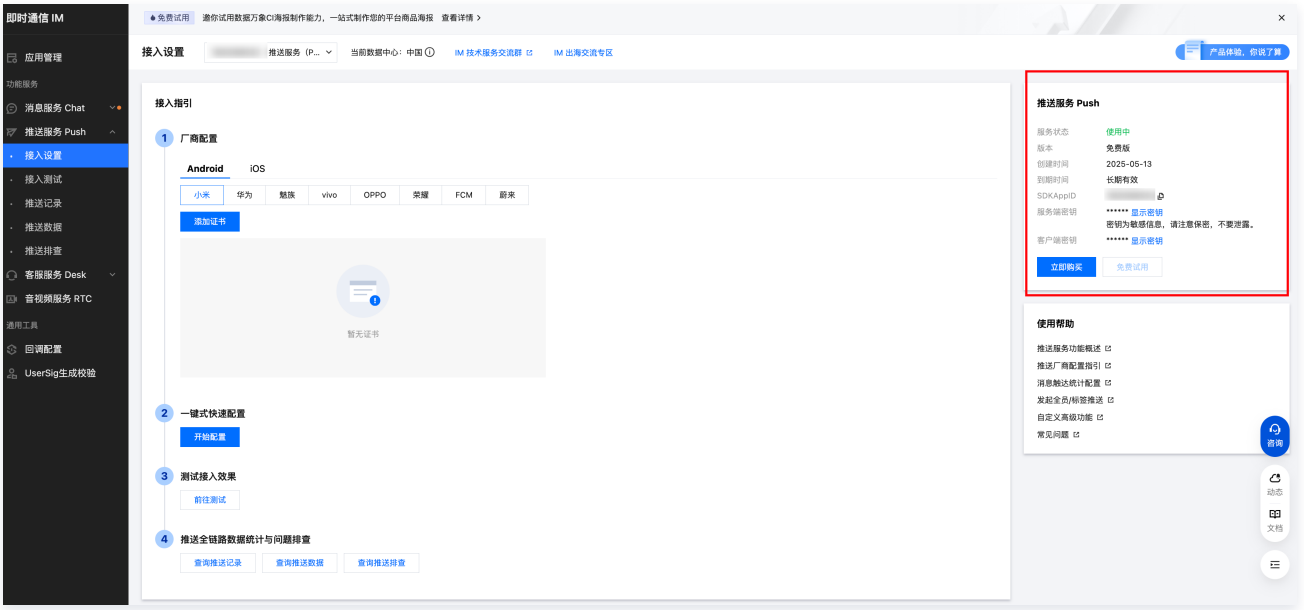
1. 登录 即时通信 IM 控制台，单击**创建新应用**。在弹窗中输入您的**应用名称**，选择**数据中心**，并单击**确定**。

2. 左侧导航栏选择 推送服务 Push 目录下任意页面，可以看到推送服务 Push 产品的介绍页面。单击**免费试用**按钮，在弹窗中单击**立即开通免费版**即可开通。





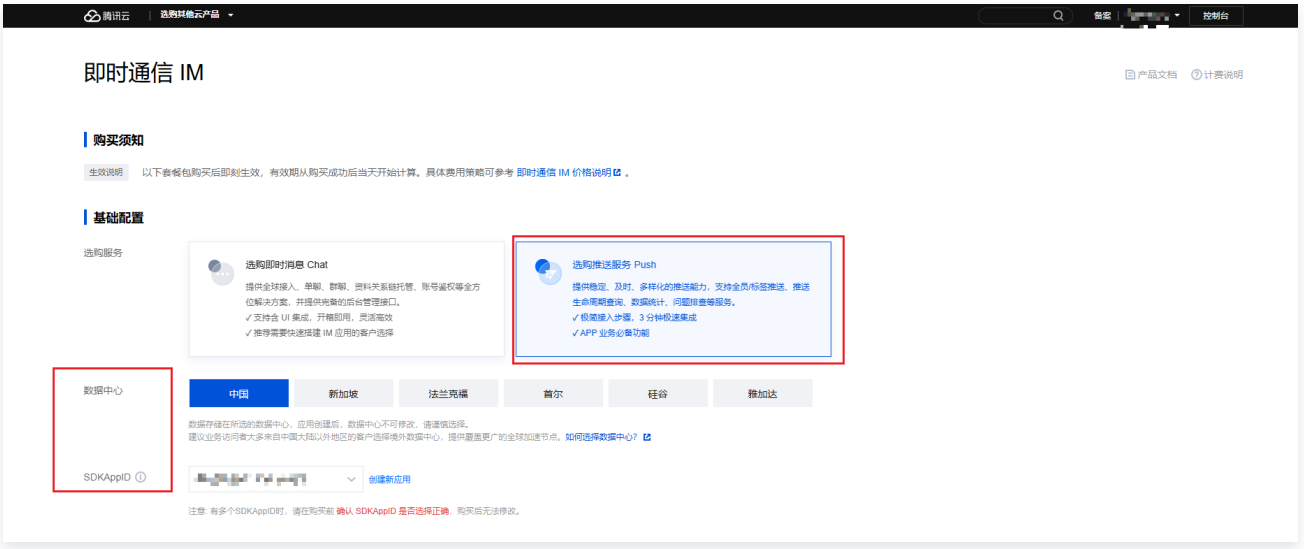
3. 开通完成后，可以看到长期免费版的基本信息，包括服务状态、版本、创建时间、到期时间、SDKAppID、服务端密钥、客户端密钥等信息。



开通正式版

本章节将详细说明如何购买推送服务（Push）预付费套餐包。计费详情请参见 功能与计费说明 。

1. 登录 即时通信 IM 购买页，切换至**选购推送服务 Push** 选项，正确选择您需要购买推送服务（Push）的 SDKAppID。



2. 选择高级版或标准版套餐包，勾选购买时长和协议条款，**推荐您同时勾选自动续费功能**。单击**立即购买** 前往下单。

3. 购买完成后，您可以回到 Push 控制台，确认套餐包购买完成情况。之后您可以根据 厂商通道接入指引 文档进行离线推送配置，或者接入测试 在线推送 功能。

续费正式版

您可对 推送服务（Push）套餐进行续费，具体操作如下：

1. 登录 续费管理控制台，选择您想要续费的推送服务（Push）套餐包，单击**续费**。
2. 按照提示完成续费即可。

功能与计费说明

推送服务（Push）的费用包含两方面：预付费套餐包费用、套餐包外超量费用。如下表所示：

计费项	计费方式	说明

套餐包费用	预付费	推送服务（Push）套餐包分为免费版、标准版、高级版，创建应用后需手动开通并领取免费版。您可以根据实际业务需求选择不同的套餐包。
		• 标准版
		  ○ 国内：1499元/月，**限时特惠 999元/月。**
		  ○ 境外：2999元/月，**限时特惠 1999元/月。**
		• 高级版
		  ○ 国内：2999元/月，**限时特惠 1999元/月。**

		○ 境外：5999元/月，**限时特惠 3999元/月。**
套餐包外超量费用	后付费	超出标准版、高级版免费额度以外的 Push DAU，所需支付的额外费用。 • 国内：1000元/万Push DAU/月 • 境外：3000元/万Push DAU/月

各版本套餐包功能详情及价格如下表所示：

套餐功能/功能点		免费版	标准版	高级版（推荐）
包月套餐包费用		免费	国内：1499元/月 （**限时特惠 999元/月**） 境外：2999元/月 （**限时特惠 1999元/月**）	国内：2999元/月 （**限时特惠 1999元/月**） 境外：5999元/月 （**限时特惠 3999元/月**）
站点	全球覆盖	支持	支持	支持
套餐资源	推送服务-峰值日活跃用户数（峰值 Push DAU）	无限制 （限时免费体验）	赠送付费额度：1万 超量后付费	赠送付费额度：3万 超量后付费
推送能力	推送速度	共享20万条/秒 **（所有免费版用户共享推送资源）**	20万条/秒	30万条/秒
	一键快速集成	✓	✓	✓
	全员/标签推送能力	✓	✓	✓
	全员/标签推送 API 调用次数	10次/天	100次/天	100次/天
	单发推送能力	✓	✓	✓
	单发推送 API 调用频率	20次/s	30次/s	40次/s
	可视化推送工具	✓	✓	✓
	离线推送可触达范围	7天内有活跃的用户	30天内有活跃的用户	30天内有活跃的用户
	创建应用数	无限制	无限制	无限制
数据分析	推送记录查询	-	✓	✓
	推送设备情况查询	-	✓	✓
	消息链路查询	-	✓	✓
	转化漏斗数据查询	-	✓	✓
	消息折损分析	-	✓	✓
	推送回调	-	✓	✓
	统计类 REST API	-	-	✓
个性化功能	自定义角标	✓	✓	✓
	自定义铃声	✓	✓	✓
	自定义单击跳转	✓	✓	✓

	自定义小图标	✓	✓	✓
	多语言智能推送	-	-	✓
音视频联动场景能力	IM 聊天消息通知能力（加购 即时通信 IM ）	-	✓	✓
	音视频通话呼叫能力（加购 TUICallKit ）	-	✓	✓
支持厂商通道		华为、荣耀、小米、OPPO、vivo、魅族、蔚来、FCM、APNs		
支持平台 / SDK		Android、iOS、微信小程序多端框架、Uni-App、React Native、Flutter		

注意：

- 推送服务-免费峰值日活跃用户数（峰值 Push DAU）的计算方式：成功注册推送服务与 Push 后台建立长链接后，Push DAU 将会加1，即单个用户当日登录推送服务（Push）计为1个 Push DAU，同一用户重复登录时不累加。
- **限时免费活动：**同时使用即时通信 IM 聊天服务和 推送服务（Push）所产生的同一用户 ID 对应的 DAU 不会重复计算和收费。

快速跑通

最近更新时间：2025-05-19 15:39:12

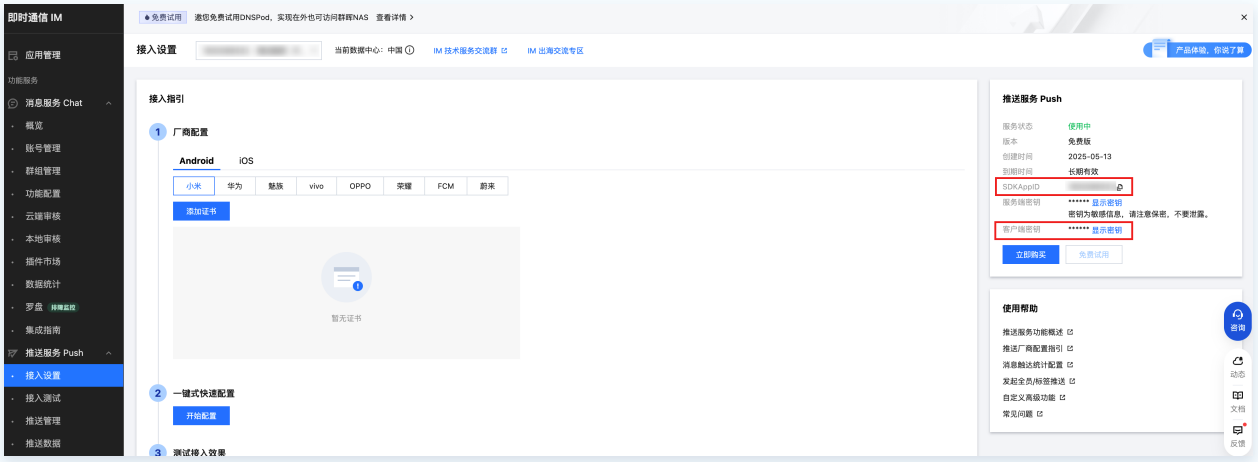
本文将介绍如何在短时间内完成 推送服务（Push）的接入，跟随本文档，您可以快速完成接入工作，并测试 App 推送的实机效果。

步骤1：开通推送服务（Push）

请参见 开通服务 文档，领取长期免费版或者购买正式版。

步骤2：集成 TIMPush 并注册推送

说明：
注册接口必填参数 SDKAppID 和客户端密钥 AppKey 的获取路径是：IM 控制台 > **推送服务 Push > 接入设置**，如下图所示：



Android

1. **集成 TIMPush**

```
// 版本号“VERSION”请前往 更新日志 中获取配置。
implementation 'com.tencent.timpush:timpush:VERSION'
implementation 'com.tencent.liteav.tuikit:tuicore:VERSION'
```

2. **注册推送（注册成功才可以收到在线推送通知）**

```
int sdkAppId = 0; //您的 sdkAppId
String appKey = ""; //客户端密钥
TIMPushManager.getInstance().registerPush(context, sdkAppId, appKey, new TIMPushCallback() {
    @Override
    public void onSuccess(Object data) {

    }

    @Override
    public void onError(int errCode, String errMsg, Object data) {

    }
});
```

说明：

1. 注册离线推送服务成功后，通过该接口 getRegistrationID 可获取推送唯一 ID 标识，即 RegistrationID，然后可以根据 RegistrationID 来向指定设备推送消息；
2. RegistrationID 是设备的推送唯一标识 ID，默认注册推送服务成功后会自动生成，卸载重装会改变。

3. 实现点击通知栏回调

收到推送消息后点击通知栏，组件会回调该点击事件和透传离线消息。

自定义点击跳转实现

注意：
注册回调时机建议放在应用 Application 的 oncreate() 函数中。

```
TIMPushManager.getInstance().addPushListener(new TIMPushListener() {
    @Override
    public void onNotificationClicked(String ext) {
        Log.d(TAG, "onNotificationClicked =" + ext);
        // 获取 ext 自定义跳转

    }
});
```

自定义点击跳转实现（旧方案）

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。

注意：
注册回调时机建议放在应用 Application 的 oncreate() 函数中。

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

//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_EXT_KEY);
            // 获取 ext 自定义跳转

        } else {
            Log.e(TAG, "onReceive ext is null");
        }
    }
```

```
}。
```

iOS

1. 集成 TIMPush

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 frameworks
  use_frameworks!
  use_modular_headers!

  # Pods for Example
  pod 'TXIMSDK_Plus_iOS_XCFramework'
  # 版本号“VERSION”请前往 更新日志 中获取配置。
  pod 'TIMPush', 'VERSION'
end
```

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
```

2. 注册推送（注册成功才可以收到在线推送通知）

```
const int sdkAppId = 0; //您的 sdkAppId
static const NSString *appKey = @""; //客户端密钥

[TIMPushManager registerPush:sdkAppId appKey:appKey succ:^(NSData * _Nonnull deviceToken) {

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

}];
```

> **说明：**
> 1. 注册离线推送服务成功后，通过该接口 getRegistrationID 可获取推送唯一 ID 标识，即 RegistrationID，然后可以根据 RegistrationID 来向指定设备推送消息；
> 2. RegistrationID 是设备的推送唯一标识 ID，默认注册推送服务成功后会自动生成，卸载重装会改变。

3. 实现点击通知栏回调

收到推送消息后点击通知栏，组件会回调该点击事件和透传离线消息。

自定义点击跳转实现

> **注意：**

> 注册回调时机建议放在应用 AppDelegate 的 didFinishLaunchingWithOptions 函数中。

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:
(NSDictionary *)launchOptions {
    [TIMPushManager addPushListener:self];
    return YES;
}
#pragma mark - TIMPushListener
- (void)onNotificationClicked:(NSString *)ext {
        // 获取 ext 自定义跳转
}
```

自定义点击跳转实现（旧方案）

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 -
navigateToBuiltInChatViewController:groupID: 方法。
    return NO;
}
```

Flutter

1. 集成 TIMPush

本插件在 pub.dev 的包名为: `tencent_cloud_chat_push` ，您可以将其引入 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
```

2. 注册推送（注册成功才可以收到在线推送通知）

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.
```

```
  }
}

TencentCloudChatPush().registerPush(onNotificationClicked: _onNotificationClicked, sdkAppId:
您的sdkAppId, appKey: "客户端密钥");
```

说明：

1. 注册离线推送服务成功后，通过该接口 getRegistrationID 可获取推送唯一 ID 标识，即 RegistrationID，然后可以根据 RegistrationID 来向指定设备推送消息；
2. RegistrationID 是设备的推送唯一标识 ID，默认注册推送服务成功后会自动生成，卸载重装会改变。

3. 实现点击通知栏回调

Android

Application 类继承 TencentCloudChatPushApplication

```
package 替换成您自己的包名（一般 Android Studio 会自动生成）

import
com.tencent.chat.flutter.push.tencent_cloud_chat_push.application.TencentCloudChatPushAppli
cation;

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。

iOS

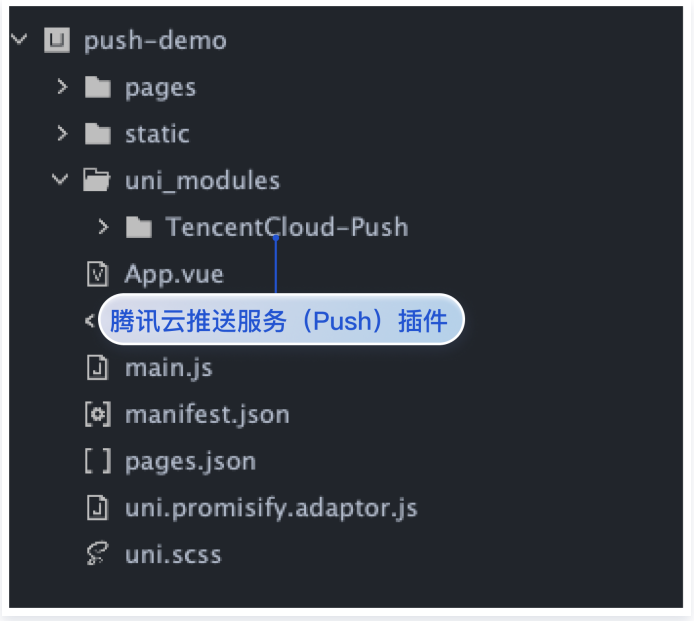
`AppDelegate` 类继承 `TIMPushDelegate`

```
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
    }
}
```

uni-app

1. **HBuilderX 4.29 有 bug，请使用 HBuilderX 4.36 或更高版本，并升级 uni-app 腾讯云推送服务（Push）到 1.1.0 或更高版本。**
2. **将 uni-app 腾讯云推送服务（Push）插件导入HbuilderX 中的工程。如图所示：**



3. **在 App.vue 中引入并注册腾讯云推送服务（Push）（注册成功才可以收到在线推送通知）**

> **说明：**
> `registerPush` 注册推送服务成功后，您可通过 `getRegistrationID` 获取推送 ID 标识，即 RegistrationID。您可以向指定的 RegistrationID 推送消息。

```
// 集成 TencentCloud-Push
import * as Push from '@/uni_modules/TencentCloud-Push';
const SDKAppID = 0; // 您的 SDKAppID
const appKey = ''; // 客户端密钥
Push.registerPush(SDKAppID, appKey, (data) => {
        console.log('registerPush ok', data);
        Push.getRegistrationID((registrationID) => {
            console.log('getRegistrationID ok', registrationID);
        });
    }, (errCode, errMsg) => {
        console.error('registerPush failed', errCode, errMsg);
    }
```

```
);

// 监听通知栏点击事件，获取推送扩展信息
Push.addPushListener(Push.EVENT.NOTIFICATION_CLICKED, (res) => {
    // res 为推送扩展信息
    console.log('notification clicked', res);
});

// 监听在线推送
Push.addPushListener(Push.EVENT.MESSAGE_RECEIVED, (res) => {
    // res 为消息内容
    console.log('message received', res);
});

// 监听在线推送被撤回
Push.addPushListener(Push.EVENT.MESSAGE_REVOKED, (res) => {
    // res 为被撤回的消息 ID
    console.log('message revoked', res);
});
```

4. **使用云端证书，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**，使用云端证书制作 Android 或 iOS 自定义调试基座。如图所示：

React Native

1. **创建一个 React Native 项目（已有项目可忽略此步骤）**

```
npx @react-native-community/cli@latest init MyReactNativeApp --version 0.75.0
```

2. **进入 MyReactNativeApp 目录，集成 @tencentcloud/react-native-push**

```
npm install @tencentcloud/react-native-push --save
```

3. **注册推送（注册成功才可以收到在线推送通知）**

   复制下面的代码到 `App.tsx`，并将 `SDKAppID` 和 `appKey` 替换为您的应用的信息。


```
import Push from '@tencentcloud/react-native-push';

const SDKAppID = 0; // 您的 SDKAppID
const appKey = ''; // 客户端密钥

if (Push) {
  // 如果您需要与 Chat 的登录 userID 打通（即向此 userID 推送消息），请使用 setRegistrationID 接口
  // Push.setRegistrationID(userID, () => {
      // console.log('setRegistrationID ok', userID);
  // });

  Push.registerPush(SDKAppID, appKey, (data) => {
      console.log('registerPush ok', data);
      Push.getRegistrationID((registrationID) => {
        console.log('getRegistrationID ok', registrationID);
      });
    }, (errCode, errMsg) => {
      console.error('registerPush failed', errCode, errMsg);
    }
  );


  // 监听通知栏点击事件，获取推送扩展信息
  Push.addPushListener(Push.EVENT.NOTIFICATION_CLICKED, (res) => {
    // res 为推送扩展信息
    console.log('notification clicked', res);
  });

  // 监听在线推送
  Push.addPushListener(Push.EVENT.MESSAGE_RECEIVED, (res) => {
    // res 为消息内容
    console.log('message received', res);
  });

  // 监听在线推送被撤回
  Push.addPushListener(Push.EVENT.MESSAGE_REVOKED, (res) => {
    // res 为被撤回的消息 ID
```

```
    console.log('message revoked', res);
  });
}
```

4. **配置 Native Modules 和相关依赖**

Android

1）使用 Android Studio 打开 `MyReactNativeApp/android` 目录。
2）修改项目入口文件。

项目入口文件是 MainApplication.kt

```
...
import com.tencent.qcloud.rntimpush.TencentCloudPushApplication

// Replace Application with TencentCloudPushApplication
class MainApplication : TencentCloudPushApplication(), ReactApplication {
  ...
  // add TencentCloudPushPackage to the list of packages returned in ReactNativeHost's
getPackages() method
  override fun getPackages(): List<ReactPackage> =
    PackageList(this).packages.apply {
        // Packages that cannot be autolinked yet can be added manually here, for
example:
        // add(MyReactNativePackage())
    }
}
```

项目入口文件是 MainApplication.java

```
...
import com.tencent.qcloud.rntimpush.TencentCloudPushApplication;

// Replace Application with TencentCloudPushApplication
public class MainApplication extends TencentCloudPushApplication implements
ReactApplication {
  ...
  // add TencentCloudPushPackage to the list of packages returned in ReactNativeHost's
getPackages() method
  @Override
  protected List<ReactPackage> getPackages() {
    List<ReactPackage> packages = new PackageList(this).getPackages();
    // Packages that cannot be autolinked yet can be added manually here, for example:
    // packages.add(new MyReactNativePackage());
    return packages;
  }
  ...
```

```
}
```

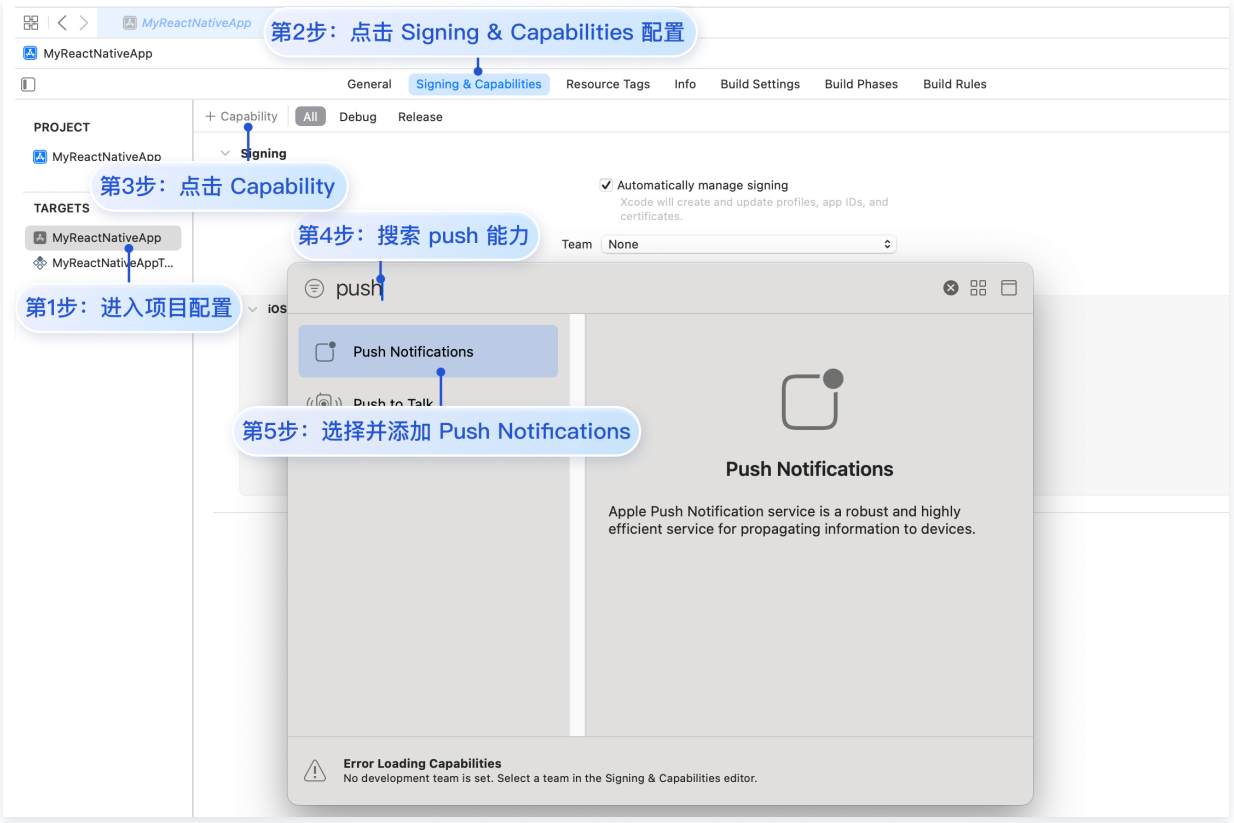
3）以上操作都完成后，选择 File > Sync Project with Gradle Files。

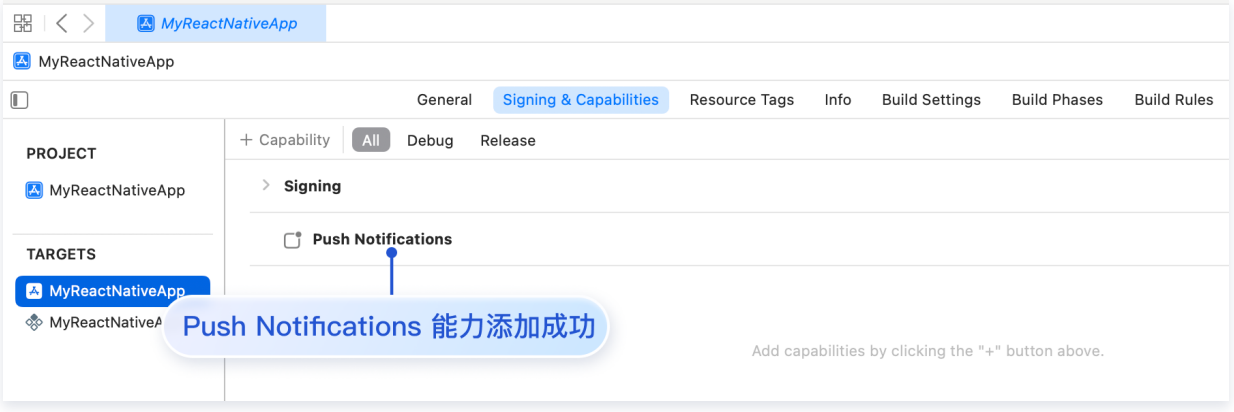
iOS

1）使用 XCode 打开 MyReactNativeApp/ios/MyReactNativeApp.xcworkspace。
2）进入 `MyReactNativeApp/ios` 目录，安装 TIMPush。

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

3）在 App 中启用推送通知功能。打开 Xcode 项目，在 **Project > Target > Capabilities** 页面选择并添加 **Push Notifications**。





5. **在真机上运行（测试前请务必打开手机通知权限，允许应用通知。）**

从项目根目录开始，在命令提示符中运行以下命令，在设备上安装并启动您的应用程序：

Android

```
npm run android
```

iOS

```
npm run ios
```

微信小程序多端框架

1. **使用最新的 Nightly 版微信开发者工具 版本 ≥ 1.06.2410152 >>下载地址**
2. **新建多端应用项目，开启消息推送功能**

   在可视化配置界面分别勾选 Android > 官方插件 > 消息推送和 iOS > 官方插件 > 消息推送。

   填写 推送插件版本号 。
3. **集成 JS API**

   在 miniprogram 目录下执行如下命令安装 @tencentcloud/donut-push：

```
npm i @tencentcloud/donut-push
```

   然后在微信开发者工具中构建 npm（微信开发者工具 > 工具 > 构建 npm）
4. **注册并使用推送功能（注册成功才可以收到在线推送通知）**

```
import Push from "@tencentcloud/donut-push"
const sdkAppID = 0; // 您的 SDKAppID
const appKey = ''; // 客户端密钥
const registrationID = ""; //用户的 registrationID
const listener = (param) => {
  console.log('onEvent', JSON.stringify(param));
}
App({
  onLaunch: function () {
    // 请确保在调用 registerPush 方法之前设置好registrationID。
    // 如果您卸载并重新安装应用，registrationID 会发生改变。
    Push.setRegistrationID(registrationID)
      .then((res) => {
          console.info("setRegistrationID", JSON.stringify(res));
          return Push.registerPush(sdkAppID, appKey);
      }).then((res) => {
          console.info("registerPush", JSON.stringify(res));
          return Push.getRegistrationID();
      }).then((res) => {
          console.info("getRegistrationID", JSON.stringify(res));
```

```
        })
        .catch((res) => {
            console.error("registerPush failed", JSON.stringify(res));
    });
    // 监听在线推送
    Push.addPushListener(Push.EventName.ON_NOTIFICATION_CLICKED, listener);
  }
})
```

步骤3：指定 RegistrationID 推送

如果您需要推送给指定某一台设备，可以在 控制台 指定 registerationID 进行在线推送测试：

说明：

1. 正式使用请参见如下方法：
   - 向全员或者被打标签的用户发送，详情请参见 全员/标签推送 。
   - 批量向指定 RegistrationID 发送，详情请参见 单发推送 。
2. 离线通道详细参见 厂商通道 配置相关。

厂商通道
厂商配置
Android

最近更新时间：2025-04-07 18:02:42

注册应用到厂商推送平台

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

说明：

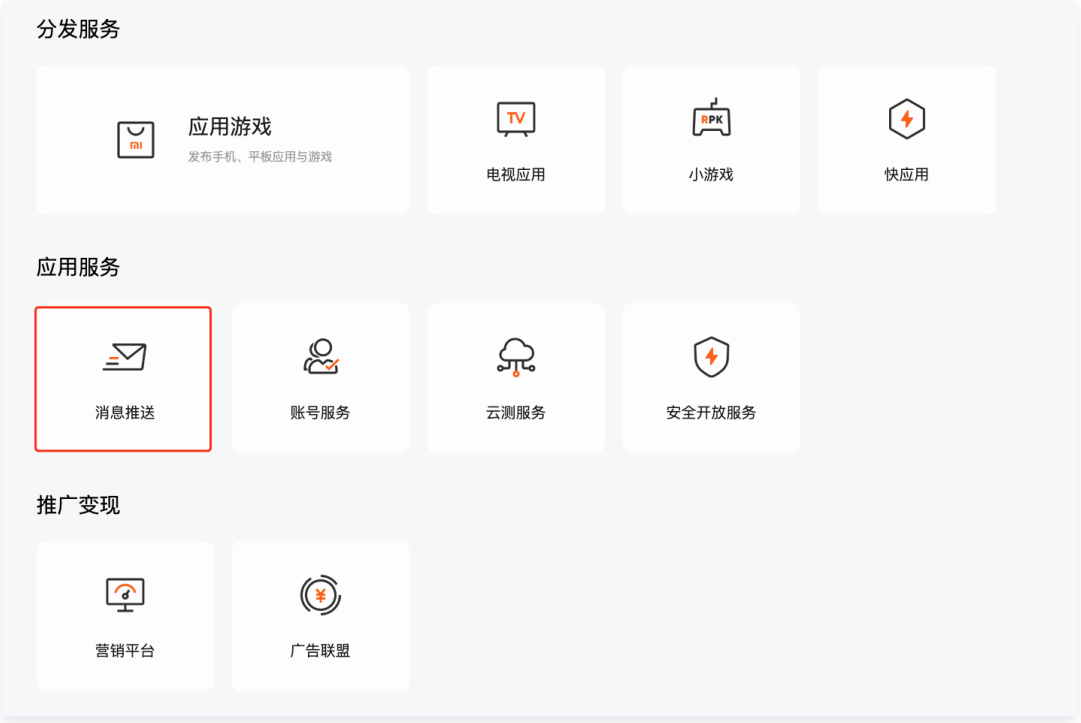
- 通知栏推送：应用需在小米软件商店上架。
- 需要使用企业账号进行推送配置。
- 小米开发平台的应用包名与插件应用包名需保持一致。

步骤1：注册小米开发者账号

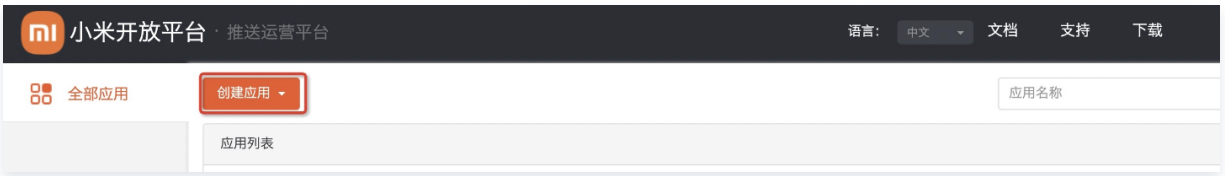
进入 小米开放平台，注册小米开发者账号，详情请参见 企业开发者账号注册流程 。

步骤2：创建应用

1. 在 小米管理控制台 单击**消息推送**。

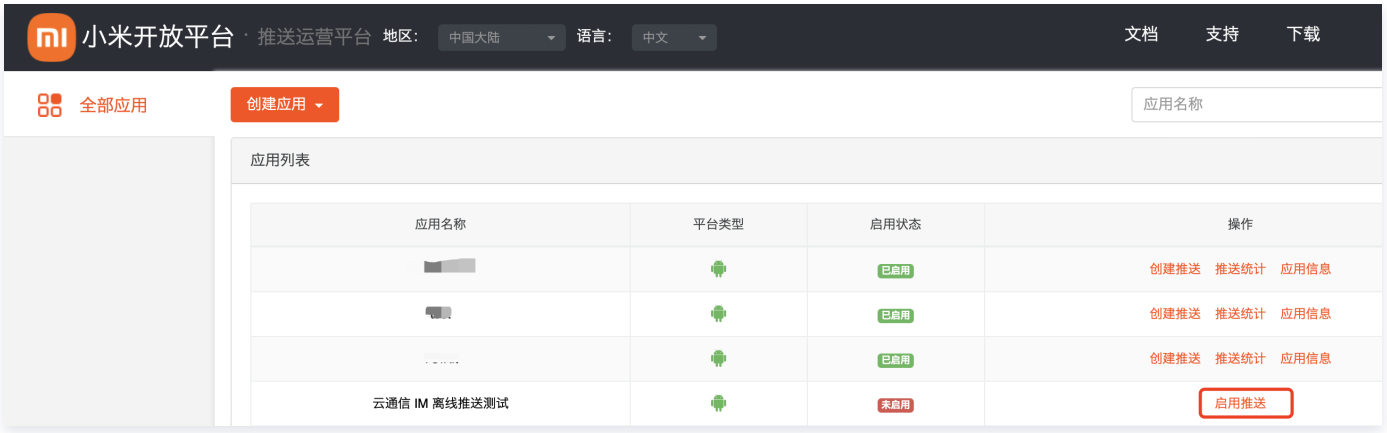


2. 单击**创建应用**，完善应用资料界面后单击**保存**。



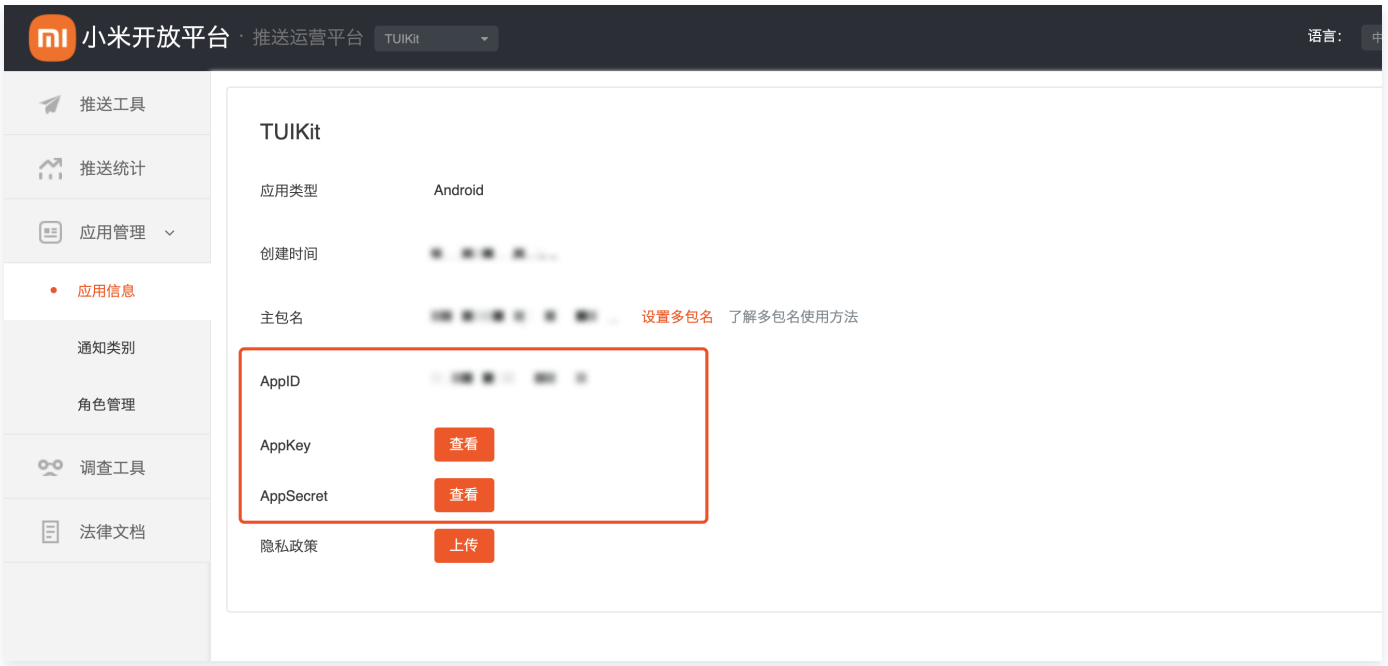
步骤3：启用推送

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。



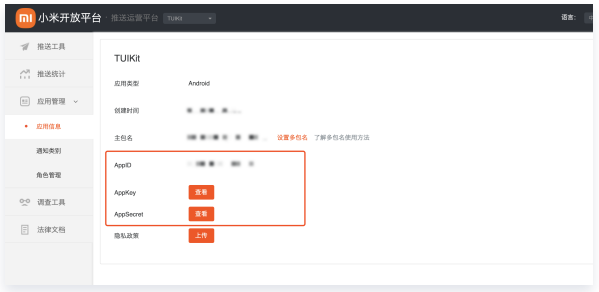
步骤4：查看获取应用信息

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
	**说明：** 应用内指定界面，请使用默认配置。 

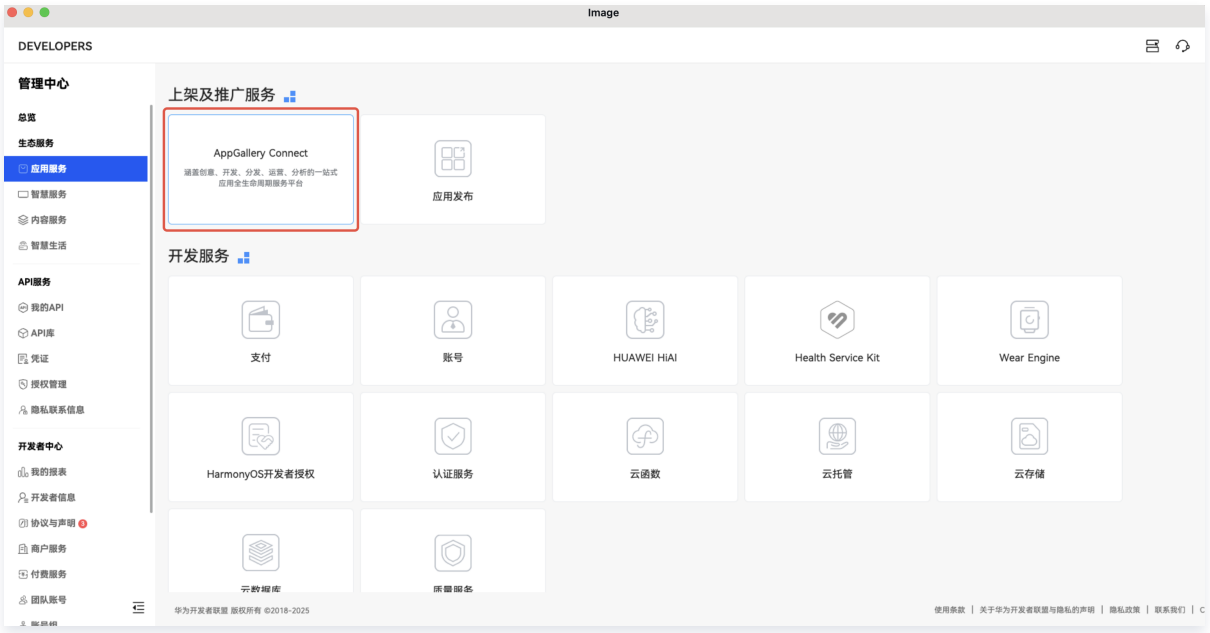
华为

步骤1：注册华为开发者账号

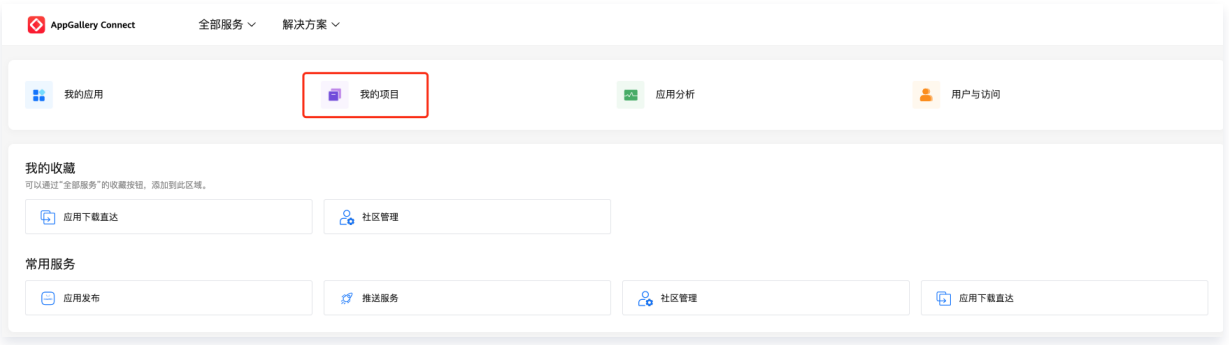
进入 华为开发者联盟，注册华为开发者账号，详情请参见 注册账号 。

步骤2：创建应用

1. 在 华为管理中心 的**应用管理中**，单击 **AppGallery Connect**，进入应用管理中心。



2. 单击**我的项目**，添加一个新的项目。



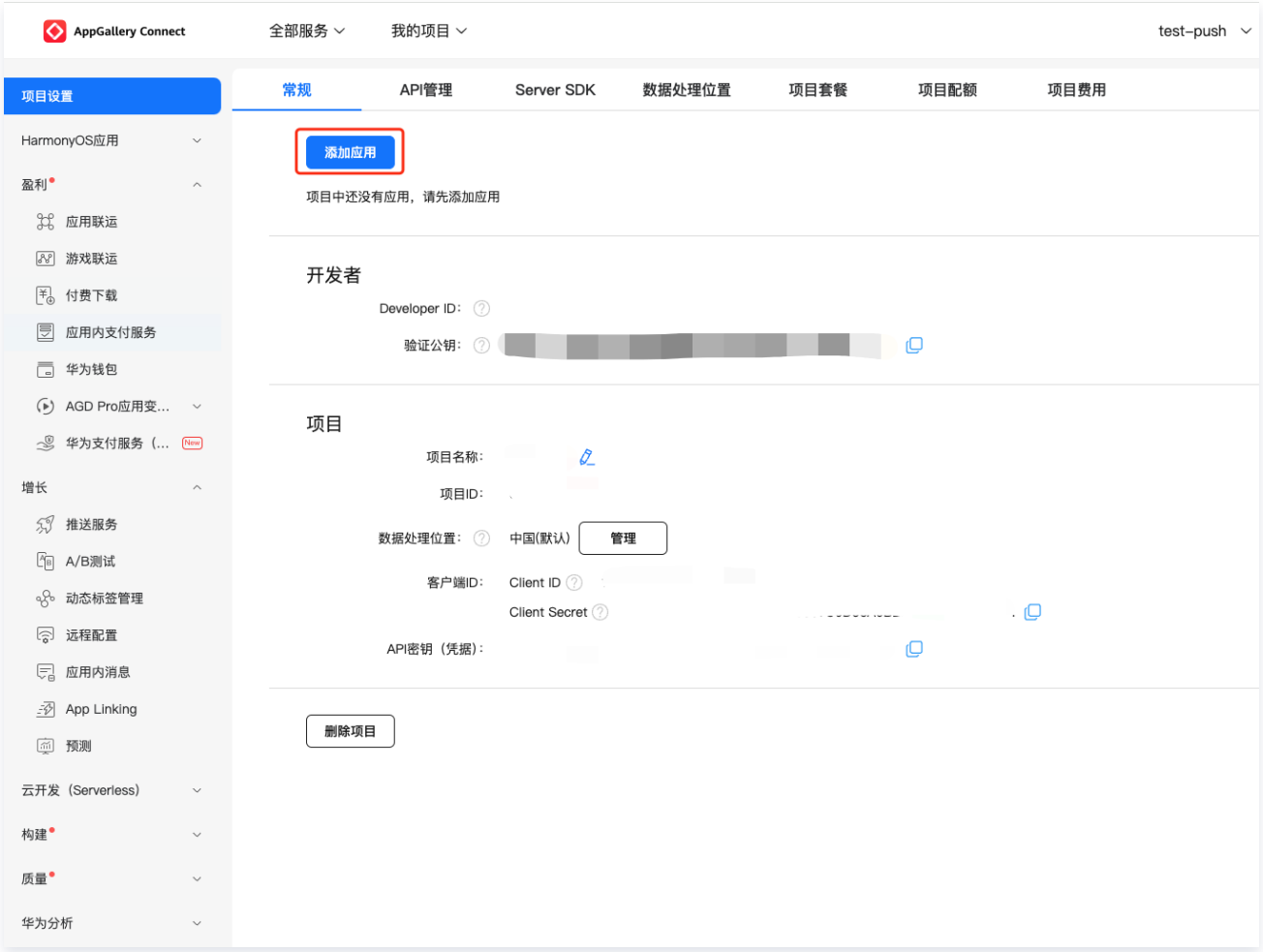
3. 在**项目设置**栏单击**推送服务 > 立即开通**。

4. 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。

> **说明：**
> 应用包名与插件应用包名保持一致。

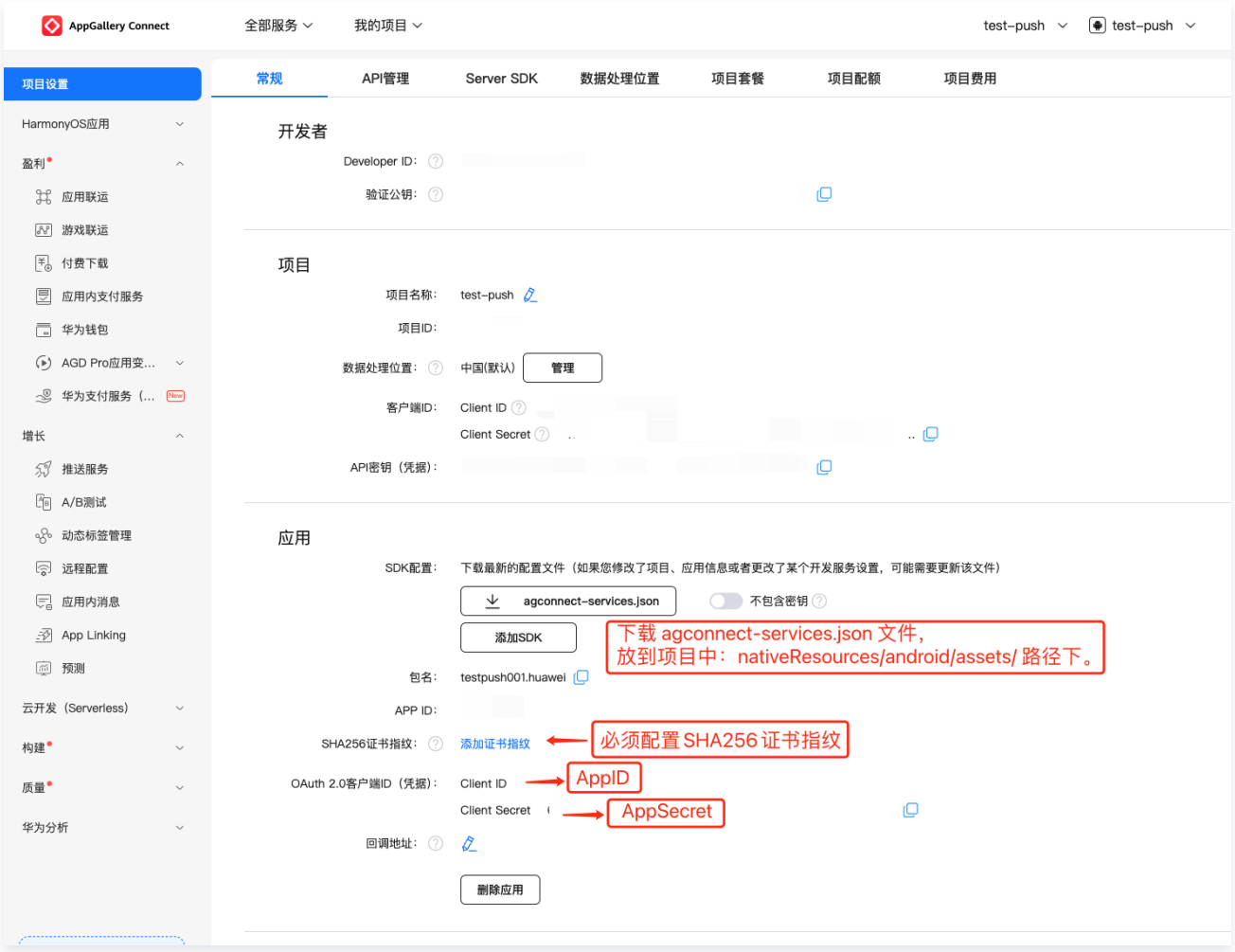


步骤4：获取应用信息

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

> **说明：**
> - Client ID 对应 AppID，Client Secret 对应 AppSecret。
> - 应用内指定界面，请使用默认配置。

回执配置请参考：消息触达统计配置 > 华为

OPPO

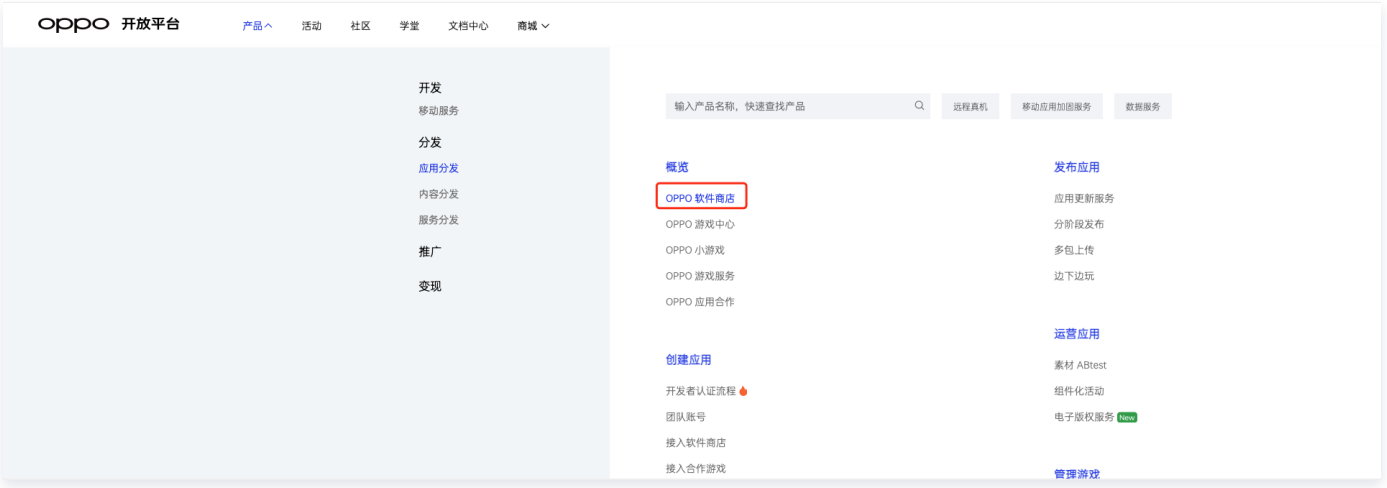
> **说明：**
> - 通知栏推送：应用需在 OPPO 软件商店上架；
> - 通知栏推送测试权限：每天仅可推送1000条消息，限测试使用。应用上架后需重新申请“通知栏推送”权限，以获得正常消息推送数量；
> - 平台将会在1个工作日内返回审核结果，开发者可以在申请页面查看审核结果，其他问题可咨询开放平台客服。

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。

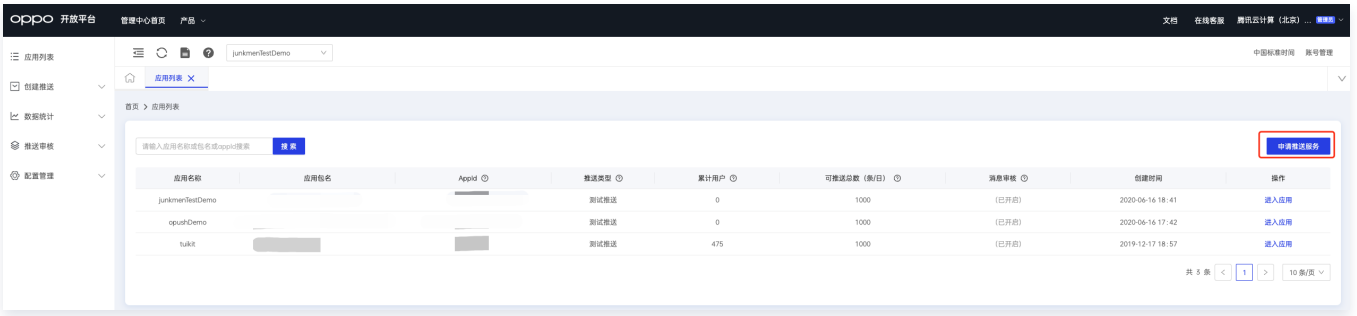


步骤3：开通 PUSH 服务

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
	**说明：** • 应用内指定界面，请使用默认配置。 

vivo

> **说明：**
> - 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。
> - vivo开发平台的应用包名与插件应用包名需保持一致。

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
	**说明：** • 应用内指定界面，请使用默认配置。 

回执配置请参见：消息触达统计配置 > vivo

魅族

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。

> **说明：**
> 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

说明：

- 应用内指定界面，请使用默认配置。

回执配置请参见：消息触达统计配置 > 魅族

荣耀

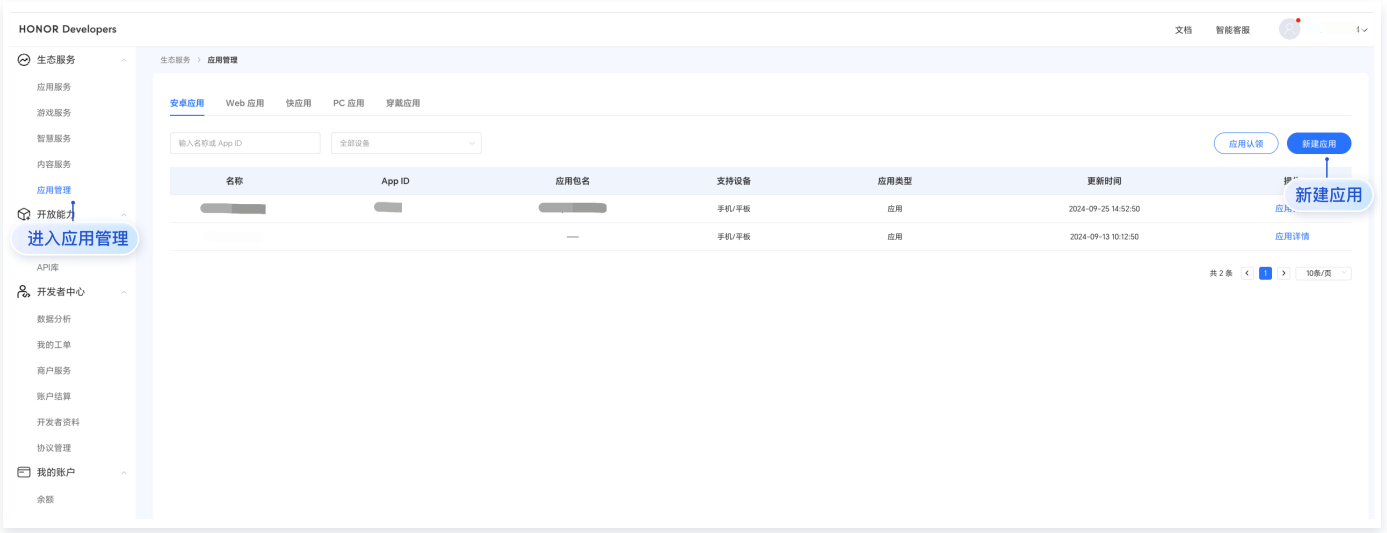
步骤1. 注册荣耀开发者账号

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: 创建应用

1. 进入**应用管理**，点击**新建应用**创建新应用。



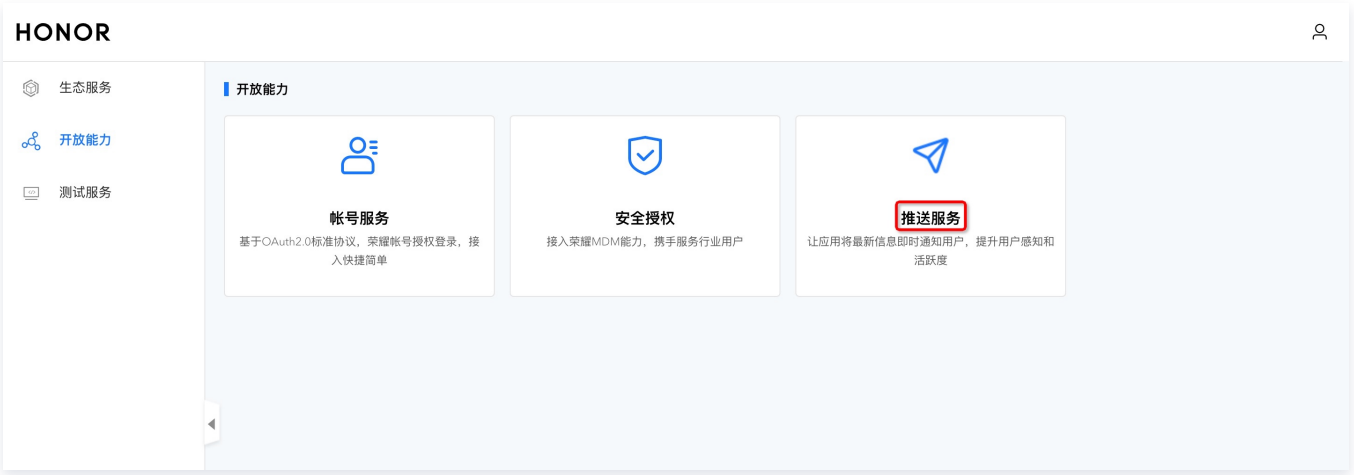
2. 进入**应用详情**，绑定**应用包名**，**下载 mcs-services.json 文件**。

说明：

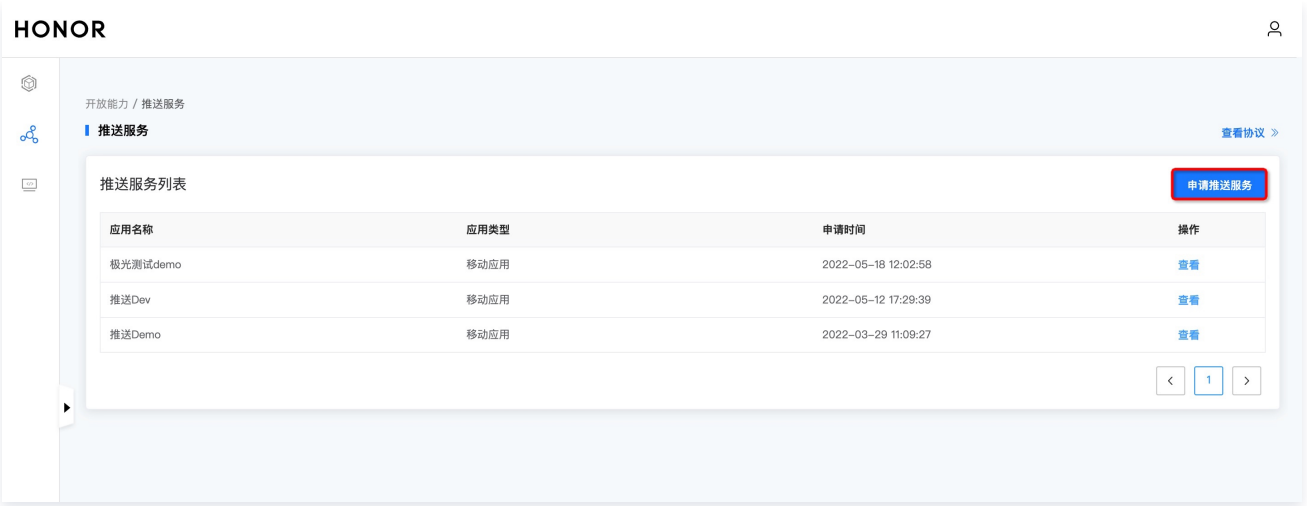
- 必须添加打包的 SHA256证书指纹，SHA256 证书指纹需与自己的打包证书一致。
- 下载 mcs-services.json 文件，放到项目中：nativeResources/android/ 路径下。
- 修改了项目、应用信息、开发服务设置，都需要重新下载配置 mcs-services.json 文件。

步骤4: 开通推送服务

1. 单击**开发能力 -> 推送服务**进入推送服务列表页面。

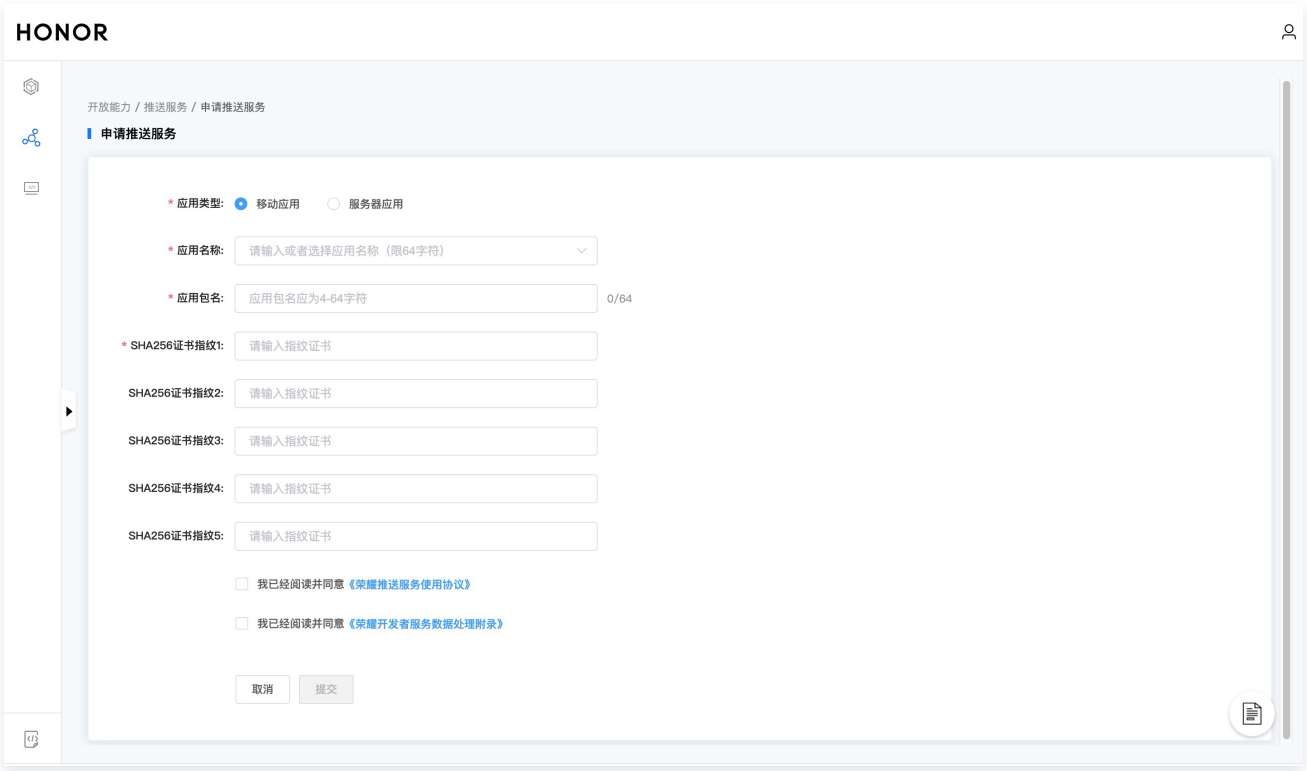


2. 单击**申请推送服务**进入应用申请页面。



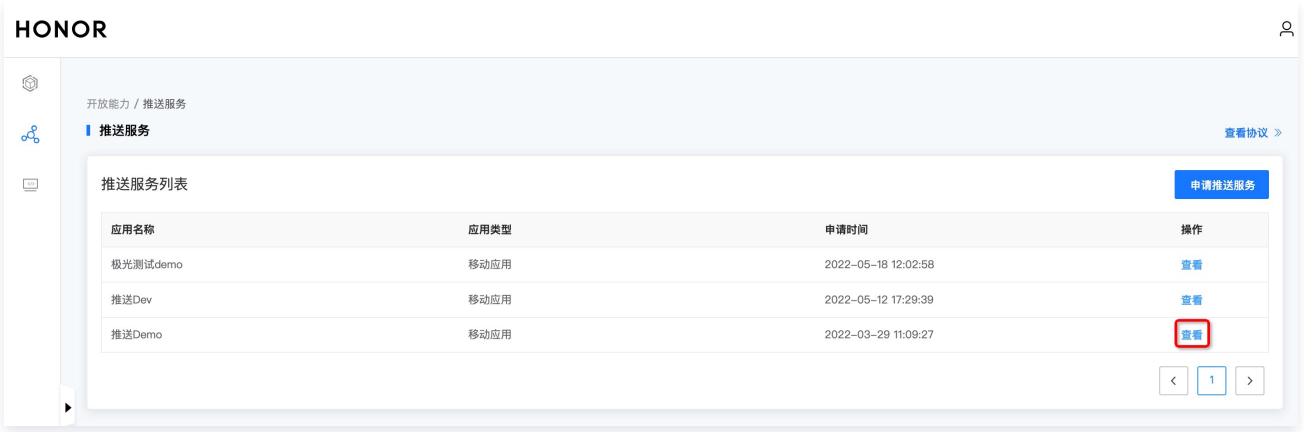
3. 选择应用类型“移动应用”，填写应用包名和证书指纹、同意推送服务协议和数据处理附录，单击**提交**。

> **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。



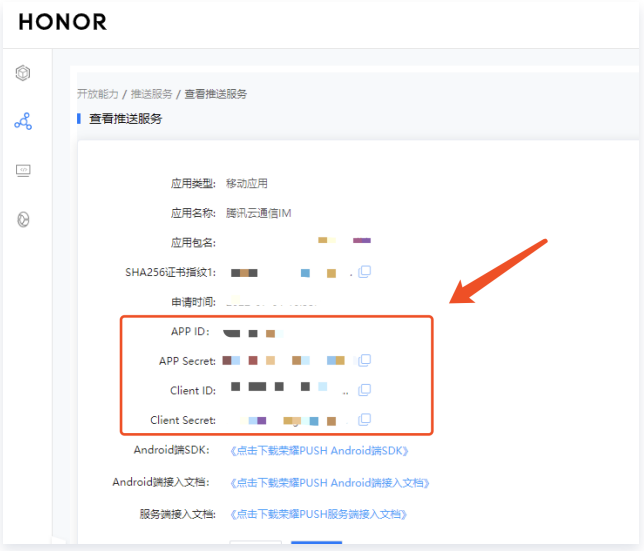
步骤5: 获取应用信息

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，不可以修改。该配置用于派发单击后离线推送插件的事件监听，不可以直接配置应用内页面的跳转。 

回执配置请参考：消息触达统计配置 > 荣耀

Google FCM

步骤1：进入Firebase 控制台

进入 Firebase 控制台，登录谷歌账号。

步骤2：创建应用

1. 单击**创建项目**，添加一个新的项目。

2. 进入Android 应用。

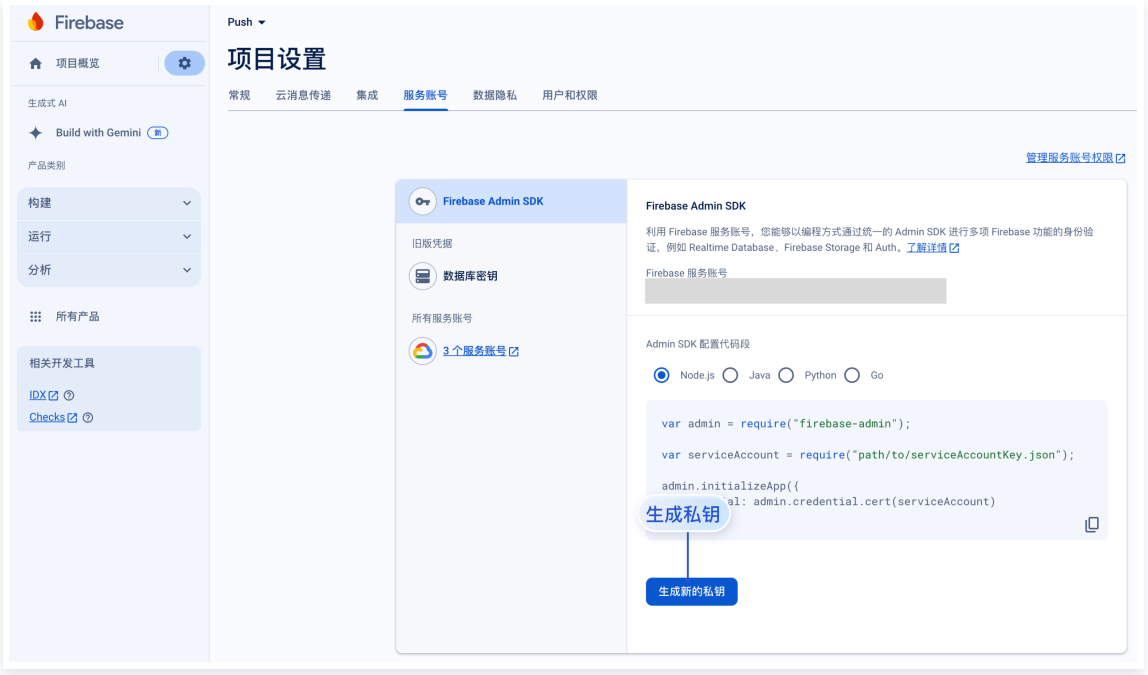
3. 输入应用信息，注册应用。

4. 下载并添加配置文件并完成注册。

步骤3. 生成证书私钥。

1. 点击应用，进入应用管理。

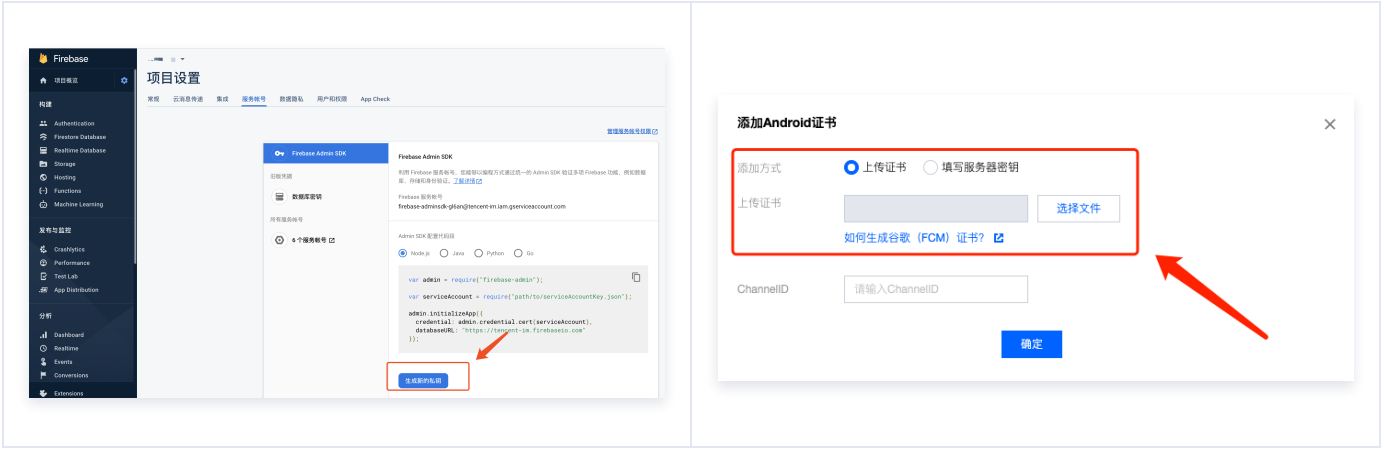
2. 在**项目设置**单击**服务账号 > 生成新的密钥**。



步骤4. 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：

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：2025-03-27 16:06:22

本文将在您拥有**开发者账号**和**开发/分发证书**的前提下，引导您创建和使用**推送证书**及**描述文件**。
如果您对“开发者账号”或“开发/分发证书”有任何疑问，可以参考附录中的相关概念。

一、推送证书配置（APNs）

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。

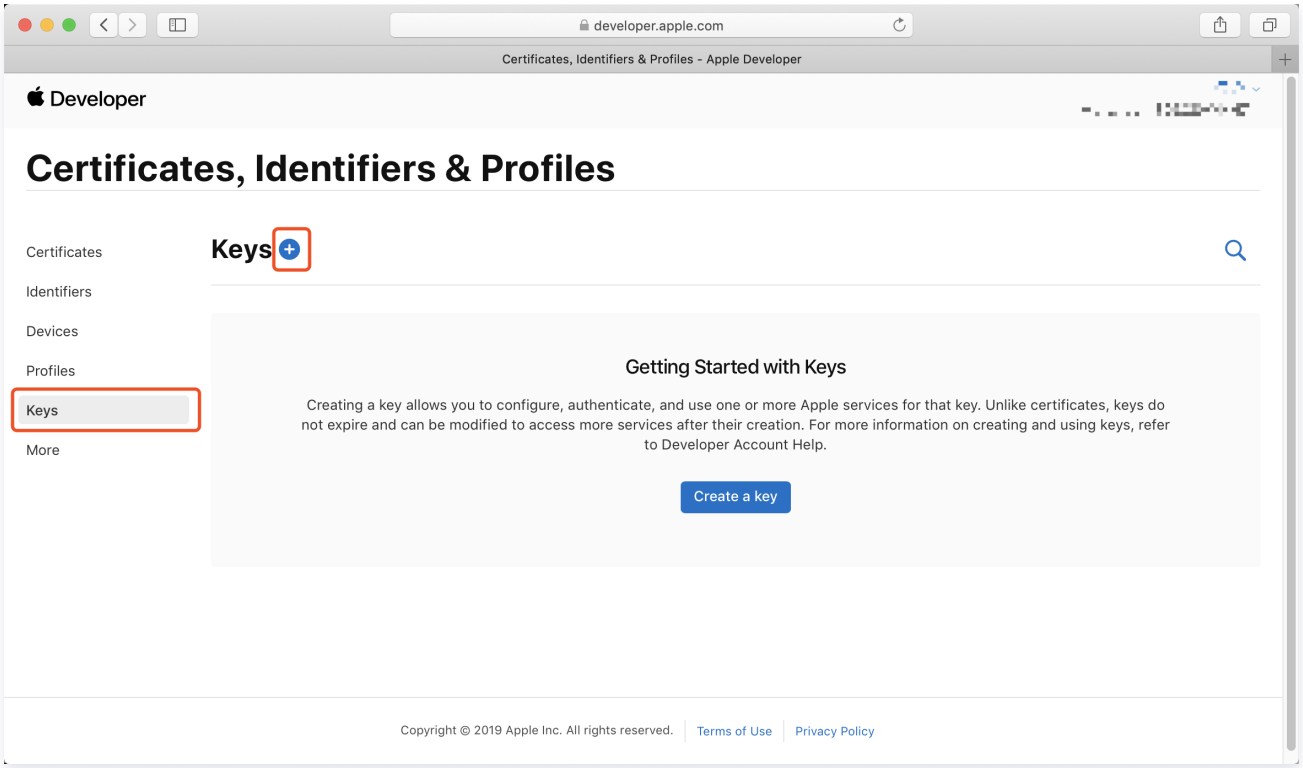
	证书类型	有效期和管理	安全性	灵动岛
p8 证书（推荐）	p8 证书是一个 Auth Key（授权密钥），用于基于令牌的身份验证。它是一个包含私钥的文本文件，扩展名为 .p8。	p8 证书没有到期日期，因此您无需担心证书过期。此外，使用 P8 证书可以简化证书管理，因为您可以使用一个 p8 证书为多个应用程序提供推送通知服务。	p8 证书使用基于令牌的身份验证，这意味着您的服务器会周期性地生成一个 JSON Web Token（JWT）来与 APNs 建立连接。这种方法更安全，因为它不需要在服务器上存储私钥。	支持灵动岛推送
p12 证书（传统）	p12 证书是一个包含公钥和私钥的二进制文件，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（推荐）

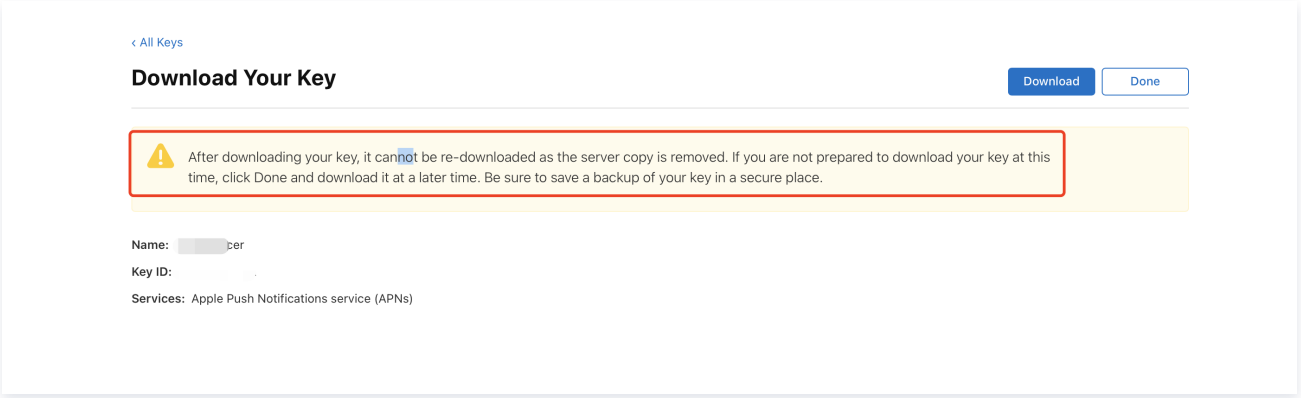
p8 证书：p8 证书没有到期日期，因此您无需担心证书过期。此外，使用 p8 证书可以简化证书管理，因为您可以使用一个p8 证书为多个应用程序提供推送通知服务。另外，p8 证书支持灵动岛推送。

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** 。



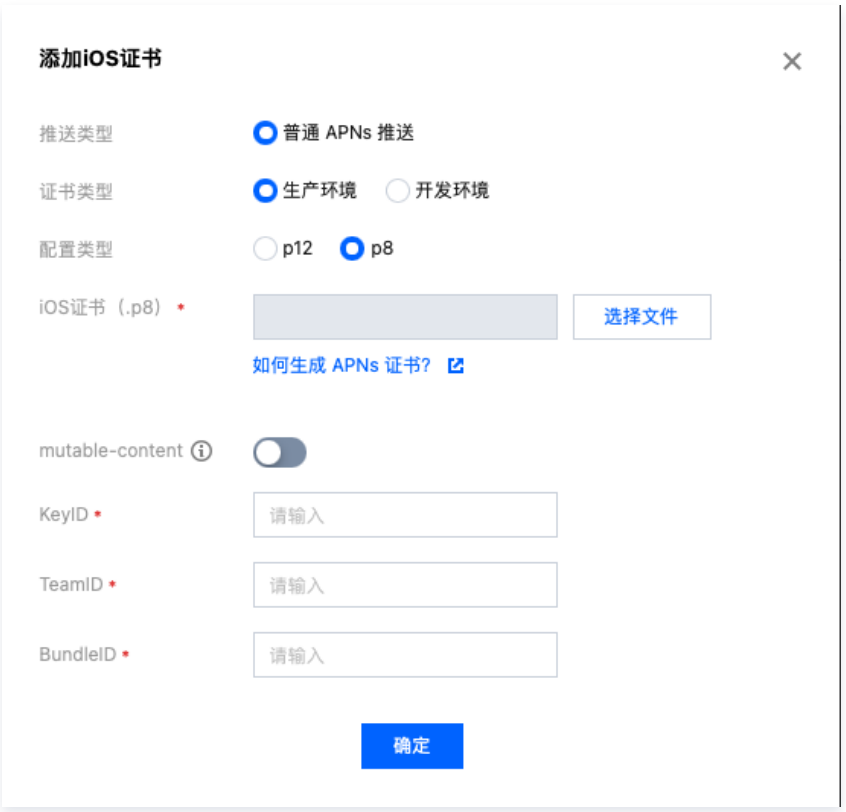
确认并生成密钥：在确认页面核对您的密钥信息，然后单击 **Register** 。接下来，您将看到一个页面提示您下载密钥。单击 **Download**，将生成的 .p8 文件保存到您的计算机上。

> **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

5. 选择生产环境、开发环境各上传一份（您需要准备的材料一致，仅由后端为您分配 APNs 的推送环境）



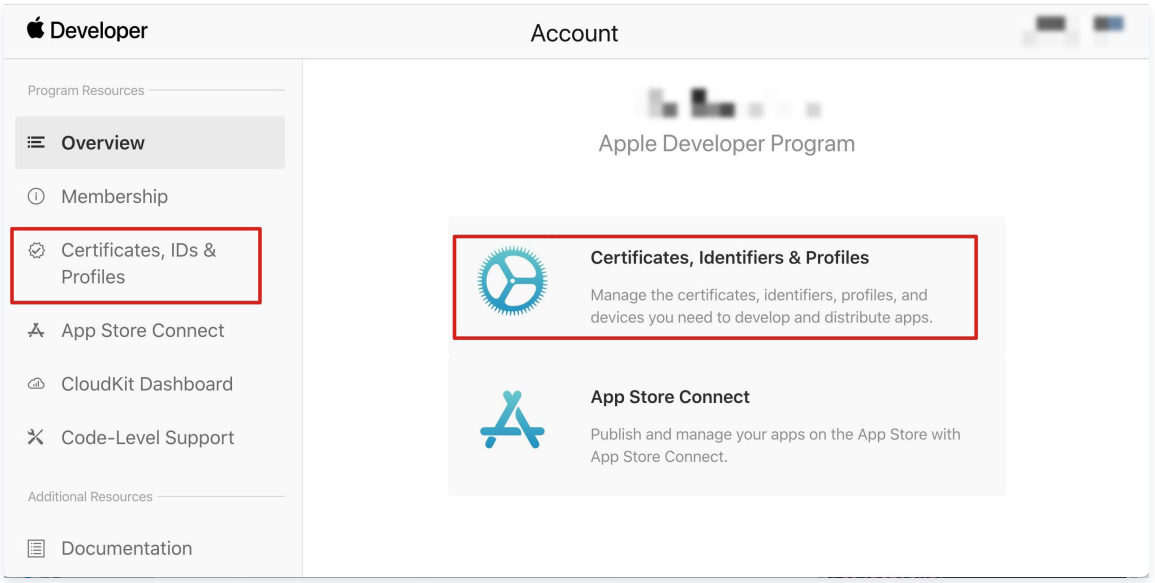
说明：

- **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。

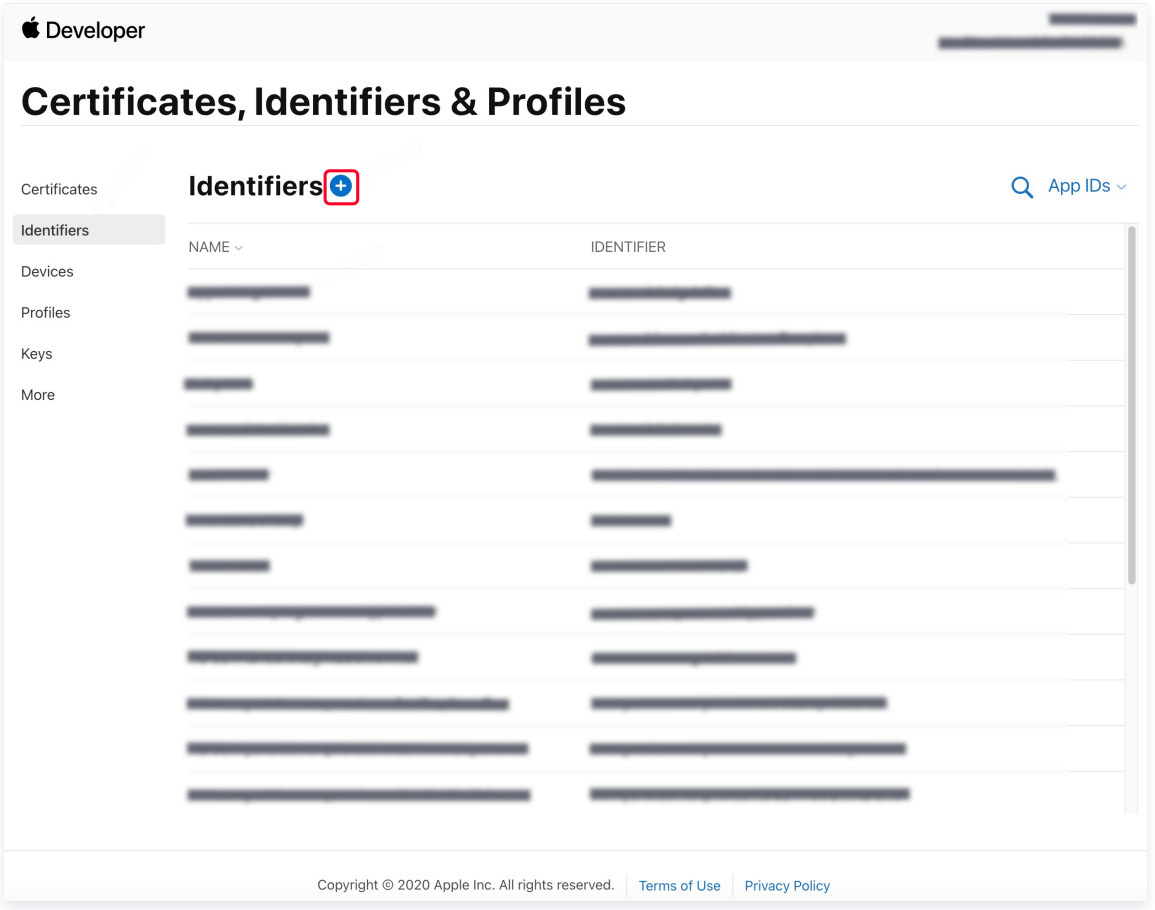
p12 证书（传统）

步骤1：申请 APNs 证书

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。



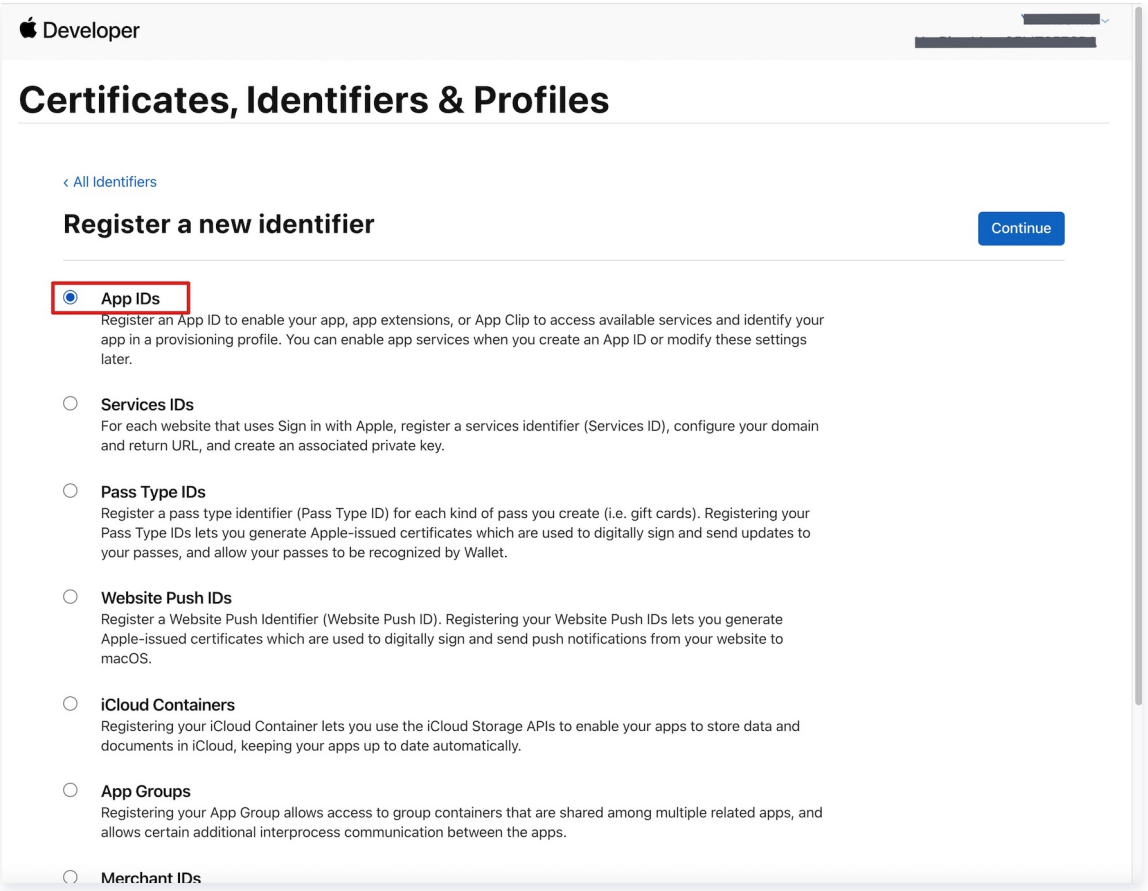
2. 单击 Identifiers 右侧的 +。



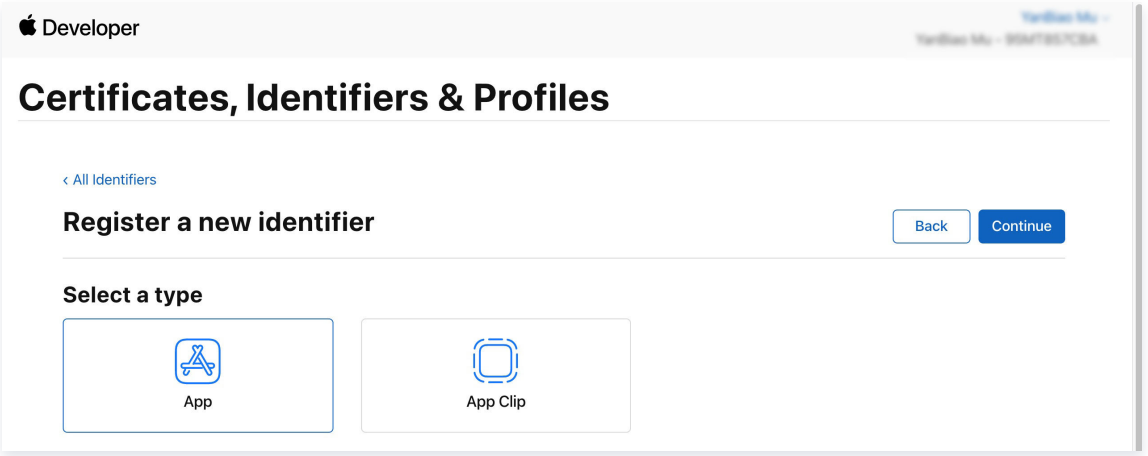
3. 您可以参见如下步骤新建一个 AppID，或者在您原有的 AppID 上增加 `Push Notification` 的 `Service`。

> **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。

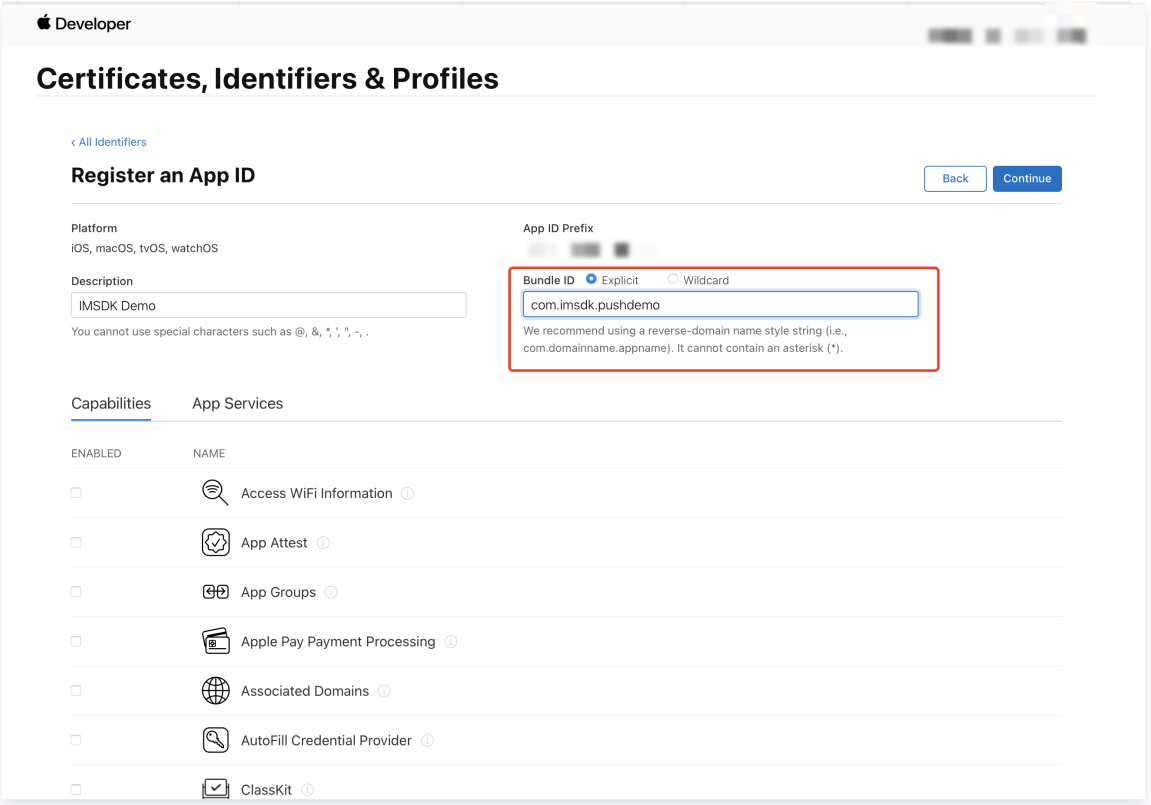
4. 勾选 App IDs，单击 Continue 进行下一步。



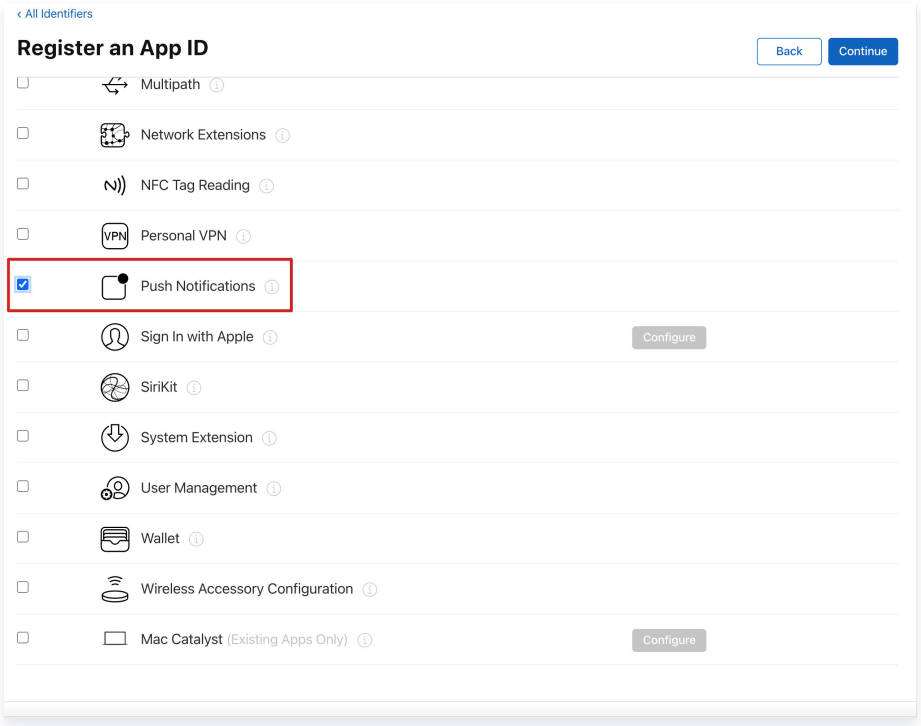
5. 选择 **App**，单击 **Continue** 进行下一步。



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

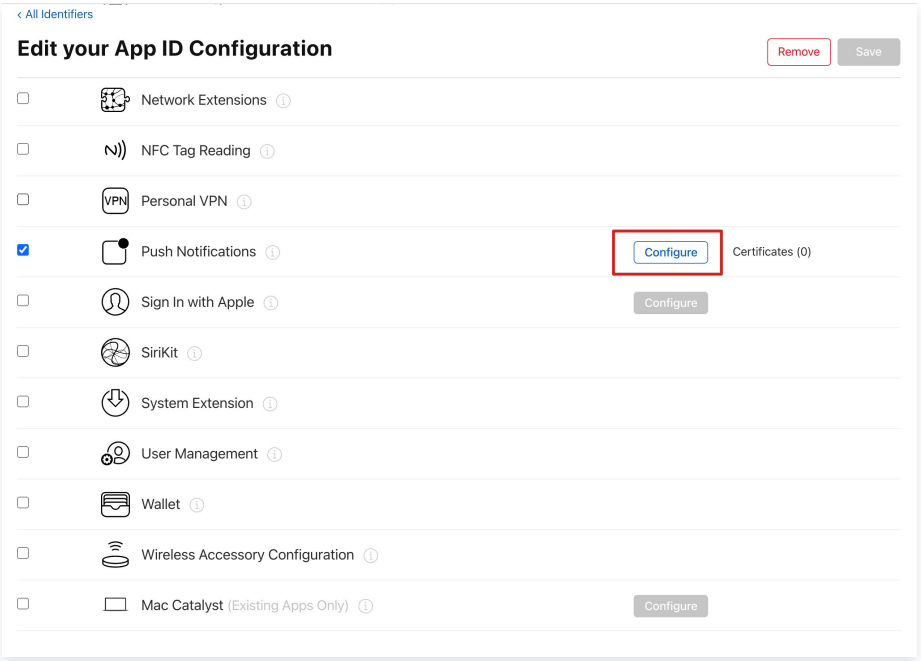


7. 勾选 **Push Notifications**，开启远程推送服务。

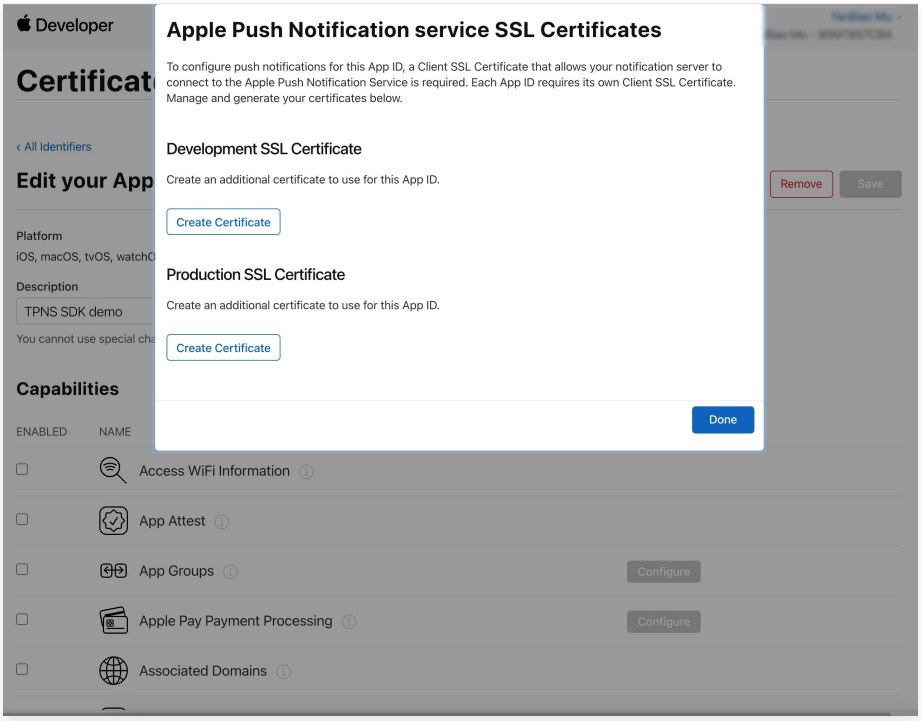


生成证书

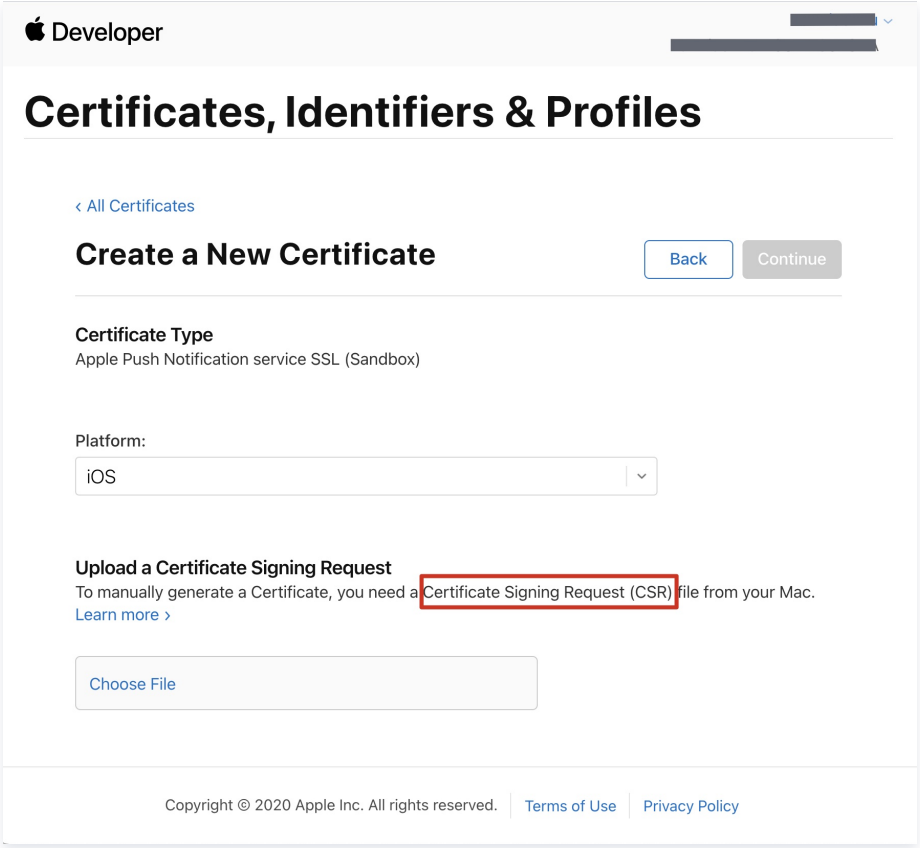
1. 选中您的 AppID，选择 **Configure**。



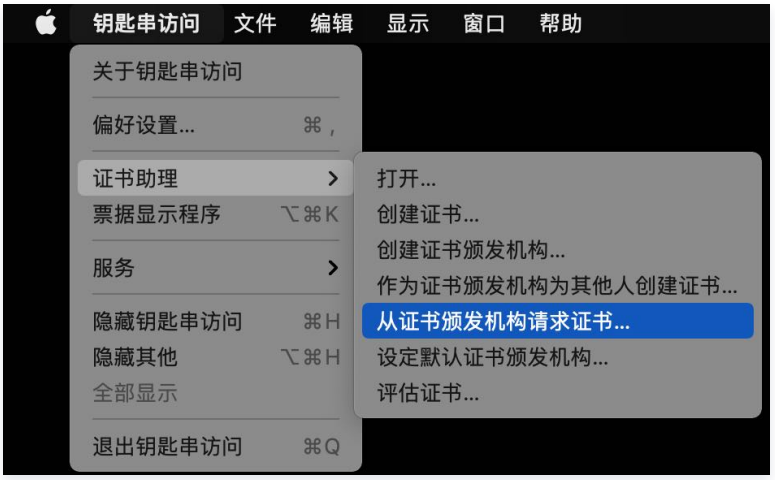
2. 可以看到在 **Apple Push Notification service SSL Certificates** 窗口中有两个 `SSL Certificate`，分别用于开发环境（Development）和生产环境（Production）的远程推送证书，如下图所示：



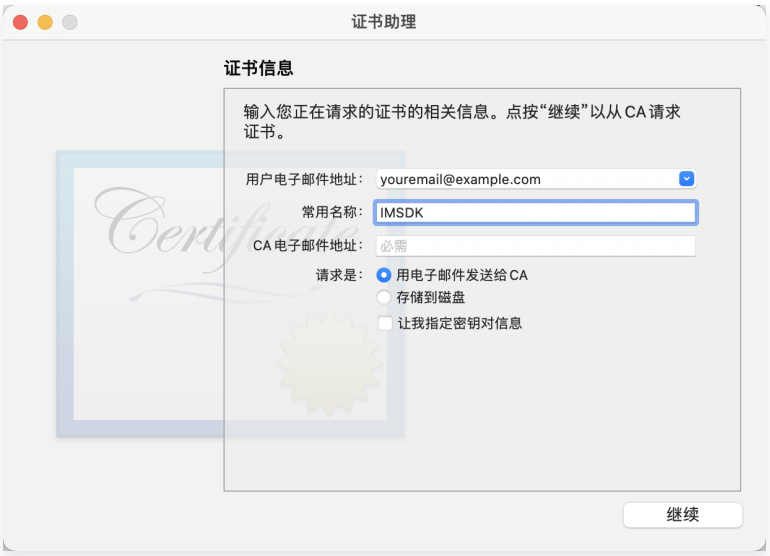
3. 我们先选择开发环境（Development）的 **Create Certificate**，系统将提示我们需要一个 Certificate Signing Request（CSR）。



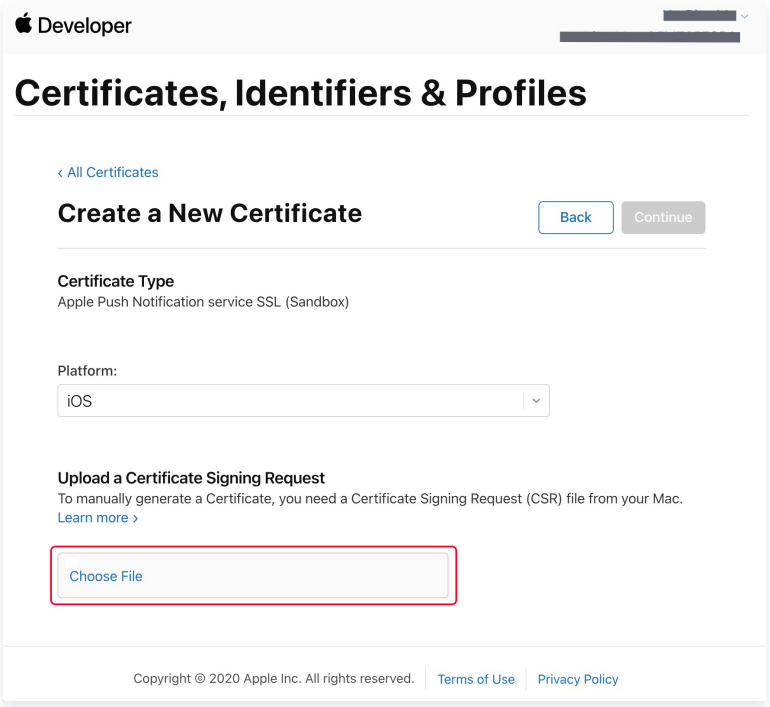
4. 在 Mac 上打开**钥匙串访问工具（Keychain Access）**，在菜单中选择**钥匙串访问 > 证书助理 > 从证书颁发机构请求证书**（`Keychain Access - Certificate Assistant - Request a Certificate From a Certificate Authority`）。



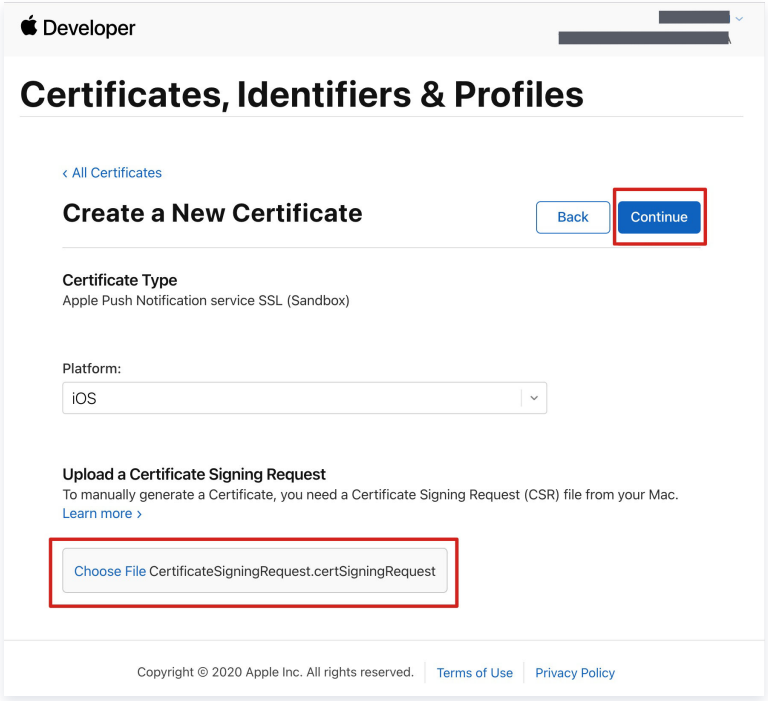
5. 输入用户电子邮件地址（您的邮箱）、常用名称（您的名称或公司名），选择**存储到磁盘**，单击**继续**，系统将生成一个 `*.certSigningRequest` 文件。



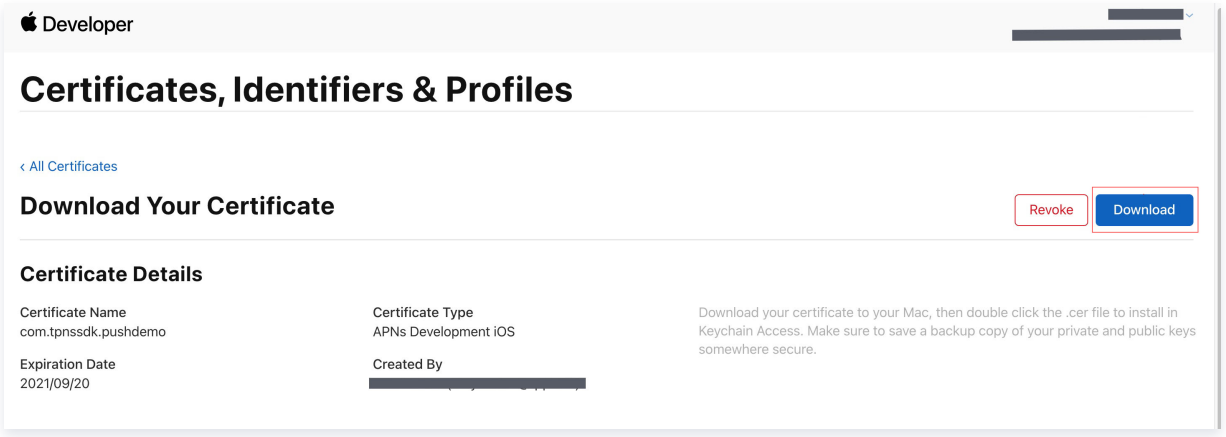
6. 返回上述 步骤3 中 Apple Developer 网站刚才的页面，单击 **Choose File** 上传生成的 `*.certSigningRequest` 文件。



7. 单击 **Continue**，即可生成推送证书。

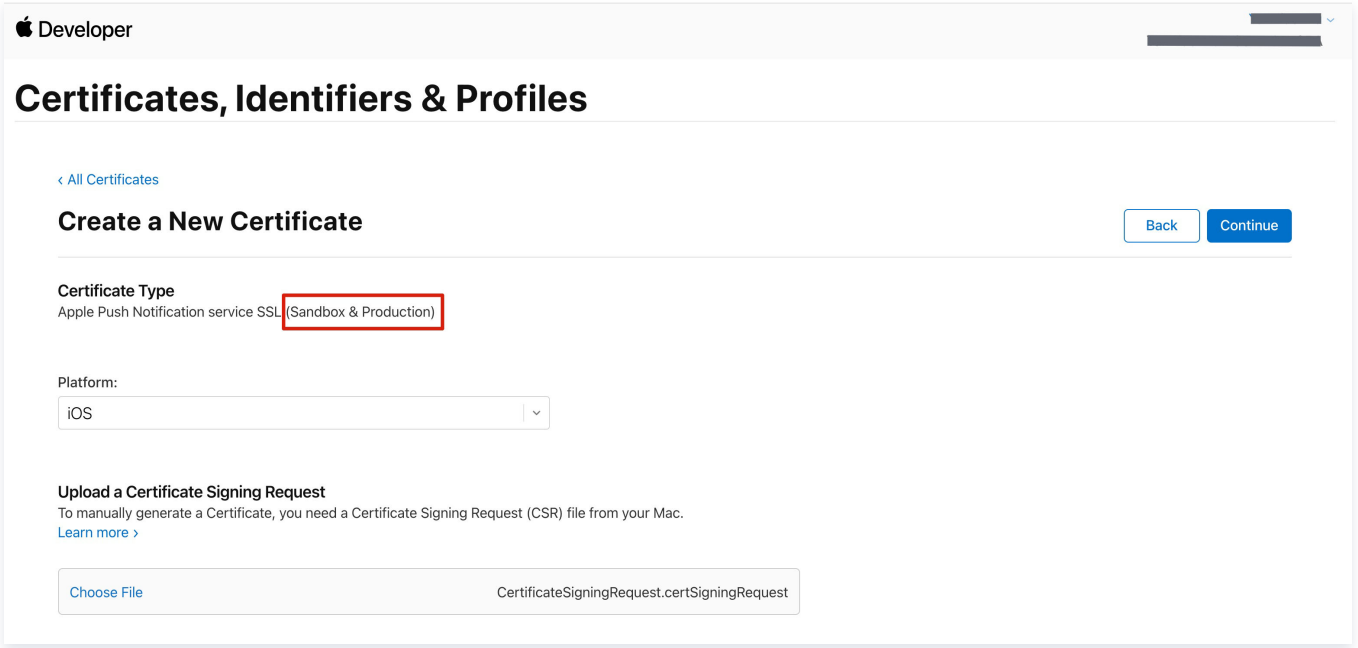


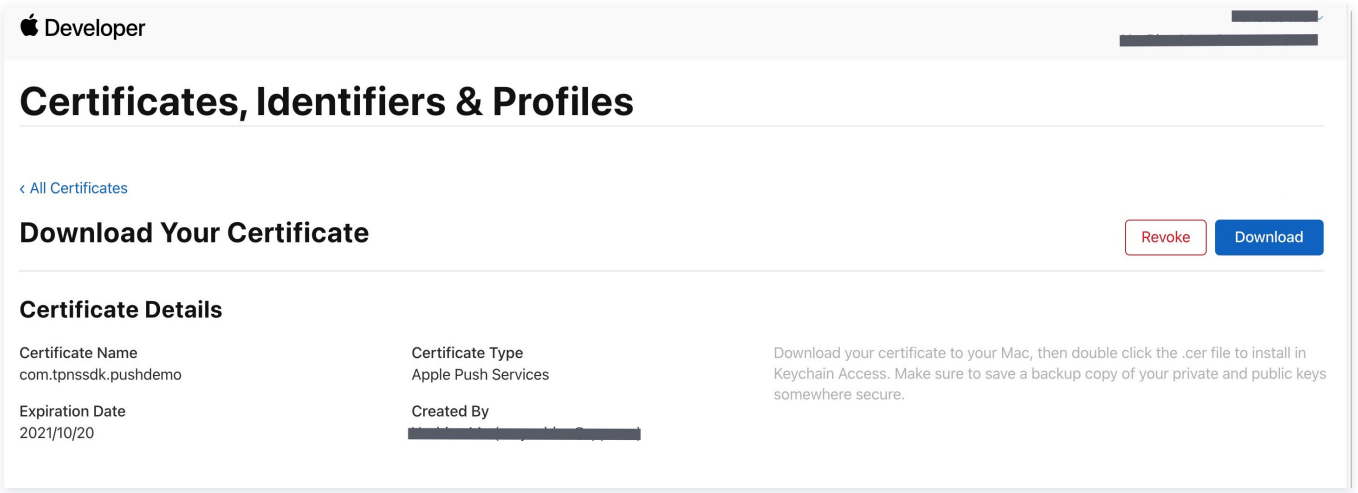
8. 单击 Download 下载开发环境的 `Development SSL Certificate` 到本地。



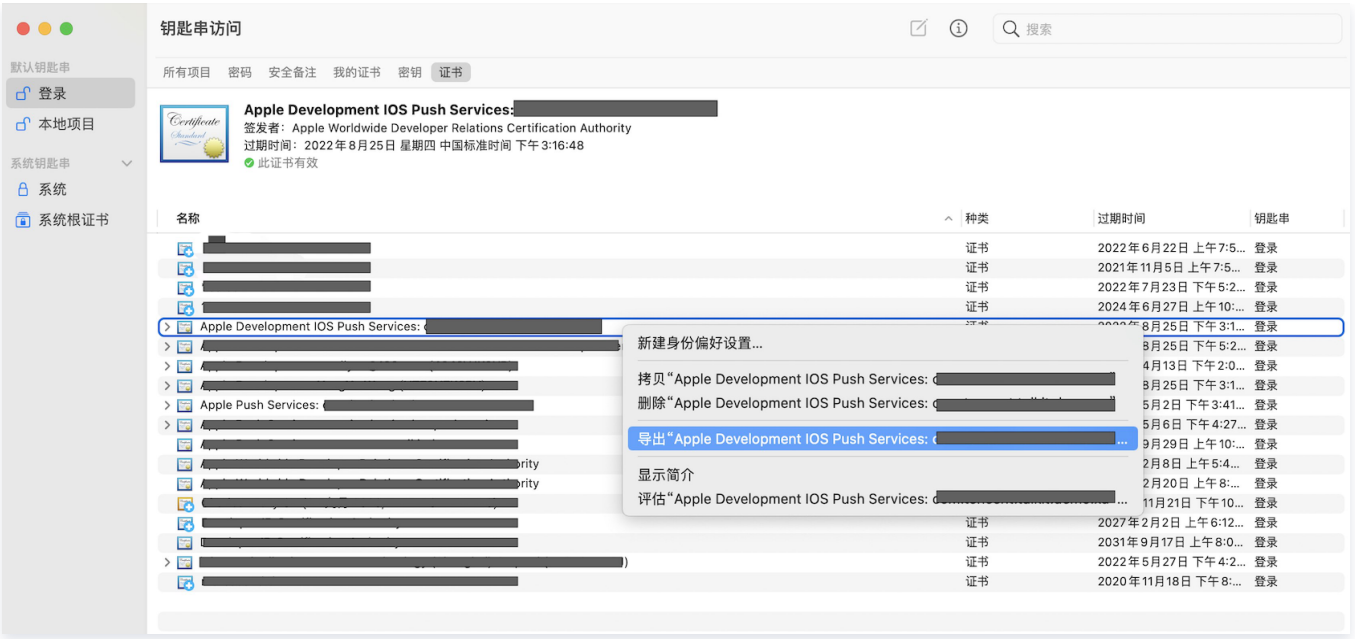
9. 再次按照上述步骤1 – 8，将生产环境的 `Production SSL Certificate` 下载到本地。

> **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。





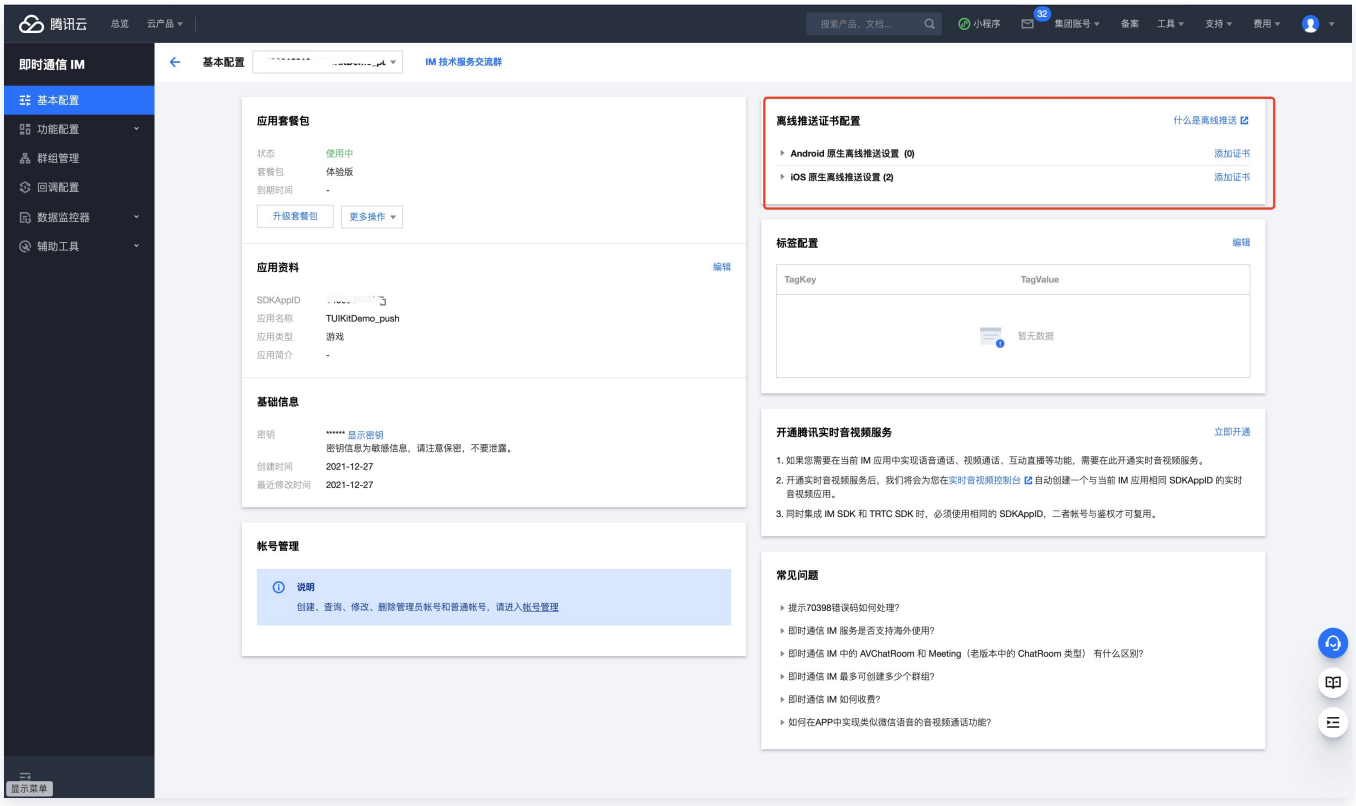
10. 双击打开下载的开发环境和生产环境的 `SSL Certificate`，系统会将其导入钥匙串中。
11. 打开钥匙串应用，在**登录 > 我的证书**，右键分别导出刚创建的开发环境（ `Apple Development IOS Push Service` ）和生产环境（ `Apple Push Services` ）的 `p12` 文件。



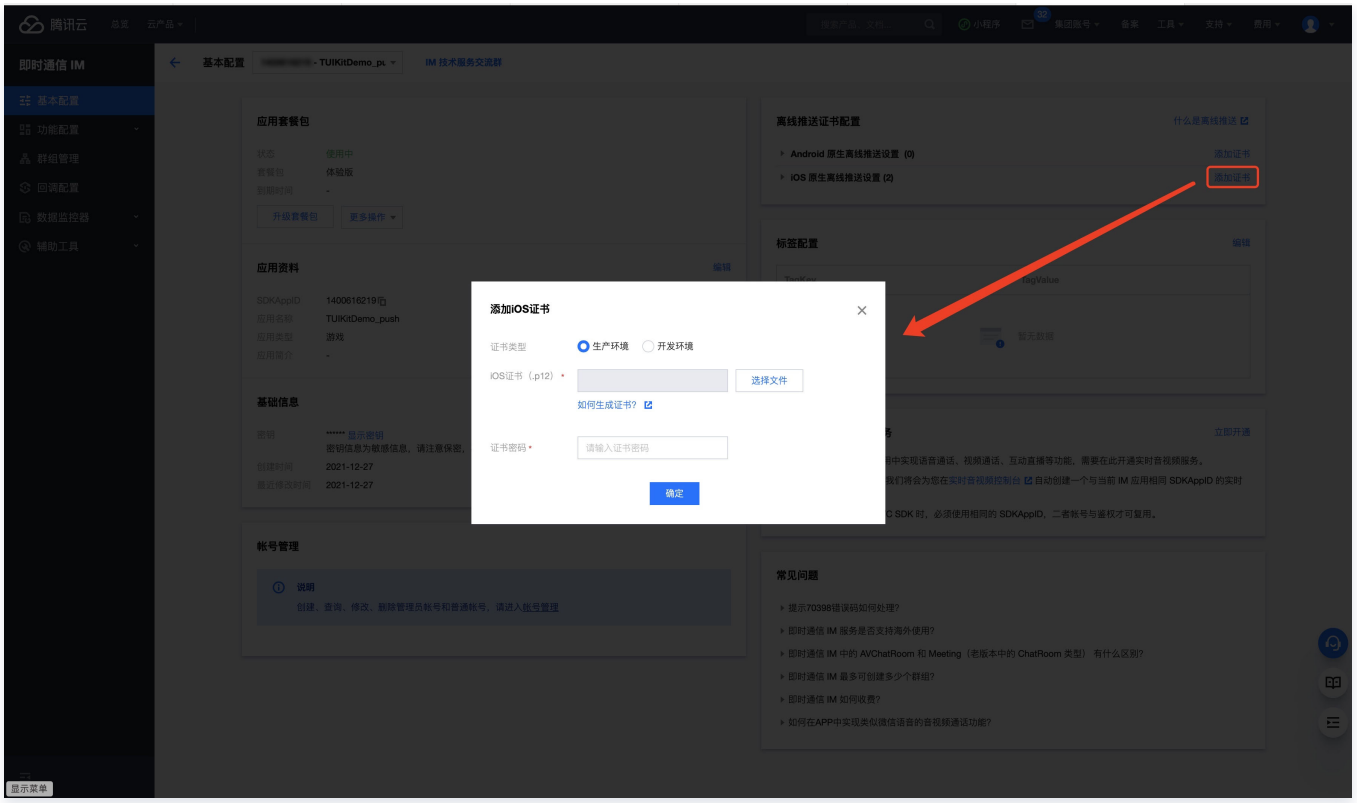
> **注意**
> 保存 `.p12` 文件时，请务必要为其设置密码。

步骤2：上传 p12 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。

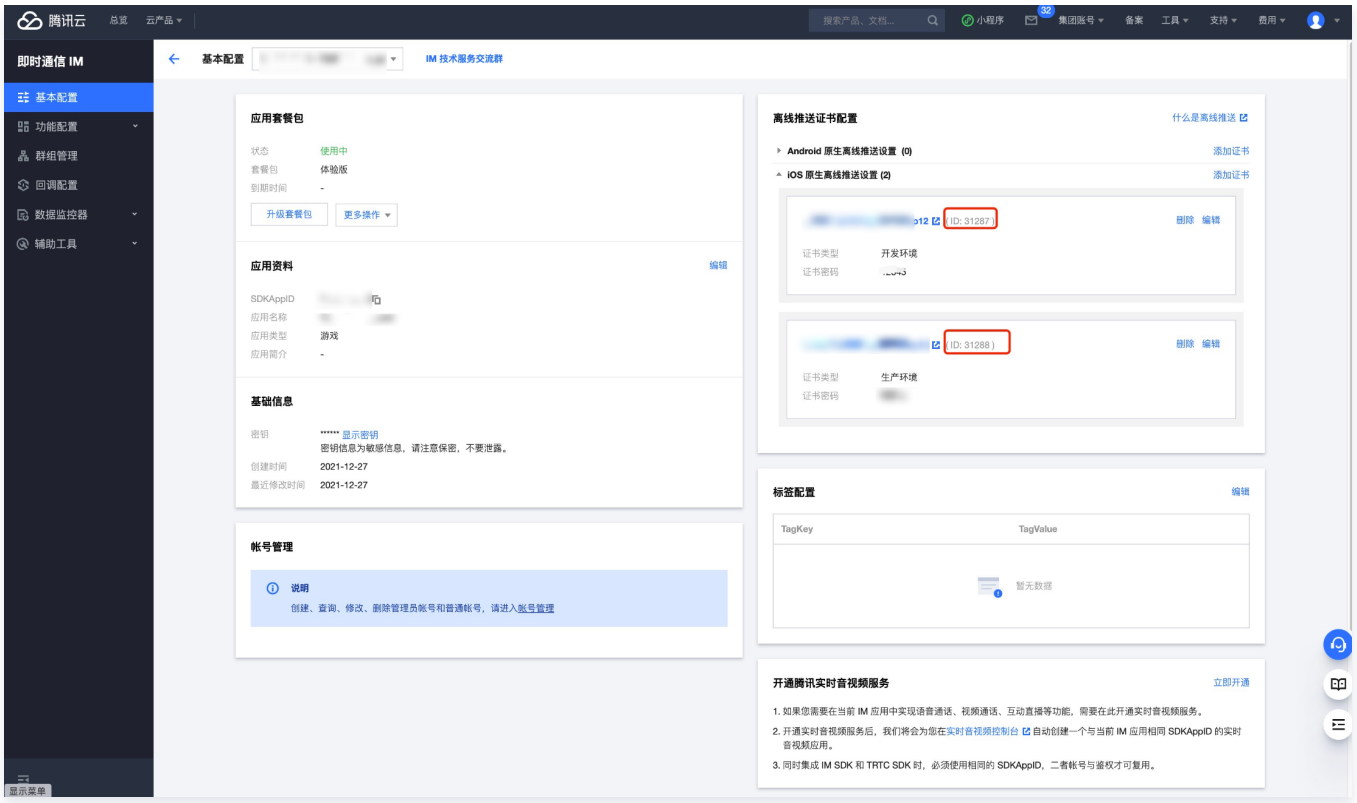


3. 单击 **iOS 原生离线推送设置**右侧的**添加证书**。
4. 选择证书类型，上传 iOS 证书（p.12），设置证书密码，单击**确认**。



> **注意：**
>
> - 上传的推送证书时需要确认下 Bundle ID 与您的主 App 的Bundle ID一致。
> - 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、创建描述文件

步骤 1: 创建 App ID

1. **登录到 Apple Developer Center:**
   - 访问 Apple Developer Center 并登录你的开发者账户。
2. **创建 App ID:**
   - 在 "Certificates, Identifiers & Profiles" 部分，选择 "Identifiers"。
   - 点击 "+" 按钮以创建新的 App ID。
   - 选择 "App IDs" 并点击 "Continue"。
   - 输入你的 App ID 名称和 Bundle ID（例如 **com.yourcompany.yourapp**）。
   - 确保选择适当的功能（如 Push Notifications、App Groups 等），然后点击 "Continue" 并确认。

步骤 2: 创建 Service Extension 的 App ID（可选）

- 重复上述步骤，创建一个新的 App ID，用于你的 Service Extension。
- Bundle ID 通常是主应用程序的 Bundle ID 后面加上扩展的标识符，例如 com.yourcompany.yourapp.extension。

步骤 3: 创建描述文件

1. **创建描述文件:**
   - 在 "Certificates, Identifiers & Profiles" 部分，选择 "Profiles"。
   - 点击 "+" 按钮以创建新的描述文件。
   - 选择 "iOS App Development" 或 "App Store"（根据你的需求），然后点击 "Continue"。
   - 选择你刚刚创建的 App ID（主应用程序的 App ID），然后点击 "Continue"。
   - 选择你的开发证书和设备（如果是开发描述文件），然后点击 "Continue"。
   - 输入描述文件的名称，然后点击 "Generate"。
2. **为 Service Extension 创建描述文件（可选）:**

- 重复上述步骤，为你的 Service Extension 创建一个新的描述文件，确保选择对应的 App ID。

步骤 4: 下载和安装描述文件

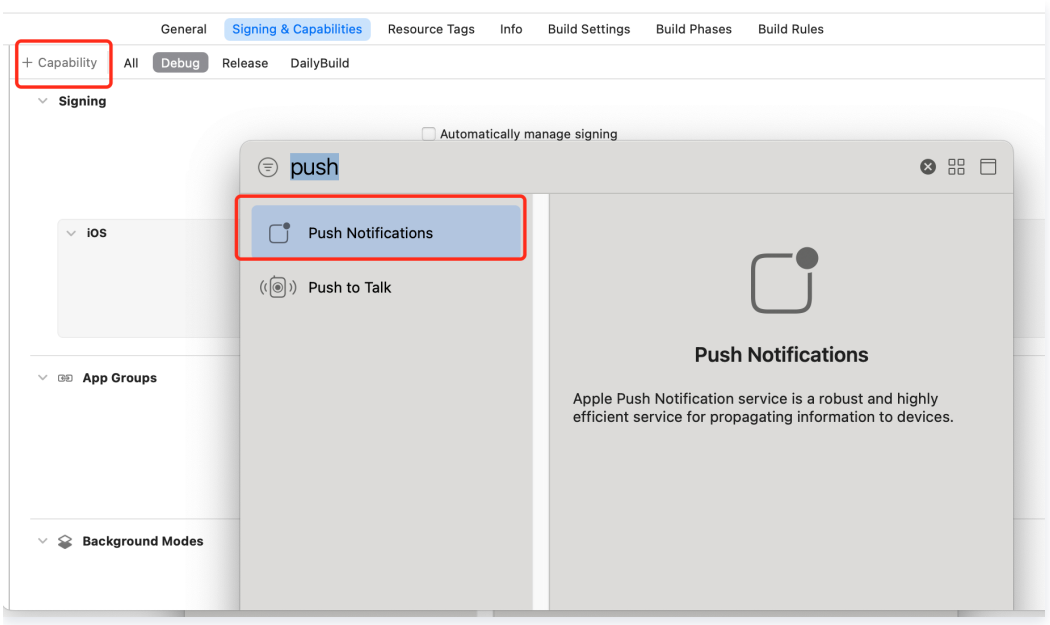
1. **下载描述文件:**
   - 在描述文件创建完成后，点击 "Download" 按钮下载描述文件。
2. **安装描述文件:**
   - 双击下载的描述文件，它会自动在 Xcode 中安装。

三、Xcode添加推送权限

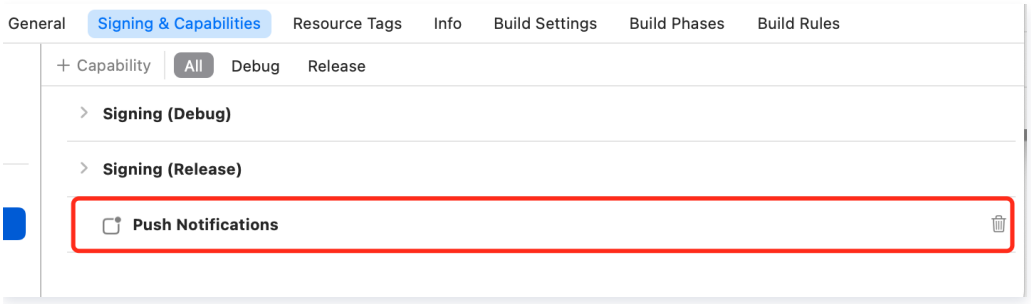
> **说明：**
> uniapp 和微信小程序多端框架内置推送权限，无需 Xcode 添加。

要在 App 中添加推送权限，请在 Xcode 项目中启用推送通知功能。

打开 Xcode 项目，在 Project > Target > Capabilities 页面中点击红框中的加号按钮，然后选择并添加 **Push Notifications** 。



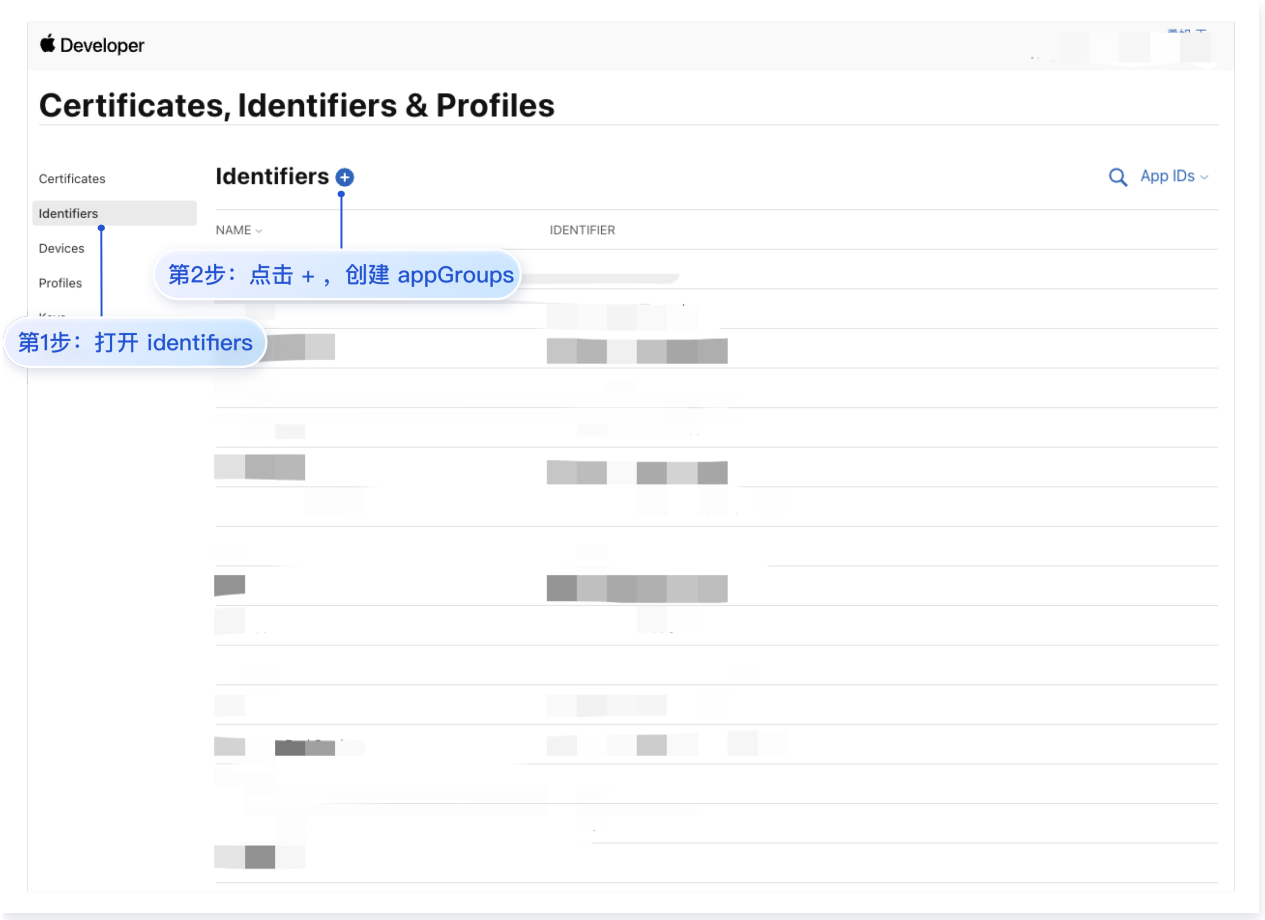
添加后的结果如图中红框所示。

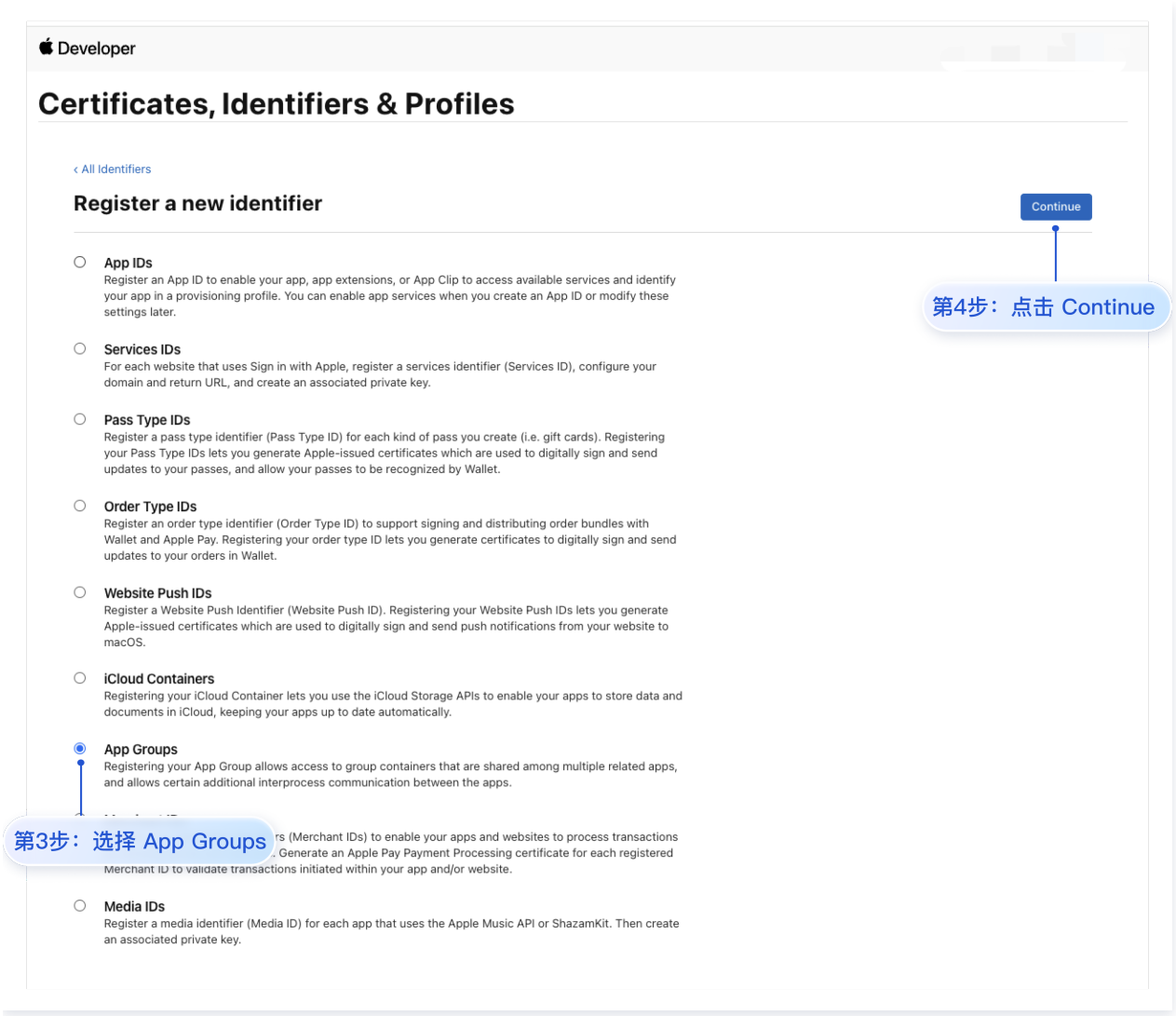


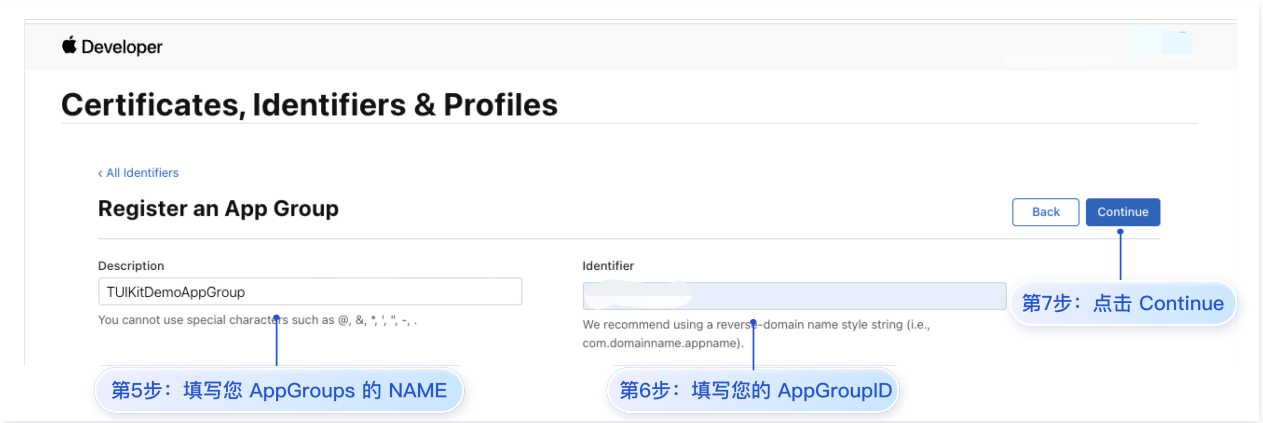
四、生成 App GroupID (可选)

当您需要使用 TIMPush 组件统计推送抵达率时，推荐您配置 TIMPushAppGroupID ，之后按照快速接入进行使用。
App GroupID 标识当前主 App 和 Extension 之间共享的 App Group，需要在主 App 和 Extension 的 Capability 中配置 App Groups 能力。

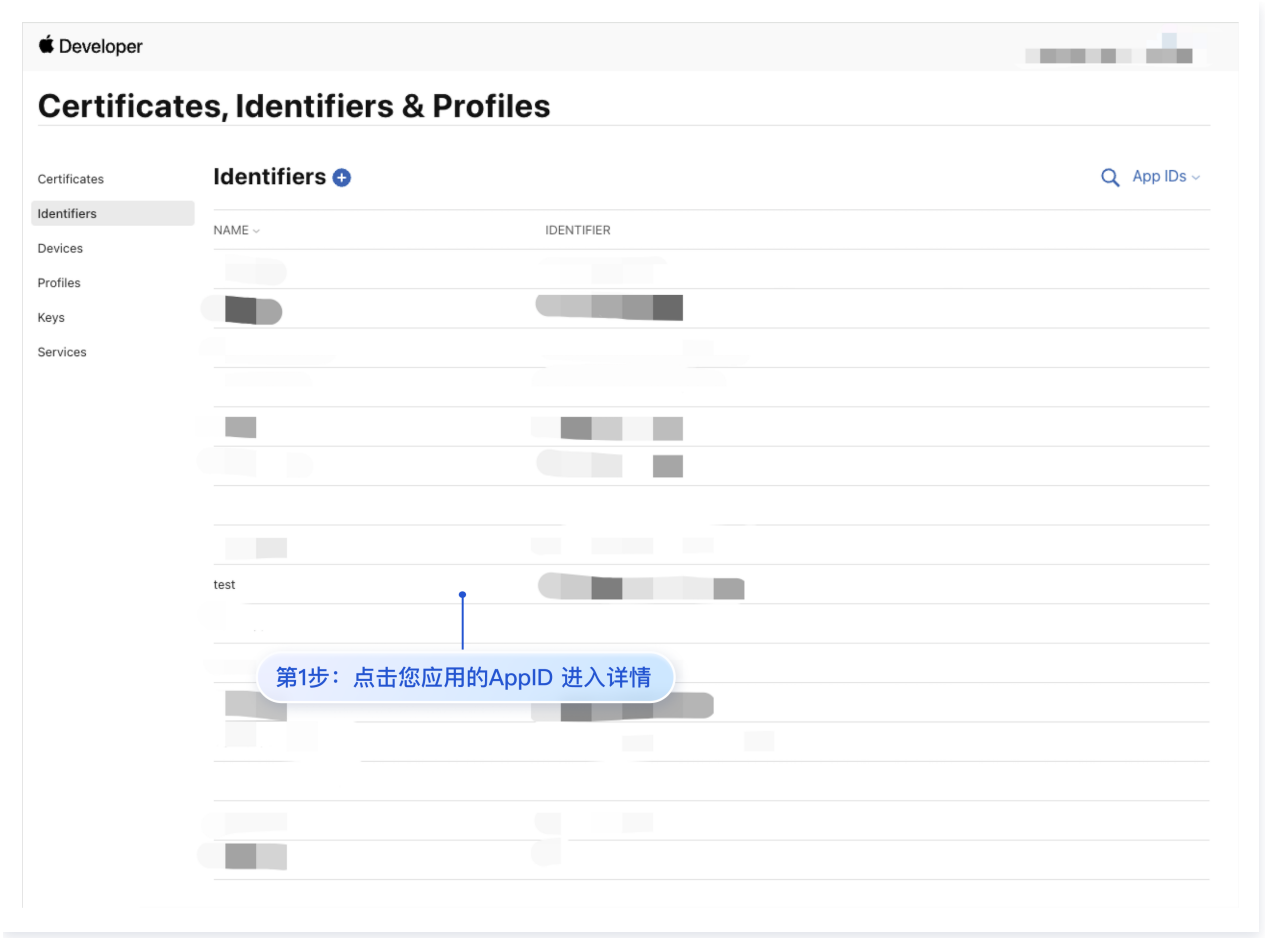
步骤1：登录苹果开发者中心网站，进入 identifiers > App Groups 创建 AppGroups。

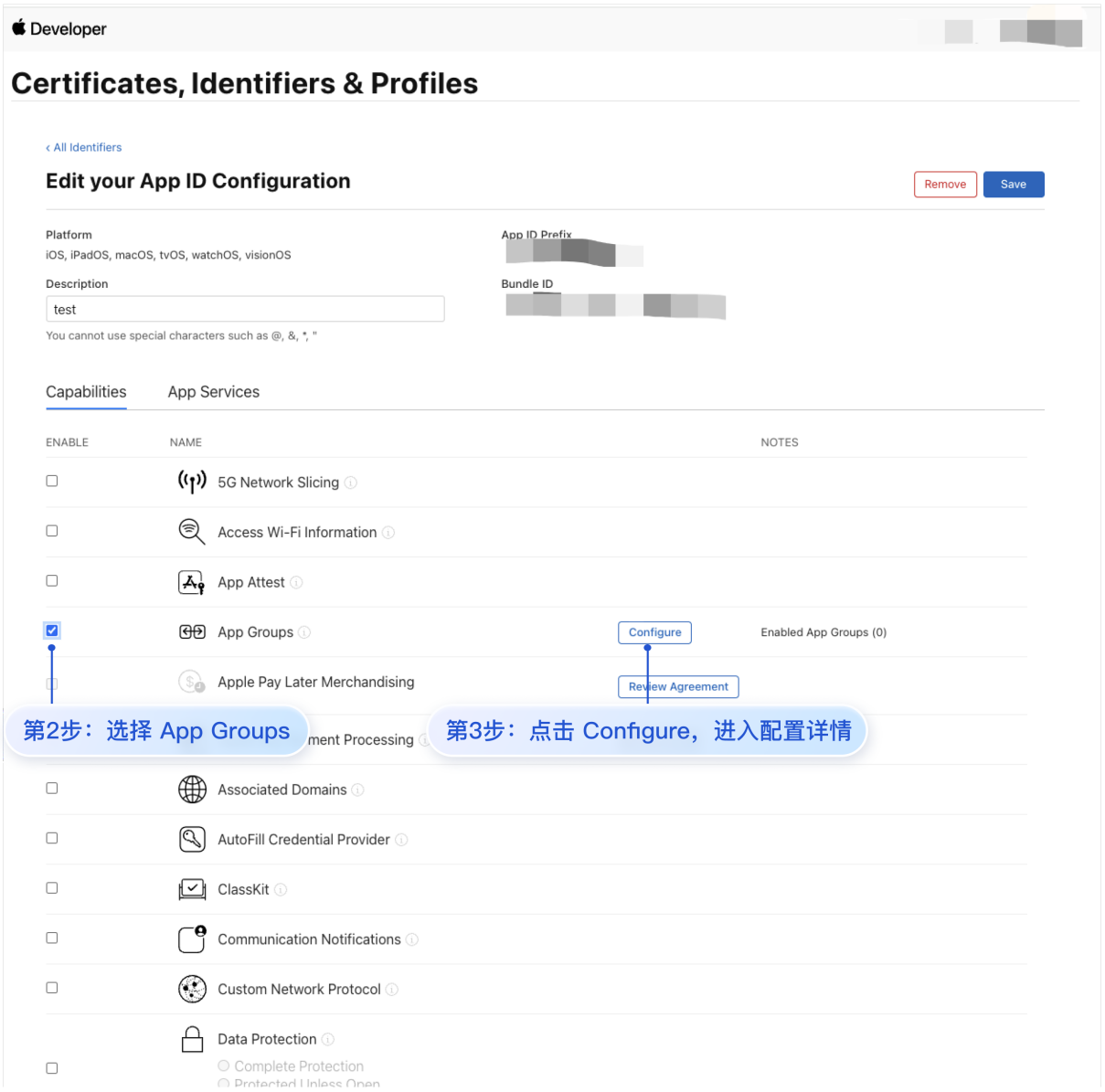

Developer
Certificates, Identifiers & Profiles
Certificates
Identifiers
Devices
Profiles
Identifiers
App IDs
NAME
IDENTIFIER
第2步：点击 + ，创建 appGroups
第1步：打开 identifiers

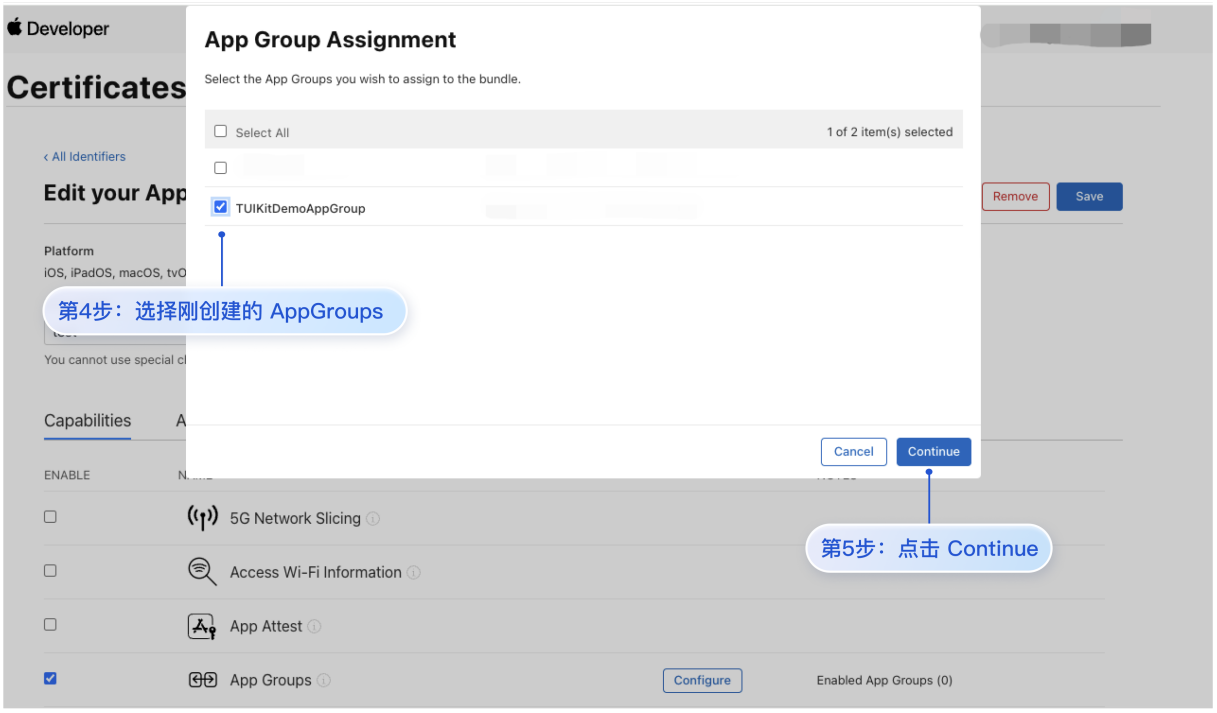




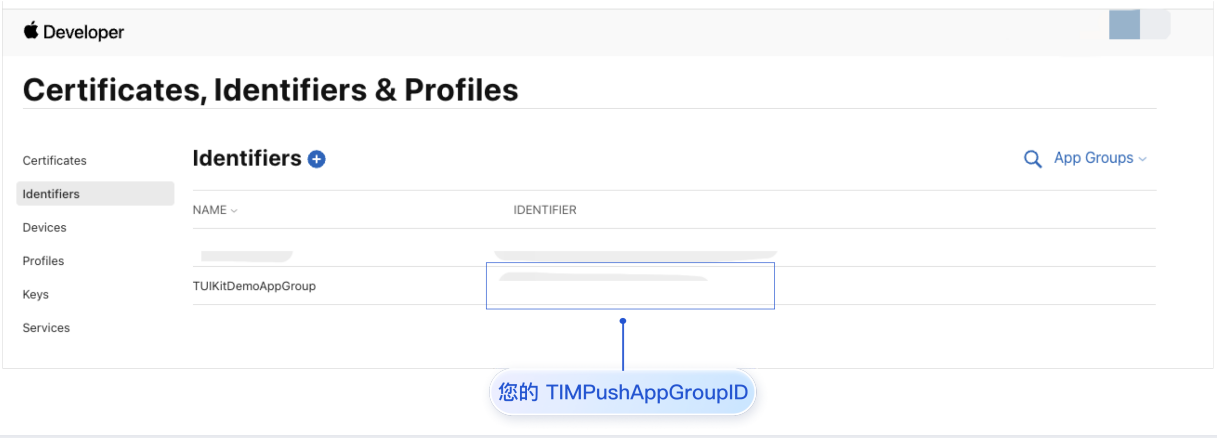
步骤2：绑定需要使用的应用的 AppID 到 AppGroups。


Developer
Certificates, Identifiers & Profiles
Certificates
Identifiers
Devices
Profiles
Keys
Services
Identifiers
App IDs
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
Remove
Save
Platform
iOS, iPadOS, macOS, tvOS, watchOS, visionOS
App ID Prefix
Description
test
You cannot use special characters such as @, &, *, "
Bundle ID
Capabilities
App Services
ENABLE
NAME
NOTES
5G Network Slicing
Access Wi-Fi Information
App Attest
App Groups
Configure
Enabled App Groups (0)
Apple Pay Later Merchandising
Review Agreement
第2步：选择 App Groups
第3步：点击 Configure，进入配置详情
Associated Domains
AutoFill Credential Provider
ClassKit
Communication Notifications
Custom Network Protocol
Data Protection
Complete Protection

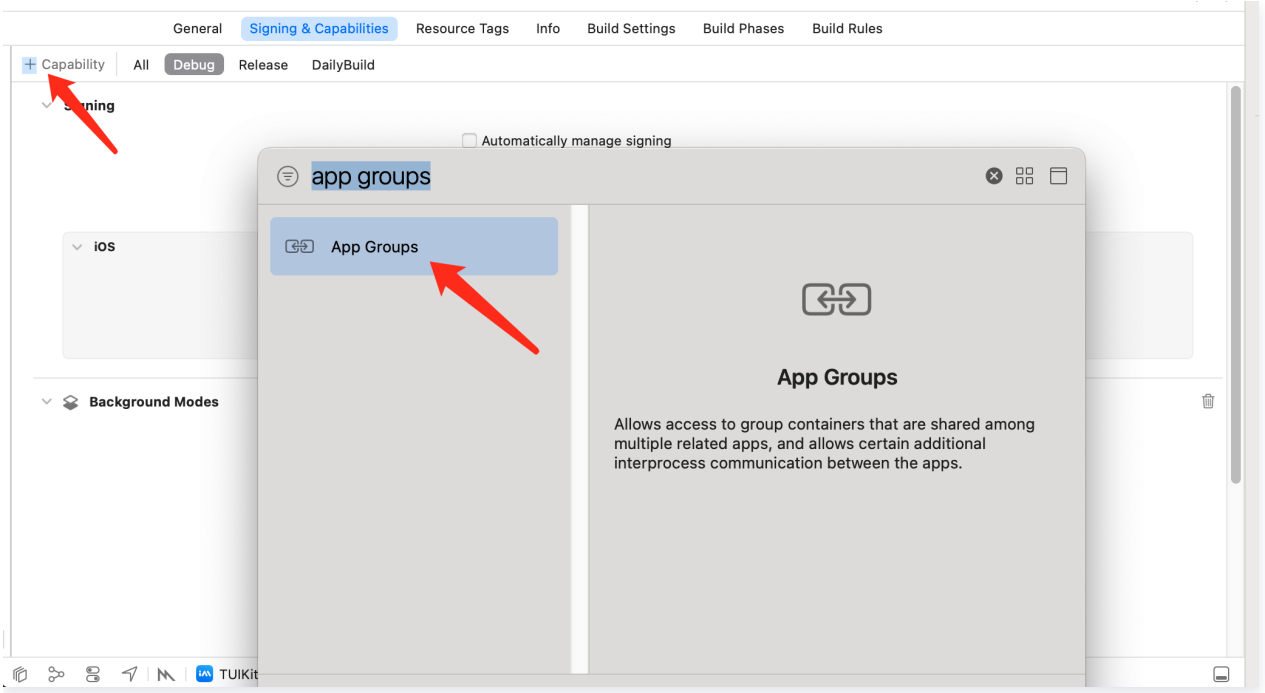


步骤3：获取您的 TIMPushAppGroupID。

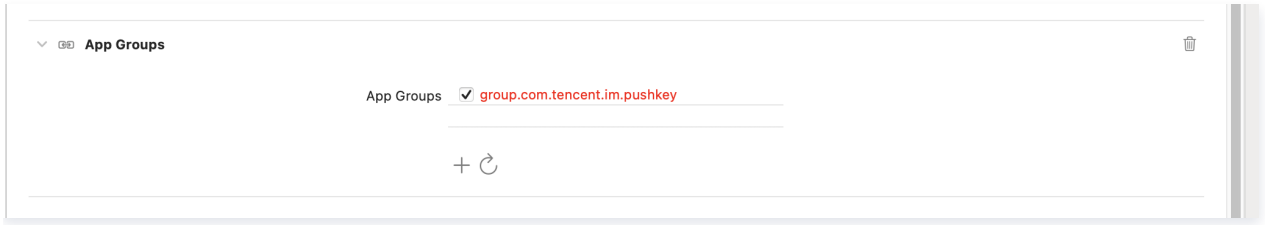


步骤4：在Xcode中配置 TIMPushAppGroupID。

打开 Xcode 项目，在 Project > Target > Capabilities 页面中点击红框中的加号按钮，然后选择并添加 App Groups。



填入第3步中配置的 Group ID，例如 group.com.tencent.im.pushkey：



最后请在 Extension 的 Capability 中进行 App Groups 的相同配置。

附录: 开发者账号、证书类型与描述文件

苹果开发者账号:

创建 iOS 证书和 Profile 需使用付费的苹果账号，苹果开发者账号有三种，个人开发者、公司开发者、企业开发者，用途各有不同。

- 个人开发者和公司开发者是 99 美元/年，企业开发者是 299 美元/年。
- 企业开发者一般是大企业开发内部应用时使用，不能上架 App Store 的。
- 请先确保你已经有一个个人开发者或公司开发者账号。

证书介绍:

证书有多种类型，用于不同的目的。以下是 iOS 证书的分类：

证书类型	区别
开发者证书（Developer Certificate）	用于在开发阶段对应用程序进行签名和调试。开发者证书由苹果开发者中心颁发，需要使用 Xcode 或者其他开发工具进行申请和管理。
分发证书（Distribution Certificate）	用于将应用程序分发给其他用户或者上传到 App Store 进行审核。分发证书由苹果开发者中心颁发，需要使用 Xcode 或者其他开发工具进行申请和管理。
推送证书（Push Certificate）	用于实现 APNs 远程推送功能，可以让应用程序接收来自服务器的推送通知。推送证书由苹果开发者中心颁发，需要使用 Xcode 或者其他开发工具进行申请和管理。
企业证书（Enterprise Certificate）	用于将应用程序分发给企业内部员工或者客户。企业证书由苹果开发者中心颁发，需要使用 Xcode 或者其他开发工具进行申请和管理。

描述文件介绍:

描述文件（Provisioning Profiles）同样也分两种，分为开发（Development）和发布（Distribution），配置文件中包含了证书、App ID、设备（Devices），
后缀名为 .mobileprovision。它在开发者账号体系中扮演着配置和验证的角色，是真机调试和打包上架必须的文件。

描述文件（Provisioning Profile）	作用
开发（Development）	• 一个 Provisioning Profile 对应一个 App ID (Bundle ID) • Provisioning Profile 决定 Xcode 用哪个证书（公钥）/私钥组合（Key Pair/Signing Identity）来签名应用程序（Signing Product），将在应用程序打包时嵌入到 .ipa 包里。 • Provisioning Profile 把这些信息全部打包在一起，方便我们在调试和发布程序打包时使用。这样，只要在不同的情况下选择不同的 Provisioning Profile 文件就可以了。 • Provisioning Profile 也分为 Development 和 Distribution 两类，有效期同 Certificate 一样。Development 版本的 ProvisioningProfile 用于开发调试，Distribution 版本的 ProvisioningProfile 主要用于提交 App Store 审核，其不指定开发测试的 Devices。
发布（Distribution）	

uni-app

最近更新时间：2025-03-17 17:25:43

uni-app 腾讯云推送服务（Push） 目前推送支持小米、华为、荣耀、OPPO、vivo、魅族、APNs、一加、realme、iQOO 、FCM 和 苹果等厂商通道。

Android

注册应用到厂商推送平台

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

观看视频

> **说明：**
> - 通知栏推送：应用需在小米软件商店上架。
> - 需要使用企业账号进行推送配置。
> - 小米开发平台的应用包名与插件应用包名需保持一致。

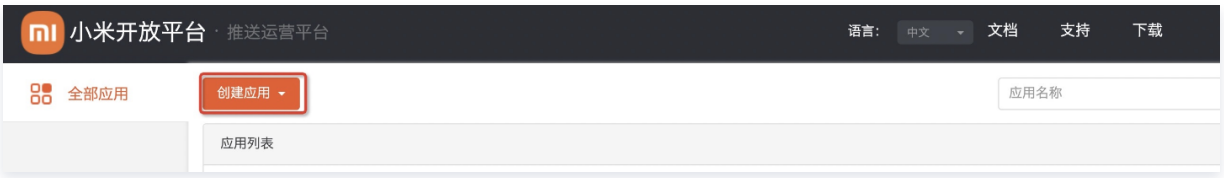
步骤1：注册小米开发者账号

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**。

2. 创建应用，完善应用资料界面，单击**保存**。

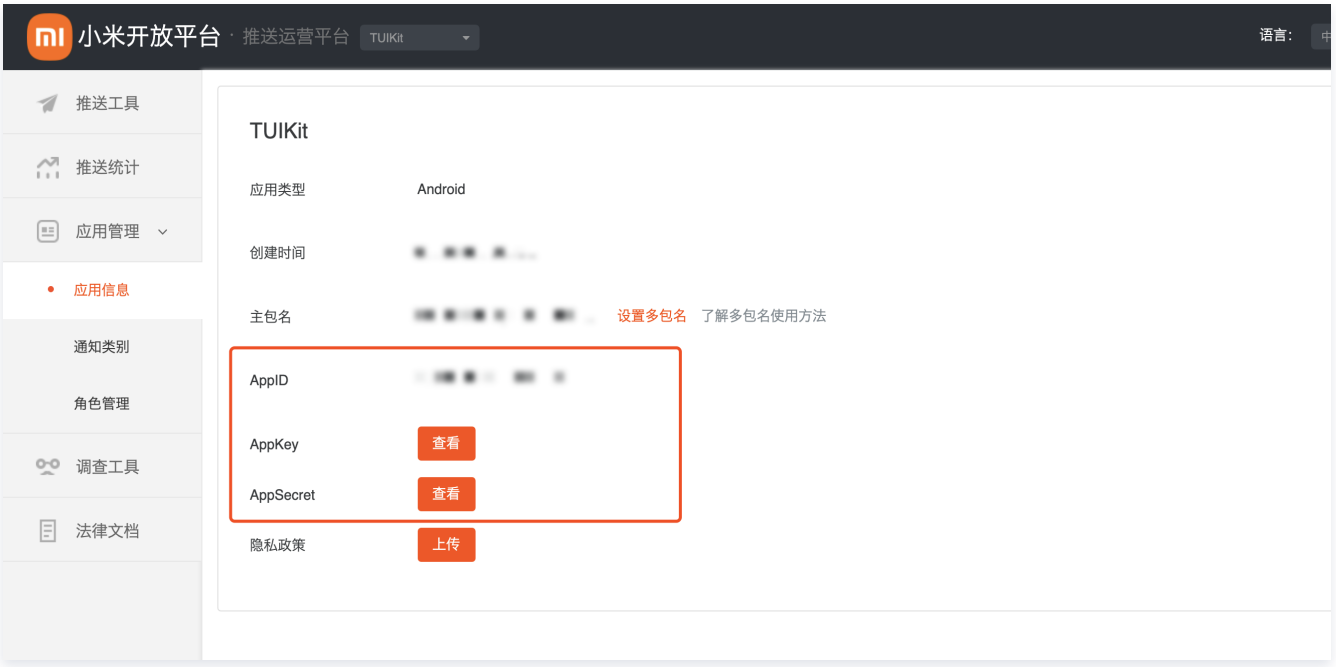


步骤3：启用推送

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。

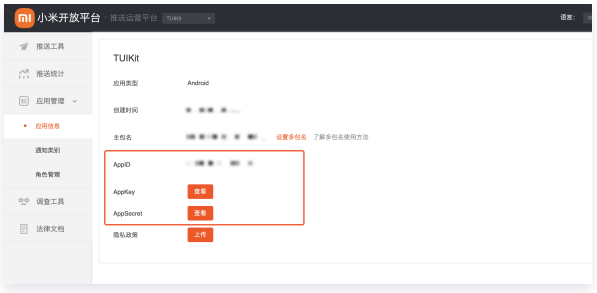
步骤4：查看获取应用信息

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
	**说明：** 应用内指定界面，请使用默认配置。

华为

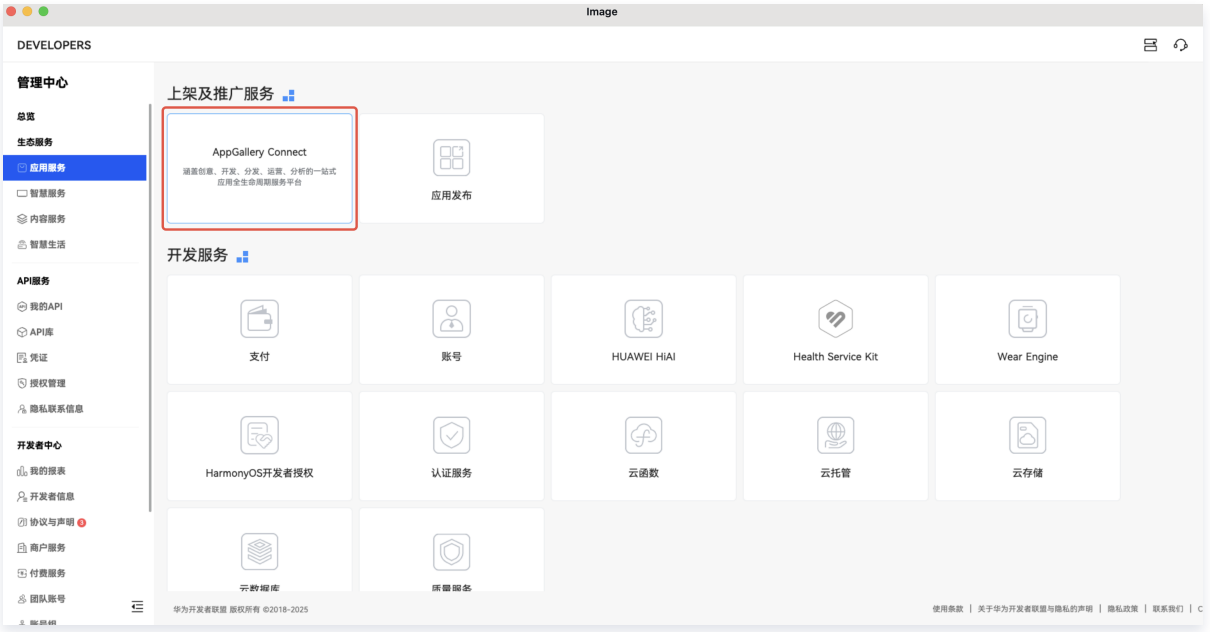
观看视频

步骤1：注册华为开发者账号

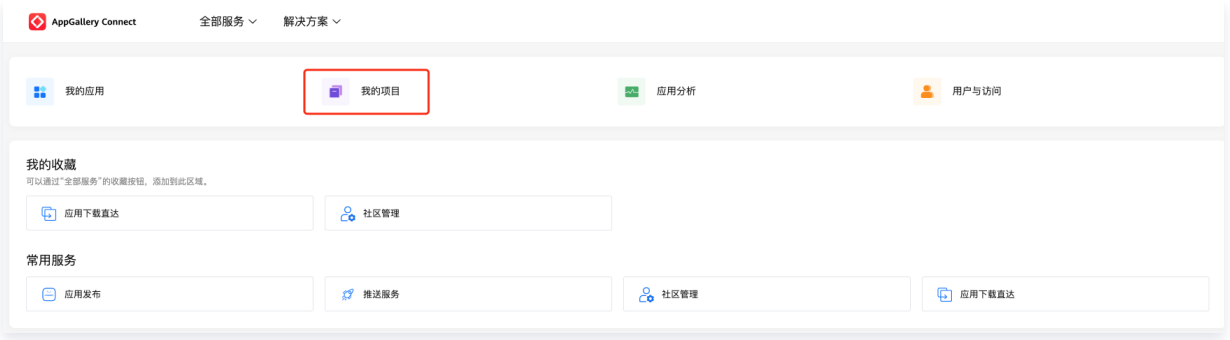
进入 华为开发者联盟，注册华为开发者账号，详情请参见 注册账号 。

步骤2：创建应用

1. 在 华为管理中心 的**应用管理中**，单击 **AppGallery Connect**，进入应用管理中心。



2. 单击**我的项目**，添加一个新的项目。



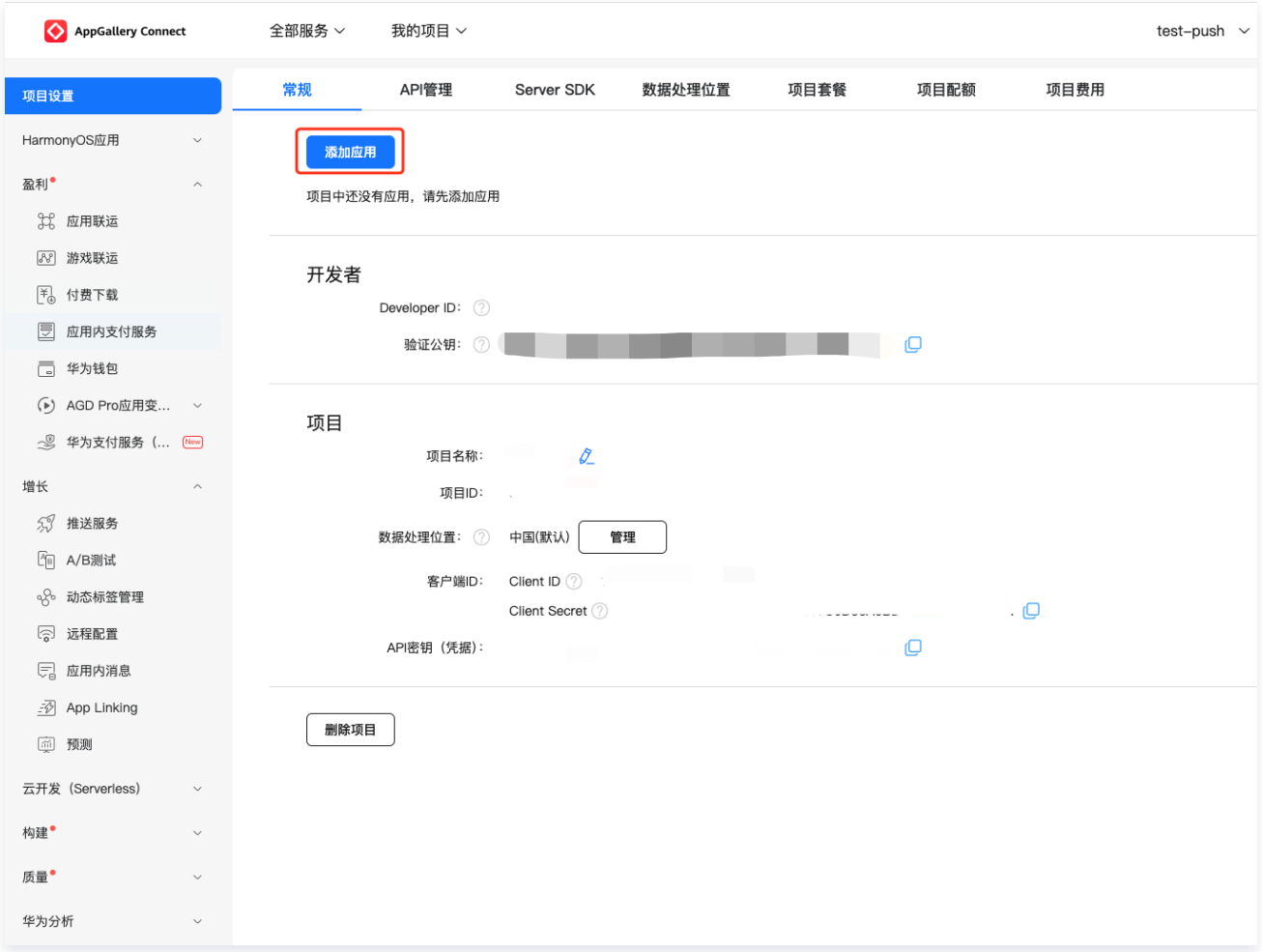
3. 在**项目设置**栏单击**推送服务 > 立即开通**。

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。

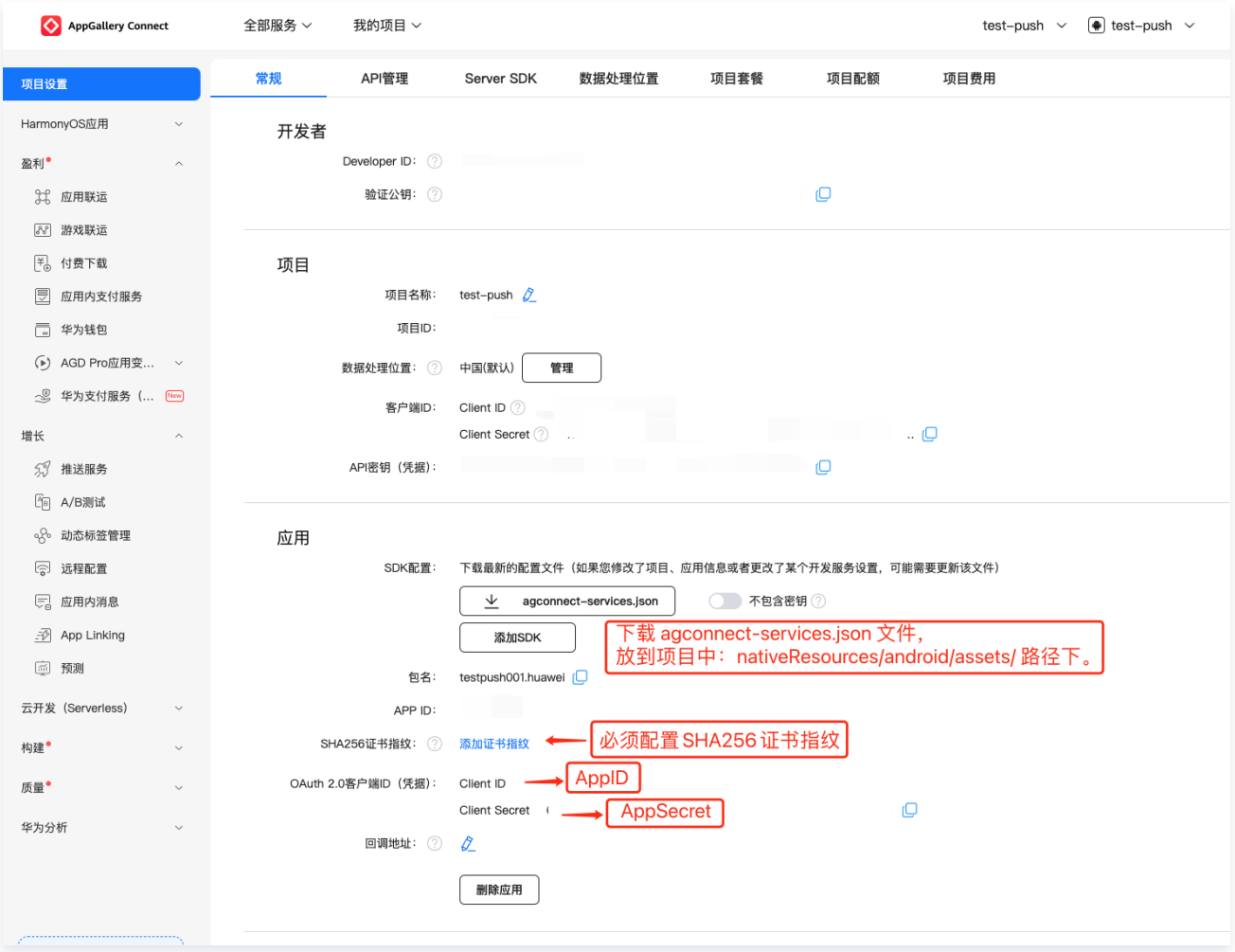
> **说明：**
> 应用包名与插件应用包名保持一致。



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
	**说明：** • Client ID 对应 AppID，Client Secret 对应 AppSecret。 • 应用内指定界面，请使用默认配置。

OPPO

观看视频

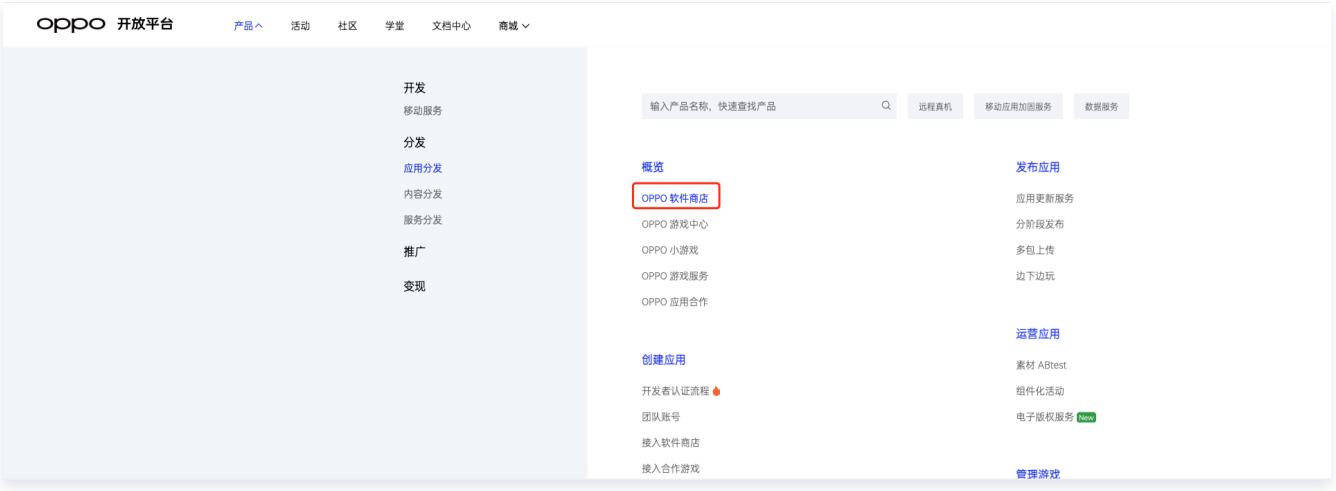
> **说明：**
> - 通知栏推送：应用需在 OPPO 软件商店上架；
> - 通知栏推送测试权限：每天仅可推送1000条消息，限测试使用。应用上架后需重新申请“通知栏推送”权限，以获得正常消息推送数量；
> - 平台将会在1个工作日内返回审核结果，开发者可以在申请页面查看审核结果，其他问题可咨询开放平台客服。

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。

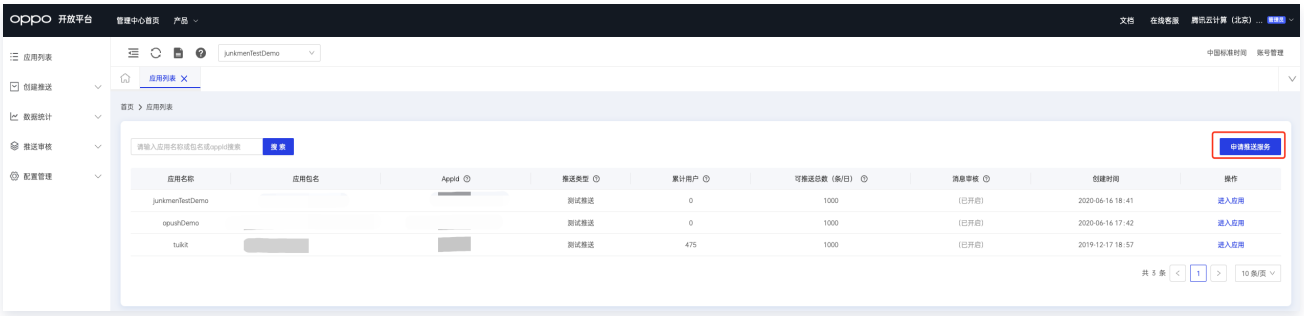


步骤3：开通 PUSH 服务

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
	**说明：** • 应用内指定界面，请使用默认配置。 

vivo

观看视频

说明：

- 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。
- vivo开发平台的应用包名与插件应用包名需保持一致。

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

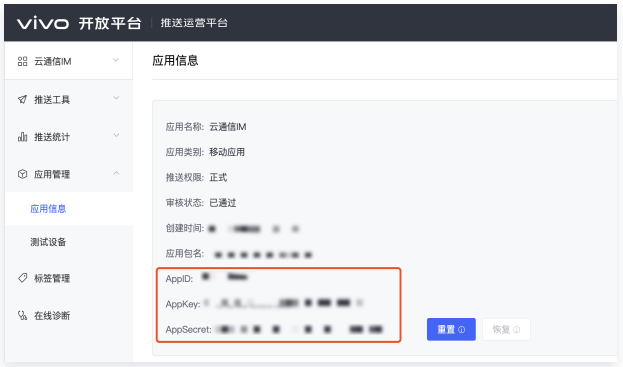
进入管理中心单击**推送服务 > 推送申请**为新建的应用申请开通推送。

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
	

回执配置请参见：消息触达统计配置 > vivo

魅族

观看视频

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。

> **说明：**
> 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
	> **说明：** • 应用内指定界面，请使用默认配置。 

回执配置请参见：消息触达统计配置 > 魅族

荣耀

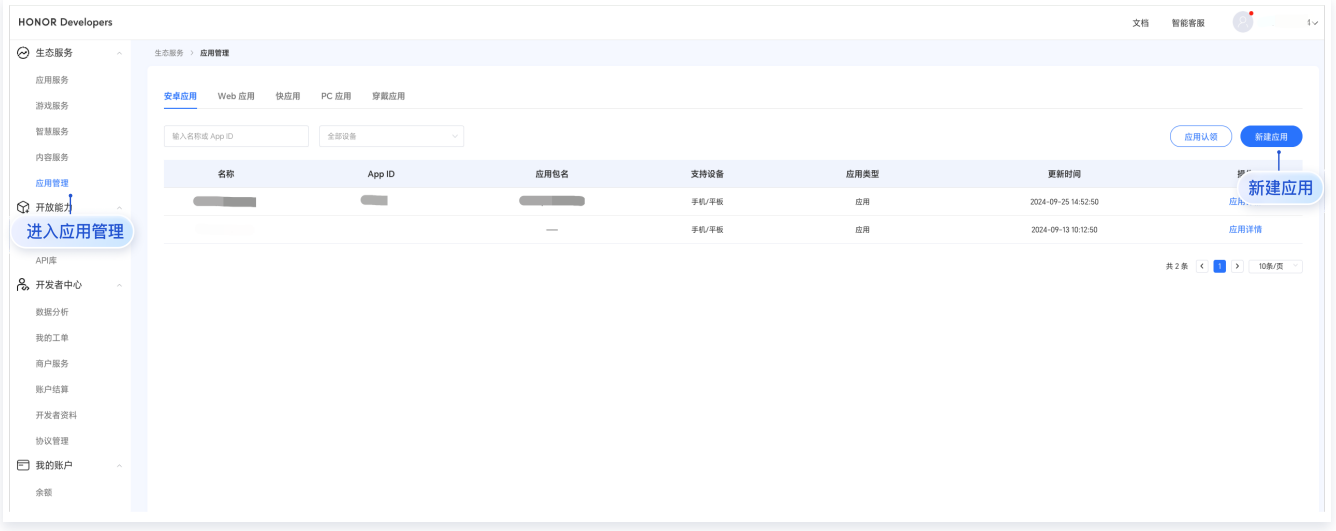
步骤1. 注册荣耀开发者账号

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: 创建应用

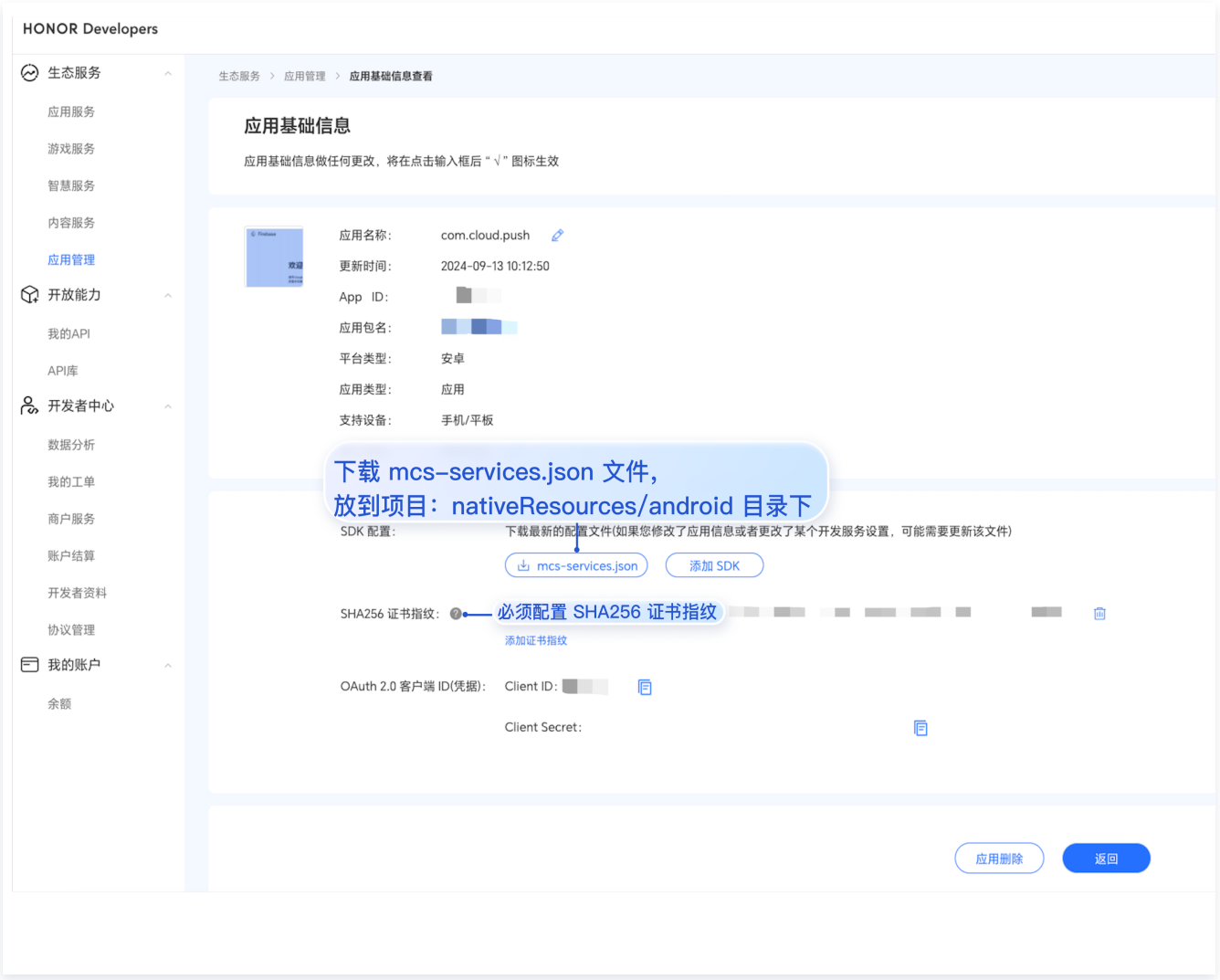
1. 进入**应用管理**，点击**新建应用**创建新应用。



2. 进入**应用详情**， 绑定**应用包名**，**下载 mcs-services.json 文件**。

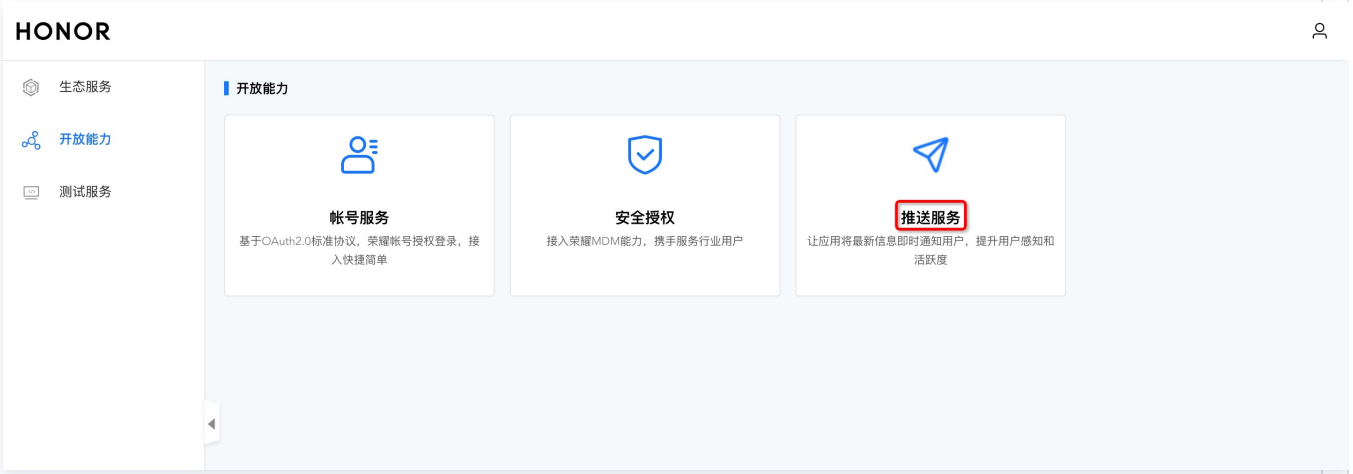
说明：

- 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 mcs-services.json 文件，放到项目中：nativeResources/android/ 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 mcs-services.json 文件。

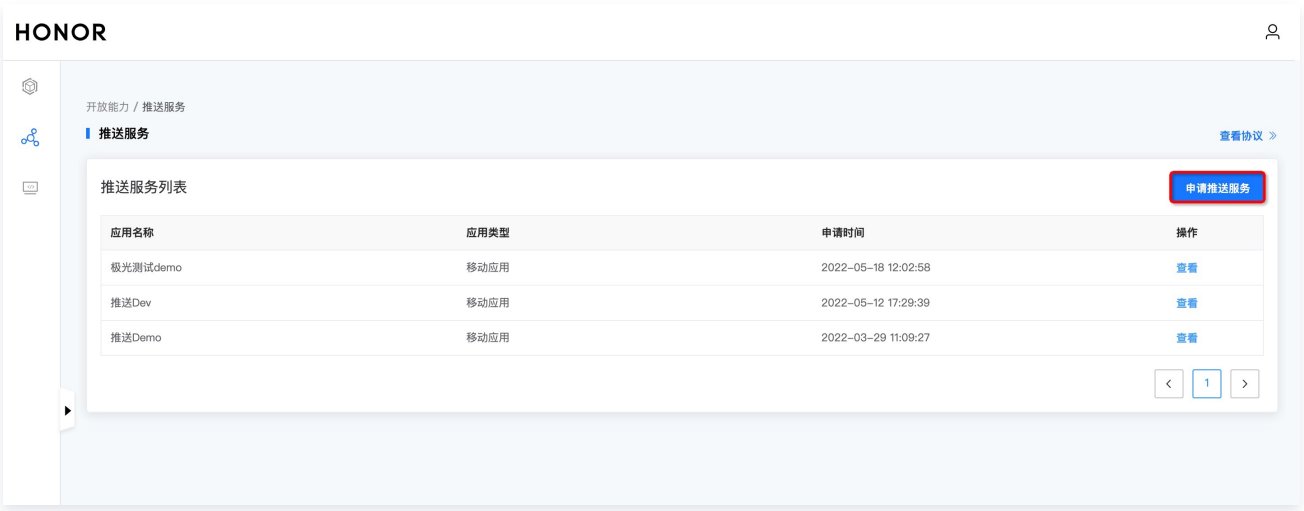


步骤4: 开通推送服务

1. 单击**开发能力 -> 推送服务**进入推送服务列表页面。

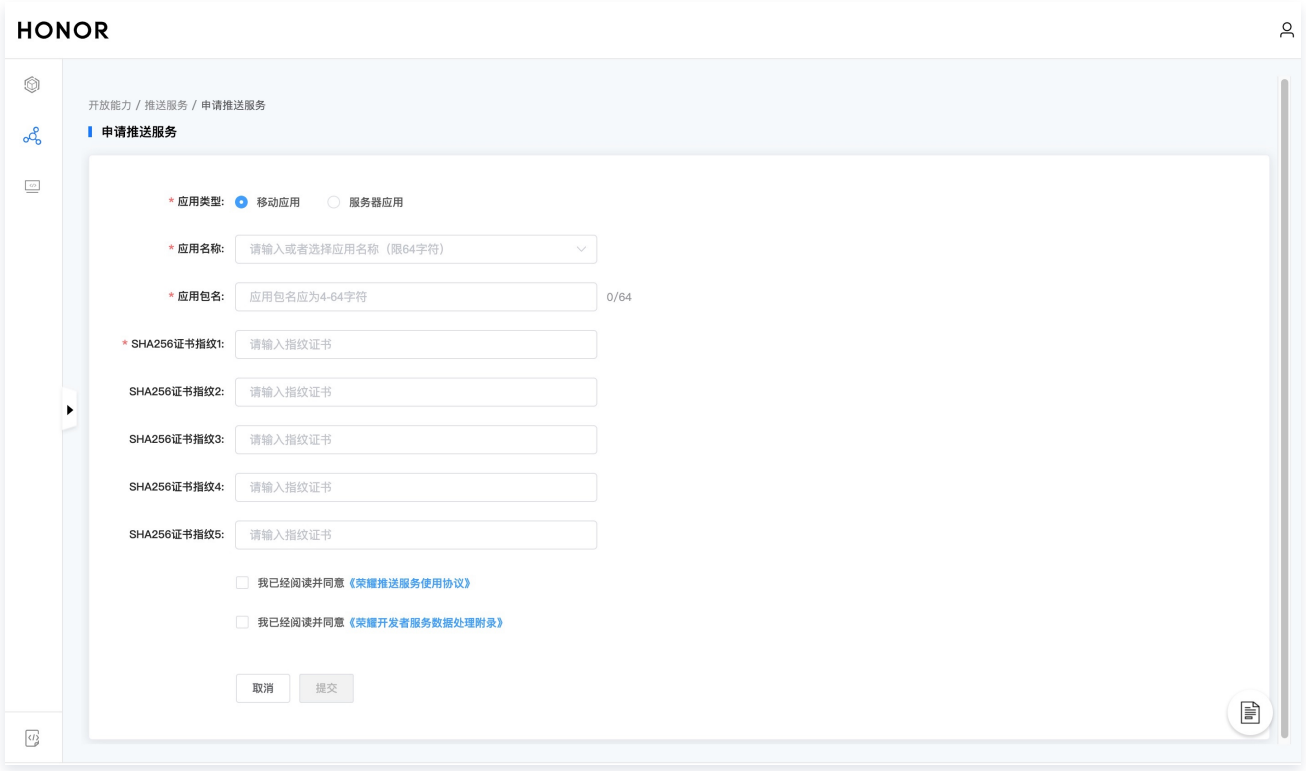


2. 单击**申请推送服务**进入应用申请页面。



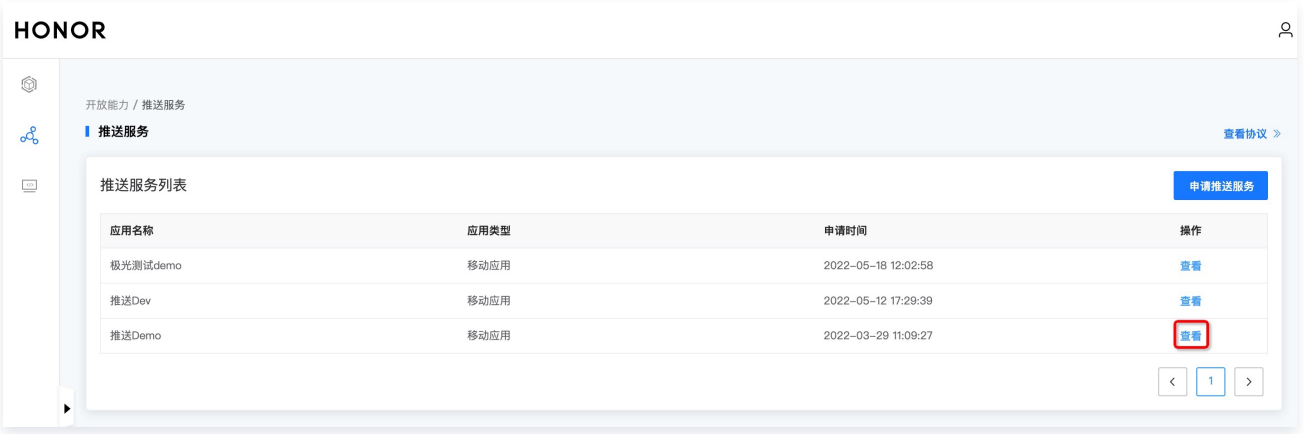
3. 选择应用类型“移动应用”，填写应用包名和证书指纹、同意推送服务协议和数据处理附录，单击**提交**。

> **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

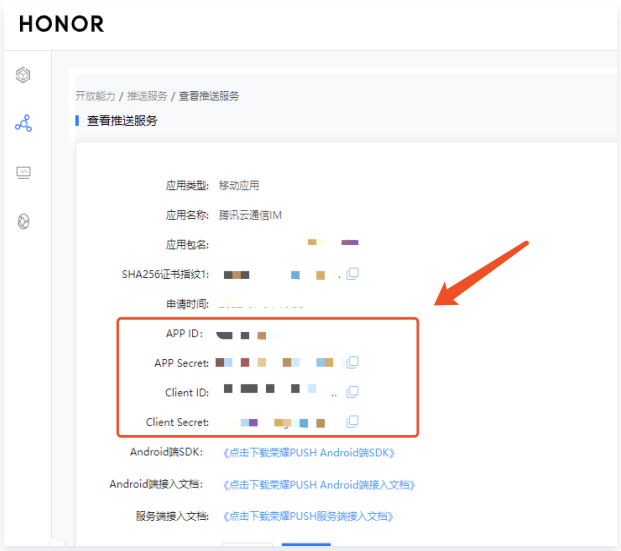
在**推送服务**列表中，单击**查看**，获取应用信息。



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，不可以修改。该配置用于派发单击后离线推送插件的事件监听，不可以直接配置应用内页面的跳转。

Google FCM

步骤1：进入Firebase 控制台

进入 Firebase 控制台，登录谷歌账号。

步骤2：创建应用

1. 单击**创建项目**，添加一个新的项目。

2. 进入Android 应用。

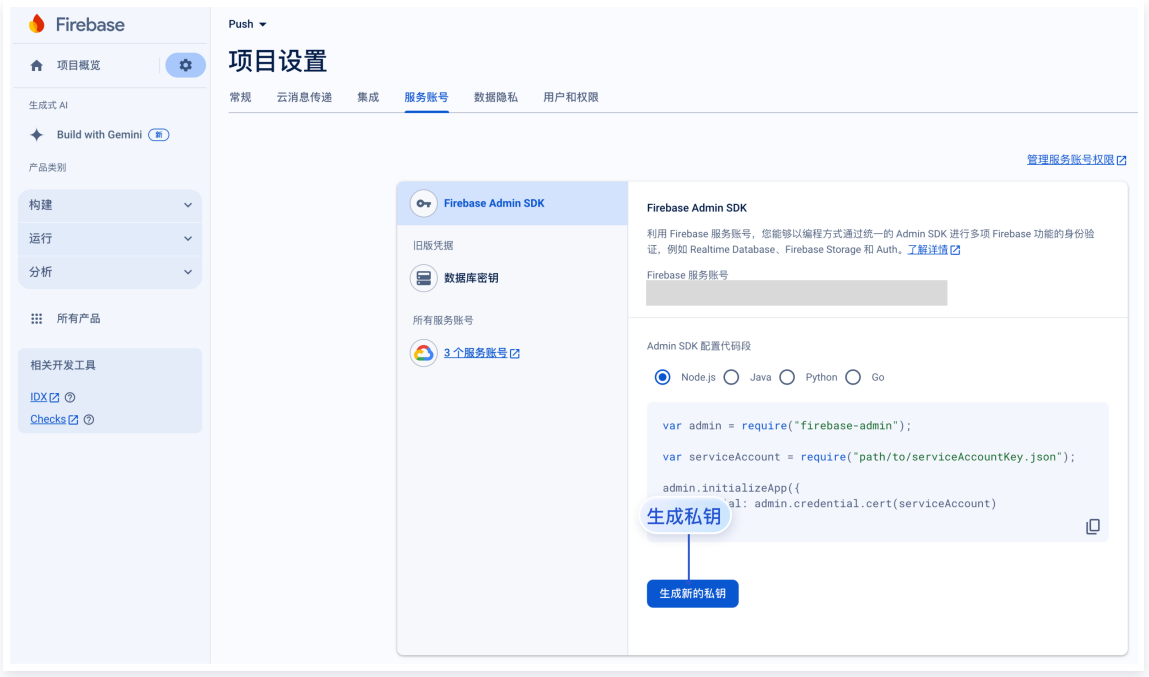
3. 输入应用信息，注册应用。

4. 下载并添加配置文件并完成注册。

步骤3. 生成证书私钥。

1. 点击应用，进入应用管理。

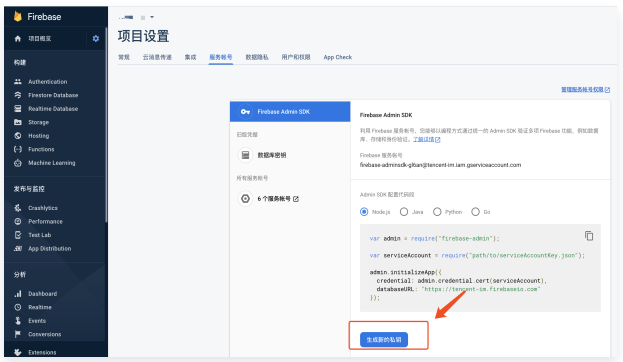
2. 在**项目设置**单击**服务账号 > 生成新的密钥**。



步骤4. 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



iOS

集成 uni-app 腾讯云推送服务（Push）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。

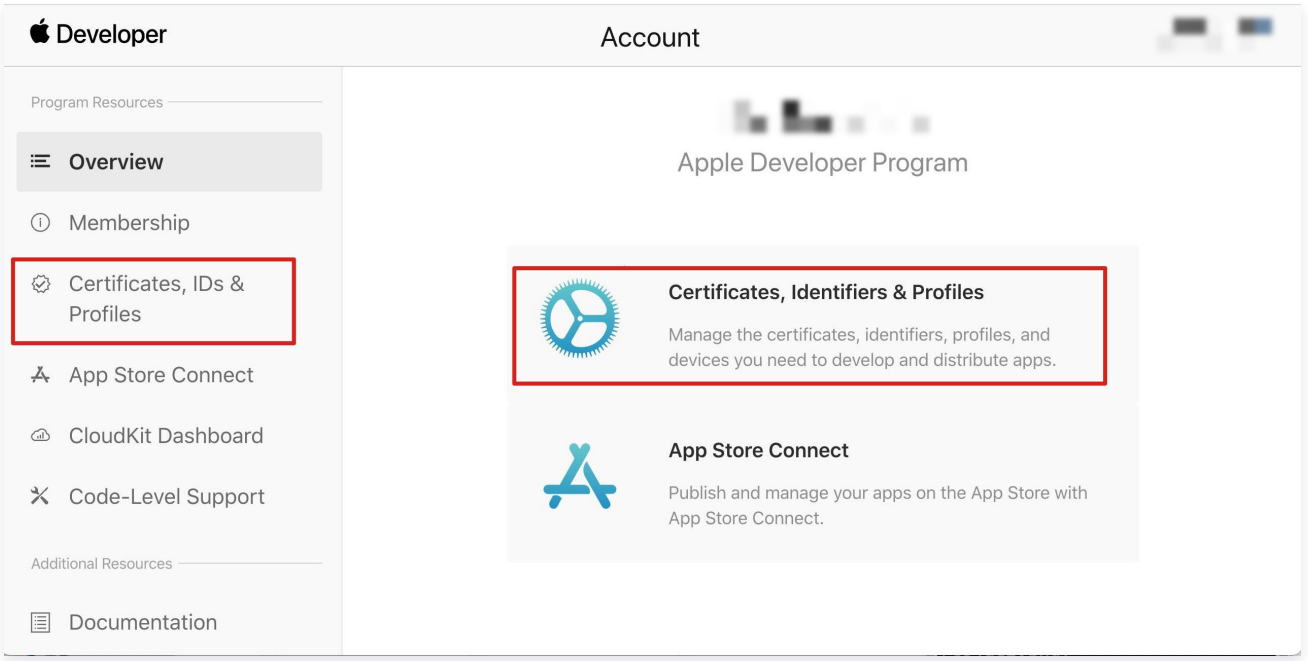
	证书类型	有效期和管理	安全性	灵动岛
p12 证书	p12 证书是一个包含公钥和私钥的二进制文件，用于基于证书的身份验证。它将公钥证书和私钥捆绑在一个文件中，扩展名为 .p12 或 .pfx。	p12 证书通常有一年的有效期，过期后需要重新生成和部署。每个应用程序都需要单独的 P12 证书来处理推送通知。	p12 证书使用基于证书的身份验证，需要在服务器上存储私钥。这可能会增加安全风险，因为私钥可能会被未经授权的用户访问。	不支持
p8 证书	p8 证书是一个 Auth Key（授权密钥），用于基于令牌的身份验证。它是一个包含私钥的文本文件，扩展名为 .p8。	p8 证书没有到期日期，因此您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

一、使用 p12 证书（传统推送证书）

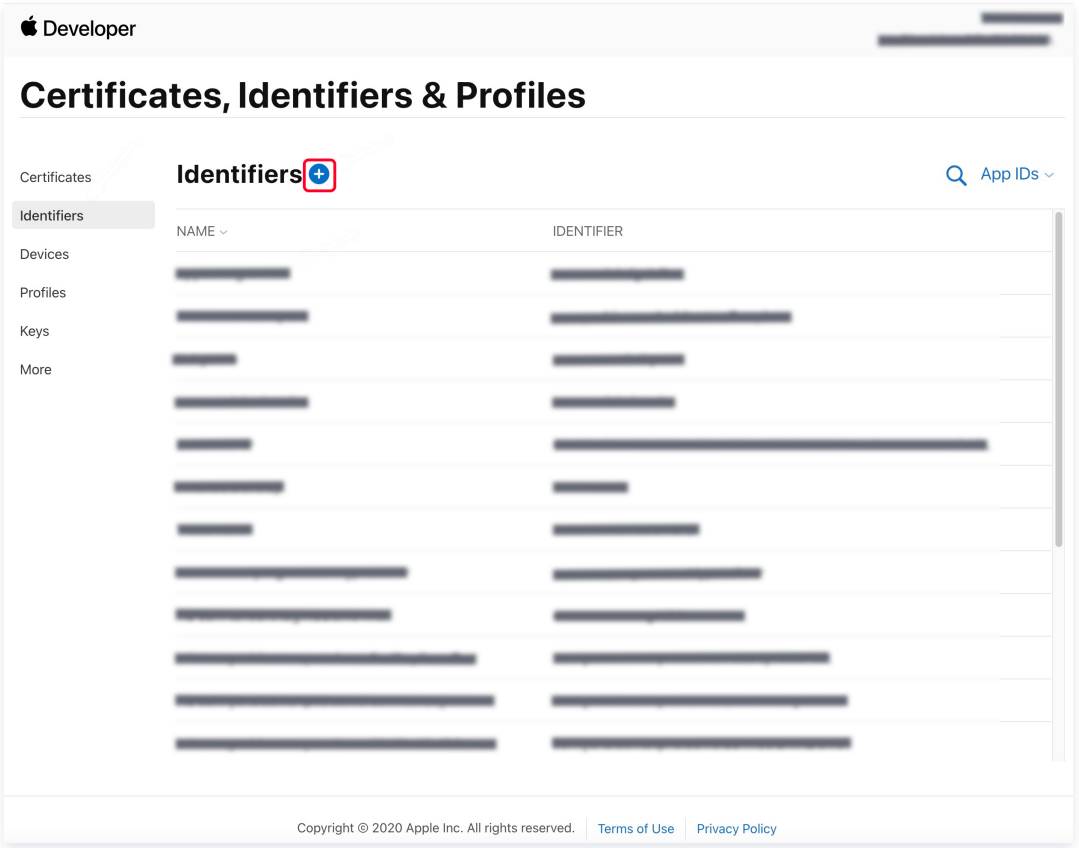
步骤1：申请 APNs 证书

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。



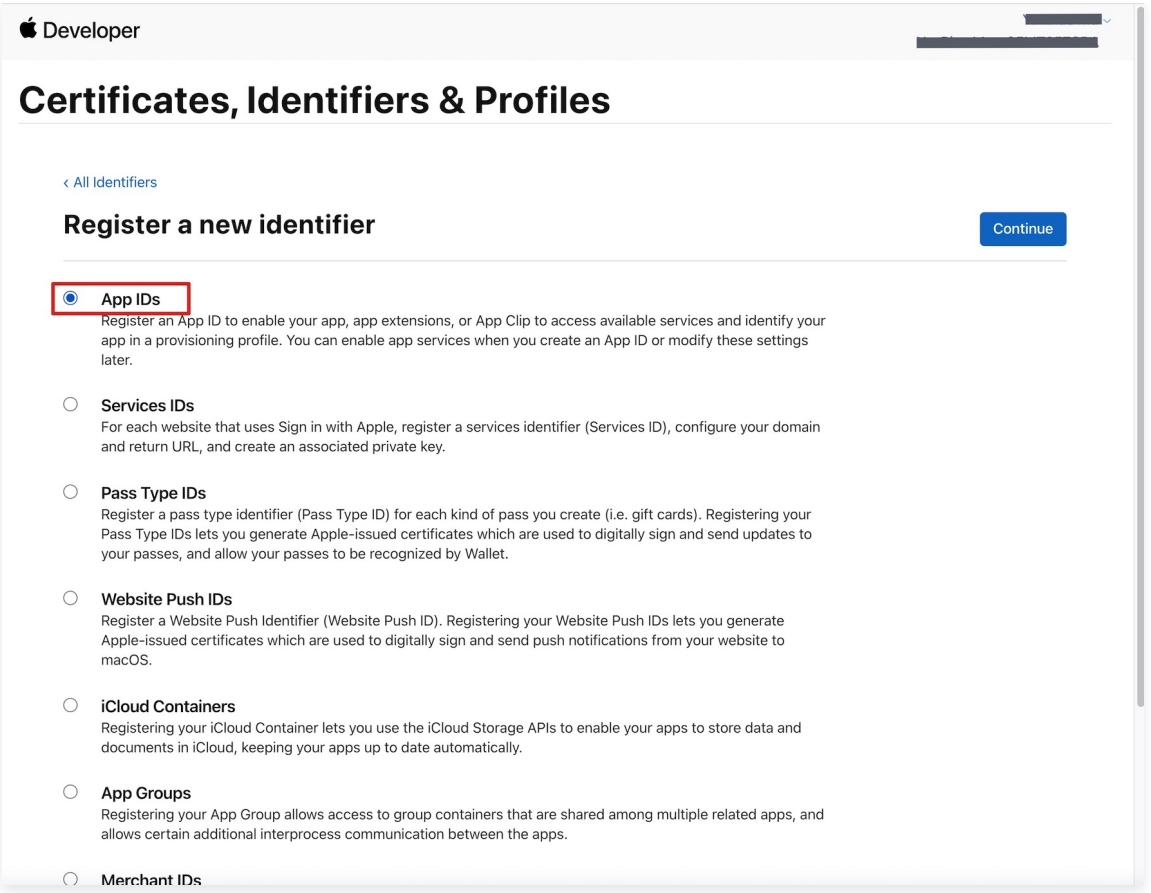
2. 单击 Identifiers 右侧的 +。



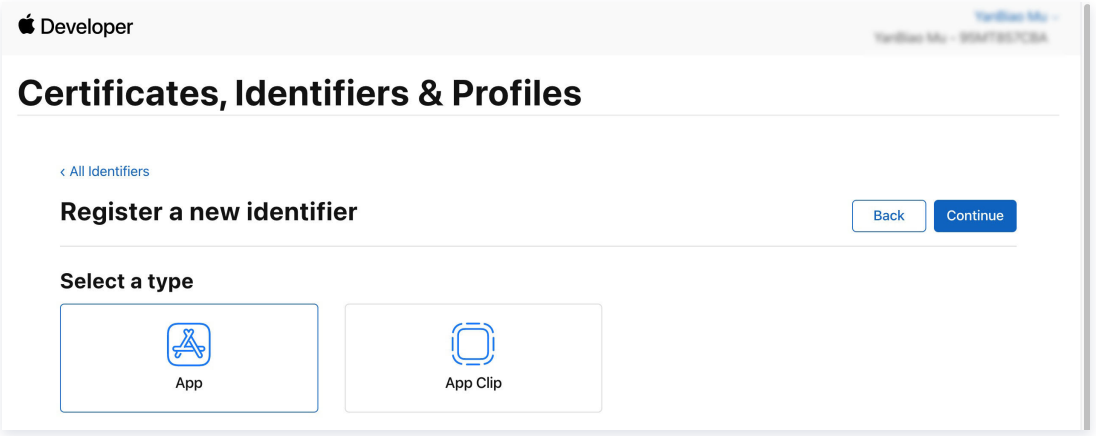
3. 您可以参见如下步骤新建一个 AppID，或者在您原有的 AppID 上增加 `Push Notification` 的 `Service`。

> **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。

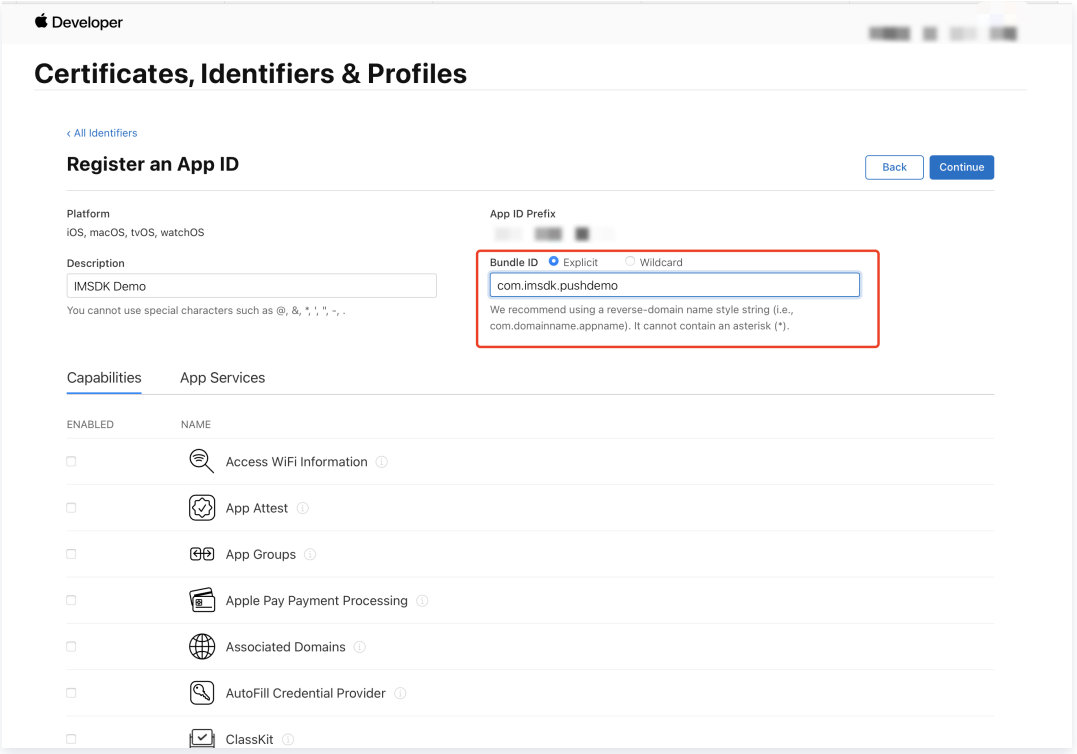
4. 勾选 **App IDs**，单击 **Continue** 进行下一步。



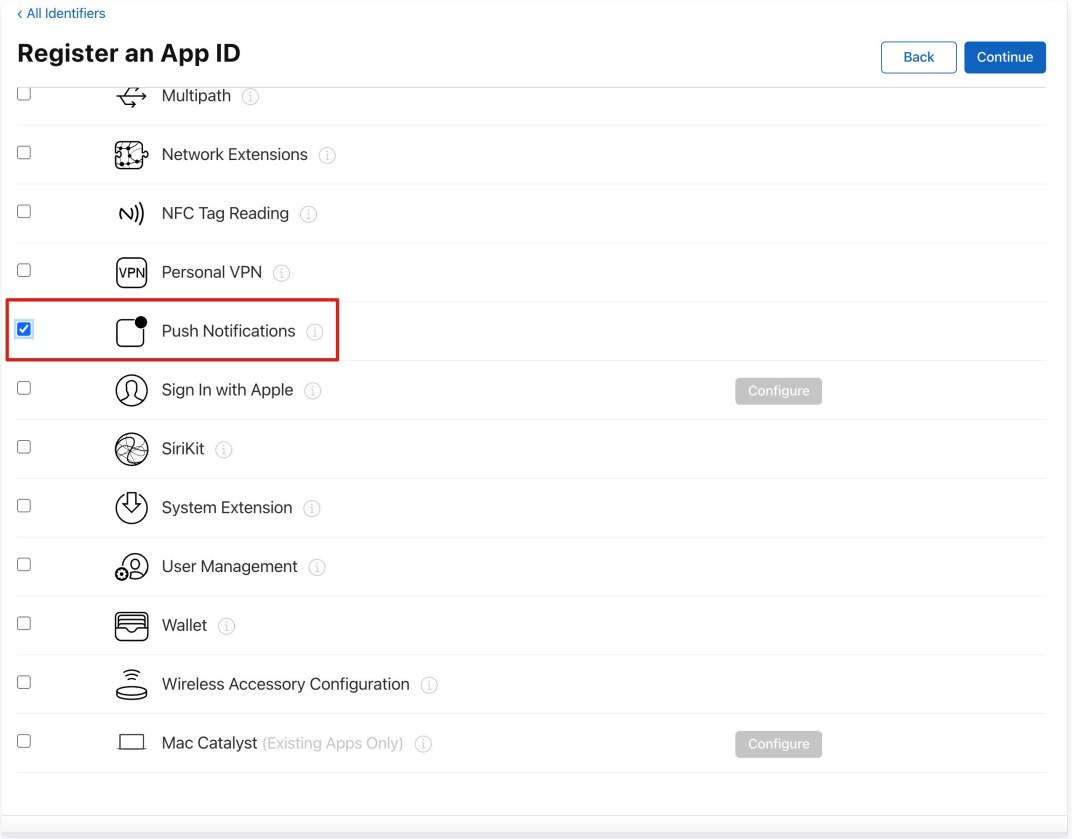
5. 选择 **App**，单击 **Continue** 进行下一步。



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

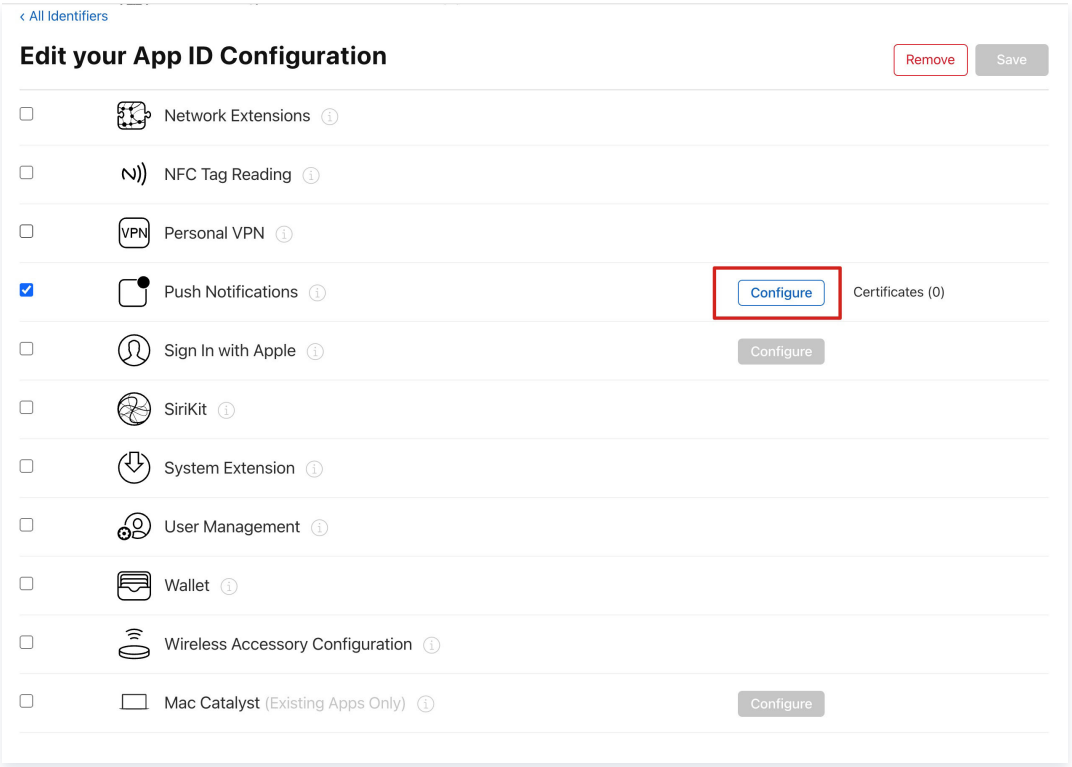


7. 勾选 **Push Notifications**，开启远程推送服务。

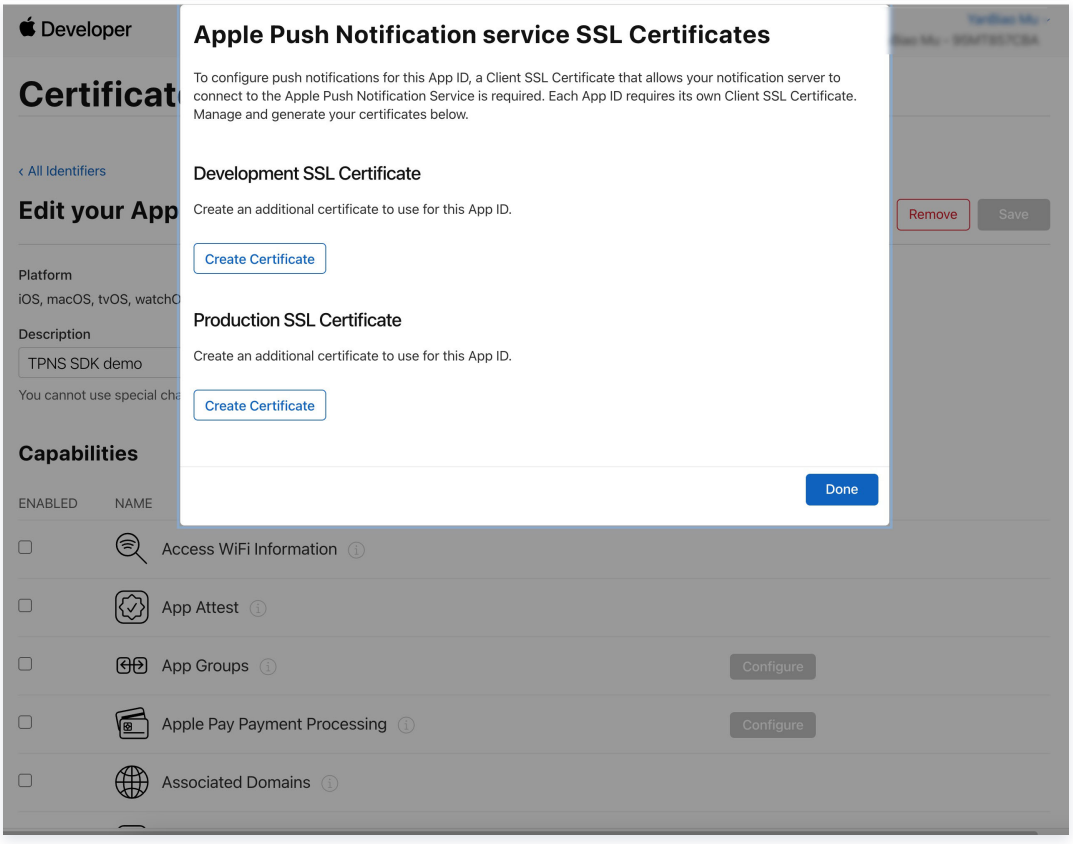


生成证书

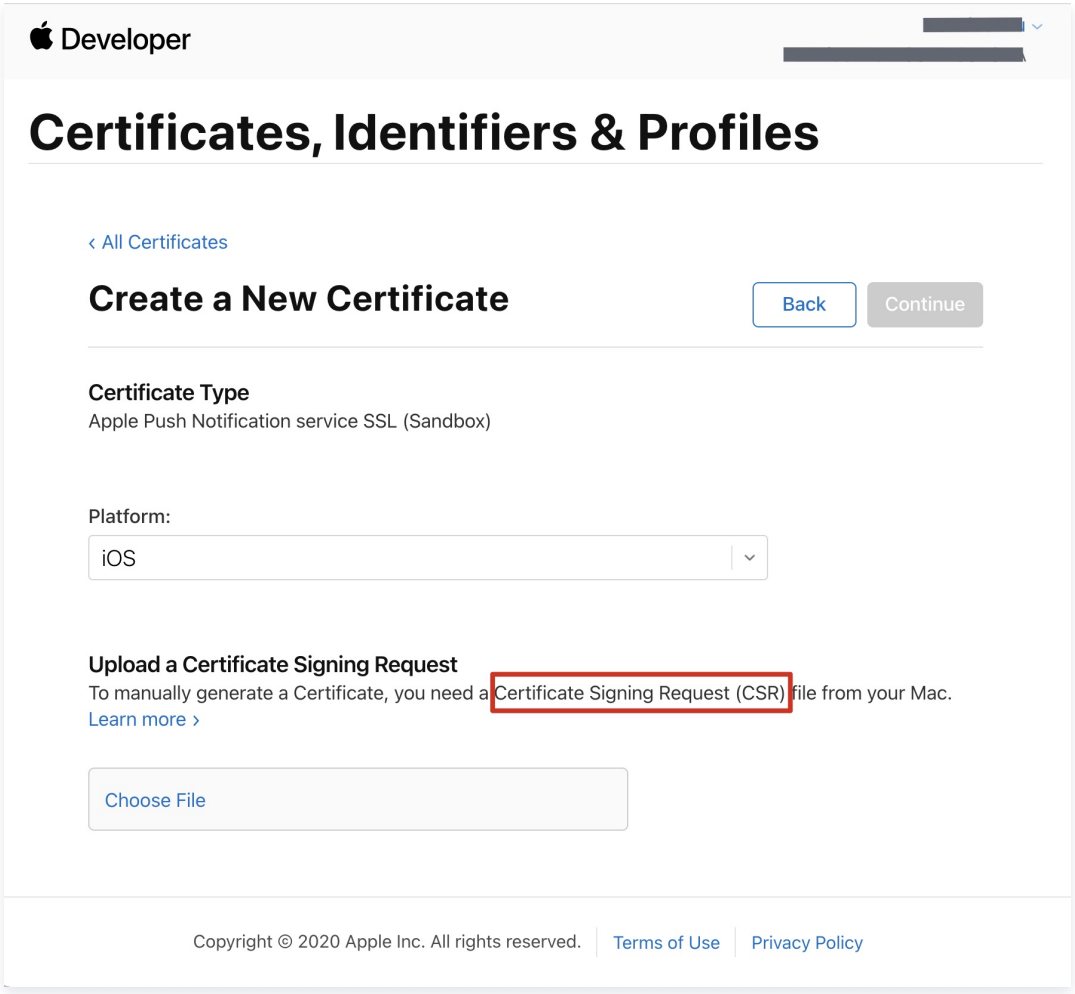
1. 选中您的 AppID，选择 **Configure**。



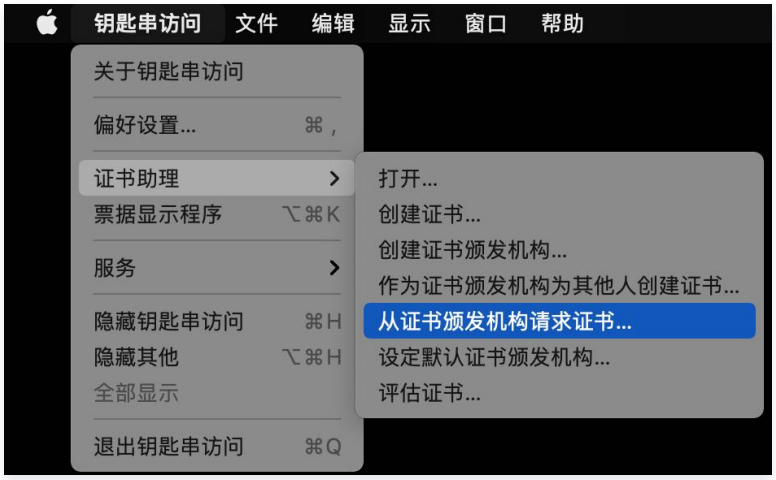
2. 可以看到在 **Apple Push Notification service SSL Certificates** 窗口中有两个 `SSL Certificate`，分别用于开发环境（Development）和生产环境（Production）的远程推送证书，如下图所示：



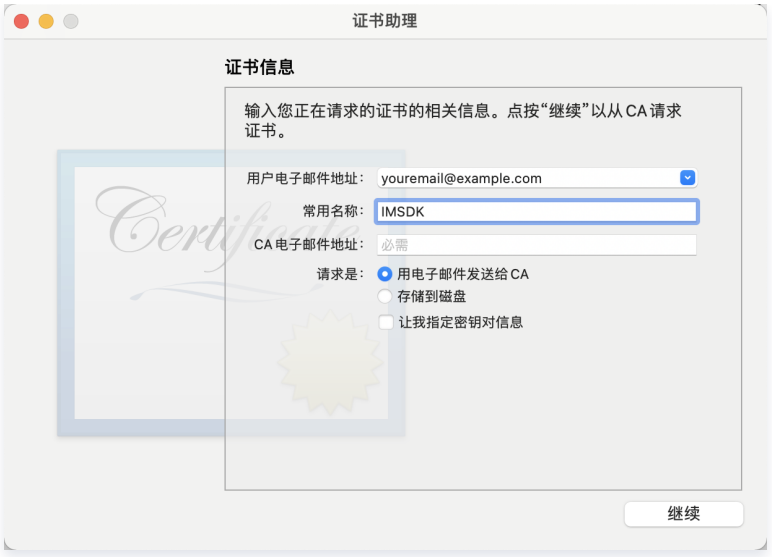
3. 我们先选择开发环境（Development）的 **Create Certificate**，系统将提示我们需要一个 Certificate Signing Request（CSR）。



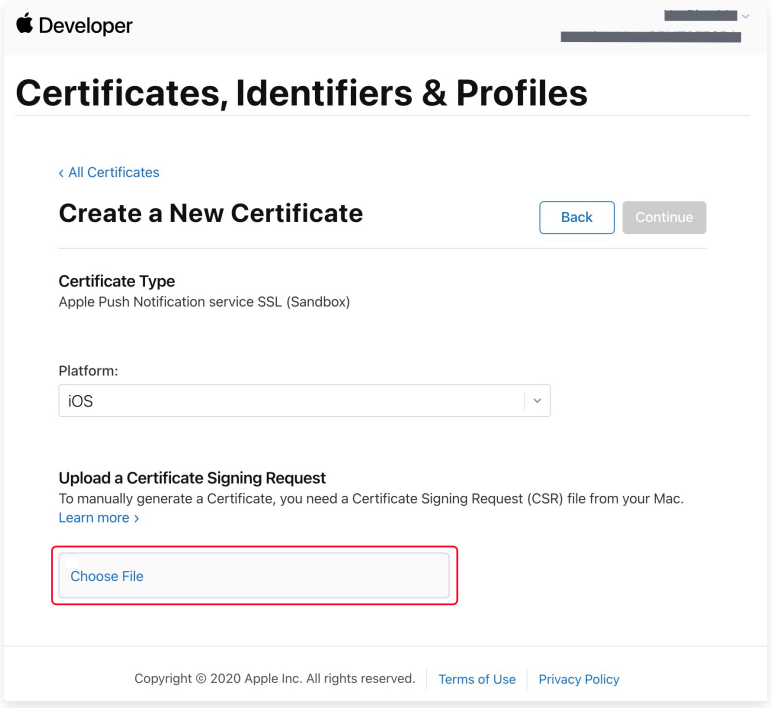
4. 在 Mac 上打开**钥匙串访问工具（Keychain Access）**，在菜单中选择**钥匙串访问 > 证书助理 > 从证书颁发机构请求证书**（`Keychain Access - Certificate Assistant - Request a Certificate From a Certificate Authority`）。



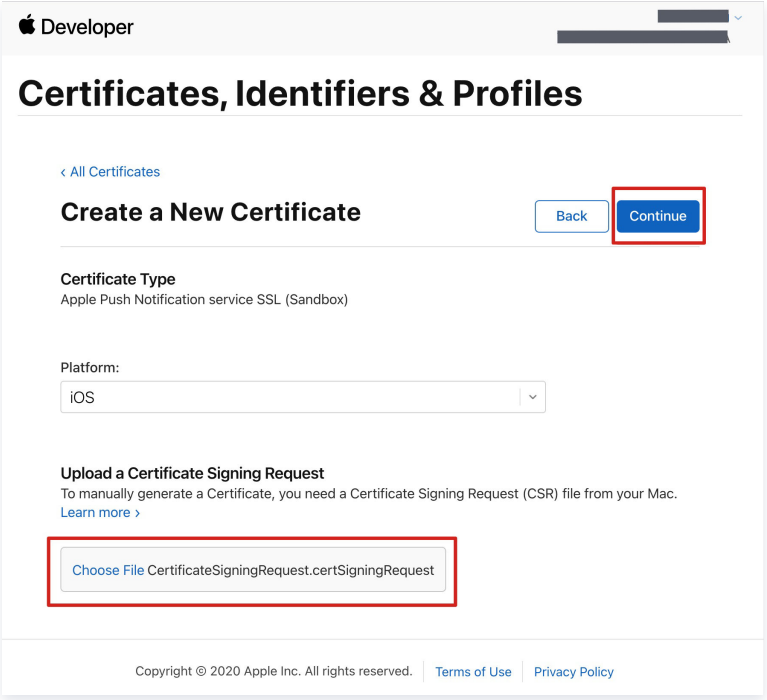
5. 输入用户电子邮件地址（您的邮箱）、常用名称（您的名称或公司名），选择**存储到磁盘**，单击**继续**，系统将生成一个 `*.certSigningRequest` 文件。



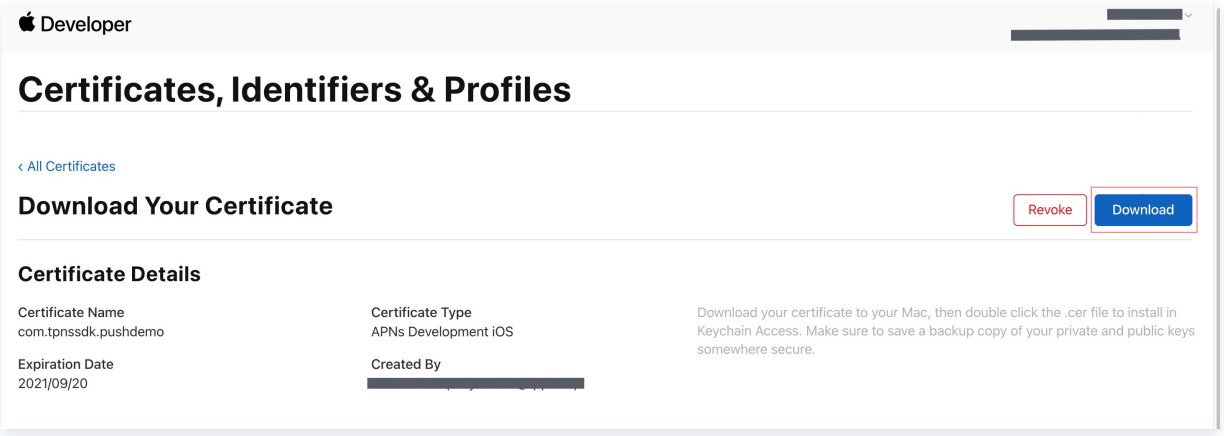
6. 返回上述 步骤3 中 Apple Developer 网站刚才的页面，单击 **Choose File** 上传生成的 `*.certSigningRequest` 文件。



7. 单击 **Continue**，即可生成推送证书。

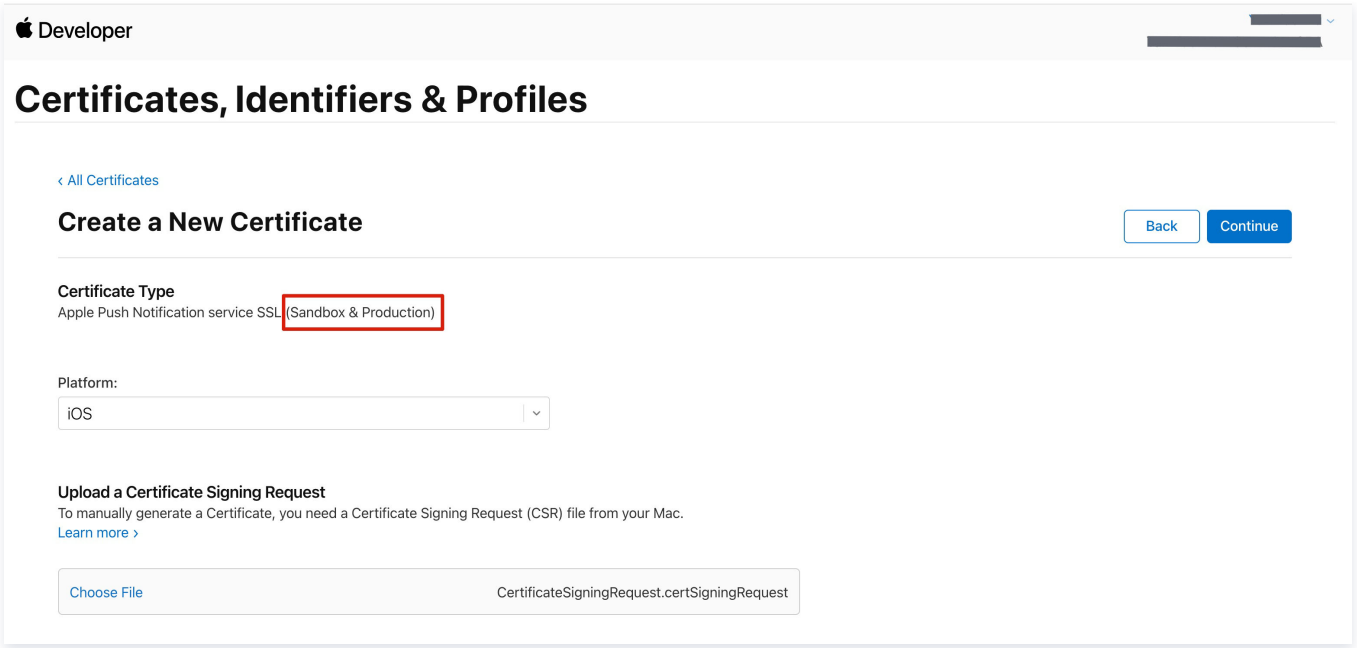


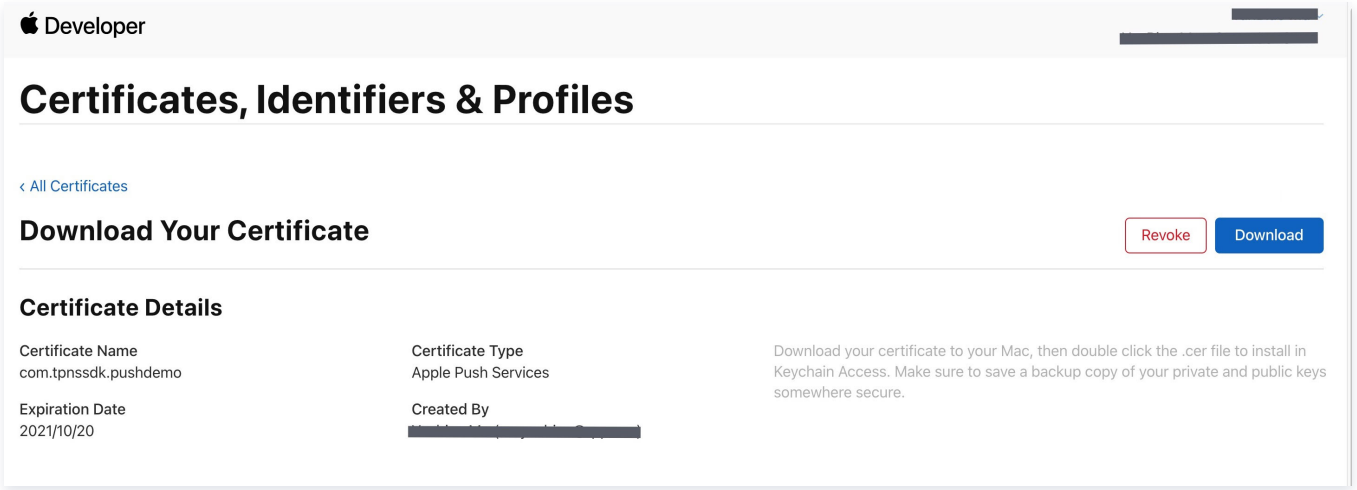
8. 单击 **Download** 下载开发环境的 `Development SSL Certificate` 到本地。



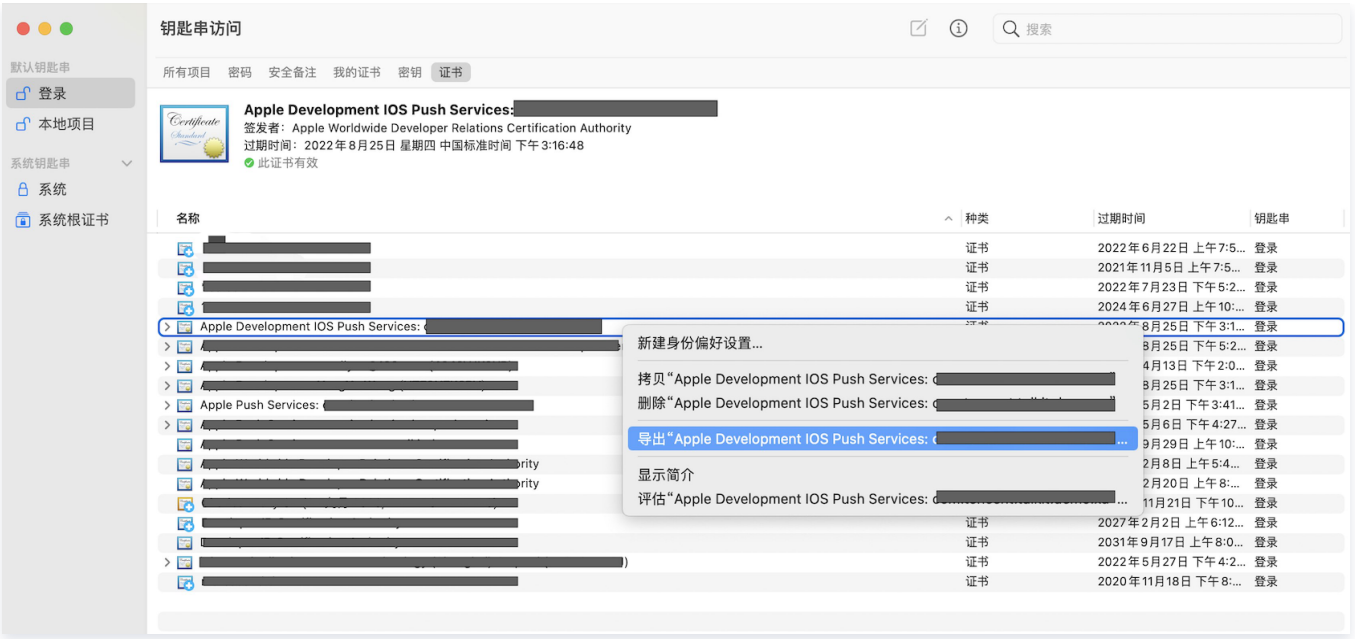
9. 再次按照上述步骤1 – 8，将生产环境的 `Production SSL Certificate` 下载到本地。

> **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。





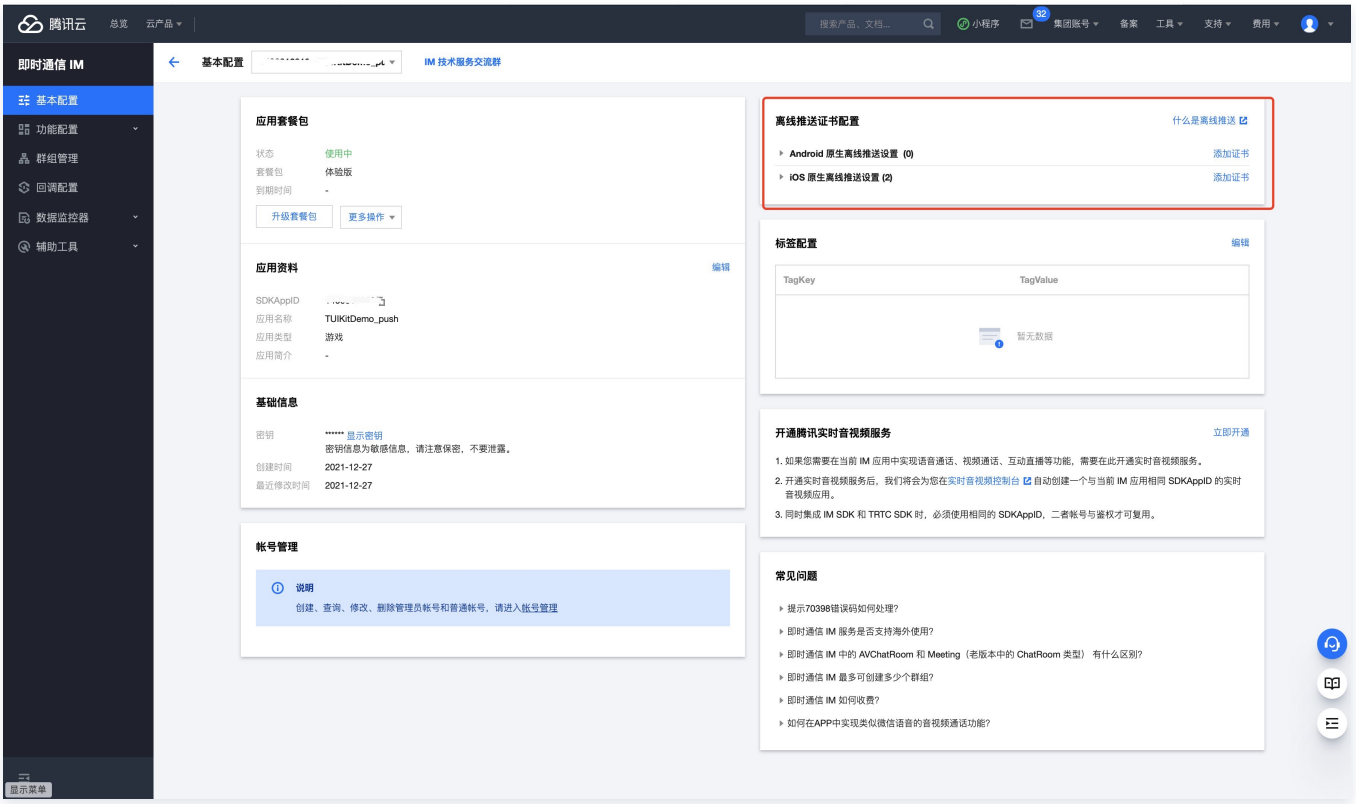
10. 双击打开下载的开发环境和生产环境的 `SSL Certificate`，系统会将其导入钥匙串中。
11. 打开钥匙串应用，在**登录 > 我的证书**，右键分别导出刚创建的开发环境（ `Apple Development IOS Push Service` ）和生产环境（ `Apple Push Services` ）的 `p12` 文件。



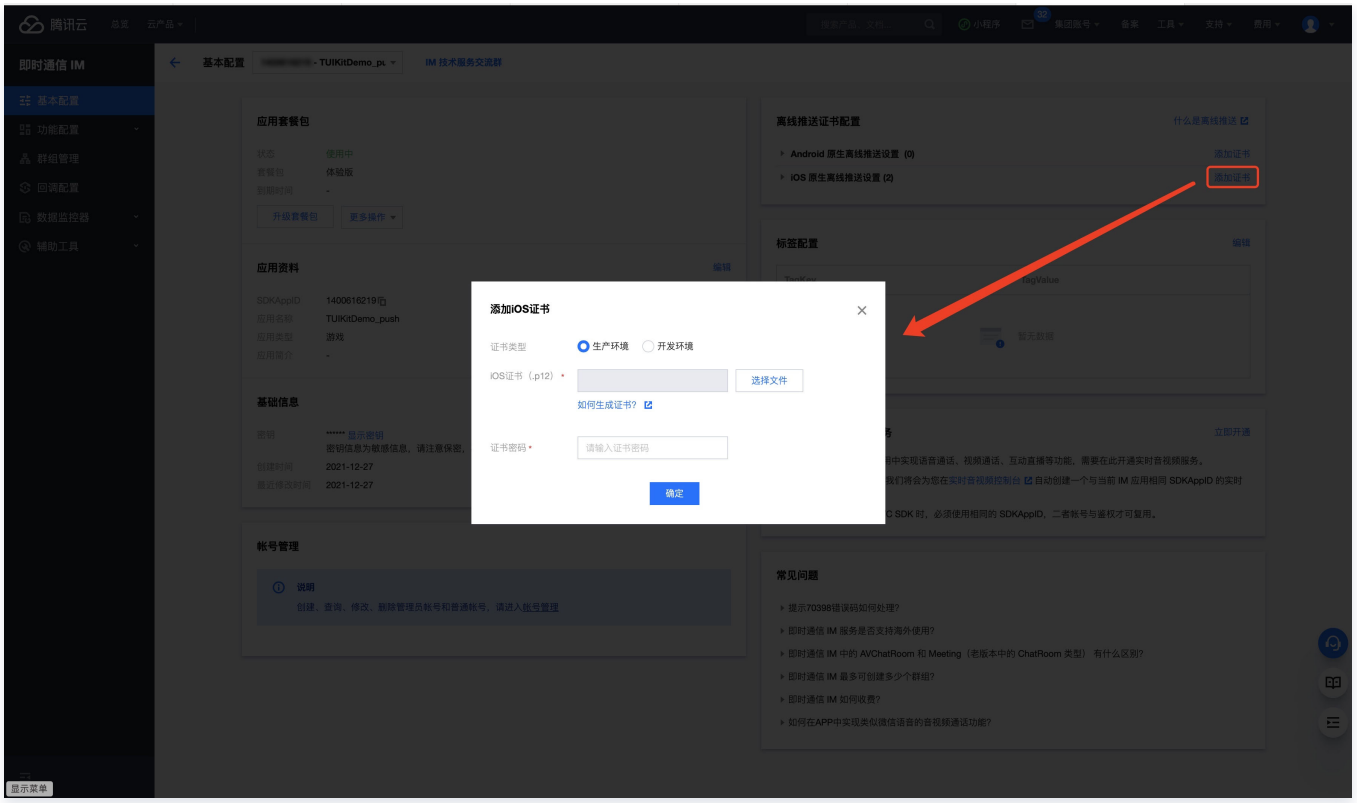
> **注意**
> 保存 `.p12` 文件时，请务必为其设置密码。

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。



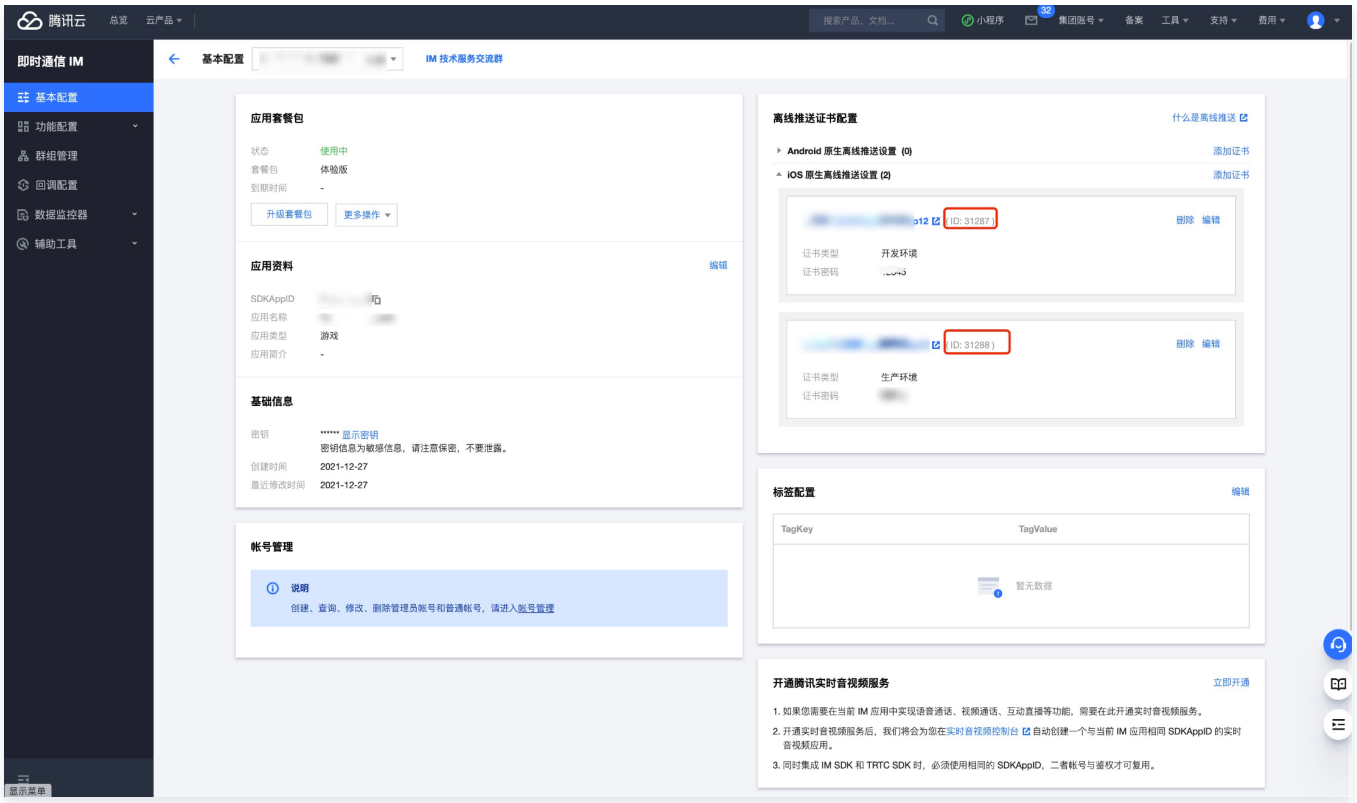
3. 单击 **iOS 原生离线推送设置**右侧的**添加证书**。
4. 选择证书类型，上传 iOS 证书（p.12），设置证书密码，单击**确认**。



说明：

- 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。

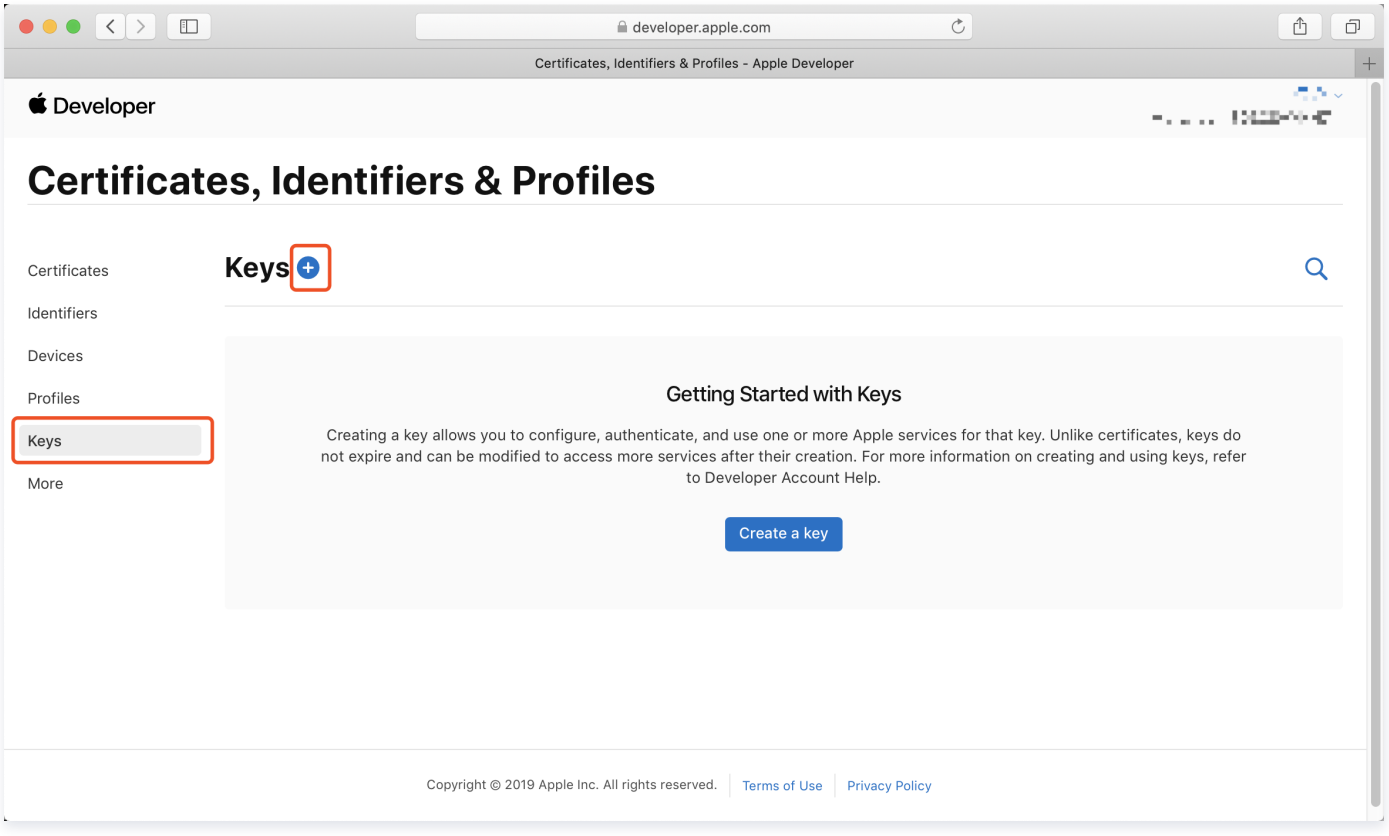


二、使用 p8 证书（支持灵动岛推送）

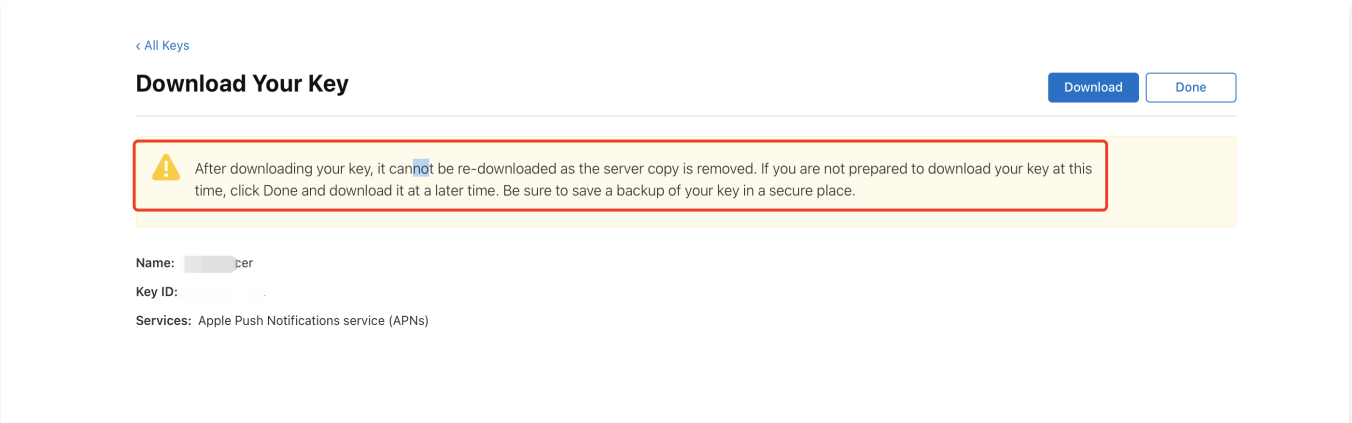
p8 证书：p8 证书没有到期日期，因此您无需担心证书过期。此外，使用 p8 证书可以简化证书管理，因为您可以使用一个p8 证书为多个应用程序提供推送通知服务。另外，p8 证书支持灵动岛推送。

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** 。

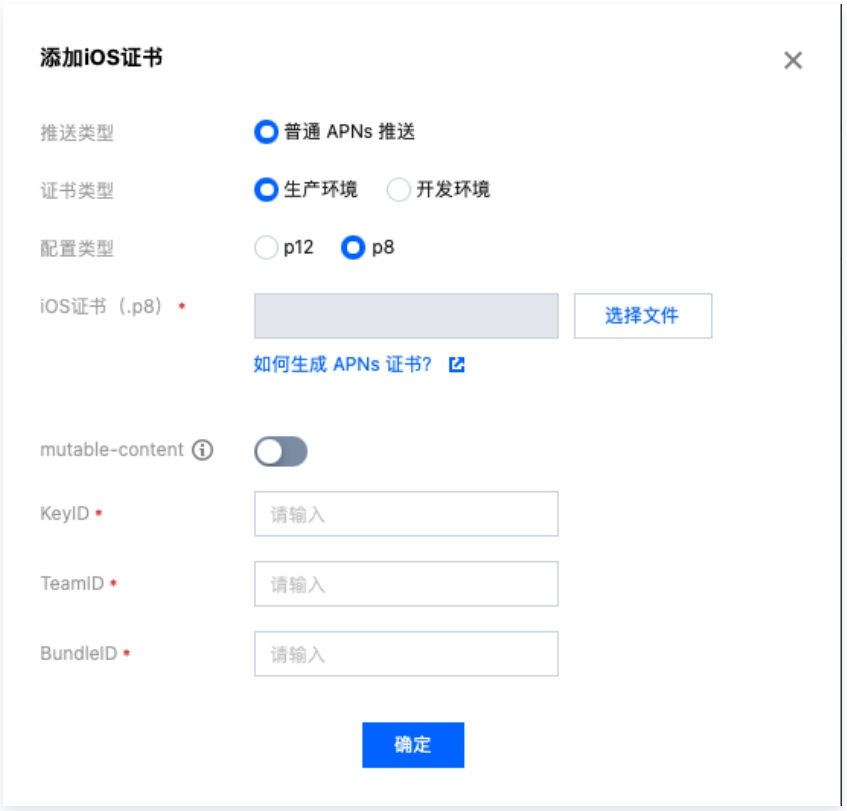


确认并生成密钥：在确认页面核对您的密钥信息，然后单击 **Register** 。接下来，您将看到一个页面提示您下载密钥。单击 **Download**，将生成的 .p8 文件保存到您的计算机上。

> **说明：**
> - 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

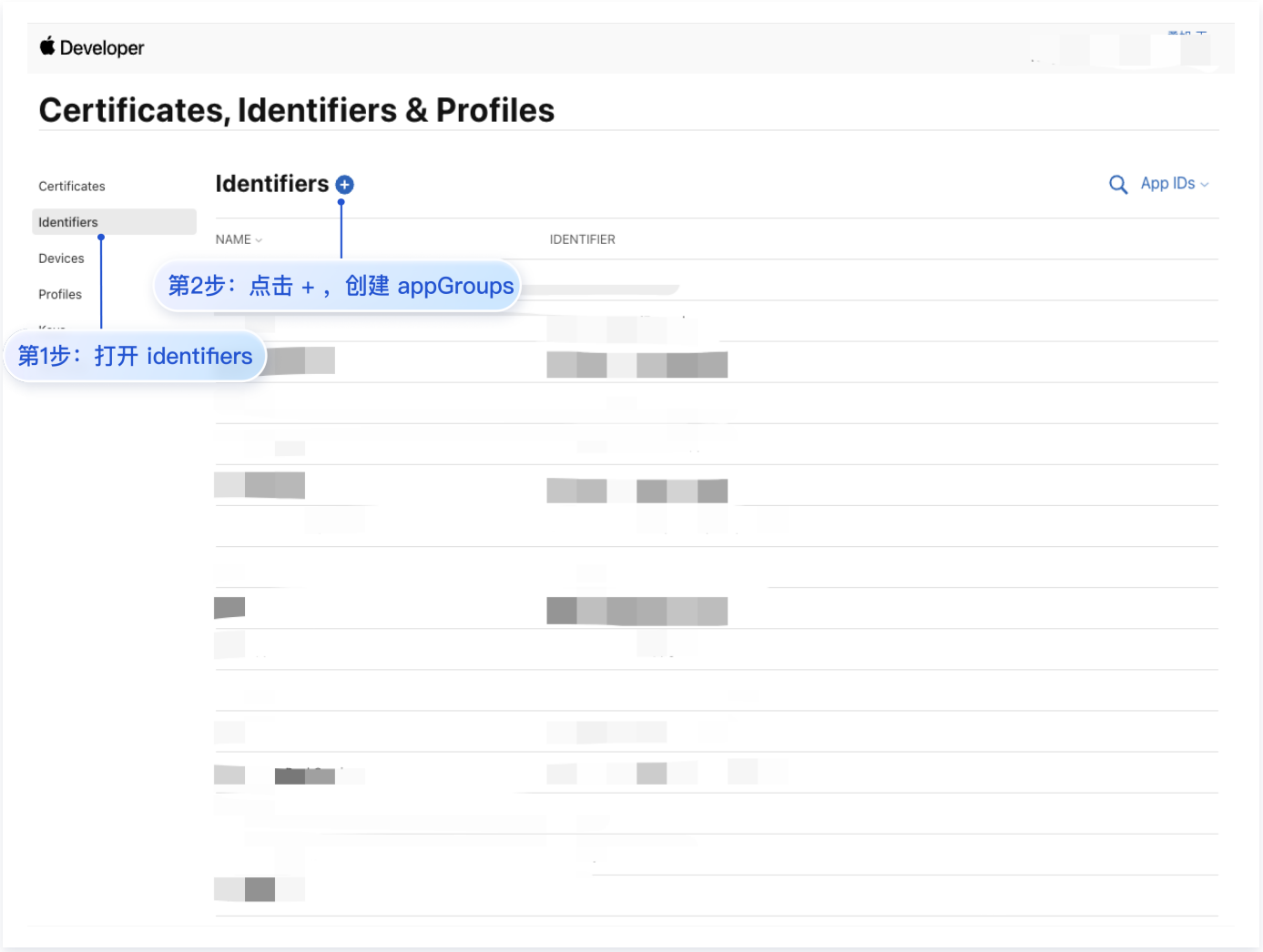


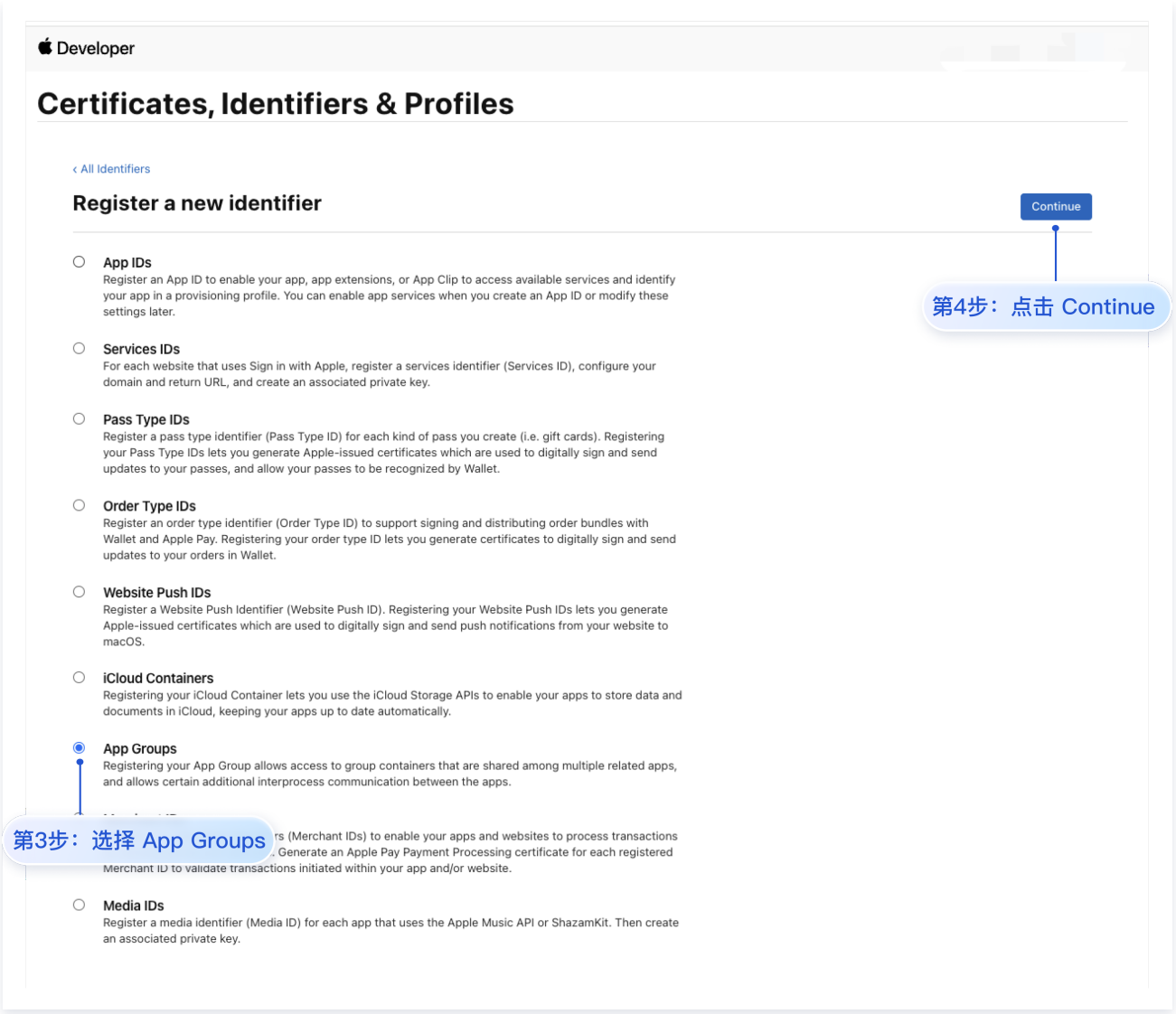
说明：

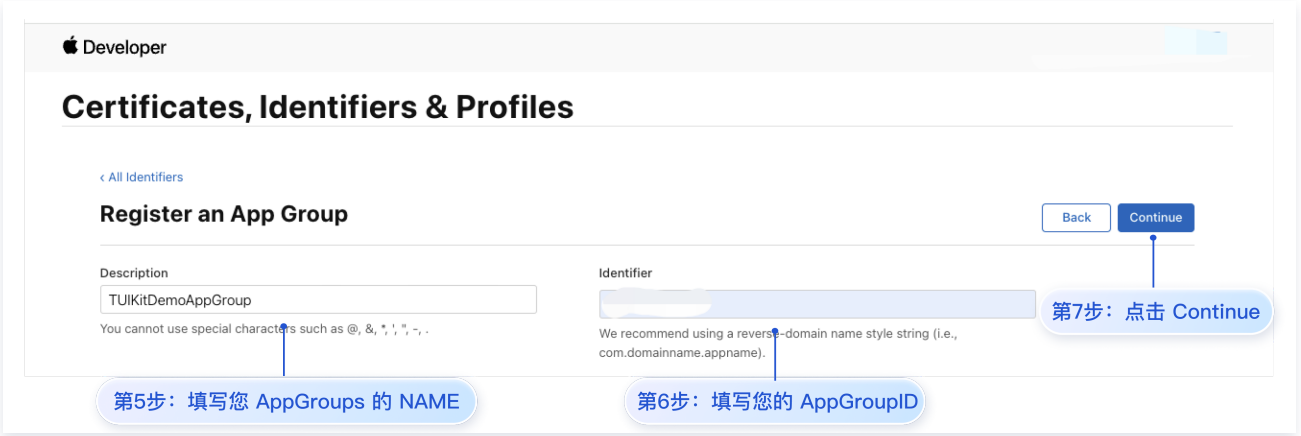
- **Key ID**：这是您的 APNs Auth Key 的唯一标识符。当您在 Apple Developer Center 创建一个新的 APNs Auth Key 时，系统会为您生成一个 Key ID。您可以在 "Certificates, Identifiers & Profiles" 部分的 "Keys" 中找到它。
- **Team ID**：这是您的开发者账户的唯一标识符。您可以在 Apple Developer Center 的账户详情页面找到它。点击右上角的 "Membership"，在 "Membership Details" 部分可以找到您的 Team ID。
- **Bundle 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

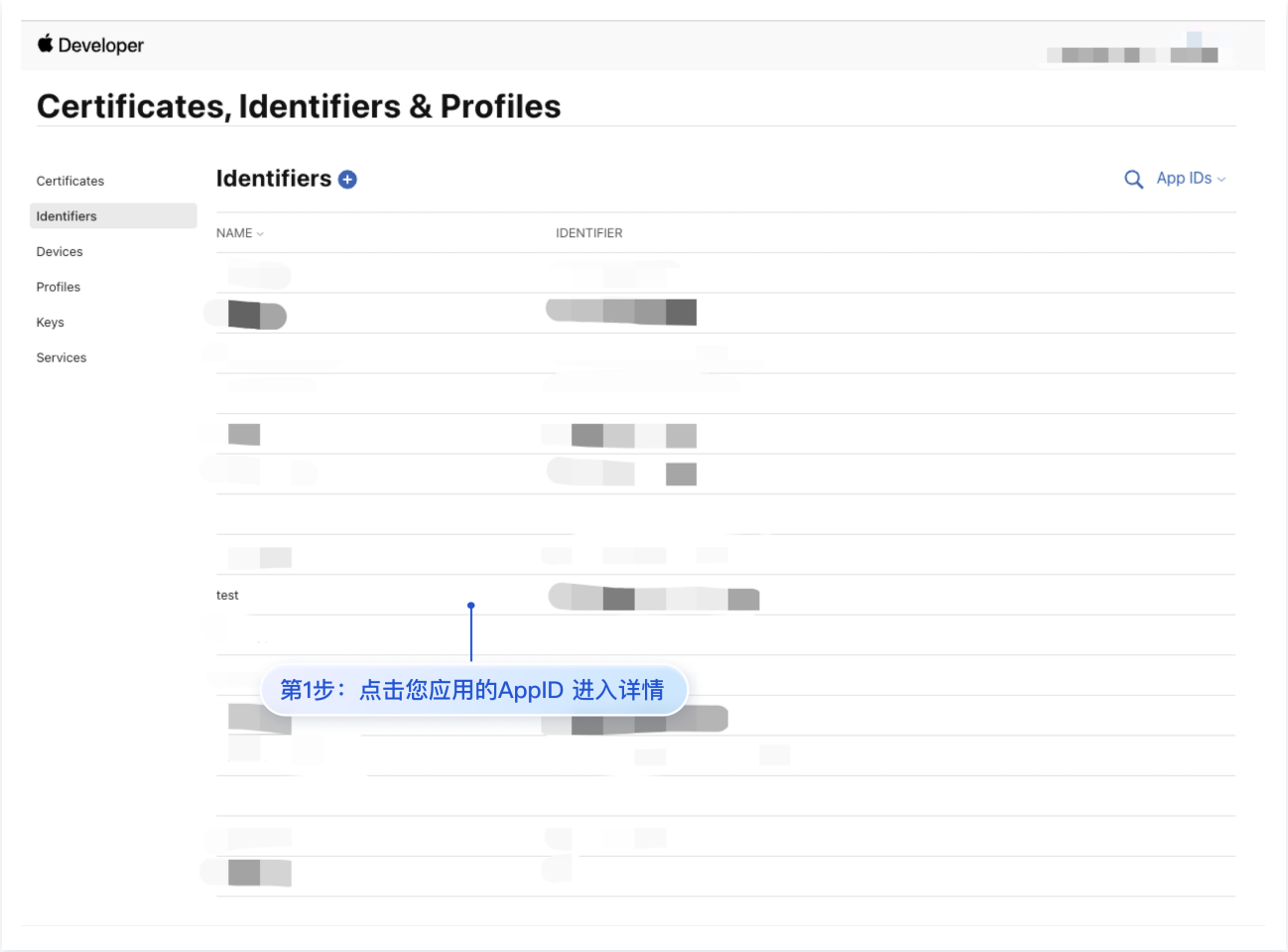
步骤1：登录苹果开发者中心网站，进入【identifiers】->【App Groups】创建 AppGroups。

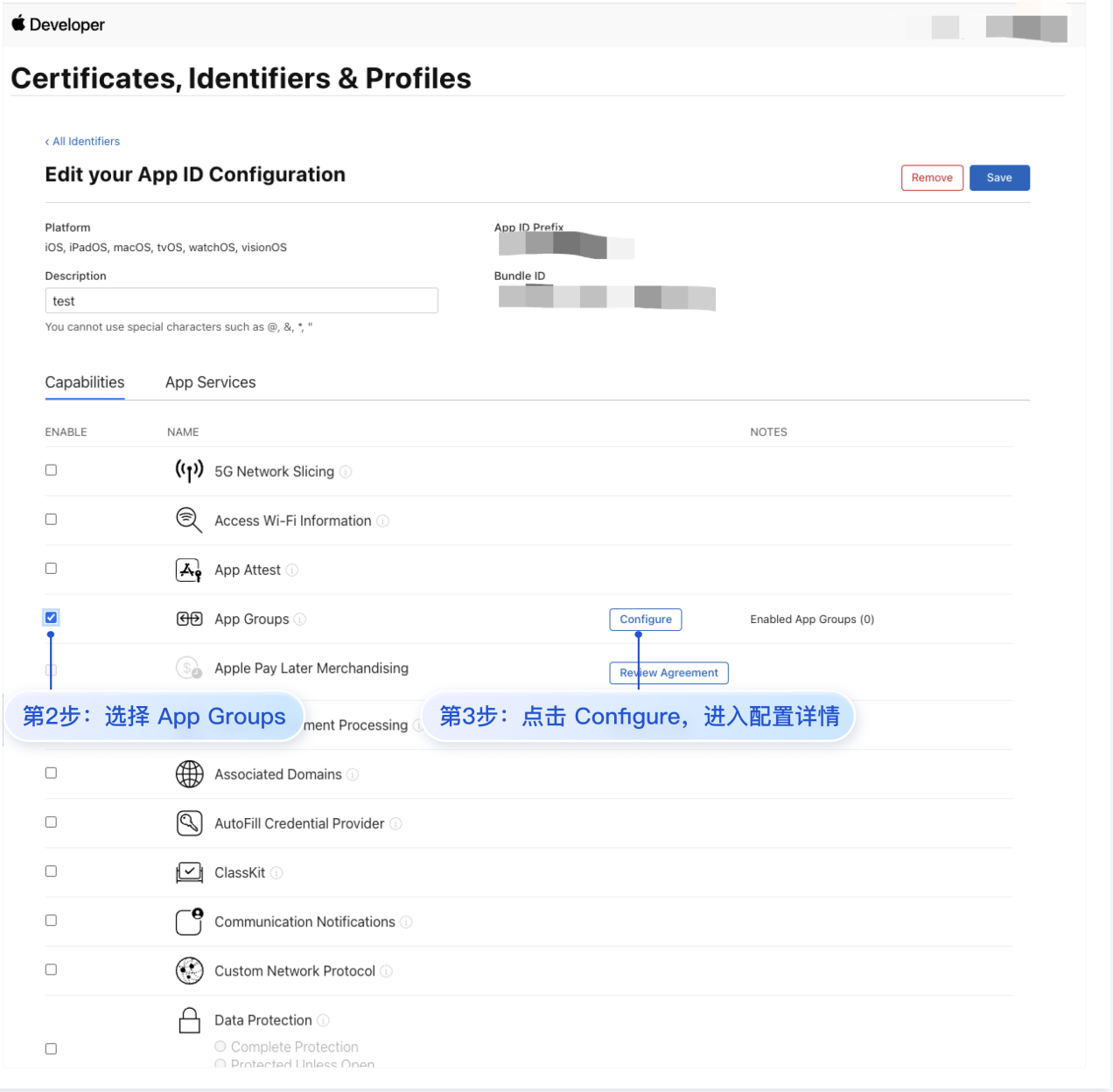

Developer
Certificates, Identifiers & Profiles
Certificates
Identifiers
Devices
Profiles
Identifiers
App IDs
NAME
IDENTIFIER
第2步：点击 + ，创建 appGroups
第1步：打开 identifiers

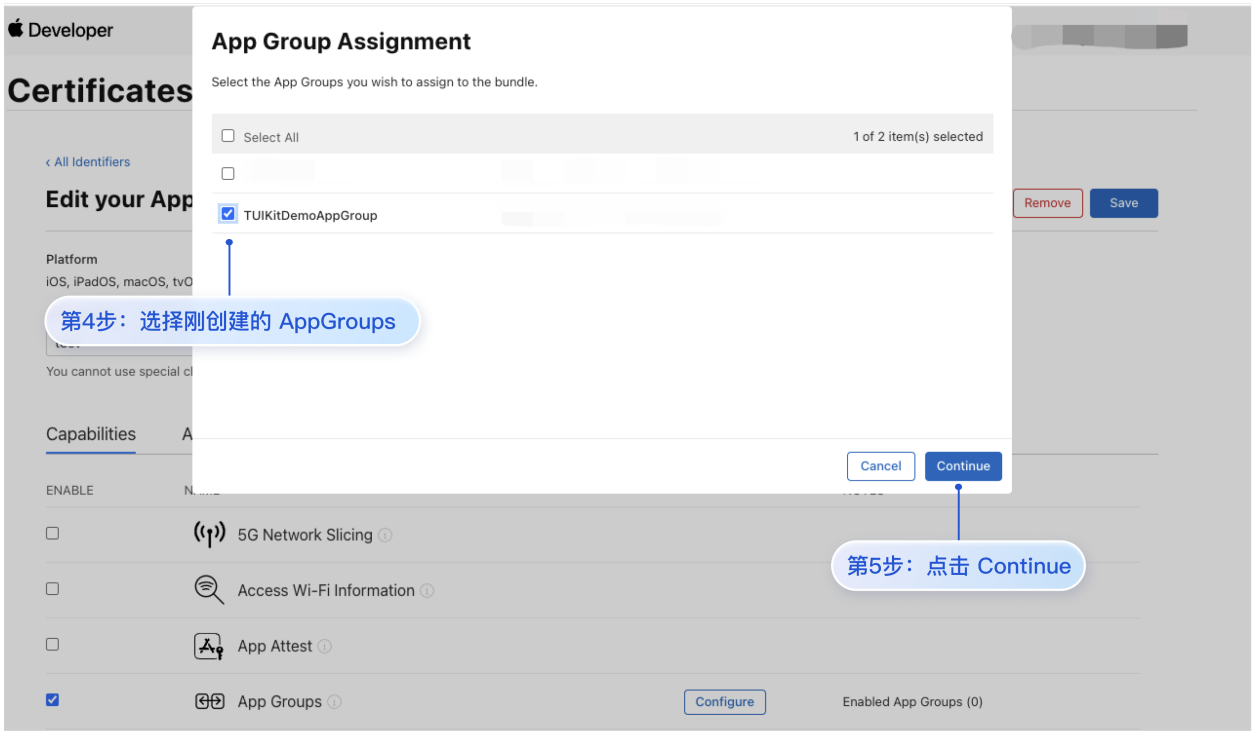




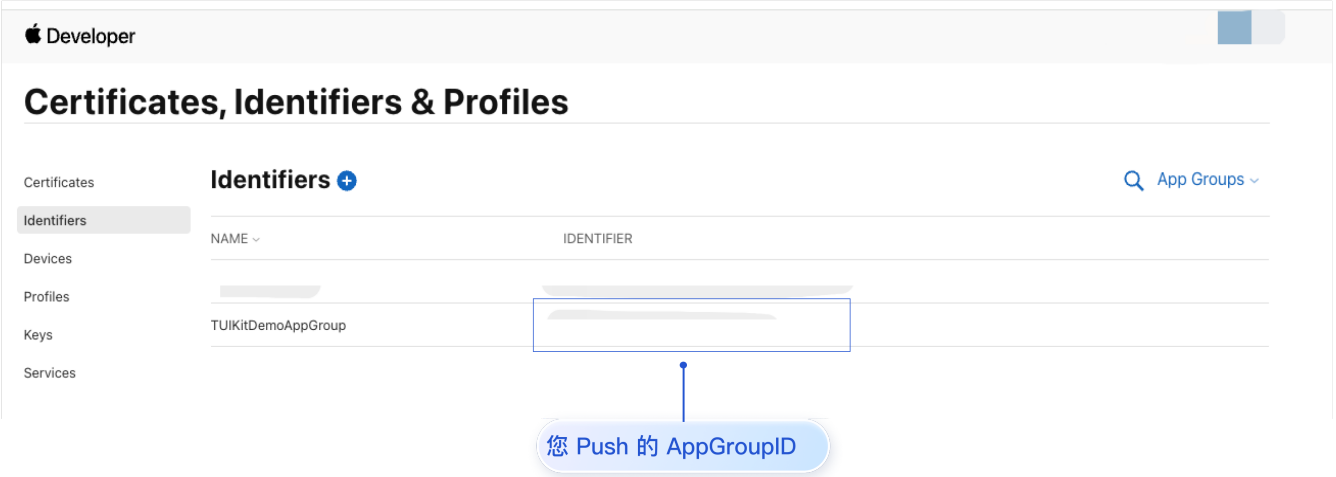
步骤2：绑定使用 TencentCloud-TIMPush 应用的 appID 到 AppGroups。


Developer
Certificates, Identifiers & Profiles
Certificates
Identifiers
Devices
Profiles
Keys
Services
Identifiers
App IDs
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
Remove
Save
Platform
iOS, iPadOS, macOS, tvOS, watchOS, visionOS
App ID Prefix
Description
test
You cannot use special characters such as @, &, *, "
Bundle ID
Capabilities
App Services
ENABLE
NAME
NOTES
5G Network Slicing
Access Wi-Fi Information
App Attest
App Groups
Configure
Enabled App Groups (0)
Apple Pay Later Merchandising
Review Agreement
第2步：选择 App Groups
第3步：点击 Configure，进入配置详情
Associated Domains
AutoFill Credential Provider
ClassKit
Communication Notifications
Custom Network Protocol
Data Protection
Complete Protection



步骤3. 获取您的 TIMPushAppGroupID。



微信小程序多端框架

最近更新时间：2025-03-17 17:25:43

微信小程序多端框架 推送服务目前推送支持小米、华为、荣耀、OPPO、vivo、魅族、一加、realme、iQOO 、FCM 和 Apple 等厂商通道。

Android

注册应用到厂商推送平台

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

观看视频

> **注意：**
> 通知栏推送：应用需在小米软件商店上架。

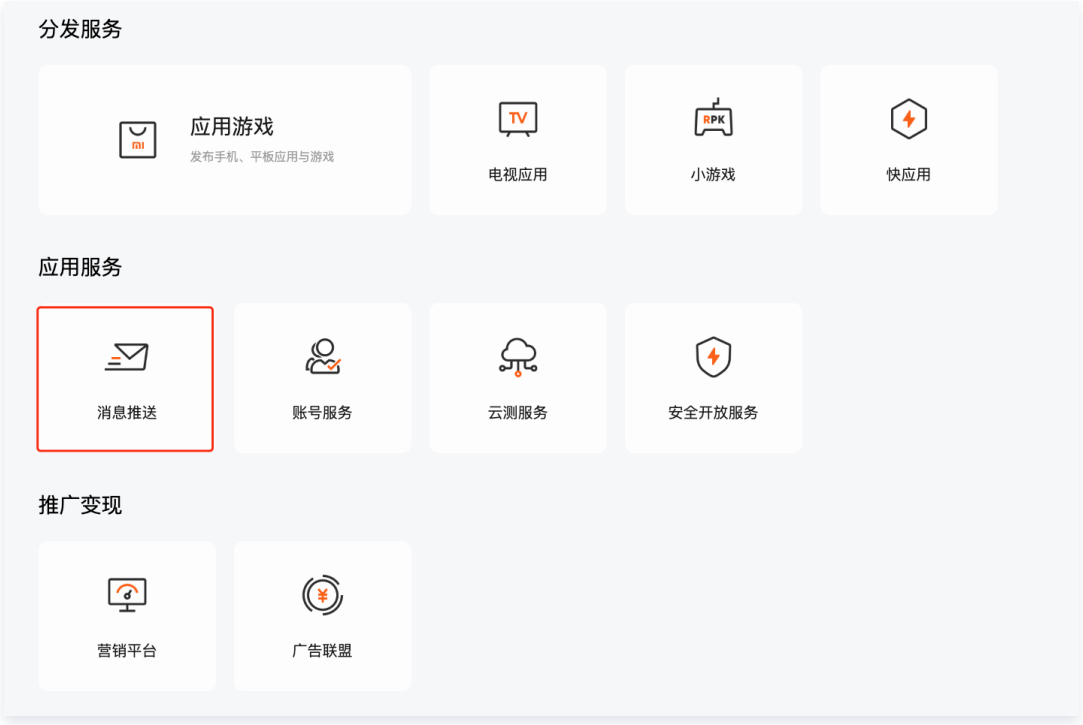
步骤1：注册小米开发者账号

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 。

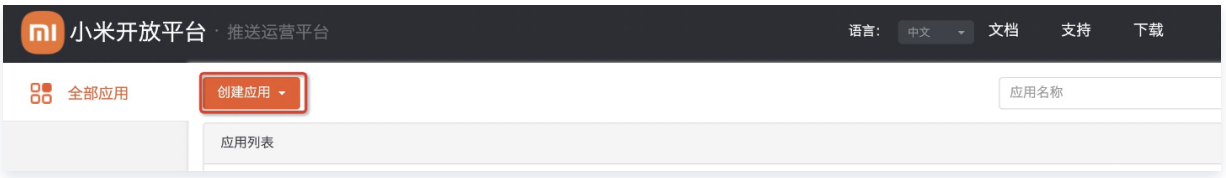
步骤2：创建应用

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**。



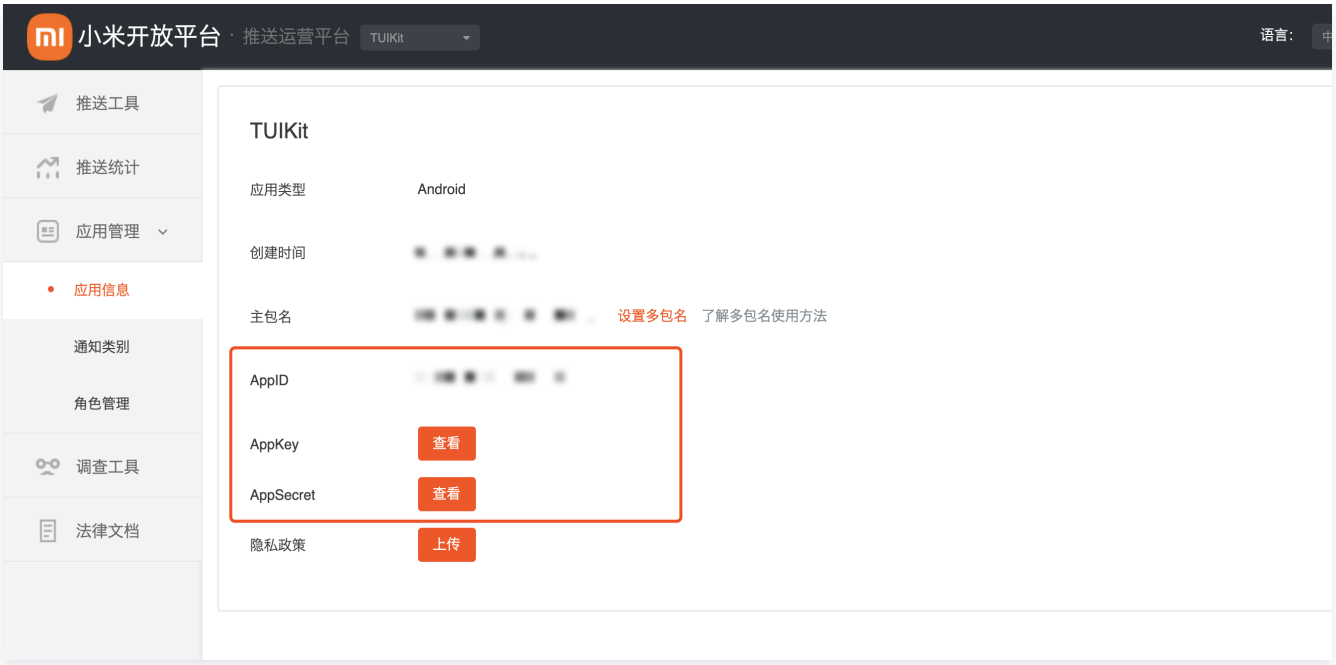
注意：
应用包名与插件应用包名保持一致。

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。

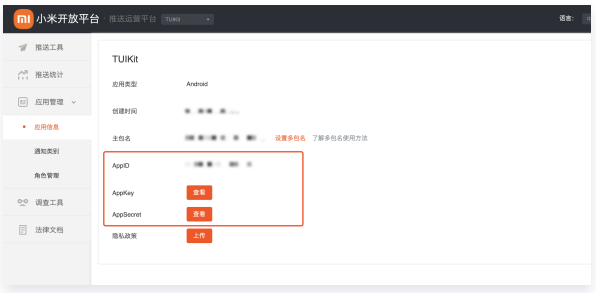
步骤4：查看获取应用信息

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。该配置用于派发单击后离线推送插件的事件监听，不可以直接配置应用内页面的跳转。 

华为

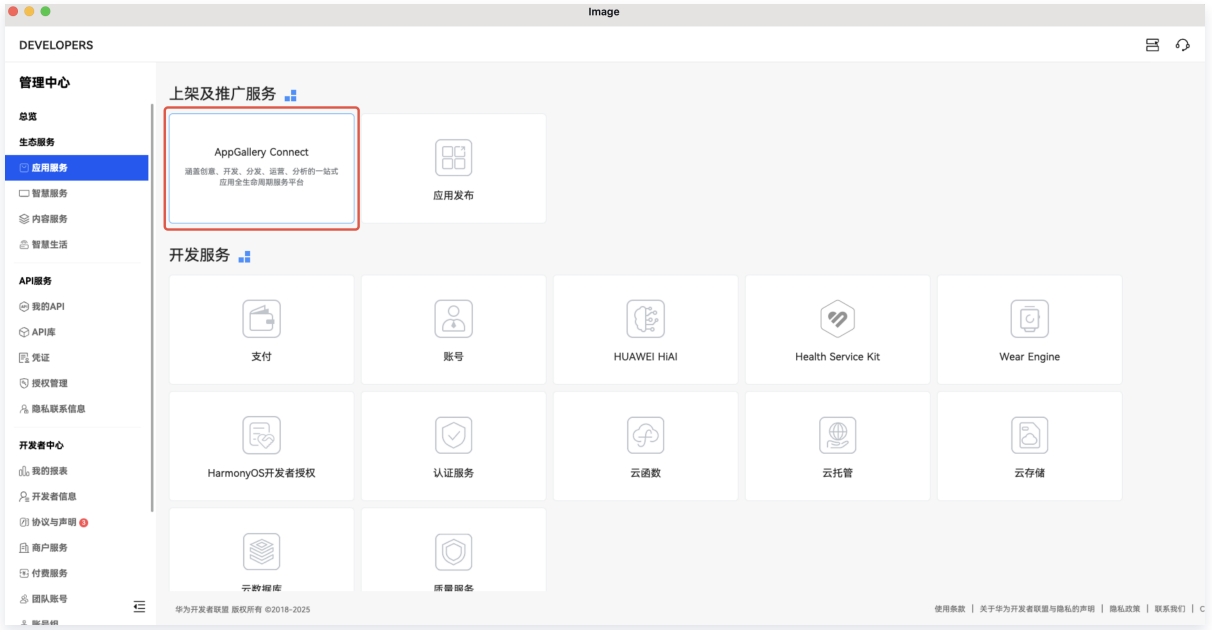
观看视频

步骤1：注册华为开发者账号

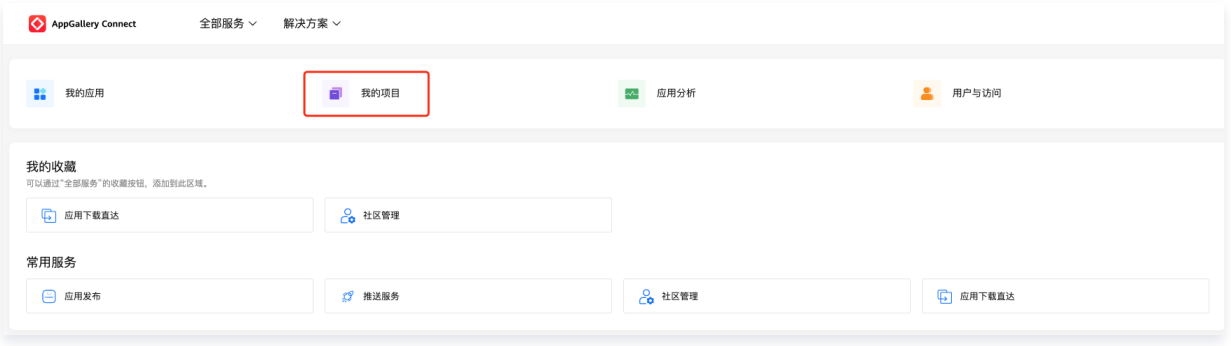
进入 华为开发者联盟，注册华为开发者账号，详情请参见 注册账号。

步骤2：创建应用

1. 在 华为管理中心 的**应用管理中**，单击 **AppGallery Connect**，进入应用管理中心。



2. 单击**我的项目**，添加一个新的项目。



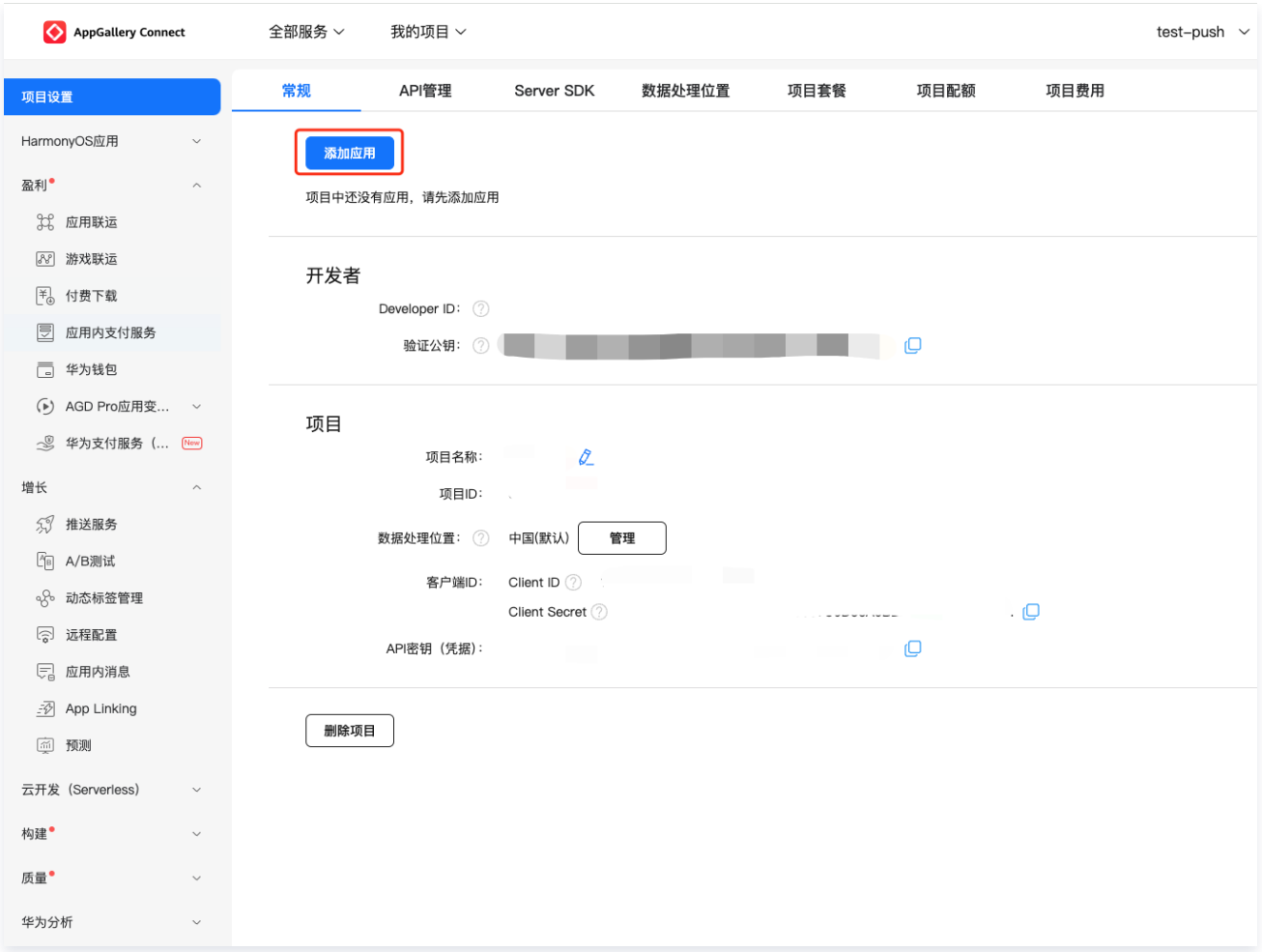
3. 在**项目设置**栏单击**推送服务 > 立即开通**。

4. **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 Android 原生资源目录中，默认在 miniapp/android/nativeResources/app/ 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 Secret 对应 AppSecret。
> - **应用内指定界面**链接，不可以修改。该配置用于派发单击后离线推送插件的事件监听，不可以直接配置应用内页面的跳转。

OPPO

观看视频

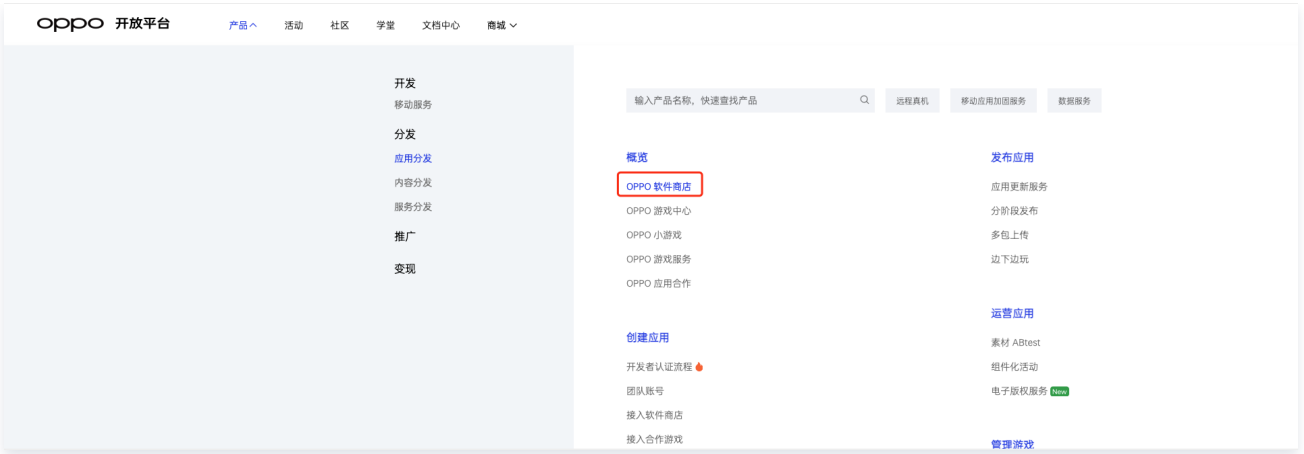
> **注意：**
> - 通知栏推送：应用需在 OPPO 软件商店上架；
> - 通知栏推送测试权限：每天仅可推送1000条消息，限测试使用。应用上架后需重新申请“通知栏推送”权限，以获得正常消息推送数量；
> - 平台将会在1个工作日内返回审核结果，开发者可以在申请页面查看审核结果，其他问题可咨询开放平台客服。

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。

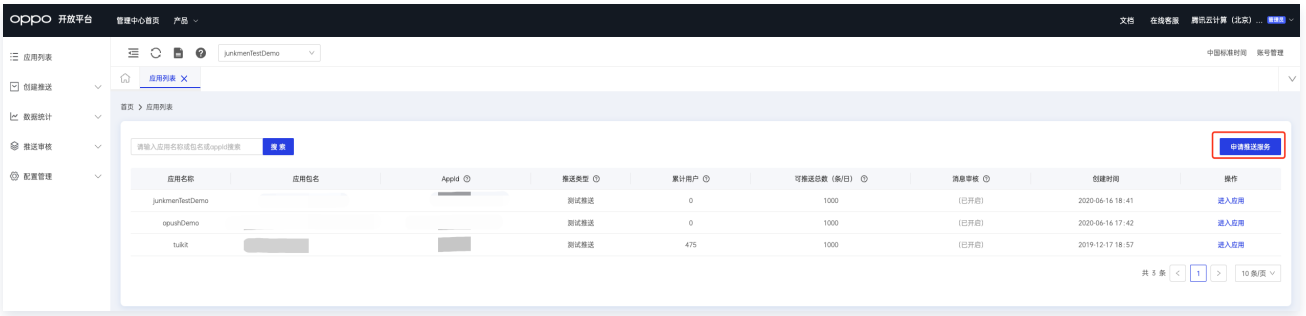


步骤3：开通 PUSH 服务

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，不可以修改。 • 该配置用于派发单击后离线推送插件的事件监听，不可以直接配置应用内页面的跳转。 

vivo

观看视频

> **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，不可以修改。该配置用于派发单击后离线推送插件的事件监听，不可以直接配置应用内页面的跳转。 

回执配置请参见：消息触达统计配置 > vivo

魅族

观看视频

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，不可以修改。该配置用于派发单击后离线推送插件的事件监听，不可以直接配置应用内页面的跳转。

回执配置请参见：消息触达统计配置 > 魅族

荣耀

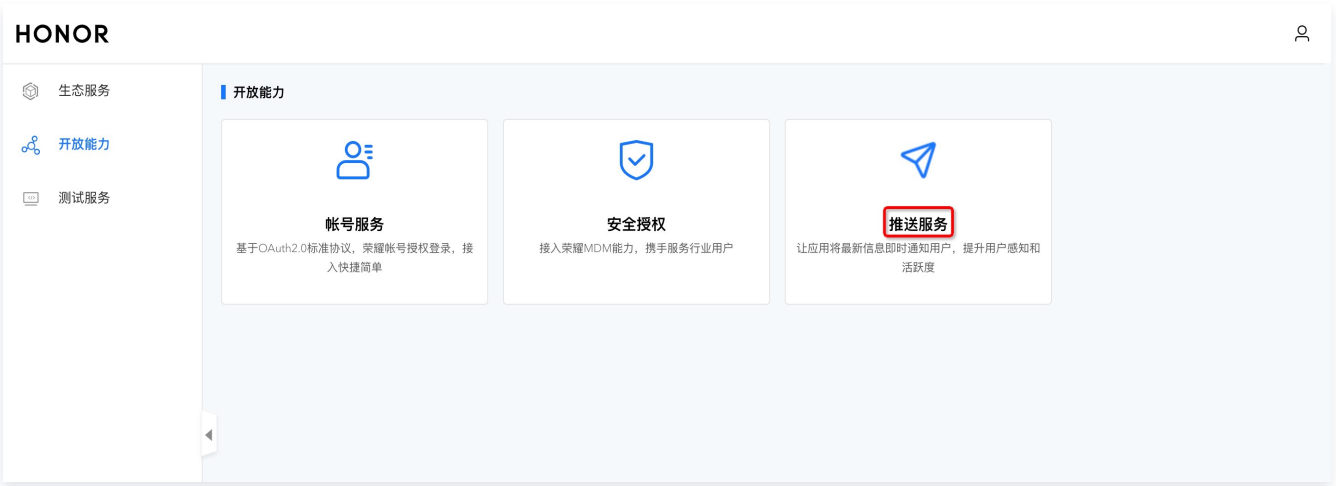
步骤1. 注册荣耀开发者账号

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。

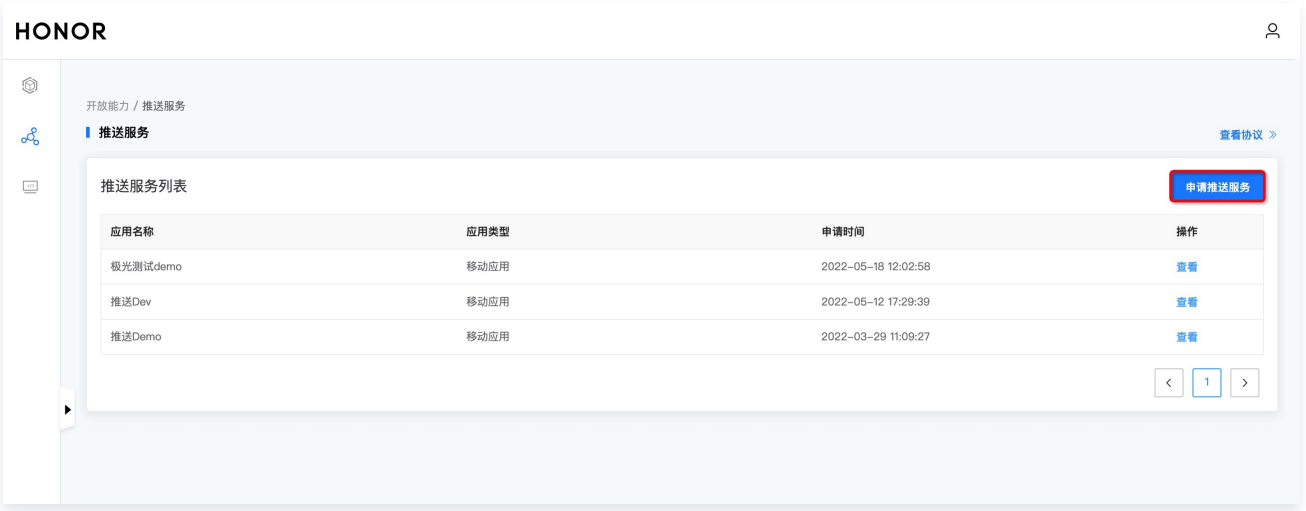
步骤3: 进入推送服务列表

单击**推送服务**进入推送服务列表页面。



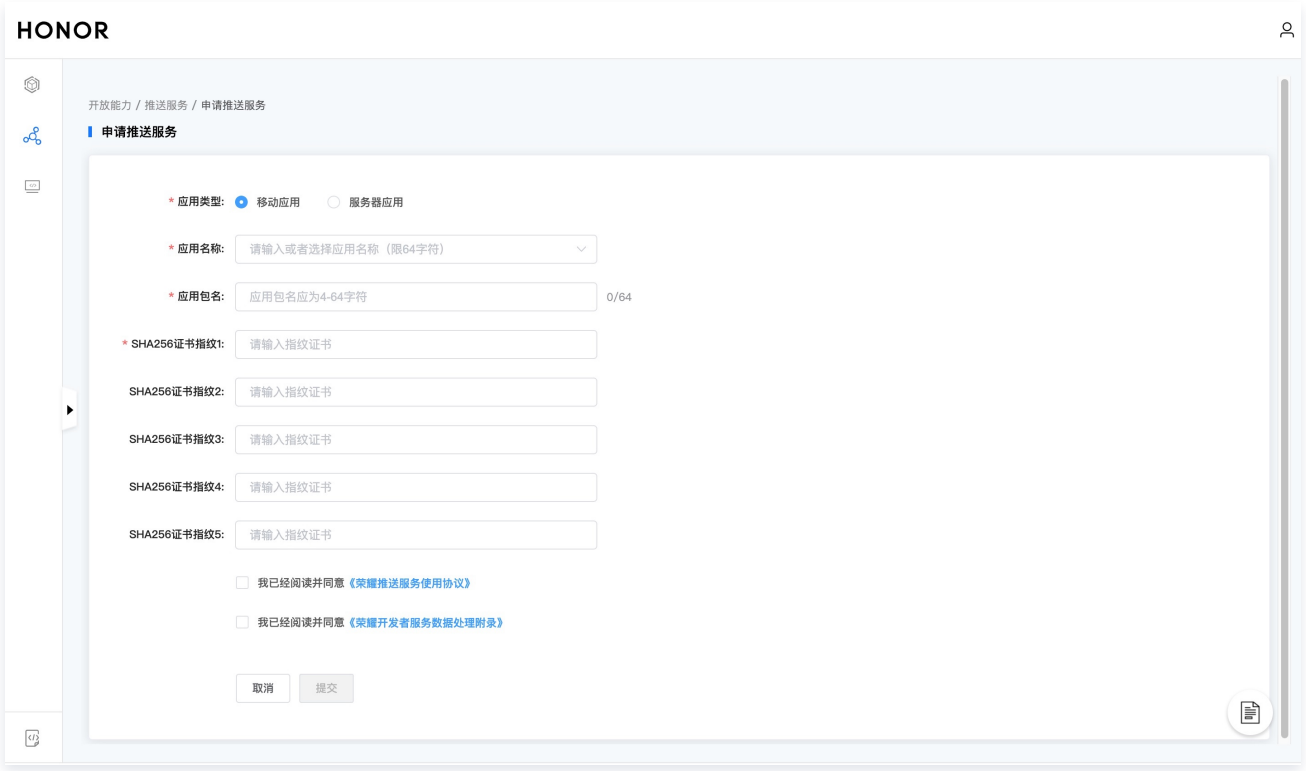
步骤4: 创建应用

1. 单击**申请推送服务**进入应用申请页面。



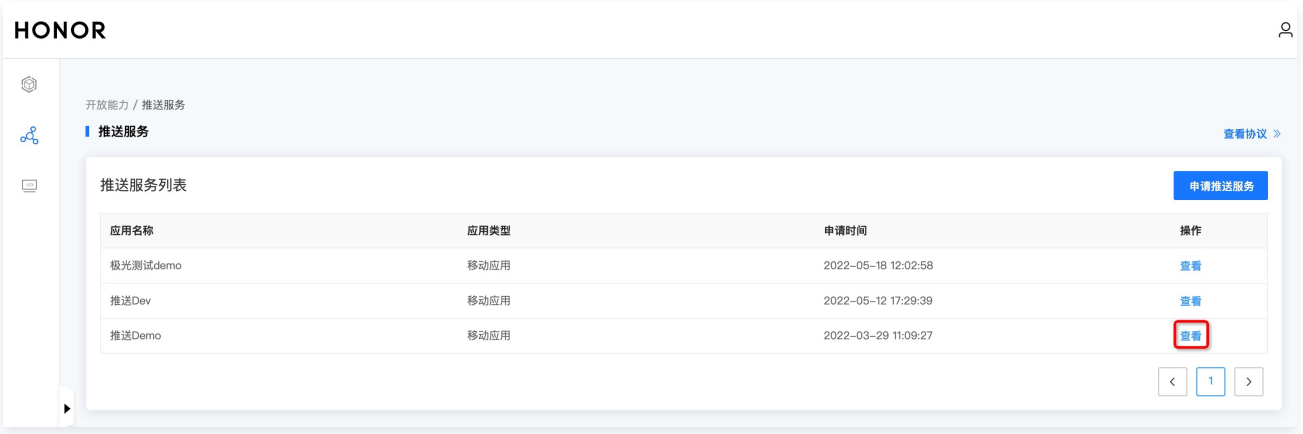
2. 选择应用类型“移动应用”，填写应用包名和证书指纹、同意推送服务协议和数据处理附录，单击**提交**。

> **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

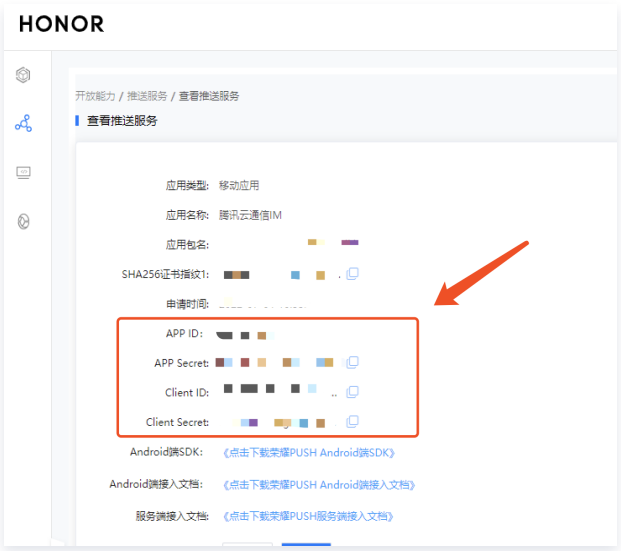
在**推送服务**列表中，单击**查看**，获取应用信息。



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，不可以修改。该配置用于派发单击后离线推送插件的事件监听，不可以直接配置应用内页面的跳转。

Google FCM

步骤1：进入Firebase 控制台

进入 Firebase 控制台，登录谷歌账号。

步骤2：创建应用

1. 单击**创建项目**，添加一个新的项目。

2. 进入Android 应用。

3. 输入应用信息，注册应用。

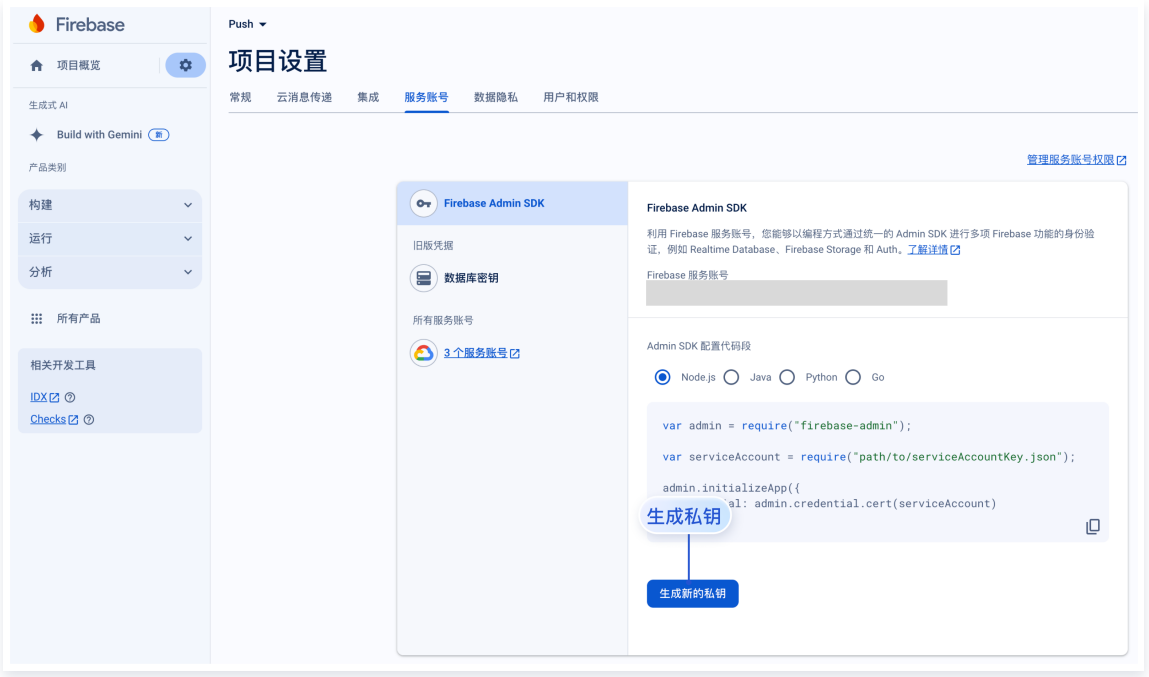
4. 下载并添加配置文件并完成注册。

   下载 google-services.json 文件，放到 Android 原生资源目录中，默认在 miniapp/android/nativeResources/app/ 路径下。

步骤3. 生成证书私钥。

1. 单击应用，进入应用管理。

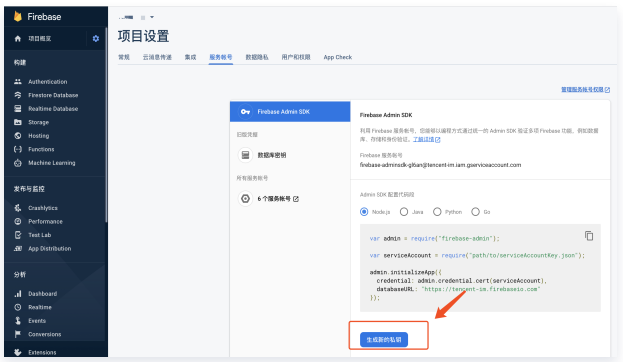
2. 在**项目设置**单击**服务账号 > 生成新的密钥**。



步骤4. 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



iOS

集成 Donut 腾讯云推送服务之前，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。

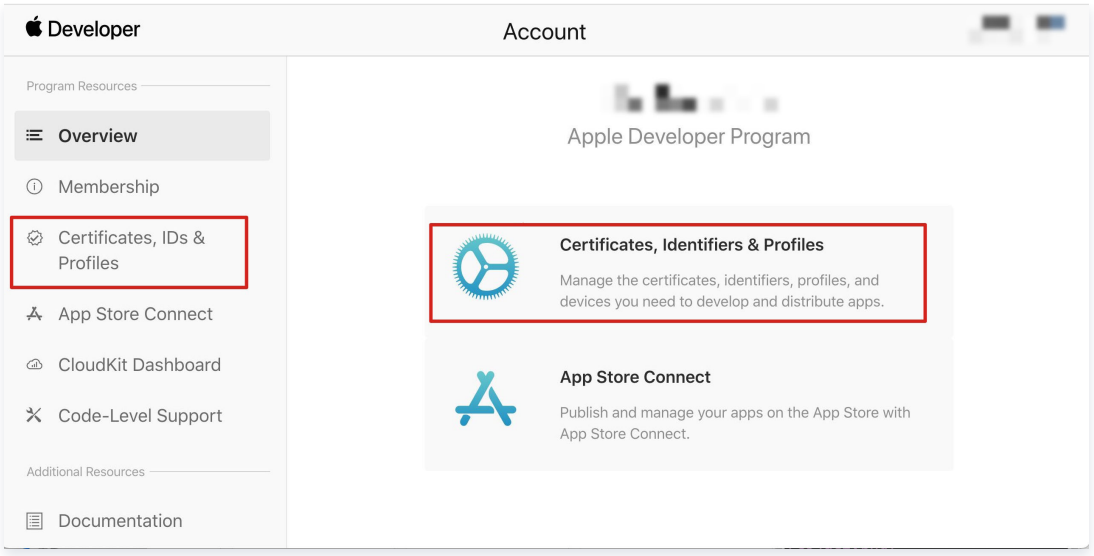
	证书类型	有效期和管理	安全性	灵动岛
p12 证书	p12 证书是一个包含公钥和私钥的二进制文件，用于基于证书的身份验证。它将公钥证书和私钥捆绑在一个文件中，扩展名为 .p12 或 .pfx。	p12 证书通常有一年的有效期，过期后需要重新生成和部署。每个应用程序都需要单独的 P12 证书来处理推送通知。	证书：p12 证书使用基于证书的身份验证，需要在服务器上存储私钥。这可能会增加安全风险，因为私钥可能会被未经授权的用户访问。	不支持
p8 证书	p8 证书是一个 Auth Key（授权密钥），用于基于令牌的身份验证。它是一个包含私钥的文本文件，扩展名为 .p8。	p8 证书没有到期日期，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

一、使用 p12 证书（传统推送证书）

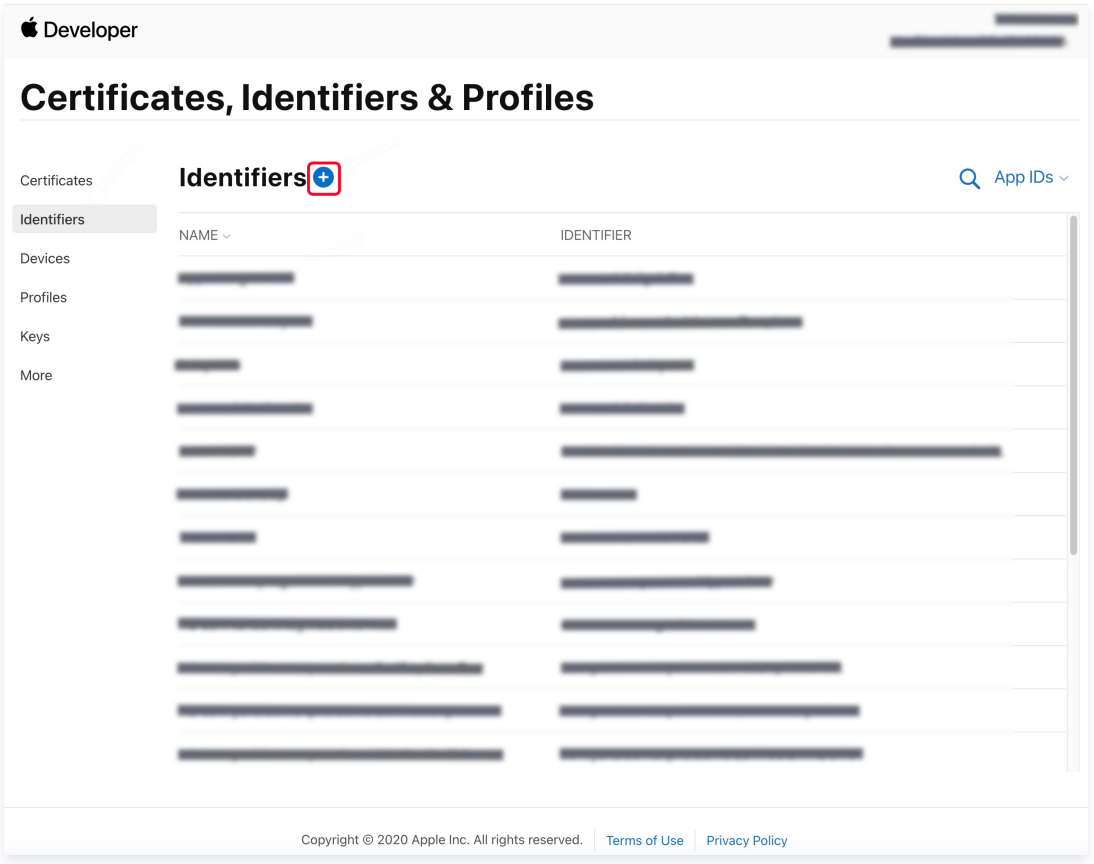
步骤1：申请 APNs 证书

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。



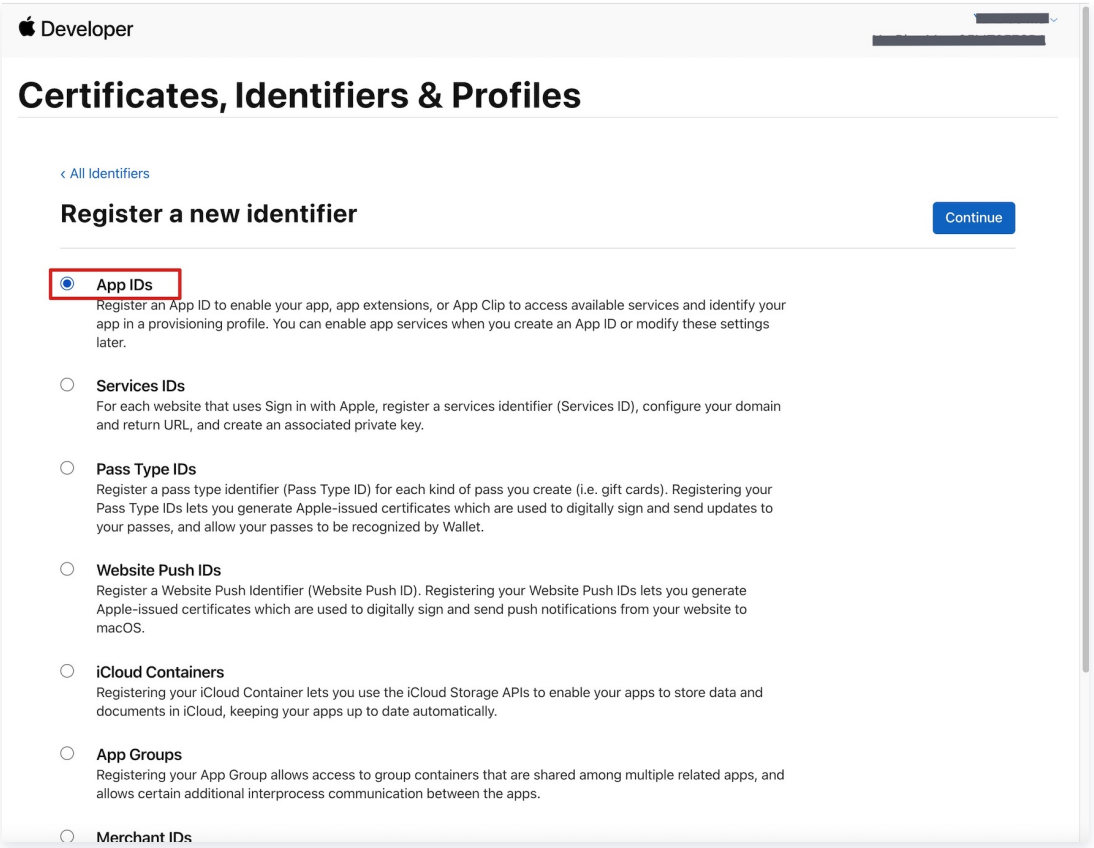
2. 单击 Identifiers 右侧的 +。



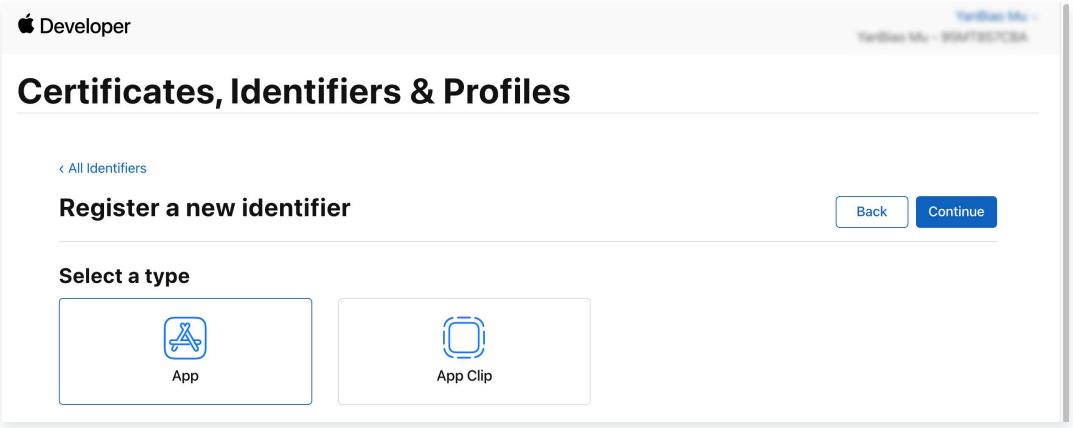
3. 您可以参见如下步骤新建一个 AppID，或者在您原有的 AppID 上增加 `Push Notification` 的 `Service`。

> **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。

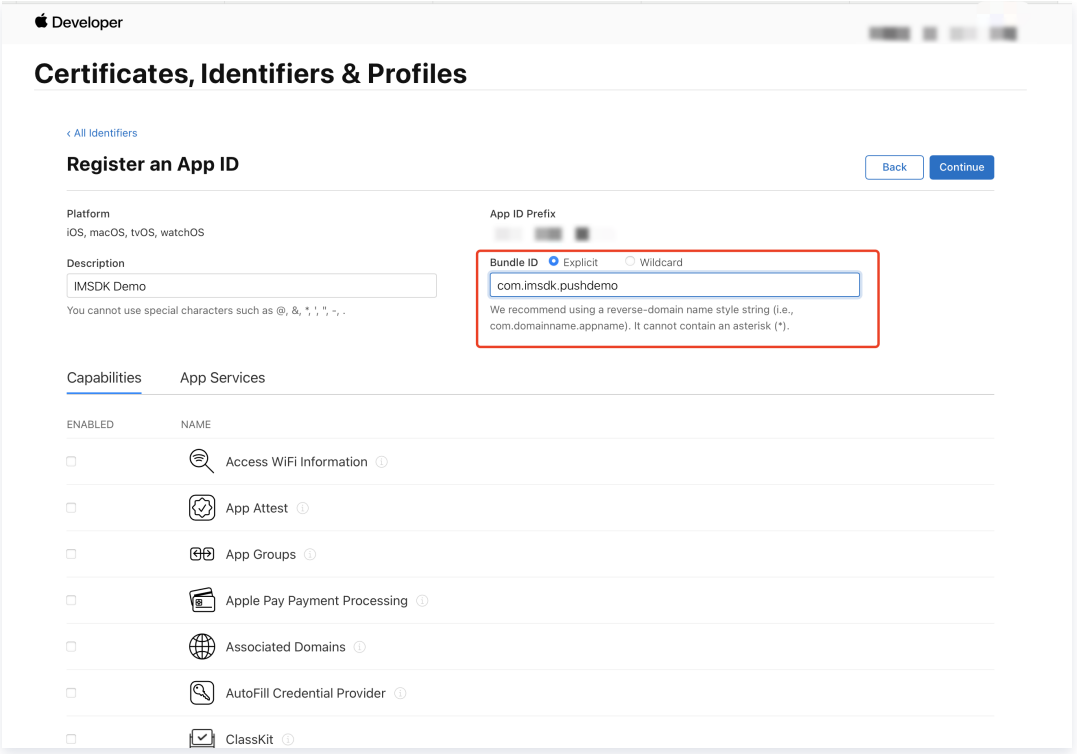
4. 勾选 App IDs，单击 Continue 进行下一步。



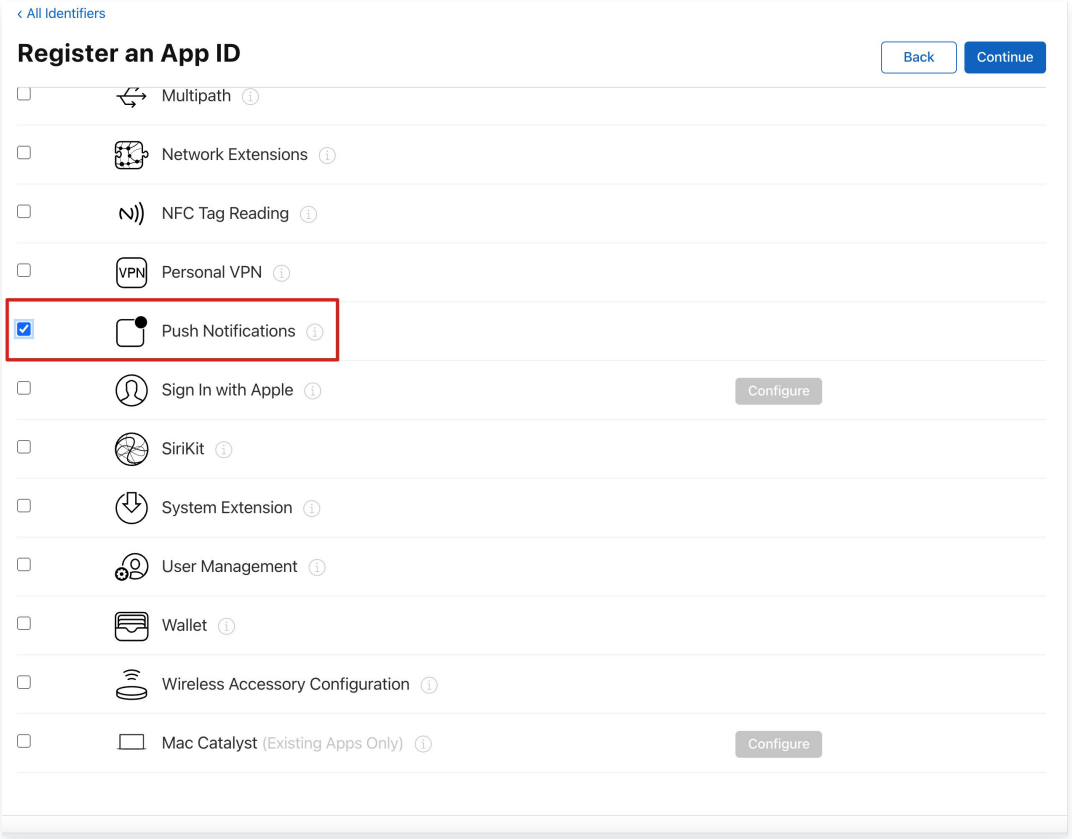
5. 选择 **App**，单击 **Continue** 进行下一步。



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

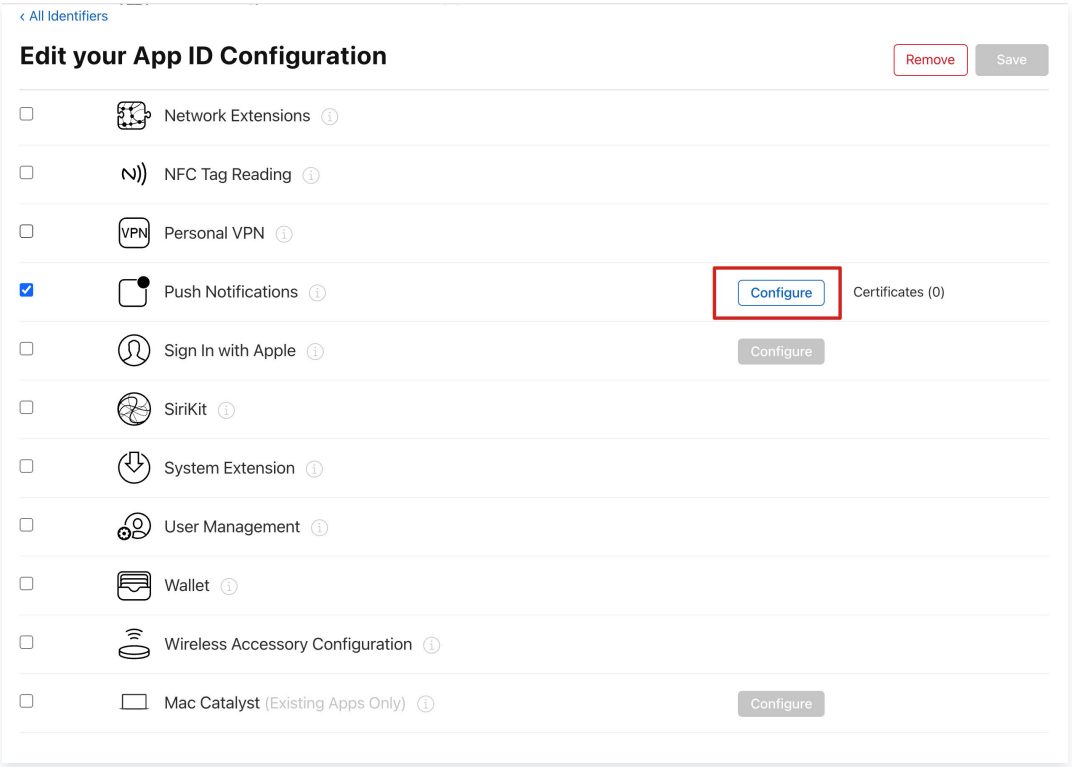


7. 勾选 **Push Notifications**，开启远程推送服务。

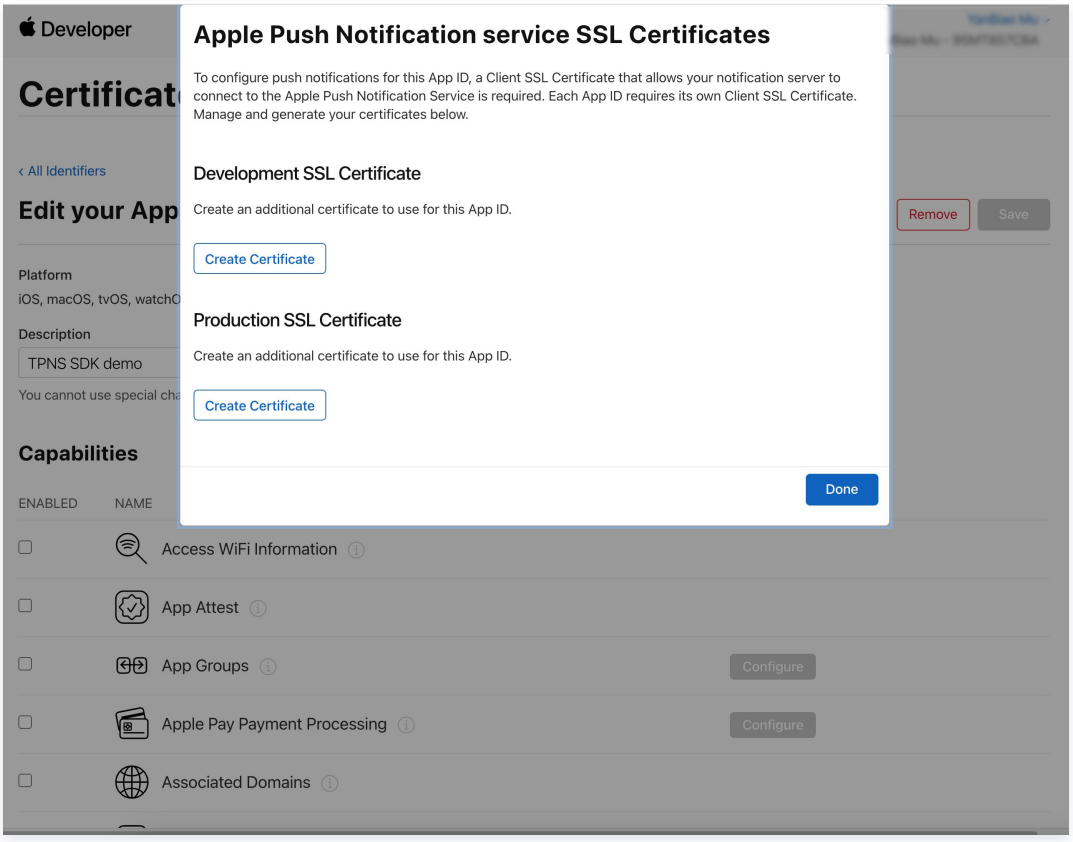


生成证书

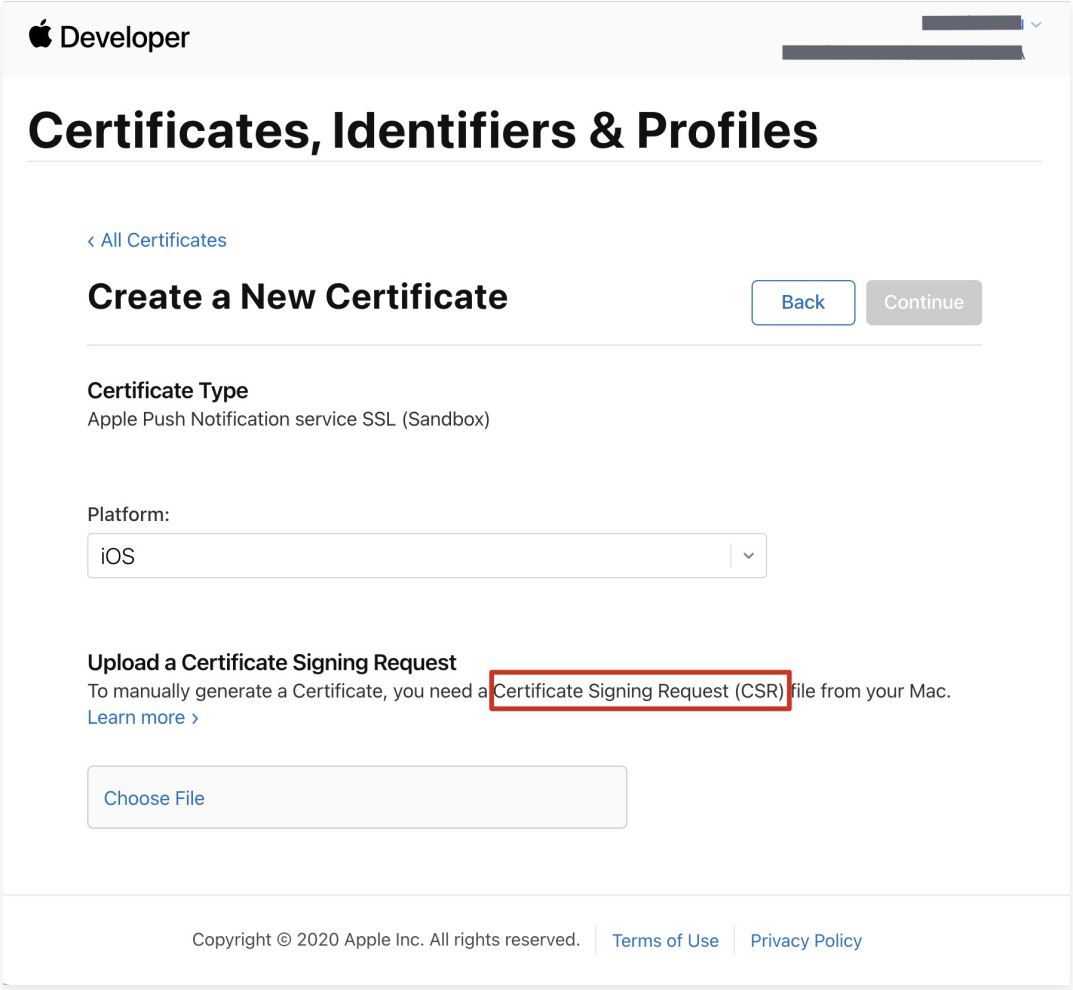
1. 选中您的 AppID，选择 **Configure**。



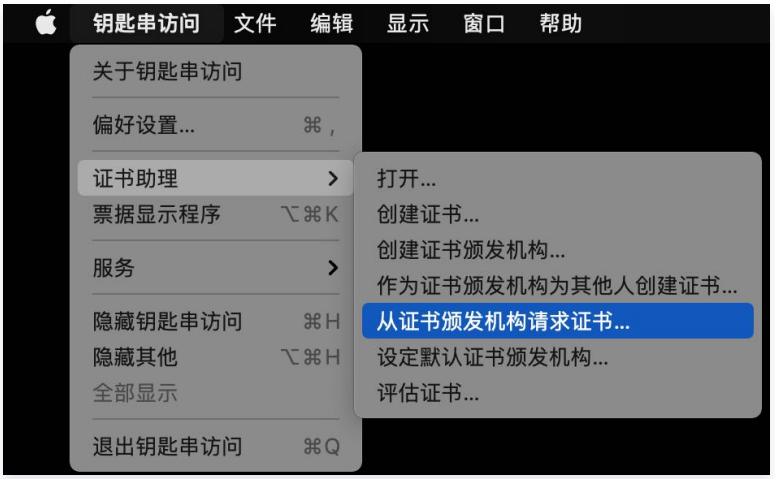
2. 可以看到在 **Apple Push Notification service SSL Certificates** 窗口中有两个 `SSL Certificate`，分别用于开发环境（Development）和生产环境（Production）的远程推送证书，如下图所示：



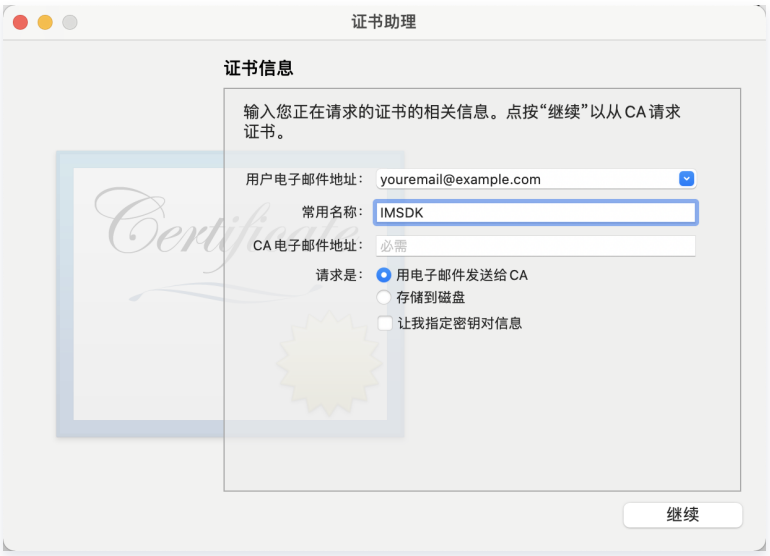
3. 我们先选择开发环境（Development）的 **Create Certificate**，系统将提示我们需要一个 Certificate Signing Request（CSR）。



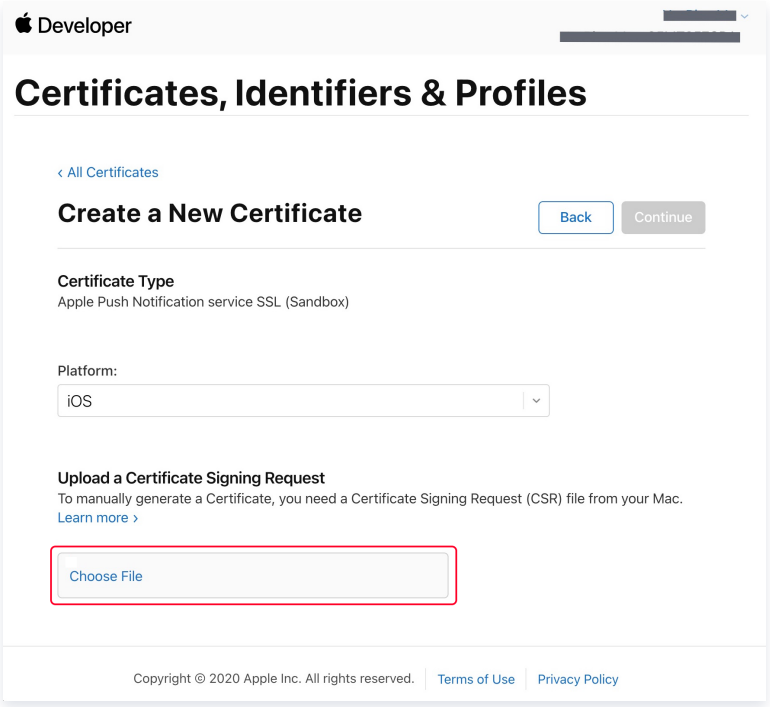
4. 在 Mac 上打开**钥匙串访问工具（Keychain Access）**，在菜单中选择**钥匙串访问 > 证书助理 > 从证书颁发机构请求证书**（`Keychain Access - Certificate Assistant - Request a Certificate From a Certificate Authority`）。



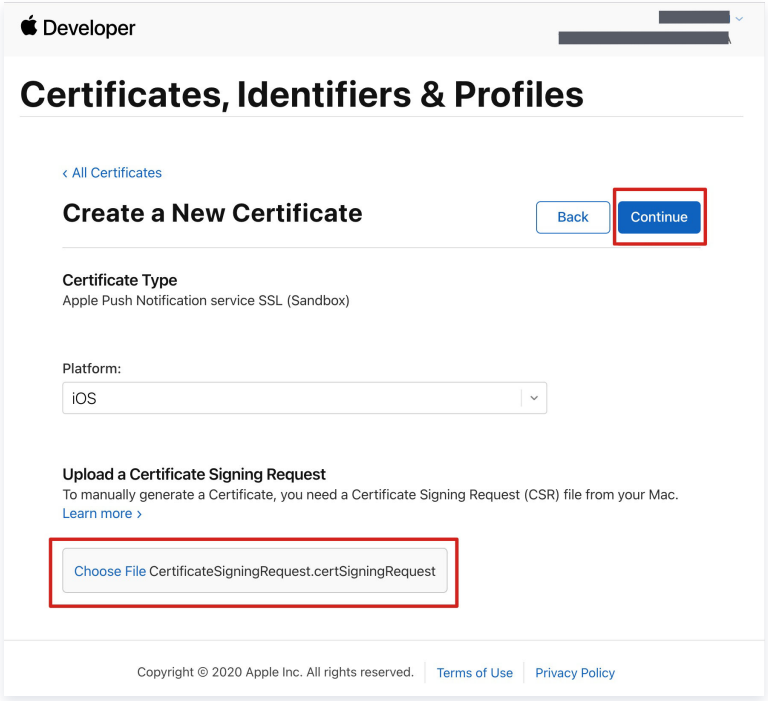
5. 输入用户电子邮件地址（您的邮箱）、常用名称（您的名称或公司名），选择**存储到磁盘**，单击**继续**，系统将生成一个 `*.certSigningRequest` 文件。



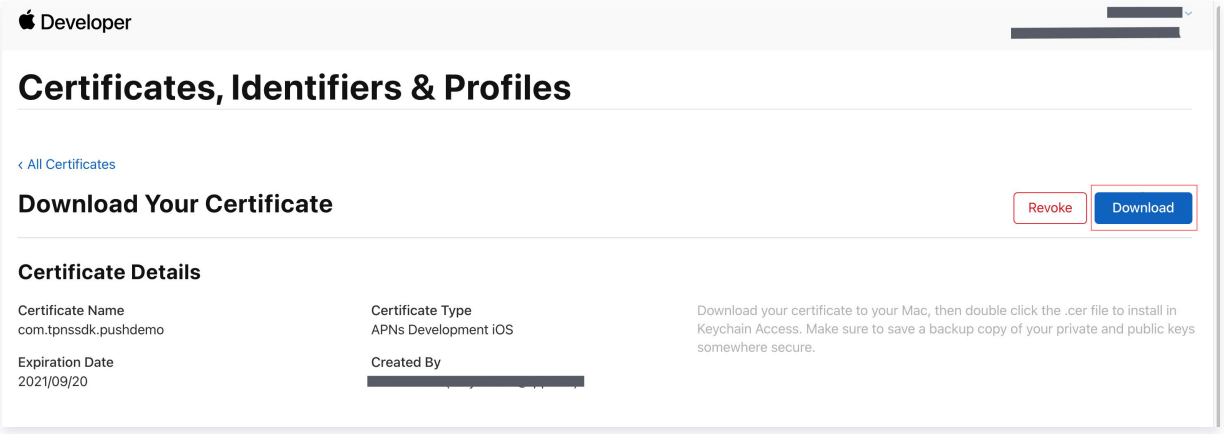
6. 返回上述 步骤3 中 Apple Developer 网站刚才的页面，单击 **Choose File** 上传生成的 `*.certSigningRequest` 文件。



7. 单击 **Continue**，即可生成推送证书。

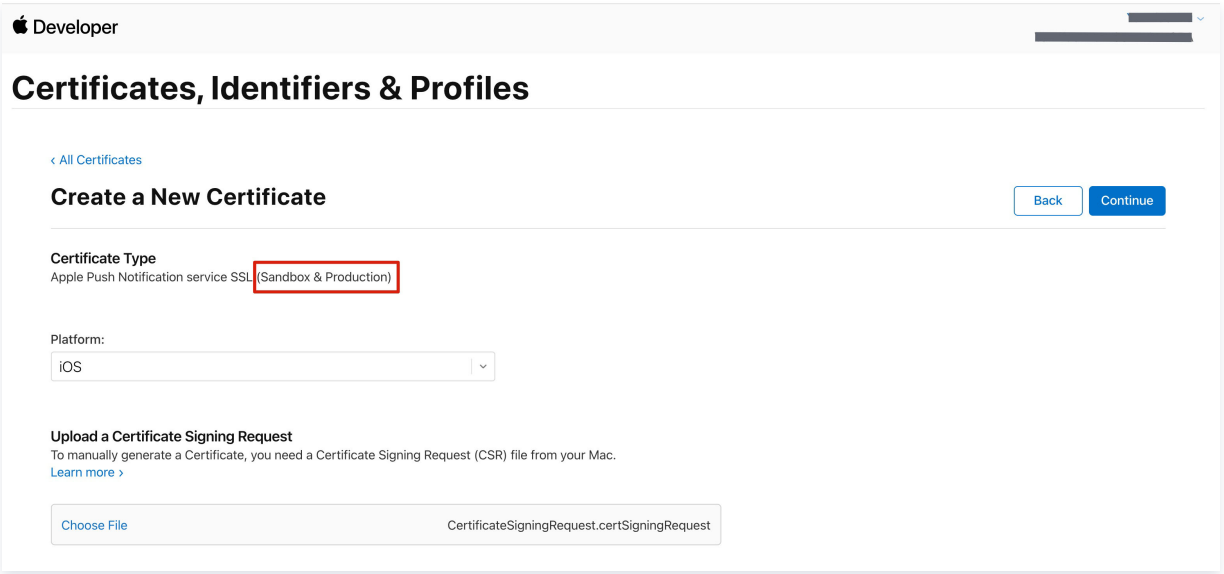


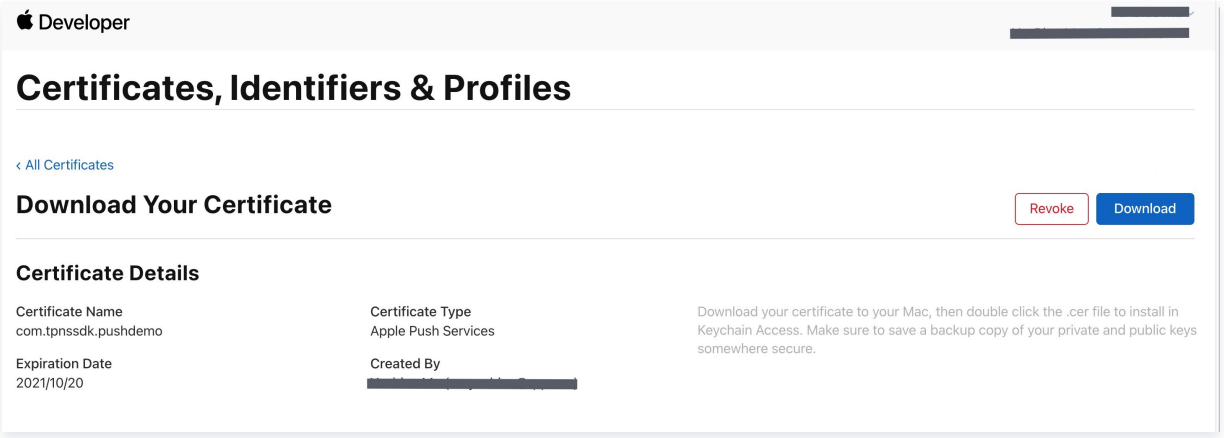
8. 单击 Download 下载开发环境的 `Development SSL Certificate` 到本地。



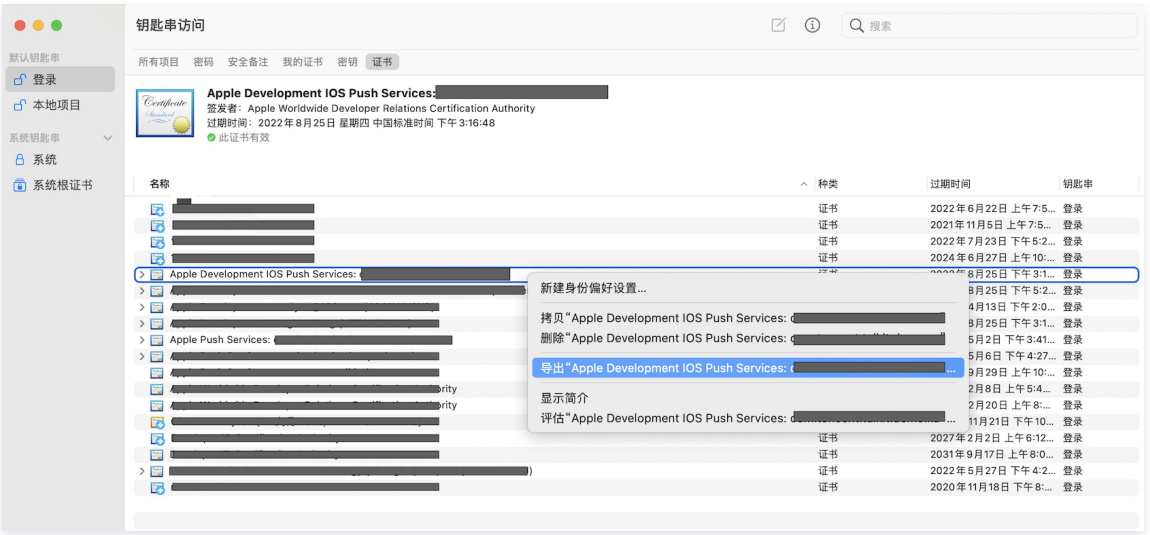
9. 再次按照上述步骤1 – 8，将生产环境的 `Production SSL Certificate` 下载到本地。

> **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。





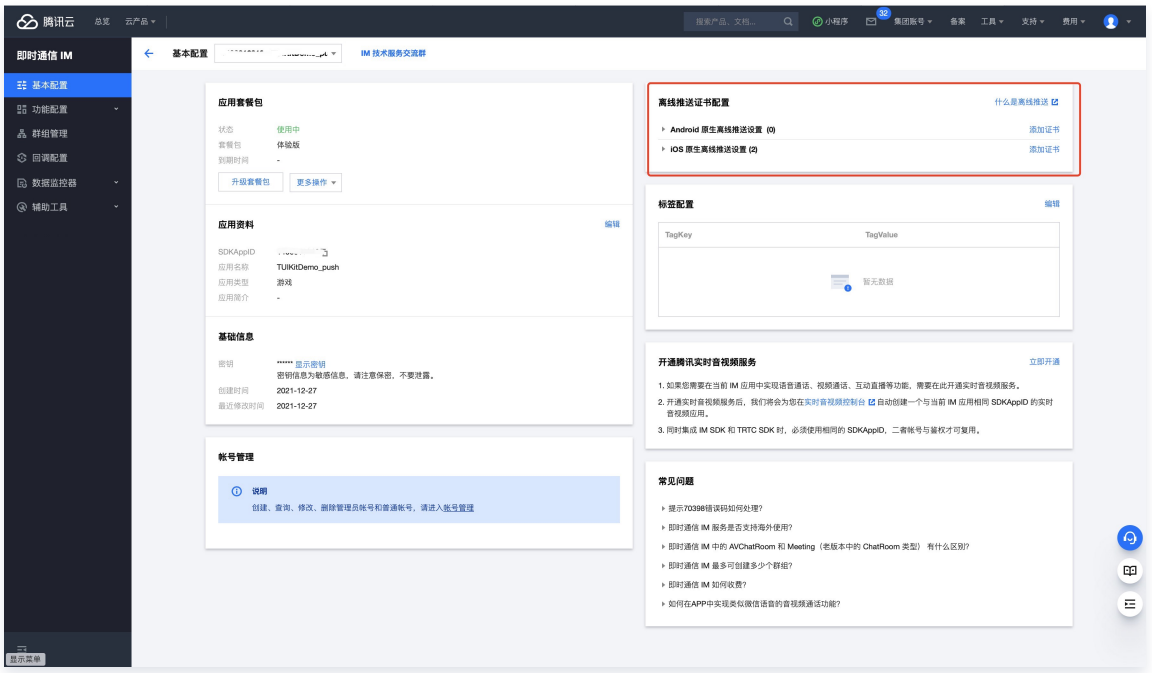
10. 双击打开下载的开发环境和生产环境的 `SSL Certificate`，系统会将其导入钥匙串中。
11. 打开钥匙串应用，在**登录 > 我的证书**，右键分别导出刚创建的开发环境（ `Apple Development IOS Push Service` ）和生产环境（ `Apple Push Services` ）的 `p12` 文件。



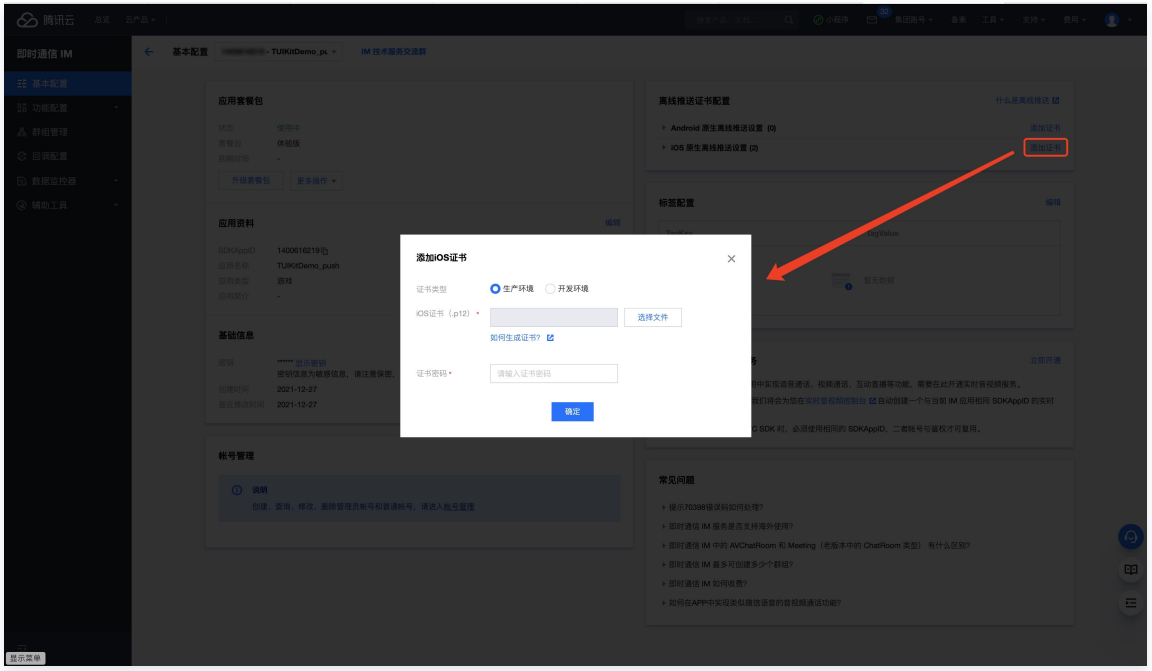
> **注意：**
> 保存 `.p12` 文件时，请务必为其设置密码。

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。

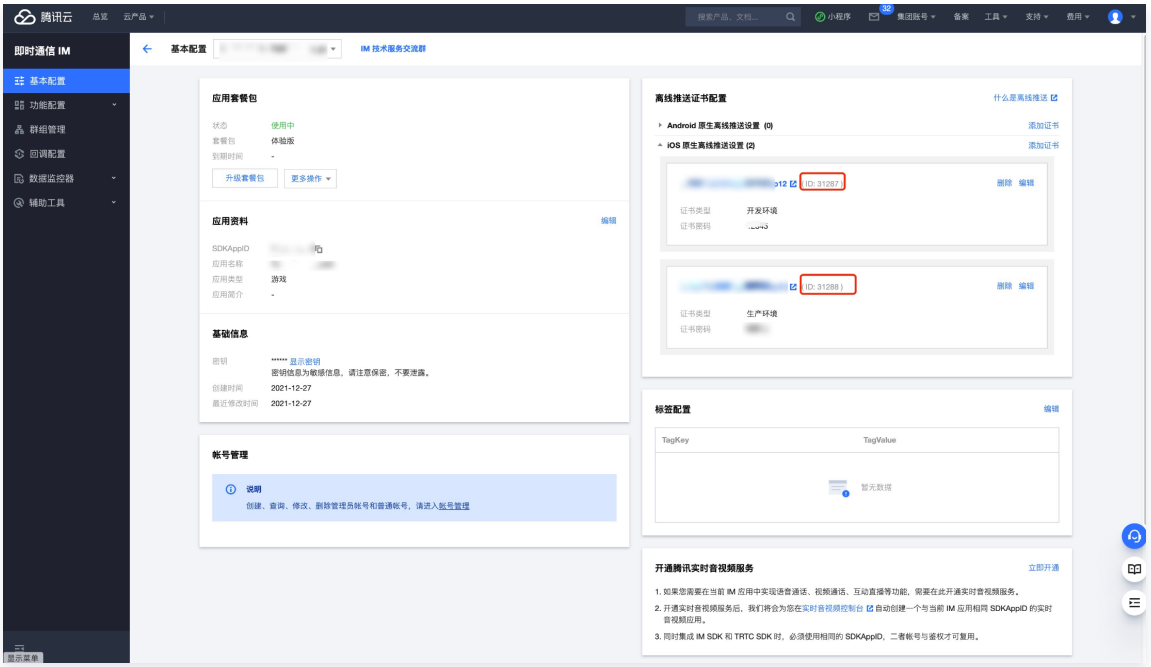


3. 单击 **iOS 原生离线推送设置**右侧的**添加证书**。
4. 选择证书类型，上传 iOS 证书（p.12），设置证书密码，单击**确认**。



> **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。

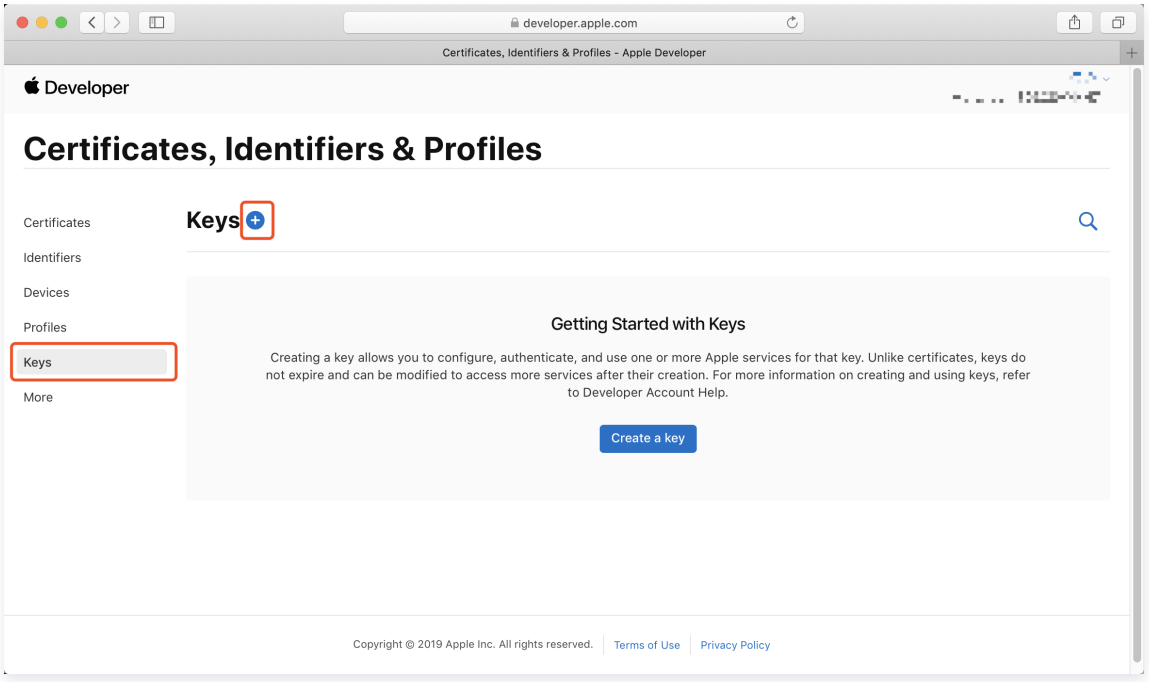


二、使用 p8 证书（支持灵动岛推送）

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。

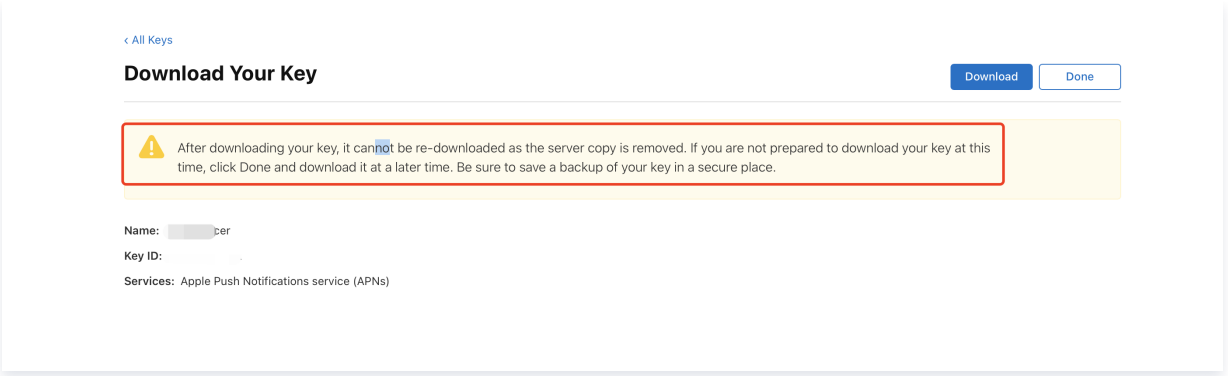
步骤1：申请 APNs 证书

要创建 p8 证书文件，首先需要登录 苹果开发者中心 。



iOS 证书配置指南

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** 。

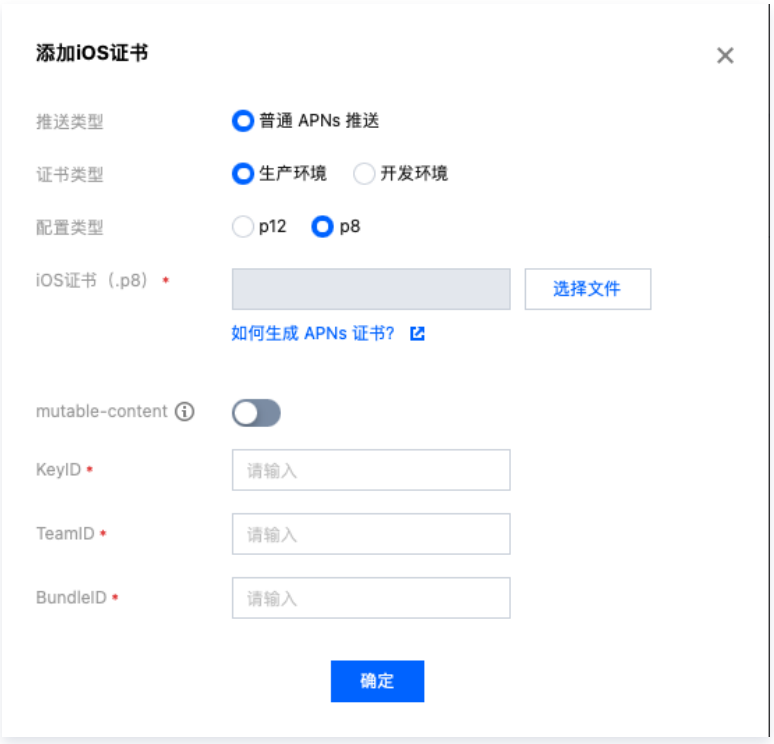


确认并生成密钥：在确认页面核对您的密钥信息，然后单击 **Register** 。接下来，您将看到一个页面提示您下载密钥。单击 **Download**，将生成的 .p8 文件保存到您的计算机上。

> ⚠ **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

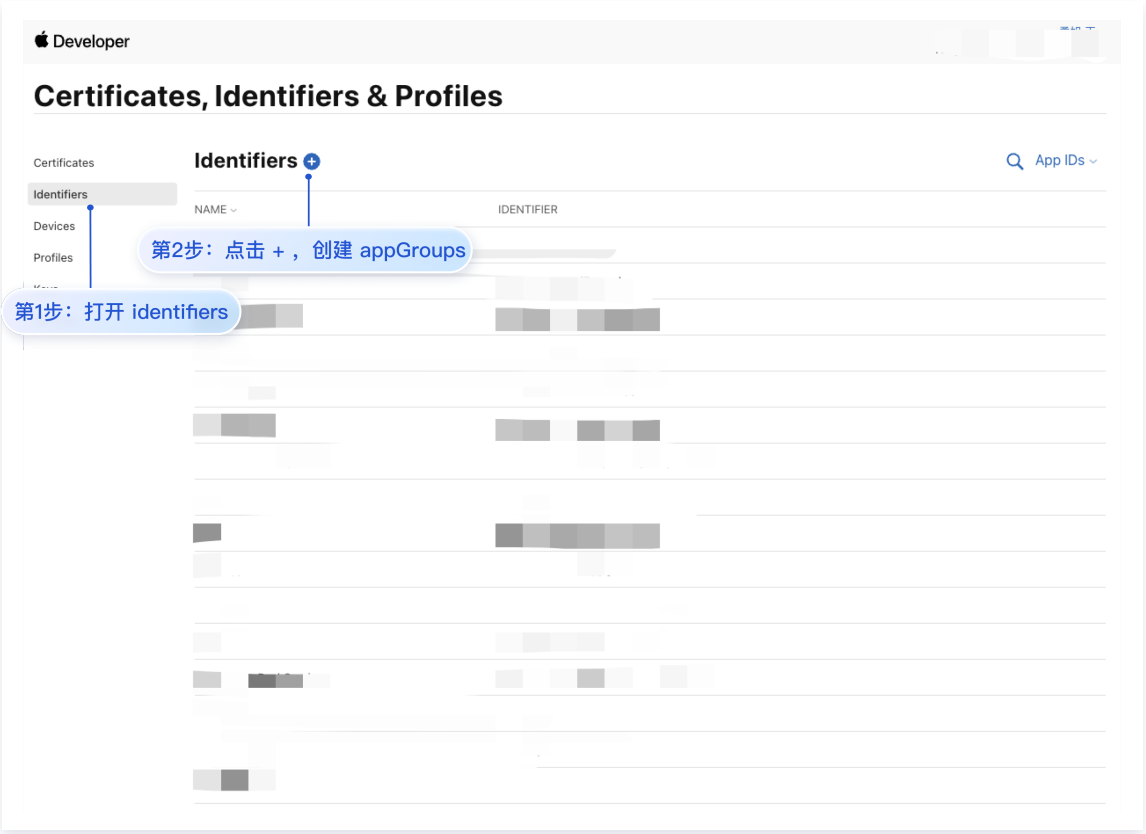


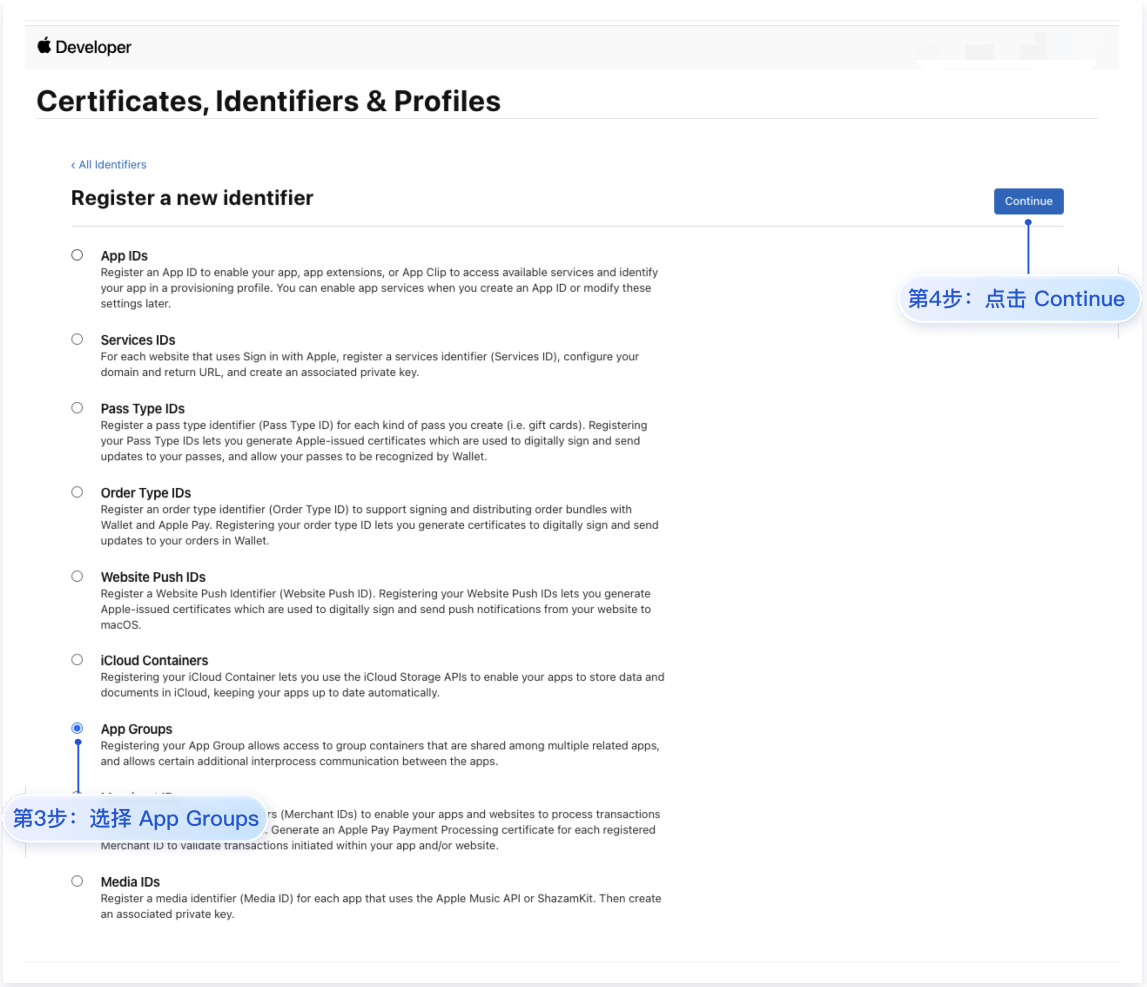
> ⓘ **说明：**
> - **Key ID**：这是您的 APNs Auth Key 的唯一标识符。当您在 Apple Developer Center 创建一个新的 APNs Auth Key 时，系统会为您生成一个 Key ID。您可以在 "Certificates, Identifiers & Profiles" 部分的 "Keys" 中找到它。
> - **Team ID**：这是您的开发者账户的唯一标识符。您可以在 Apple Developer Center 的账户详情页面找到它。点击右上角的 "Membership"，在 "Membership Details" 部分可以找到您的 Team ID。

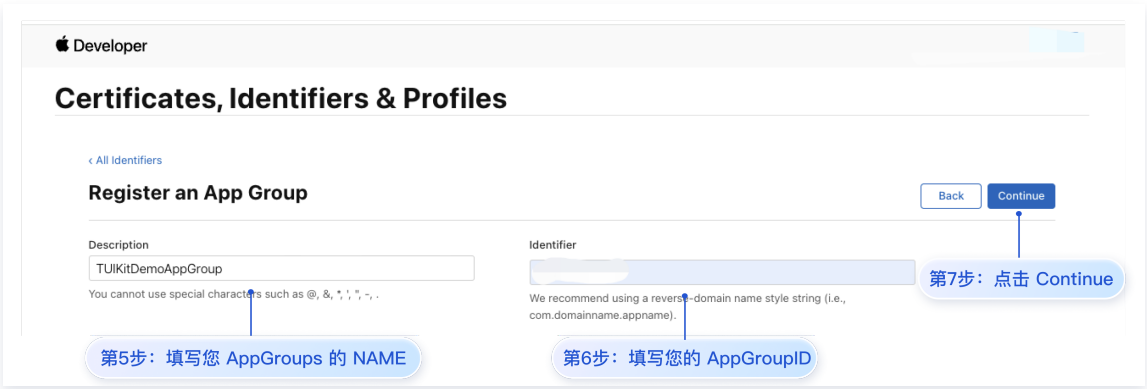
- **Bundle ID**：这是您的应用程序的唯一标识符，也称为应用程序 ID。您可以在 Apple Developer Center 的 "Certificates, Identifiers & Profiles" 部分找到它。选择 "Identifiers"，然后在您的应用程序列表中找到对应的 Bundle ID。

三. 生成 TIMPushAppGroupID

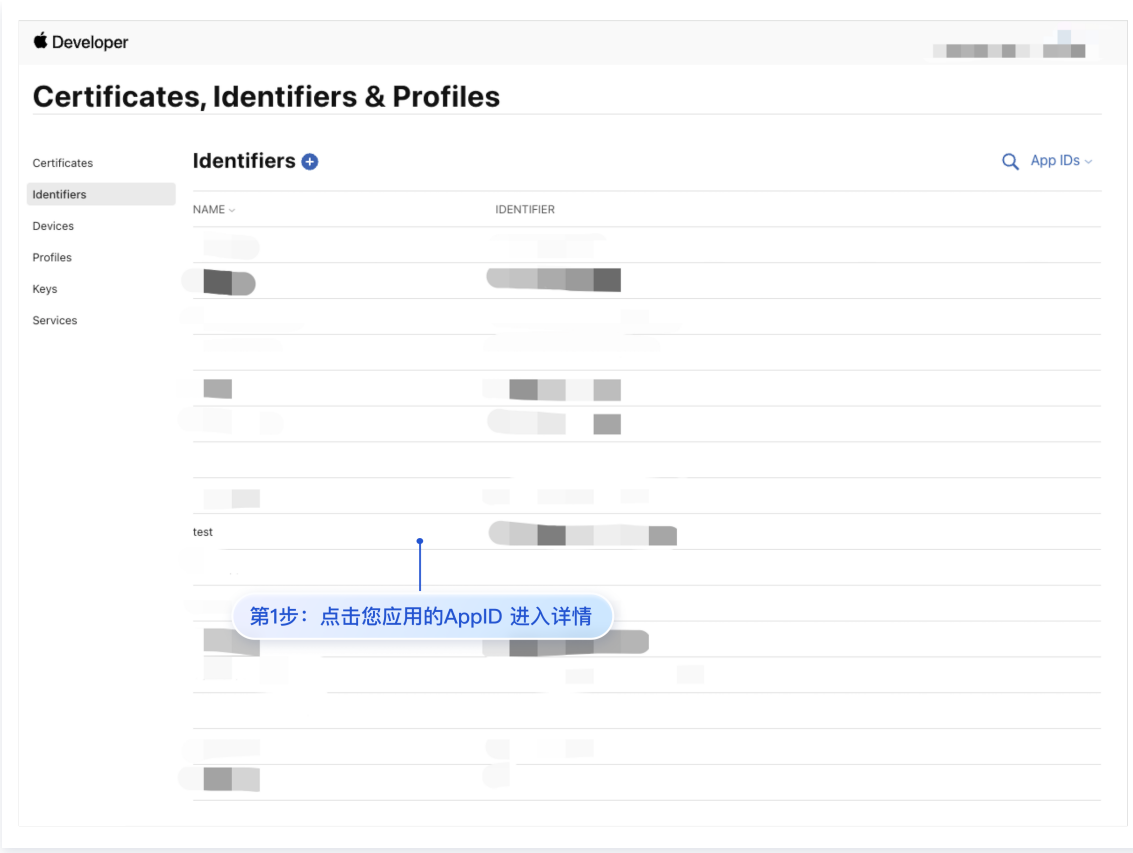
步骤1：登录苹果开发者中心网站，进入【identifiers】->【App Groups】创建 AppGroups。

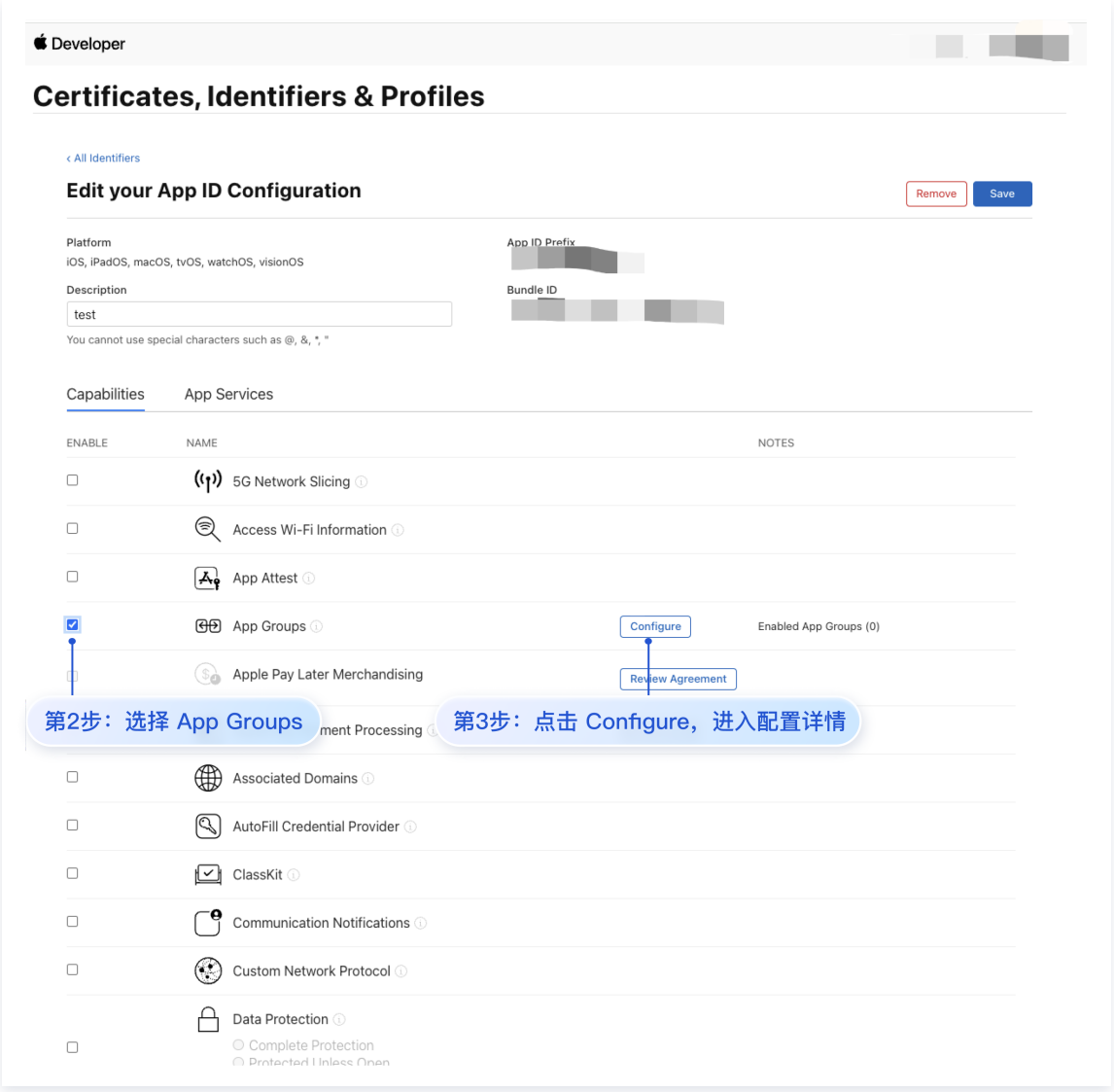


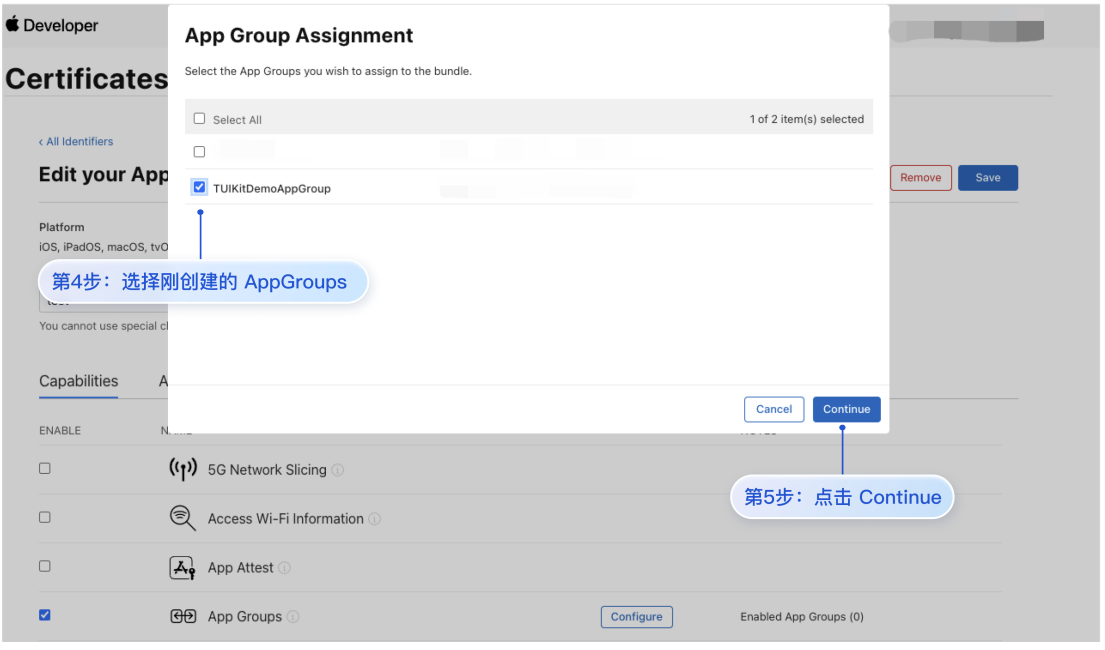




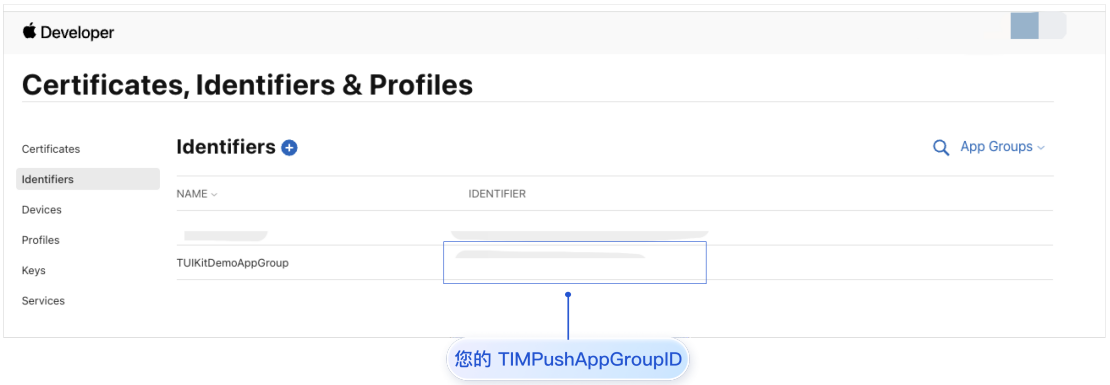
步骤2：绑定使用 TencentCloud-TIMPush 应用的 appID 到 AppGroups。


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





步骤3. 获取您的 TIMPushAppGroupID。


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

Flutter

最近更新时间：2024-12-02 18:25:42

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

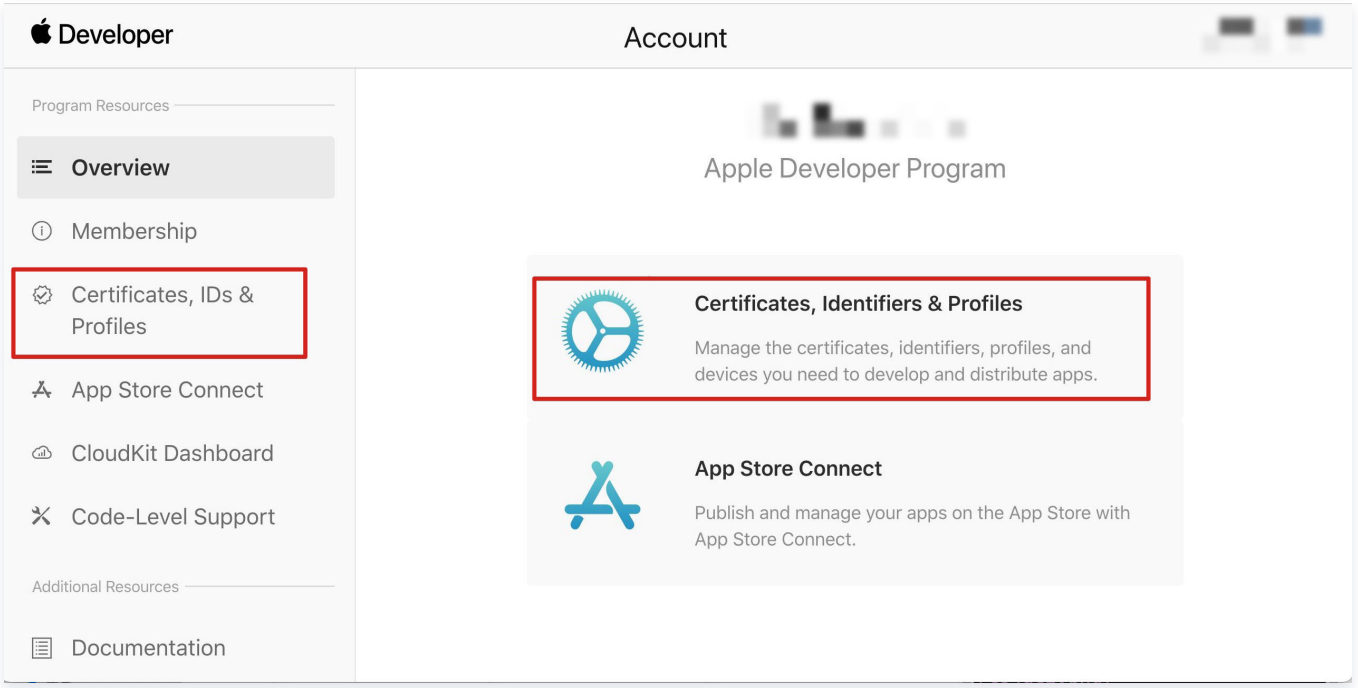
集成 消息推送 插件之前，需要先向 Apple 申请 APNs 推送证书，然后上传推送证书到 IM 控制台 。之后按照快速接入步骤接入即可。

操作步骤

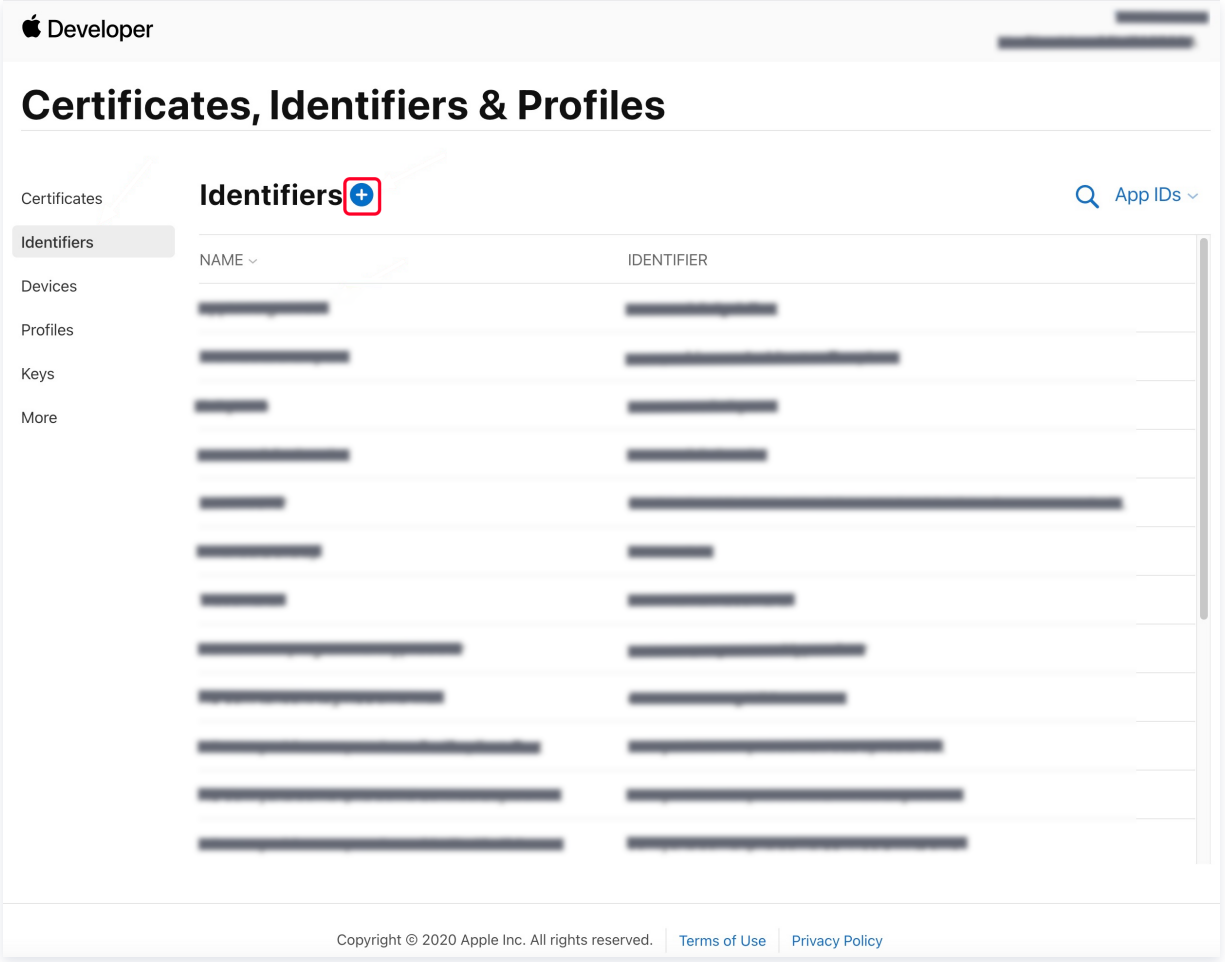
步骤1：申请 APNs 证书

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。



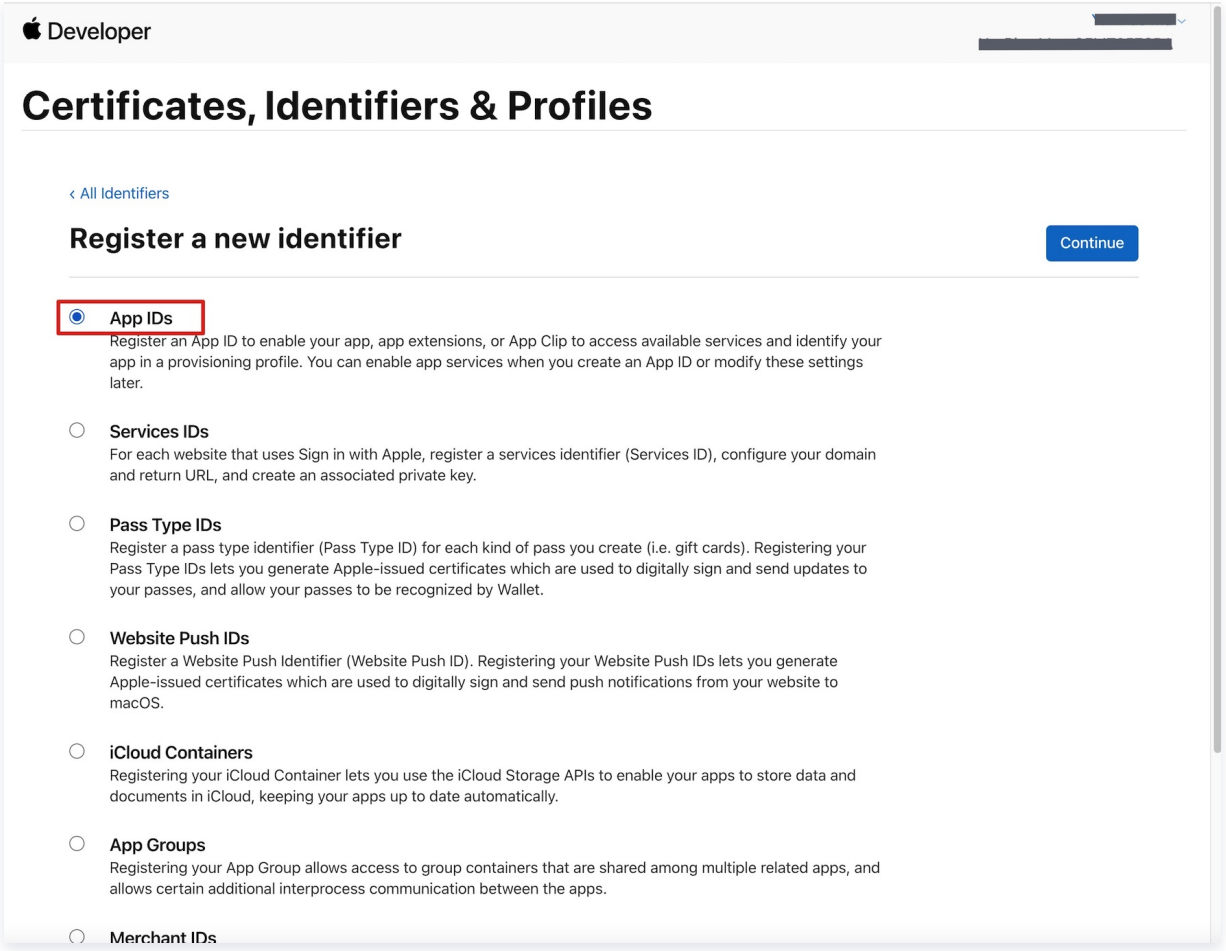
2. 单击 Identifiers 右侧的 +。



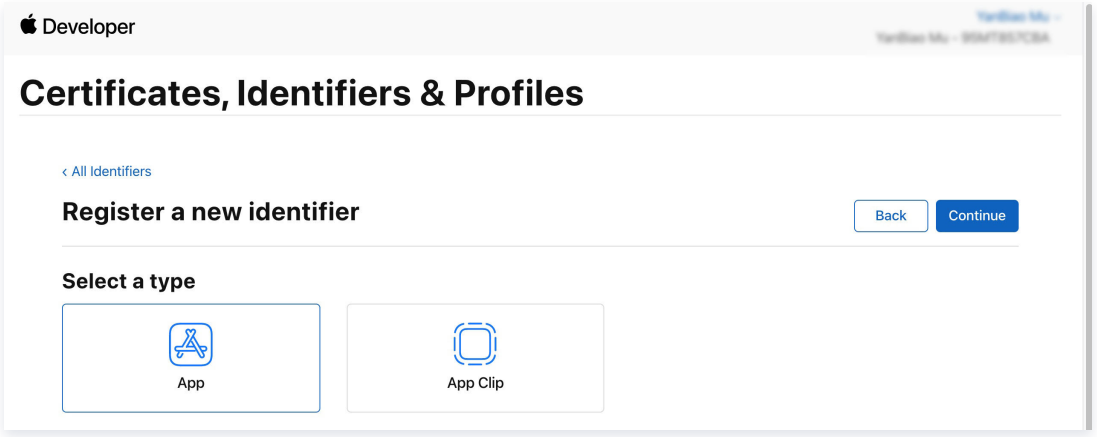
3. 您可以参见如下步骤新建一个 AppID，或者在您原有的 AppID 上增加 `Push Notification` 的 `Service`。

> **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。

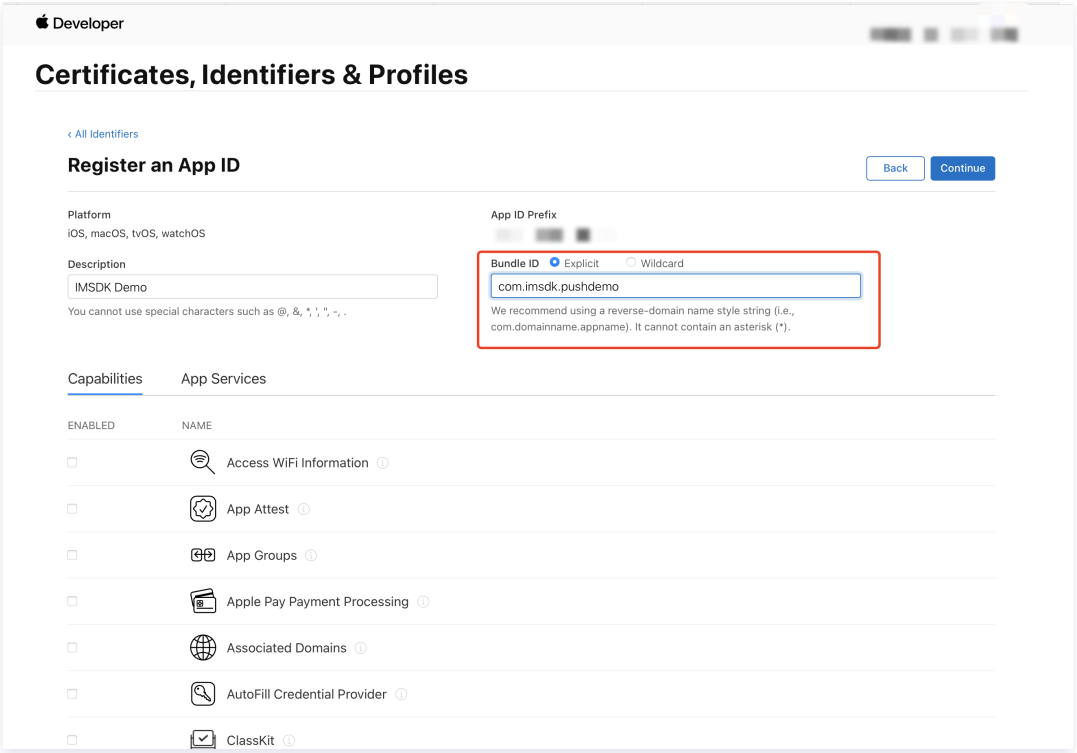
4. 勾选 **App IDs**，单击 **Continue** 进行下一步。



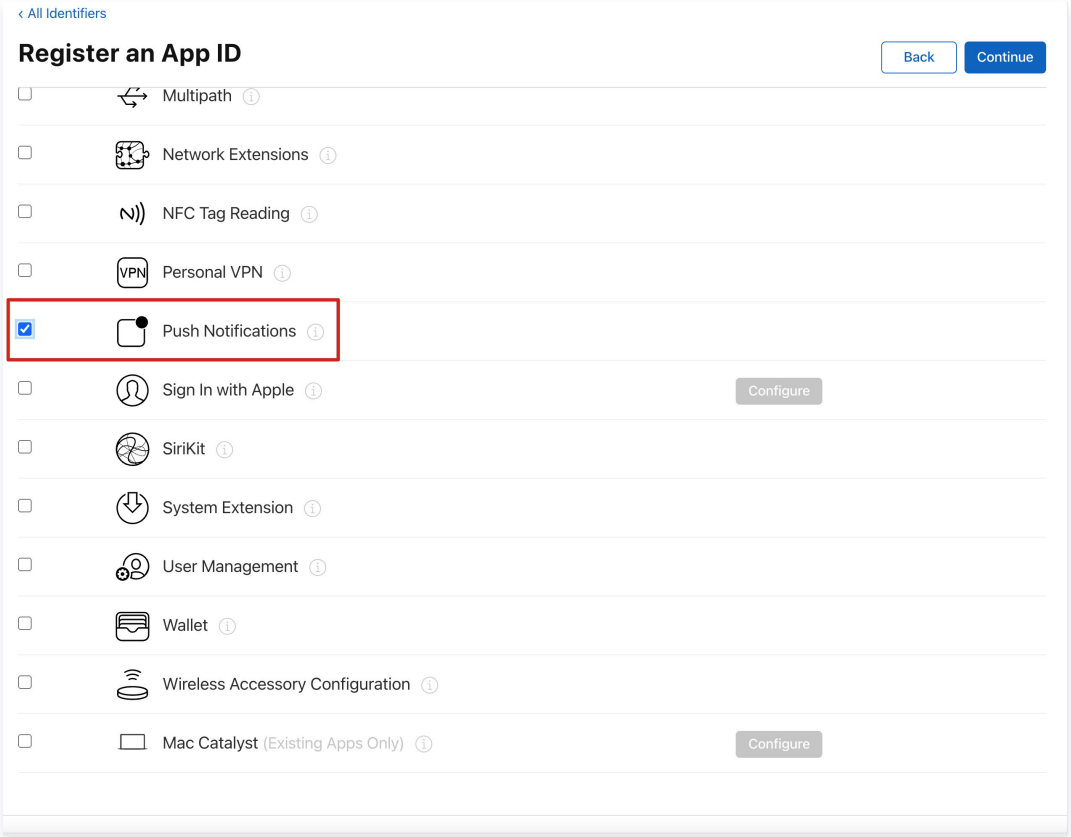
5. 选择 **App**，单击 **Continue** 进行下一步。



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

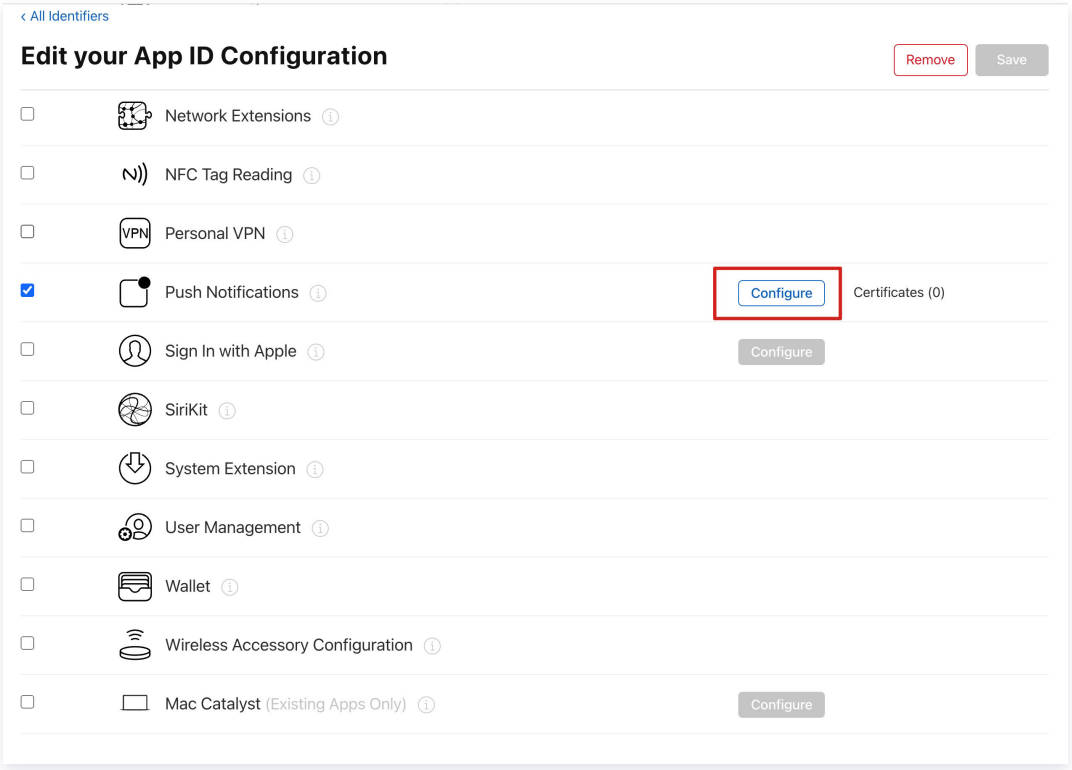


7. 勾选 **Push Notifications**，开启远程推送服务。

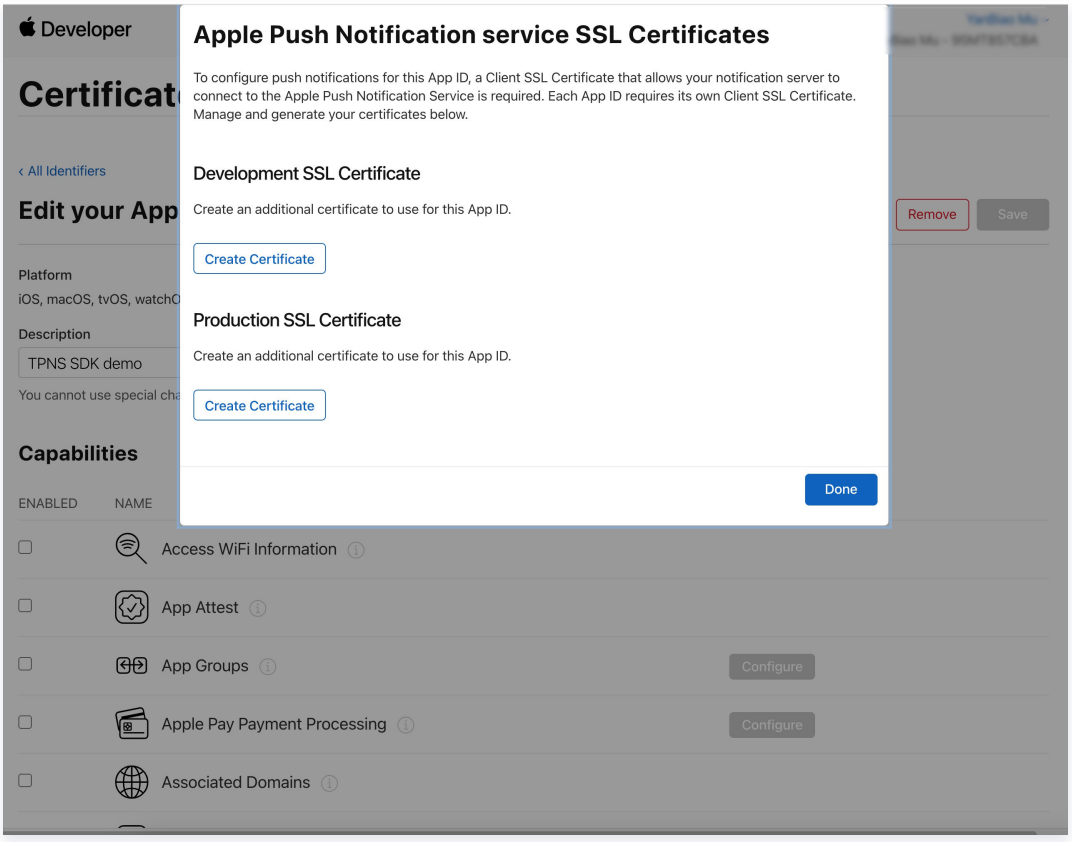


生成证书

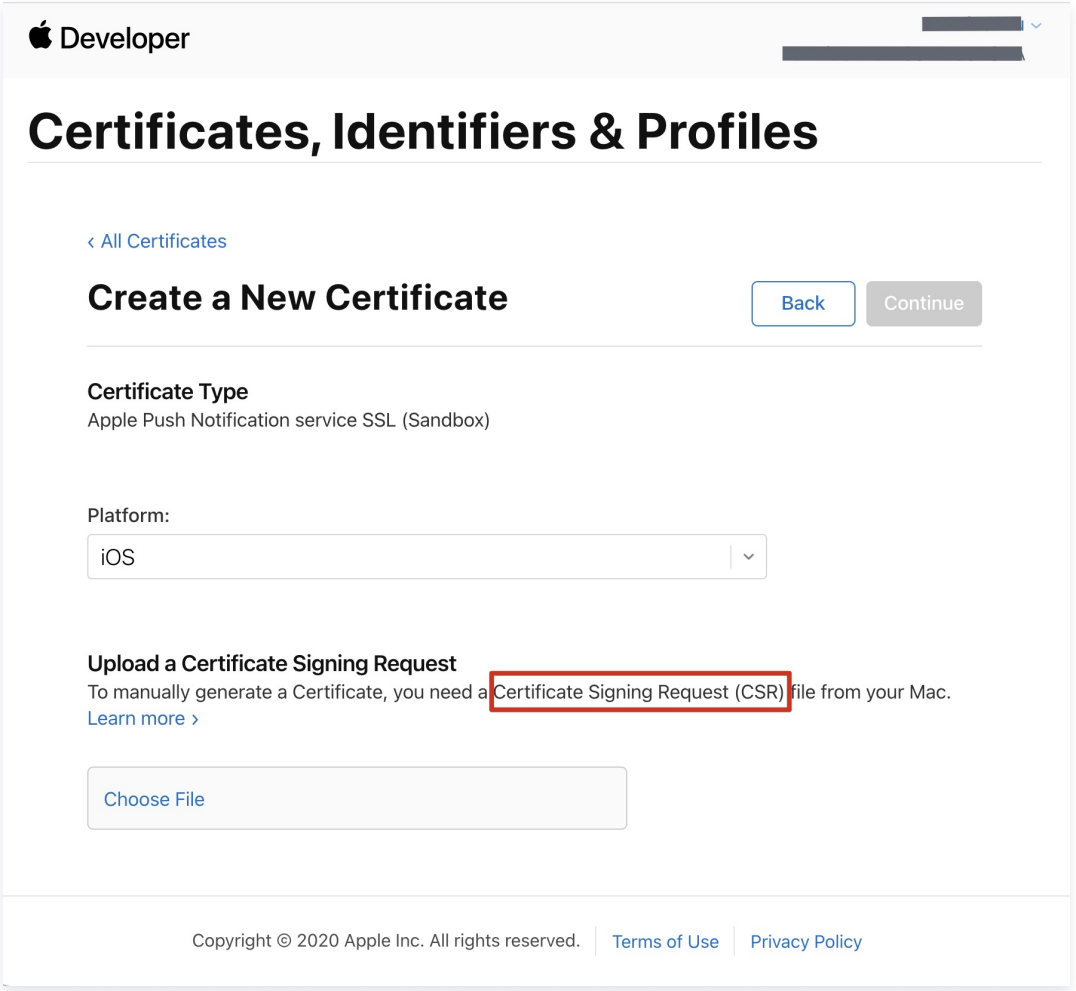
1. 选中您的 AppID，选择 **Configure**。



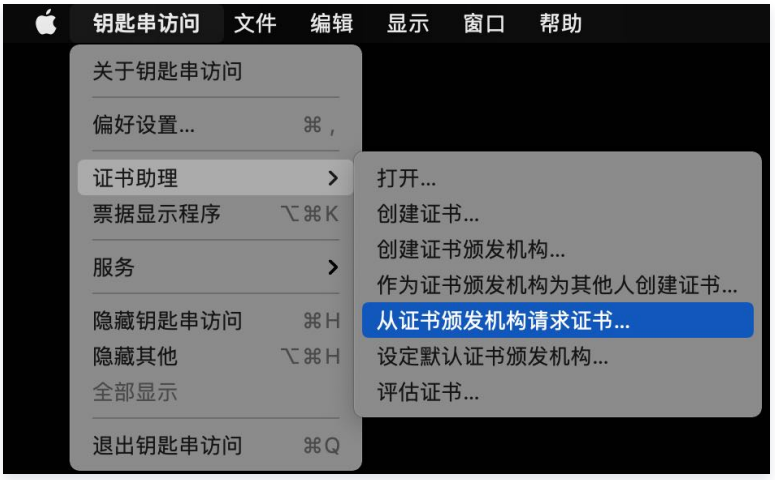
2. 可以看到在 **Apple Push Notification service SSL Certificates** 窗口中有两个 `SSL Certificate`，分别用于开发环境（Development）和生产环境（Production）的远程推送证书，如下图所示：



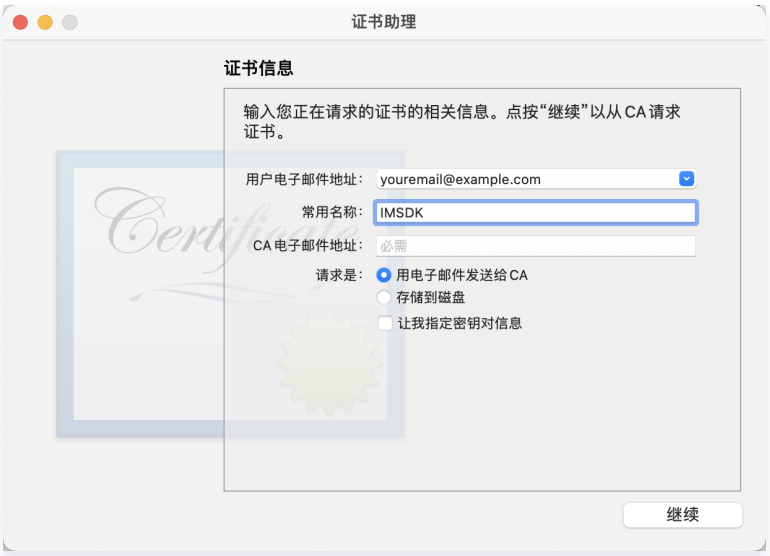
3. 我们先选择开发环境（Development）的 **Create Certificate**，系统将提示我们需要一个 Certificate Signing Request（CSR）。



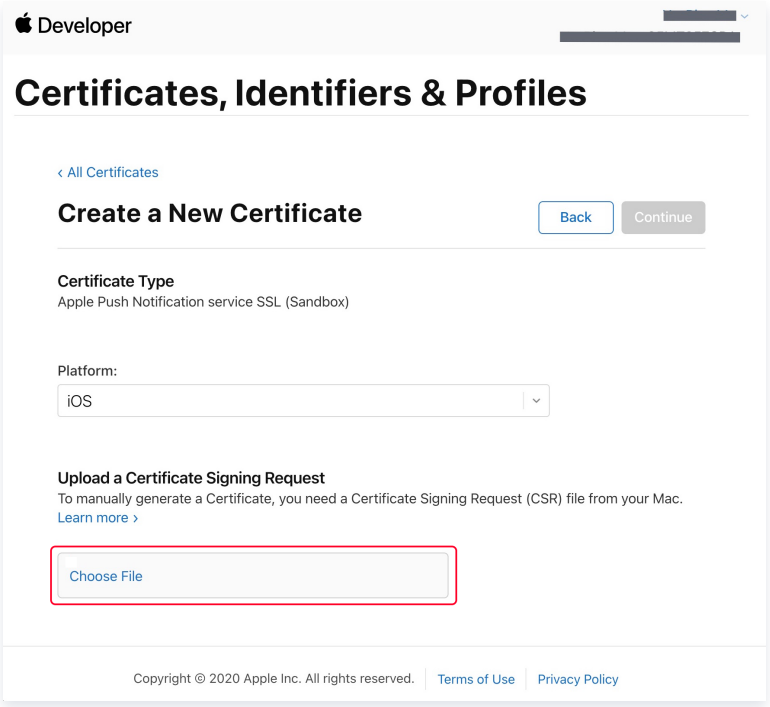
4. 在 Mac 上打开**钥匙串访问工具（Keychain Access）**，在菜单中选择**钥匙串访问 > 证书助理 > 从证书颁发机构请求证书**（`Keychain Access - Certificate Assistant - Request a Certificate From a Certificate Authority`）。



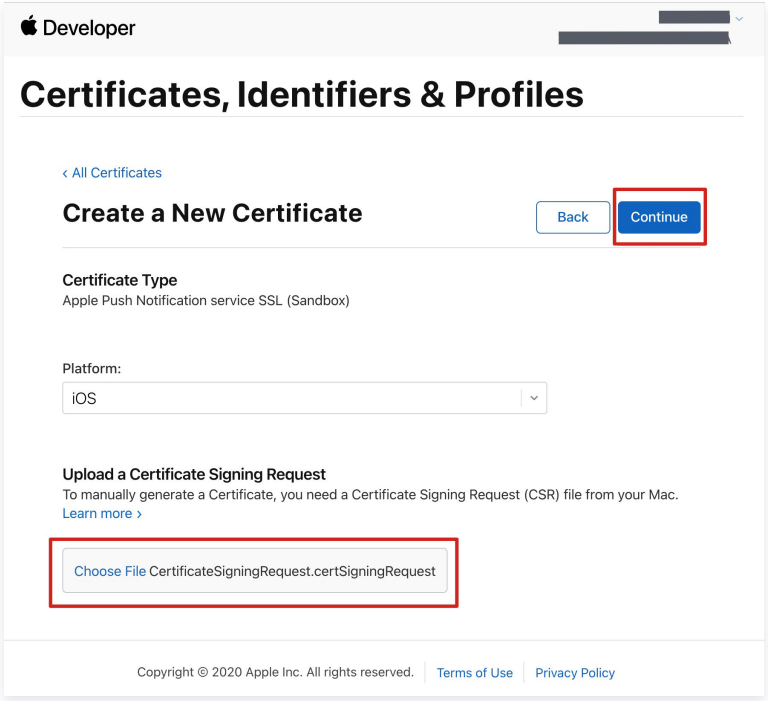
5. 输入用户电子邮件地址（您的邮箱）、常用名称（您的名称或公司名），选择**存储到磁盘**，单击**继续**，系统将生成一个 `*.certSigningRequest` 文件。



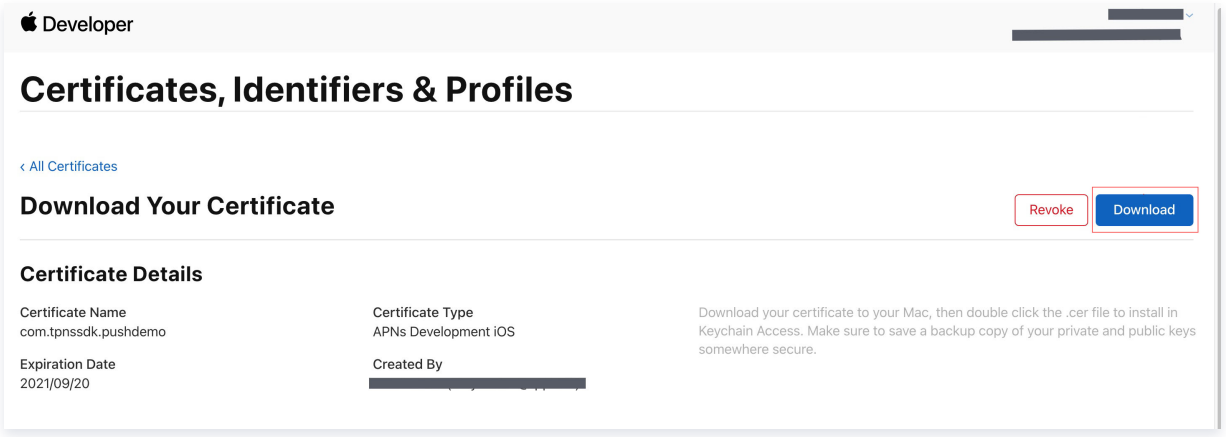
6. 返回上述 步骤3 中 Apple Developer 网站刚才的页面，单击 **Choose File** 上传生成的 `*.certSigningRequest` 文件。



7. 单击 **Continue**，即可生成推送证书。

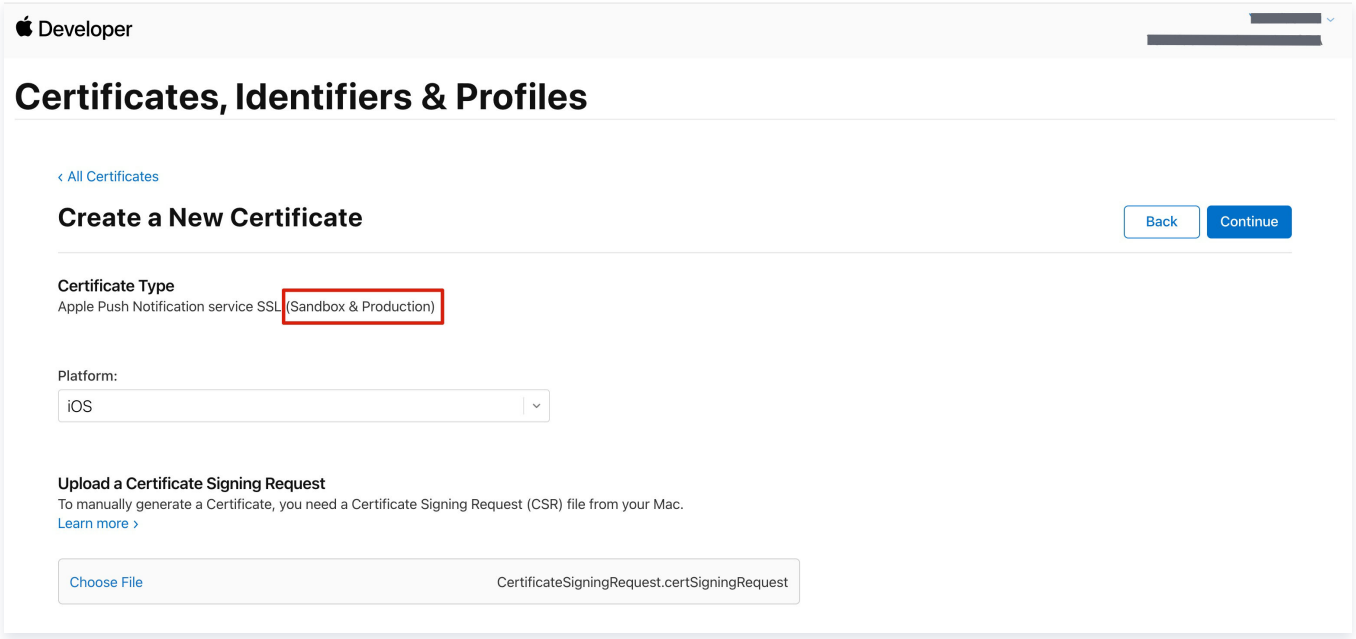


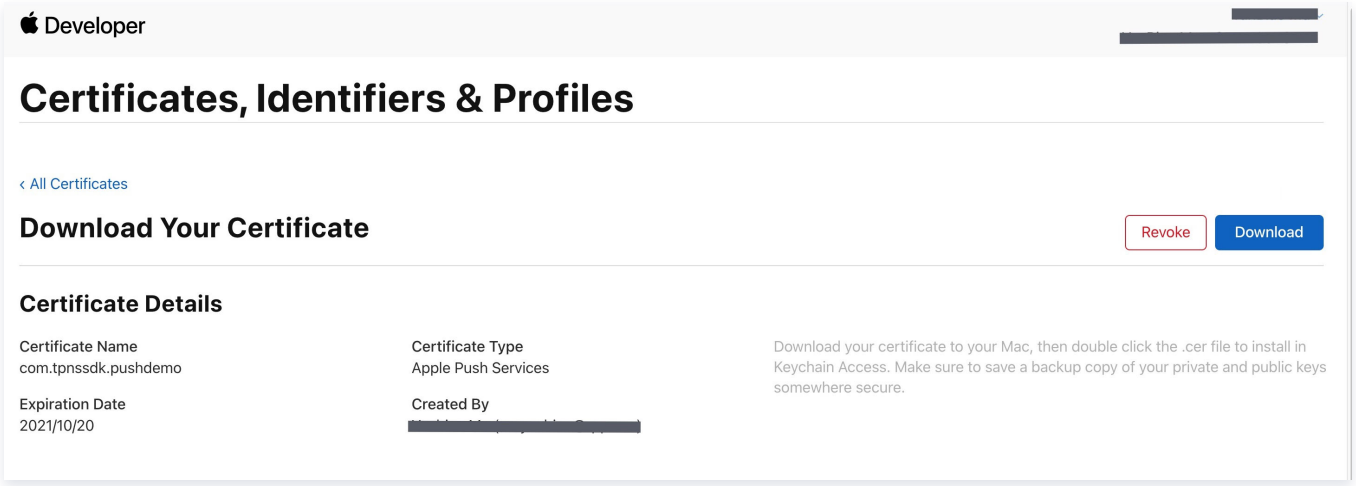
8. 单击 Download 下载开发环境的 `Development SSL Certificate` 到本地。



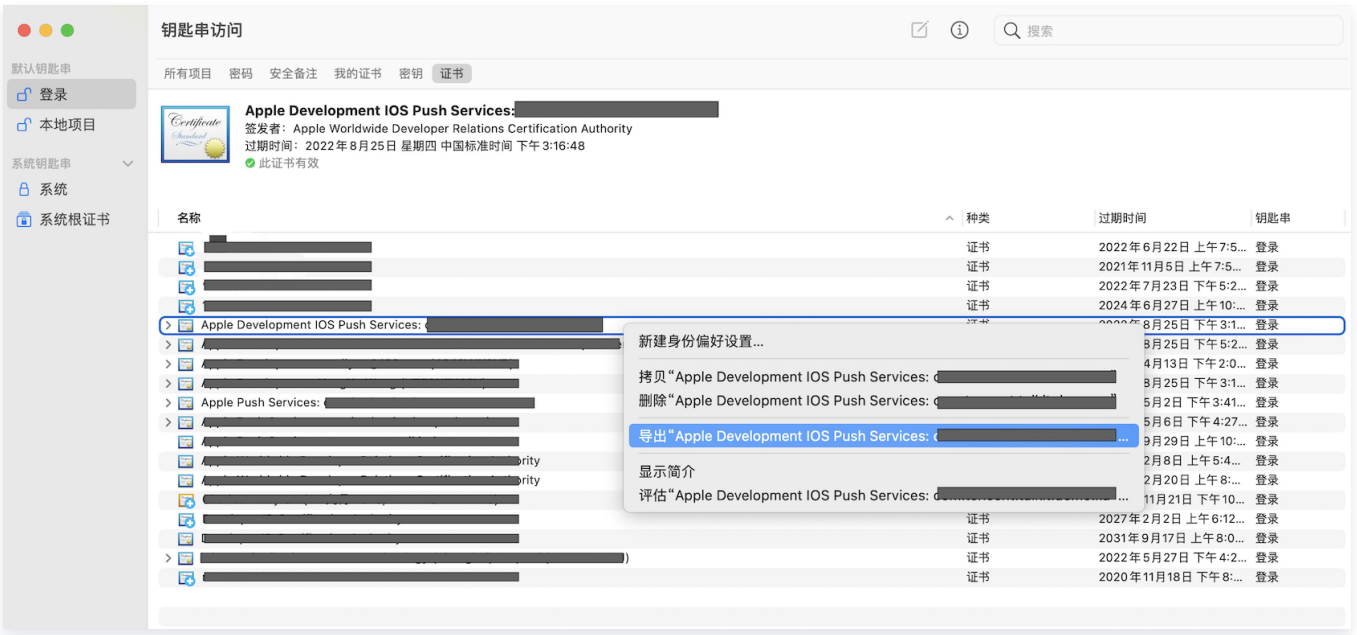
9. 再次按照上述步骤1 – 8，将生产环境的 `Production SSL Certificate` 下载到本地。

> **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。





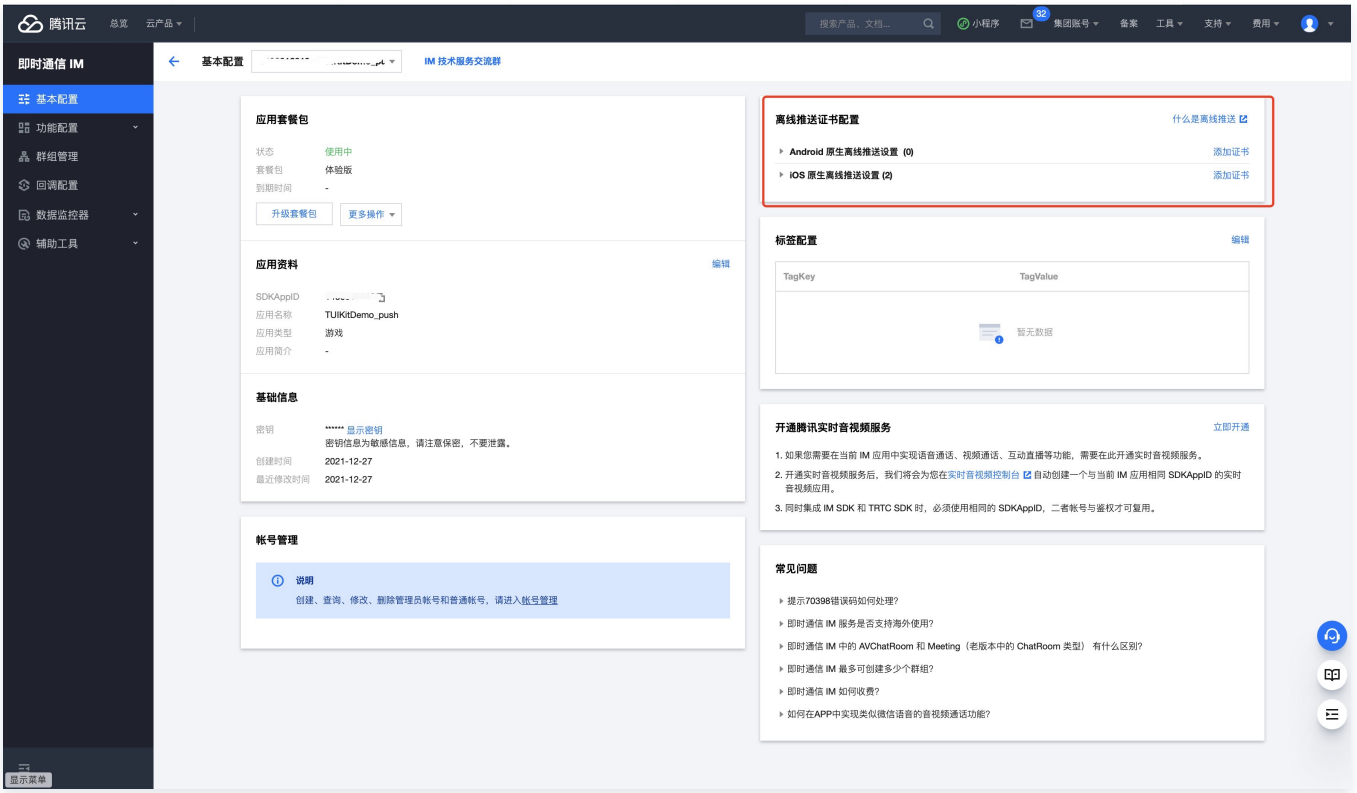
10. 双击打开下载的开发环境和生产环境的 `SSL Certificate`，系统会将其导入钥匙串中。
11. 打开钥匙串应用，在**登录 > 我的证书**，右键分别导出刚创建的开发环境（`Apple Development IOS Push Service`）和生产环境（`Apple Push Services`）的 `P12` 文件。



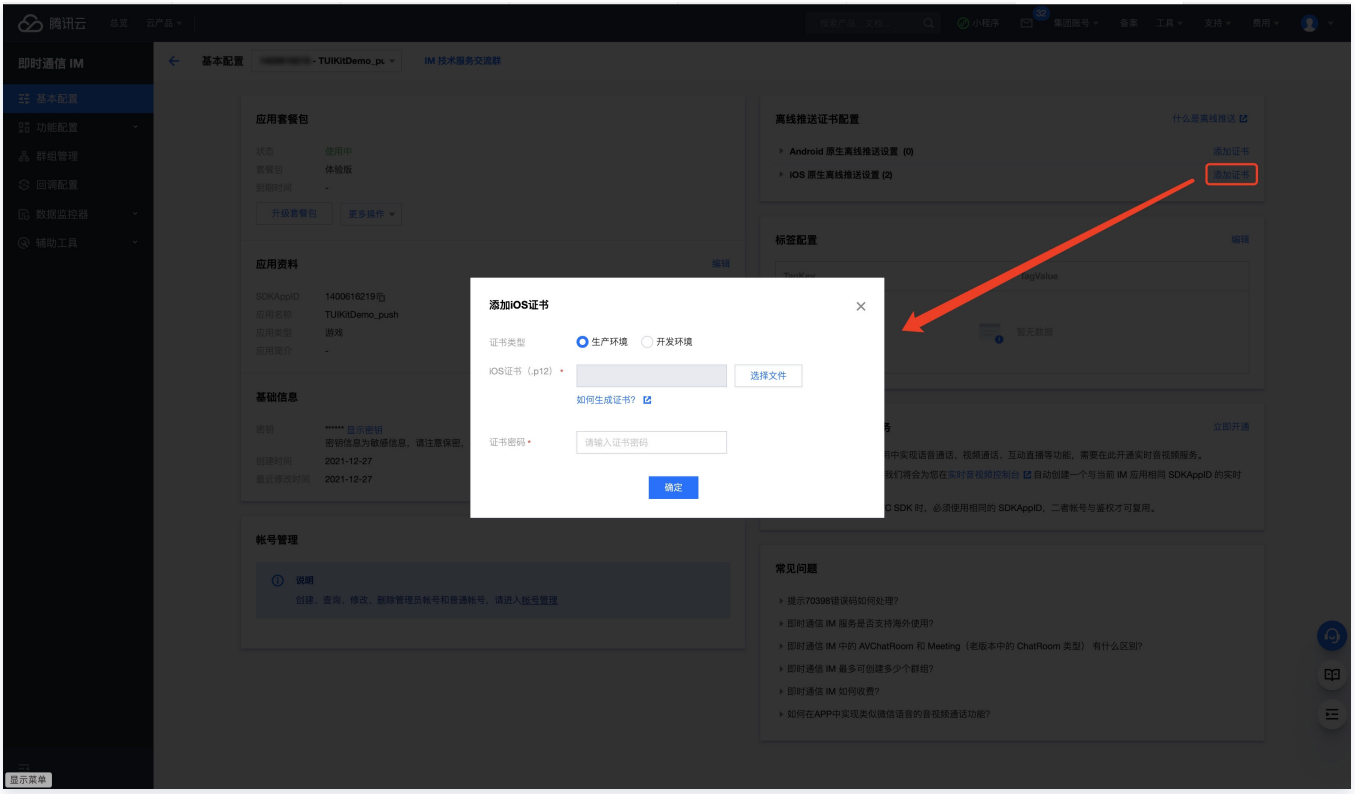
> **注意**
>
> 保存 `P12` 文件时，请务必要为其设置密码。

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。

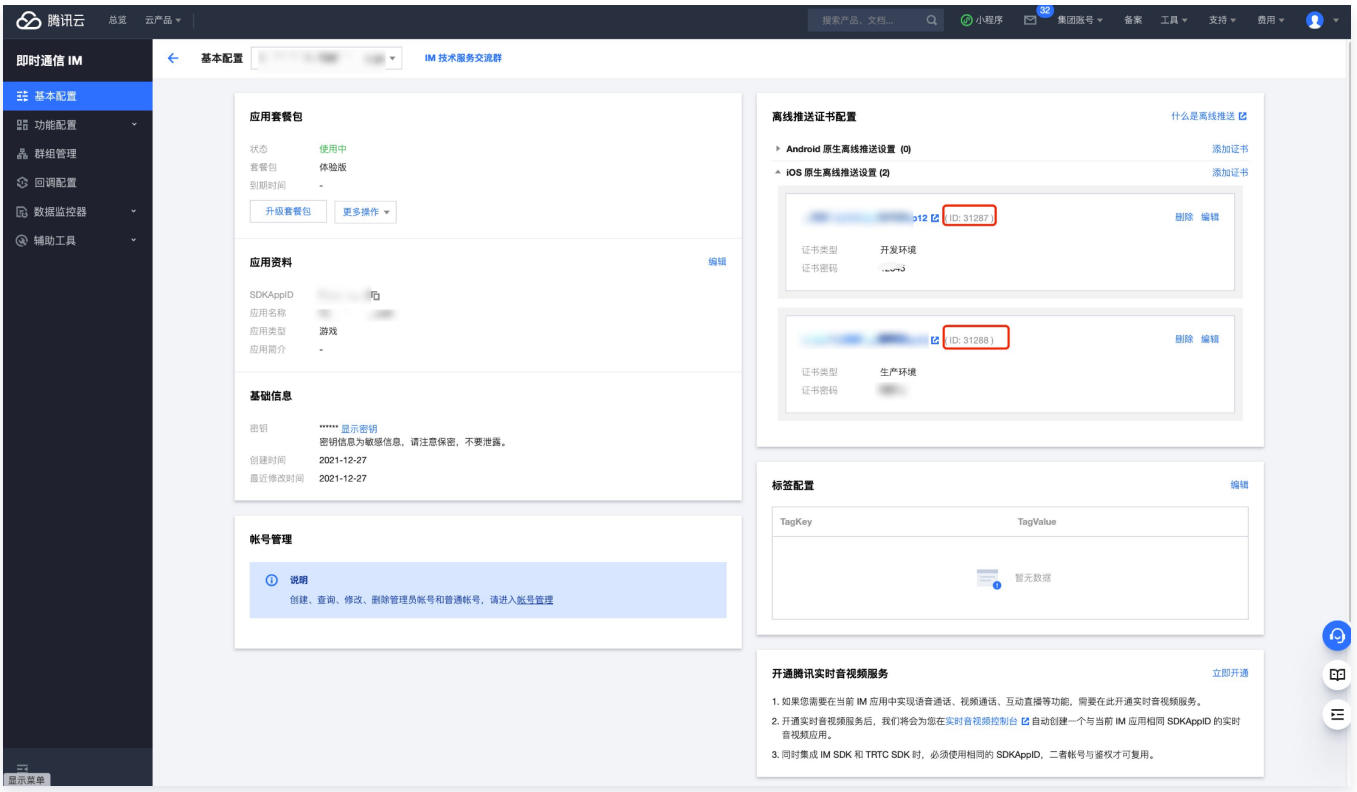


3. 单击 **iOS 原生离线推送设置**右侧的**添加证书**。
4. 选择证书类型，上传 iOS 证书（p.12），设置证书密码，单击**确认**。



> **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。



Android

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。

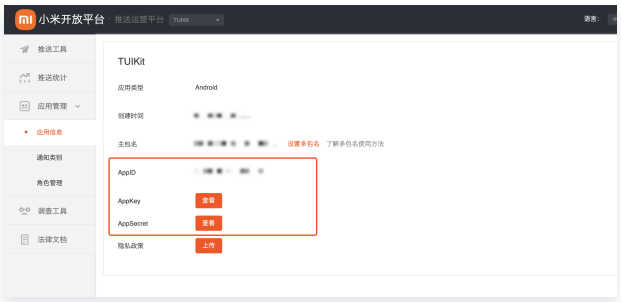
步骤2：IM 控制台配置

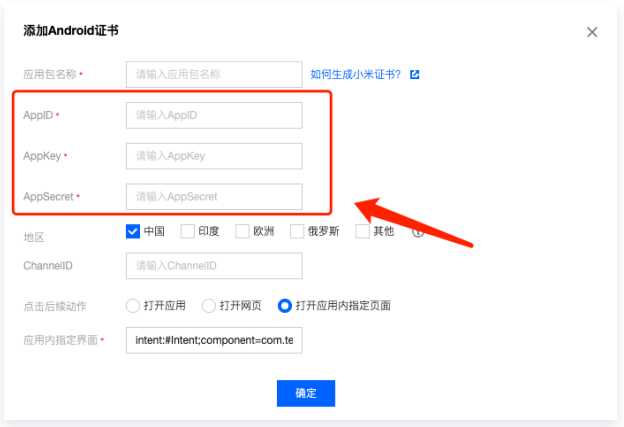
登录腾讯云 即时通信 IM 控制台 ，在**推送管理** > **接入设置**功能栏添加各个厂商推送证书，并将您在步骤一中获取的各厂商的 AppId、AppKey、AppSecret 等参数配置给添加的推送证书。

> **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

厂商推送平台	IM 控制台配置





华为

厂商推送平台

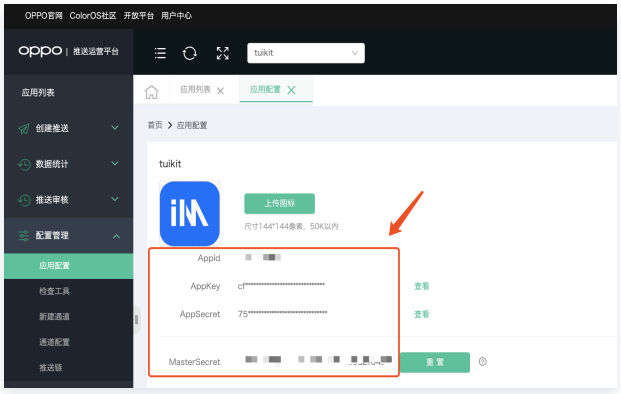
IM 控制台配置

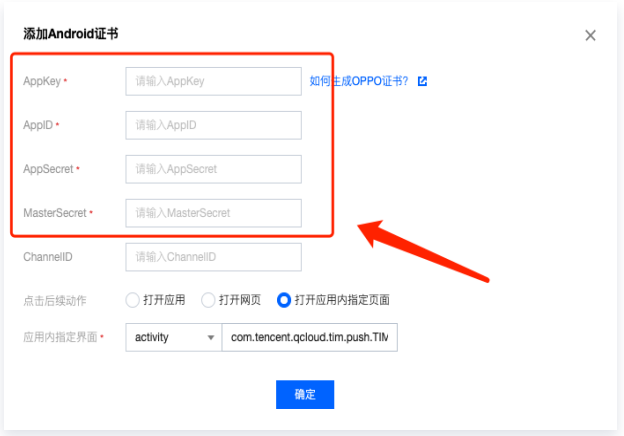
说明：
Client ID 对应 AppID，Client Secret 对应 AppSecret。

OPPO

厂商推送平台

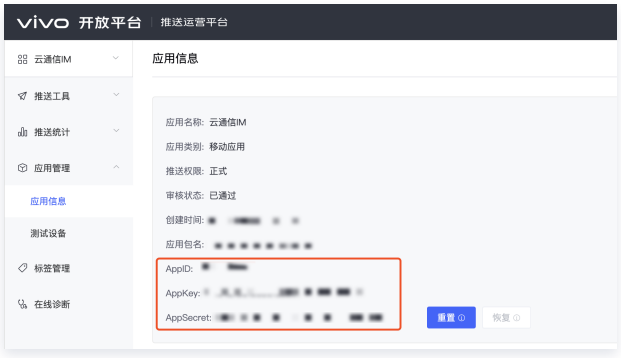
IM 控制台配置





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-vivo

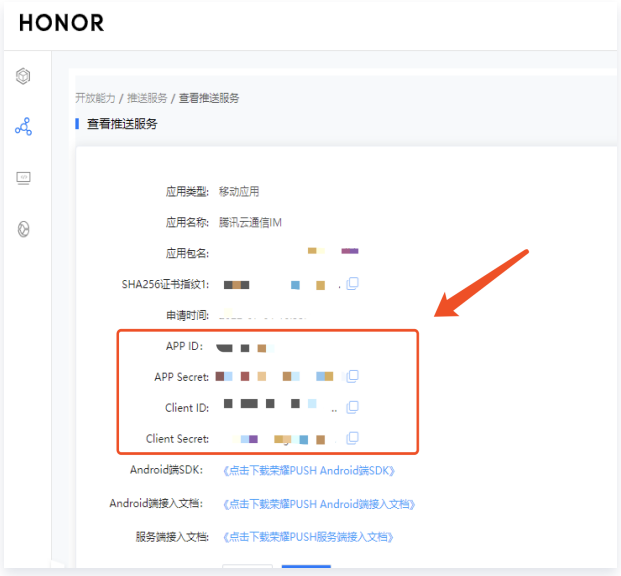
魅族

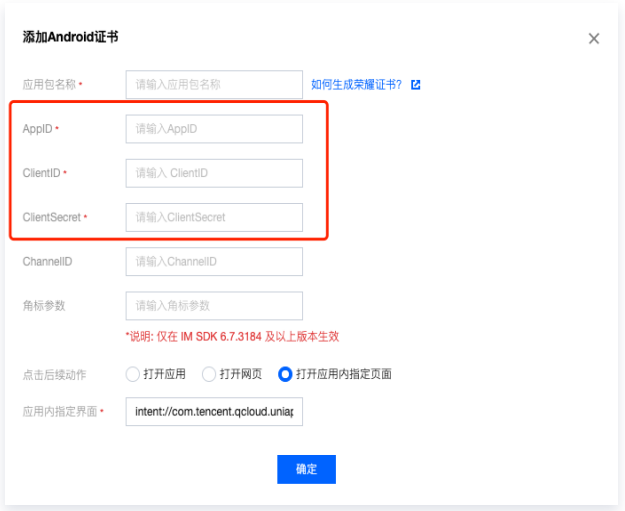
厂商推送平台	IM 控制台配置

回执配置请参考：消息触达统计配置 > 魅族

荣耀

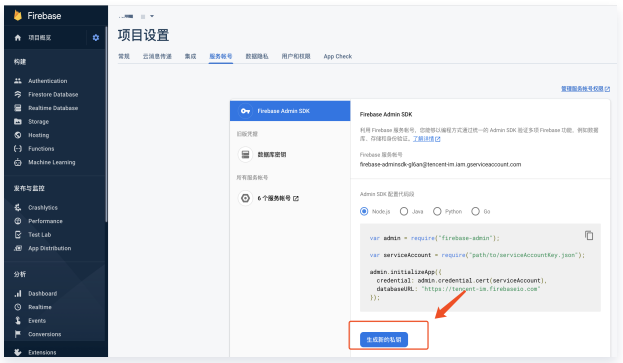
厂商推送平台	IM 控制台配置





Google FCM

厂商推送平台	IM 控制台配置


Firebase
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
Firebase Admin SDK
Firebase Admin SDK
数据库密钥
6 个服务帐号
Node.js
Java
Python
Go
生成新的私钥


添加Android证书
添加方式
上传证书
填写服务器密钥
上传证书
选择文件
如何生成谷歌（FCM）证书?
ChannelID
请输入ChannelID
确定

React Native

最近更新时间：2024-11-21 16:41:12

React Native 推送目前支持小米、华为、荣耀、OPPO、vivo、魅族、APNs、一加、realme、iQOO 、FCM 和 苹果等厂商通道。

iOS

集成 @tencentcloud/react-native-push 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。

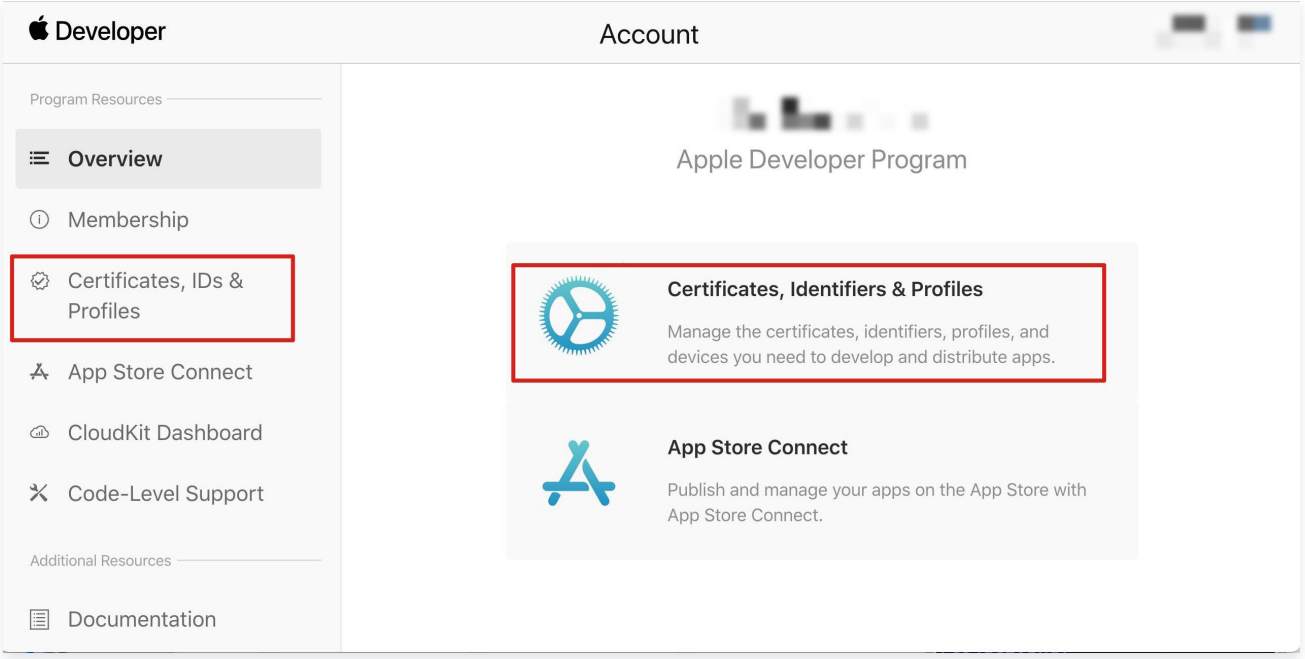
	证书类型	有效期和管理	安全性	灵动岛
p12 证书	p12 证书是一个包含公钥和私钥的二进制文件，用于基于证书的身份验证。它将公钥证书和私钥捆绑在一个文件中，扩展名为 .p12 或 .pfx。	p12 证书通常有一年的有效期，过期后需要重新生成和部署。每个应用程序都需要单独的 P12 证书来处理推送通知。	p12 证书使用基于证书的身份验证，需要在服务器上存储私钥。这可能会增加安全风险，因为私钥可能会被未经授权的用户访问。	不支持
p8 证书	p8 证书是一个 Auth Key（授权密钥），用于基于令牌的身份验证。它是一个包含私钥的文本文件，扩展名为 .p8。	p8 证书没有到期日期，因此您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

一、使用 p12 证书（传统推送证书）

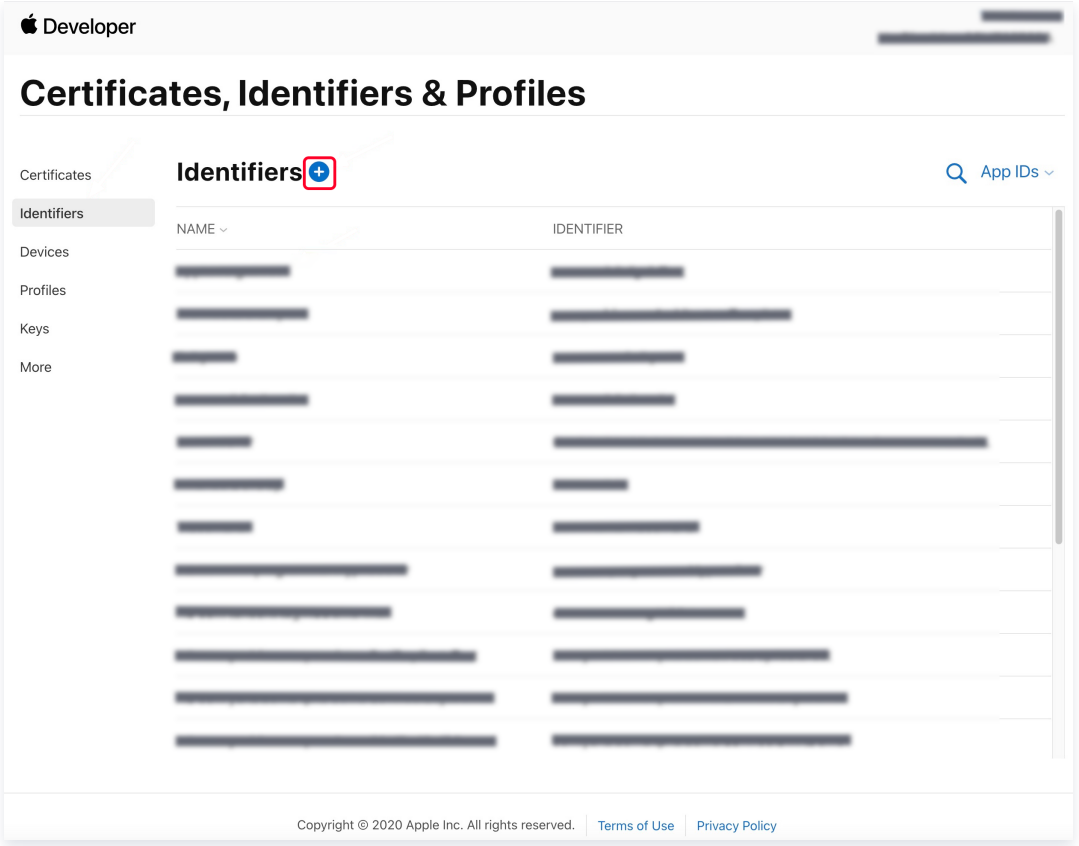
步骤1：申请 APNs 证书

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。



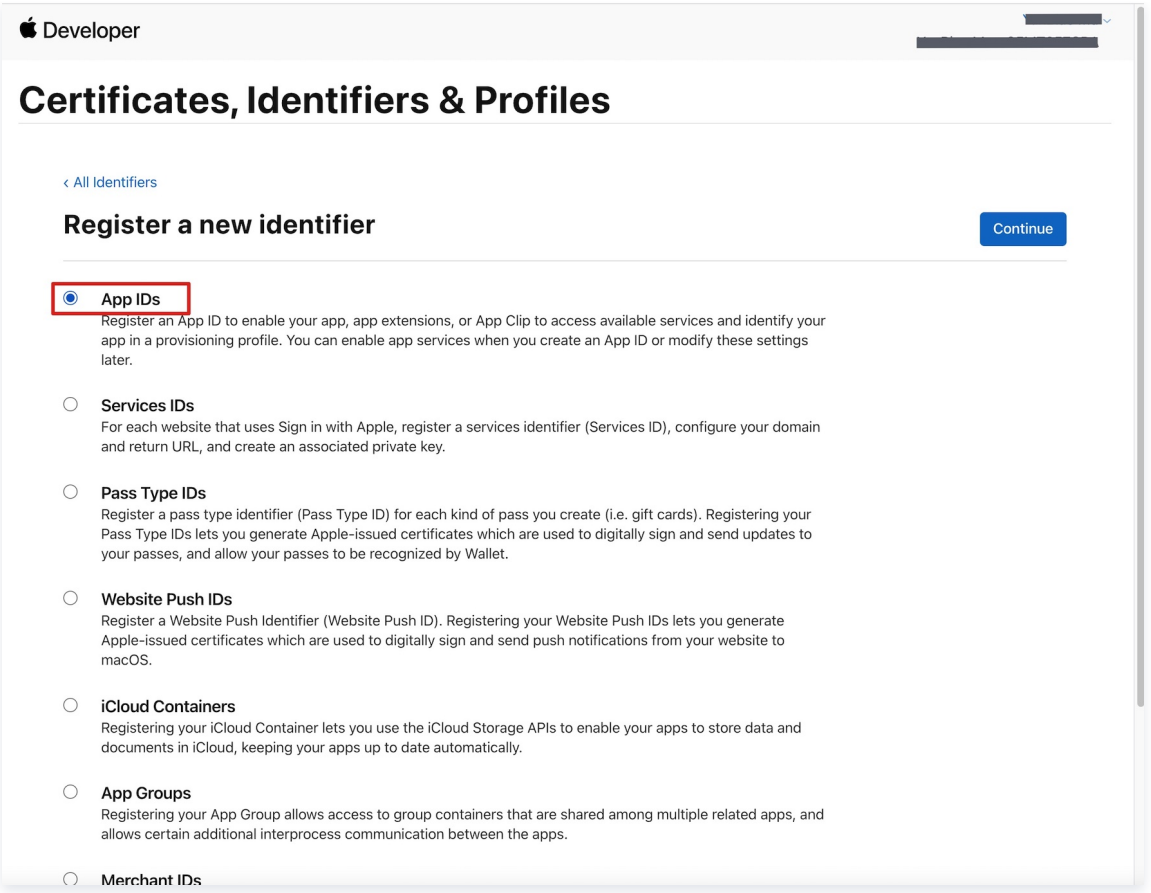
2. 单击 Identifiers 右侧的 +。



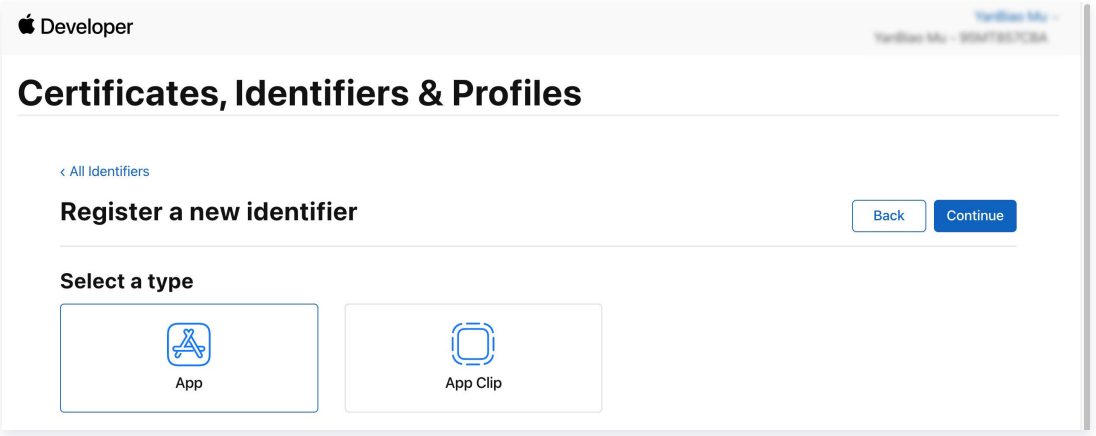
3. 您可以参见如下步骤新建一个 AppID，或者在您原有的 AppID 上增加 `Push Notification` 的 `Service`。

> **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。

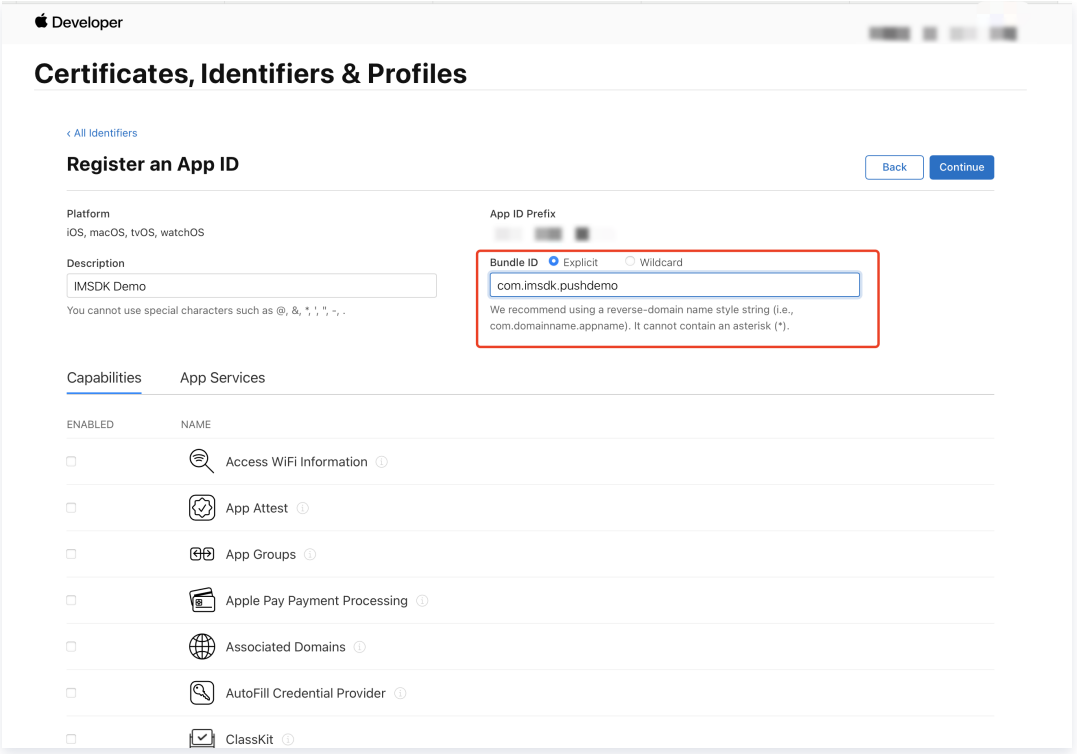
4. 勾选 **App IDs**，单击 **Continue** 进行下一步。



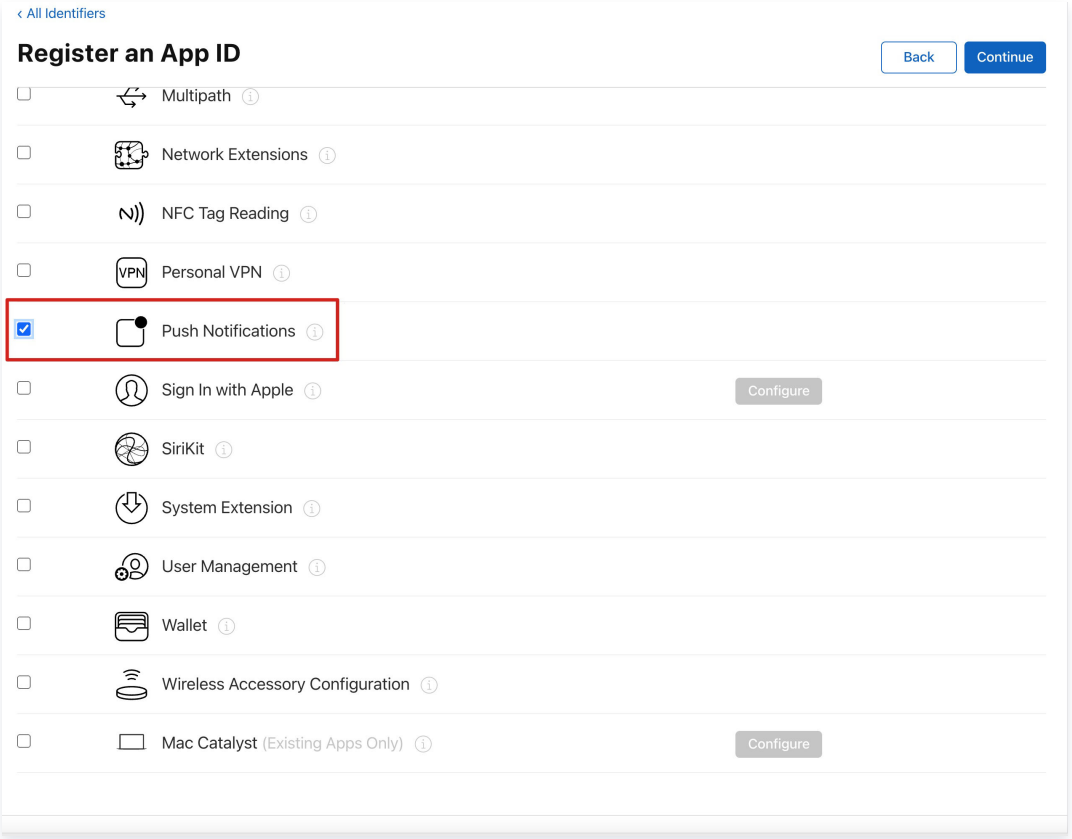
5. 选择 **App**，单击 **Continue** 进行下一步。



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

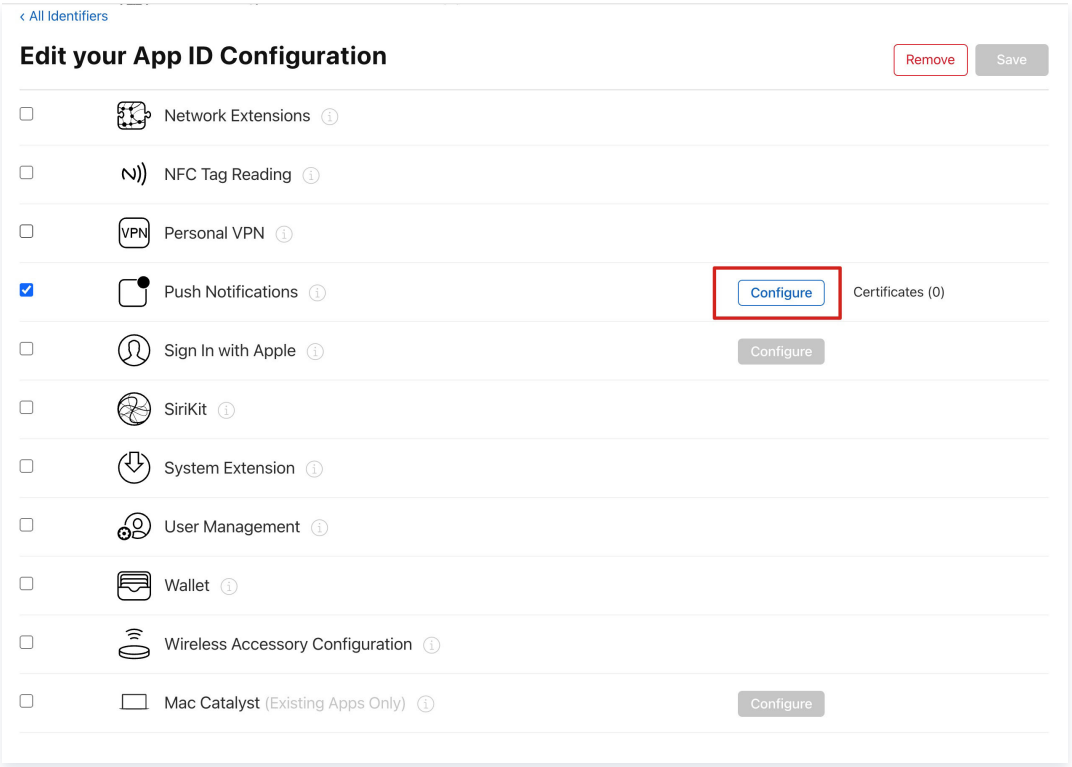


7. 勾选 **Push Notifications**，开启远程推送服务。

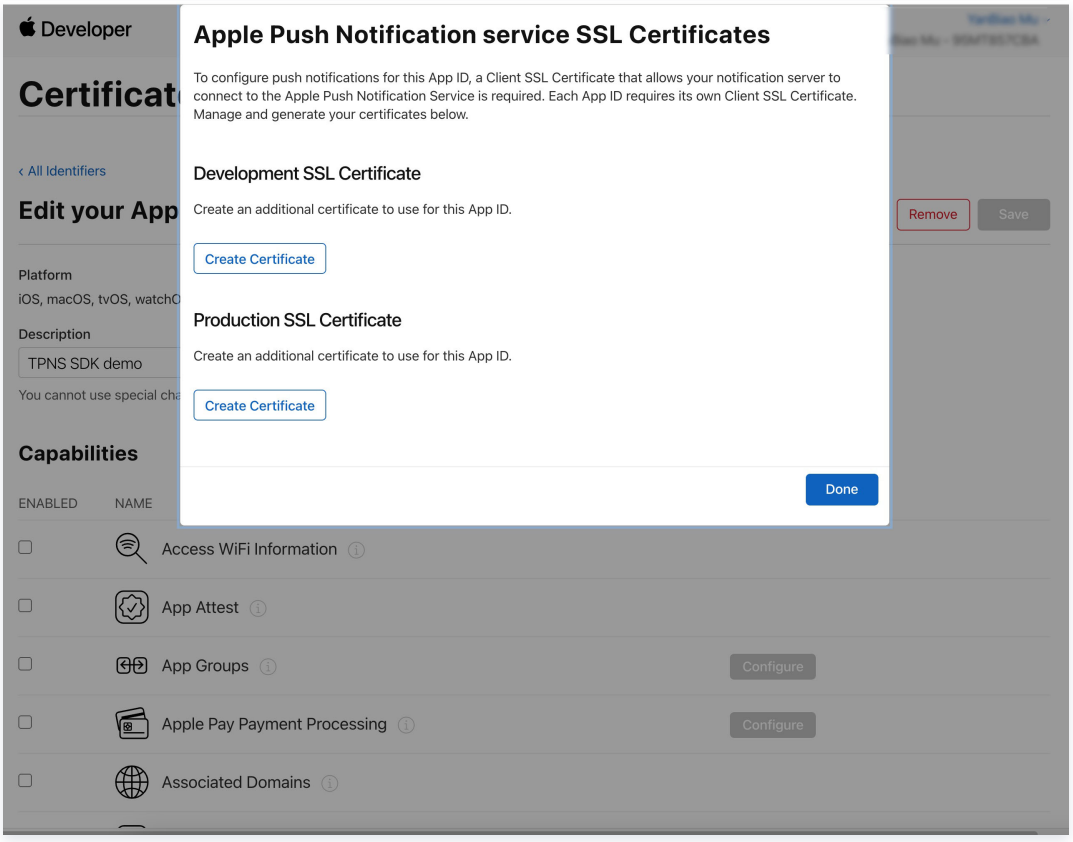


生成证书

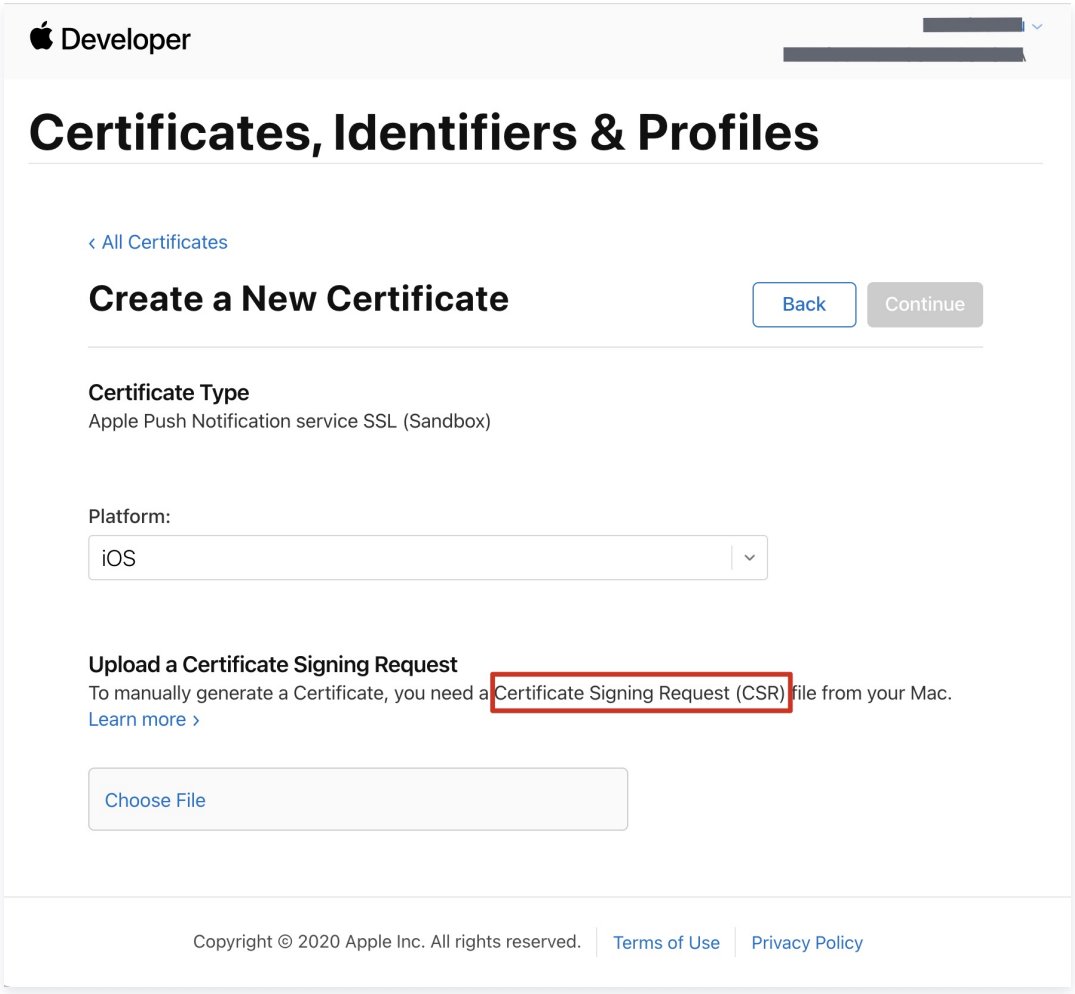
1. 选中您的 AppID，选择 **Configure**。



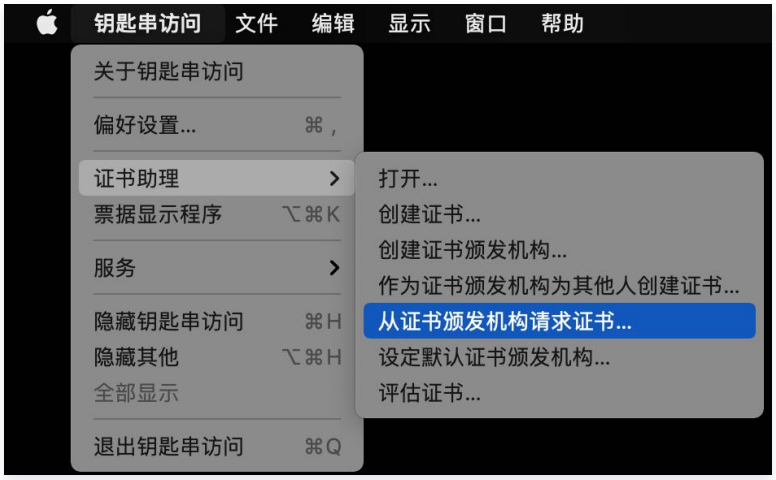
2. 可以看到在 **Apple Push Notification service SSL Certificates** 窗口中有两个 `SSL Certificate`，分别用于开发环境（Development）和生产环境（Production）的远程推送证书，如下图所示：



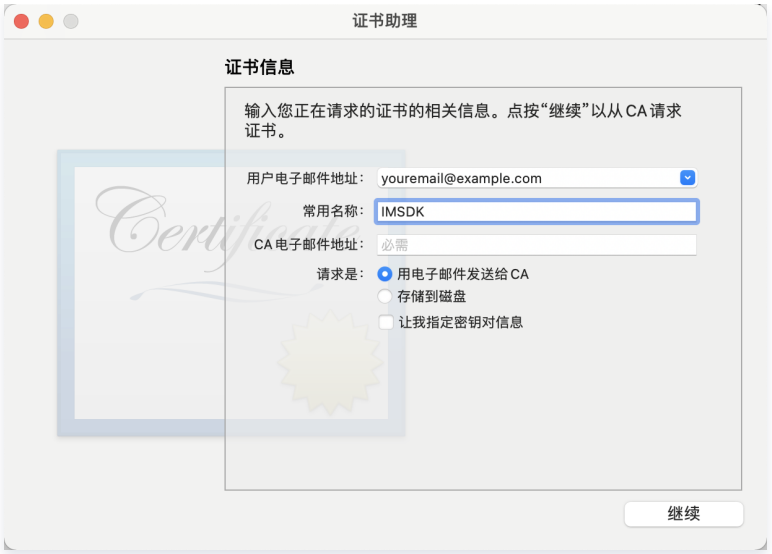
3. 我们先选择开发环境（Development）的 **Create Certificate**，系统将提示我们需要一个 Certificate Signing Request（CSR）。



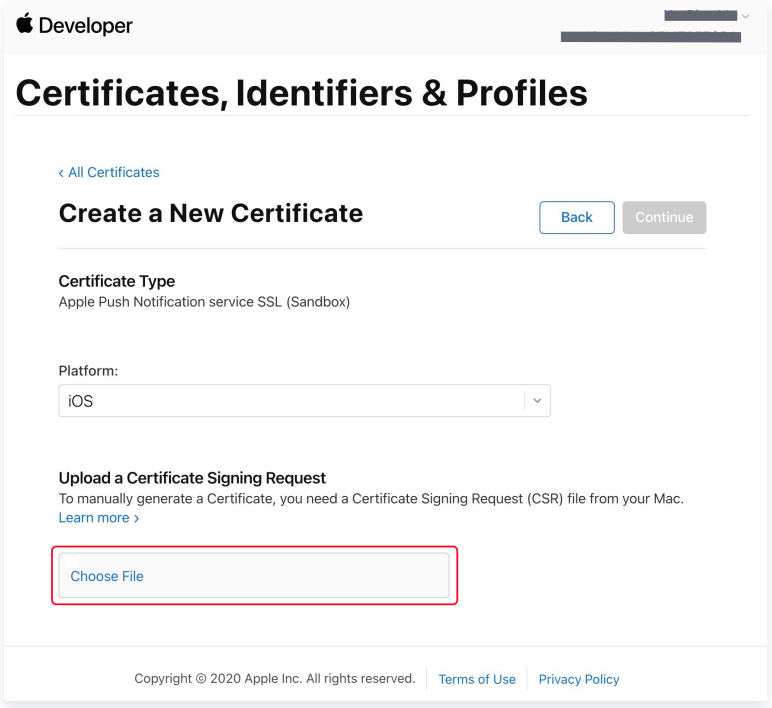
4. 在 Mac 上打开**钥匙串访问工具（Keychain Access）**，在菜单中选择**钥匙串访问 > 证书助理 > 从证书颁发机构请求证书**（`Keychain Access - Certificate Assistant - Request a Certificate From a Certificate Authority`）。



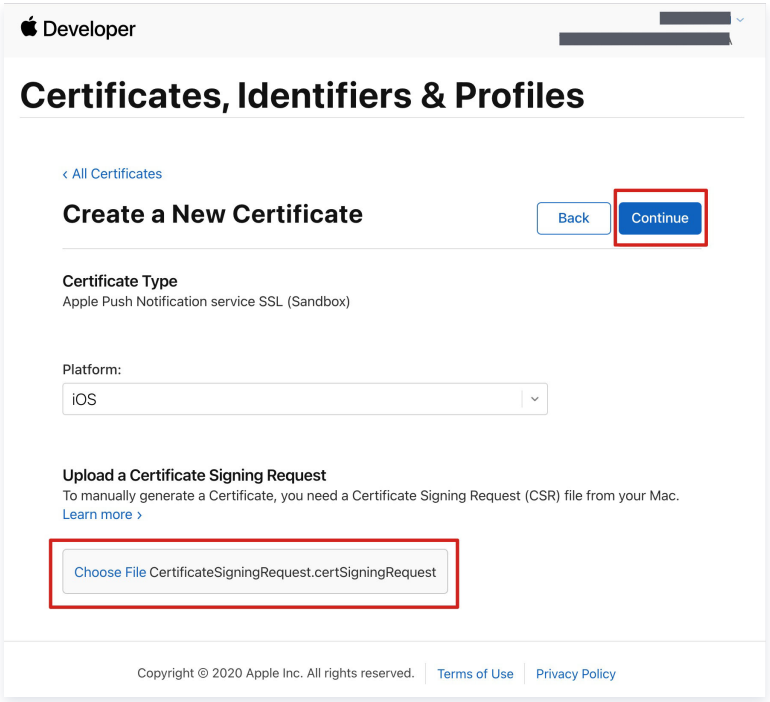
5. 输入用户电子邮件地址（您的邮箱）、常用名称（您的名称或公司名），选择**存储到磁盘**，单击**继续**，系统将生成一个 `*.certSigningRequest` 文件。



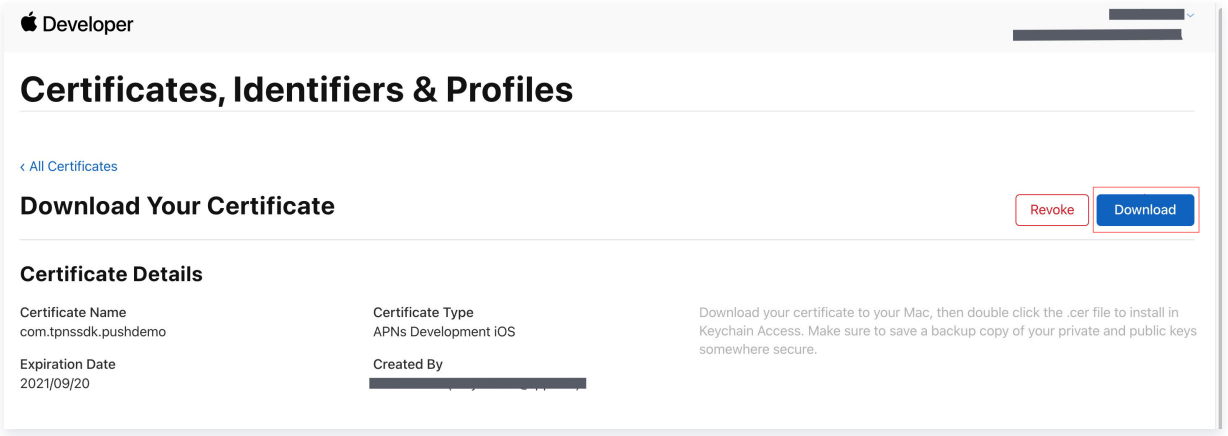
6. 返回上述 步骤3 中 Apple Developer 网站刚才的页面，单击 **Choose File** 上传生成的 `*.certSigningRequest` 文件。



7. 单击 **Continue**，即可生成推送证书。

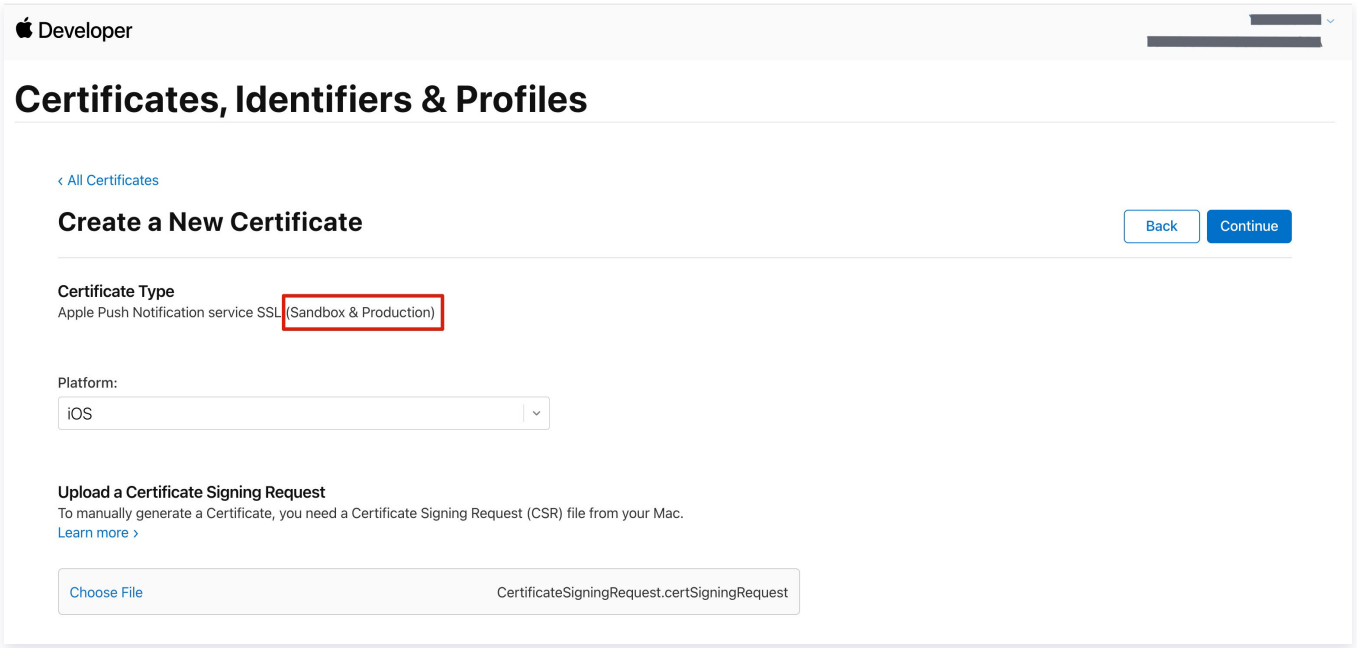


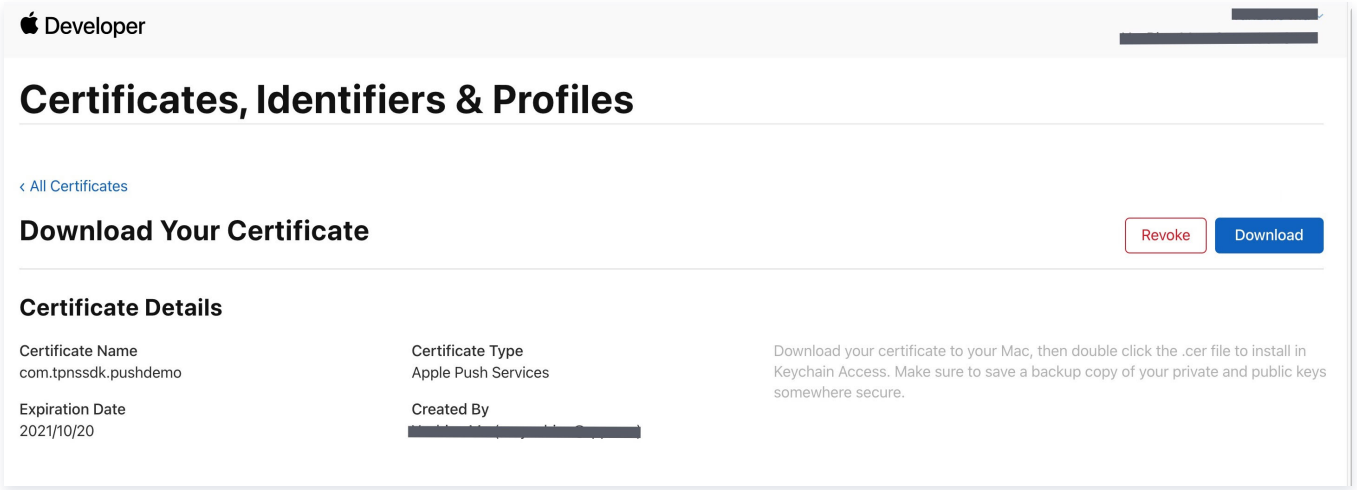
8. 单击 **Download** 下载开发环境的 `Development SSL Certificate` 到本地。



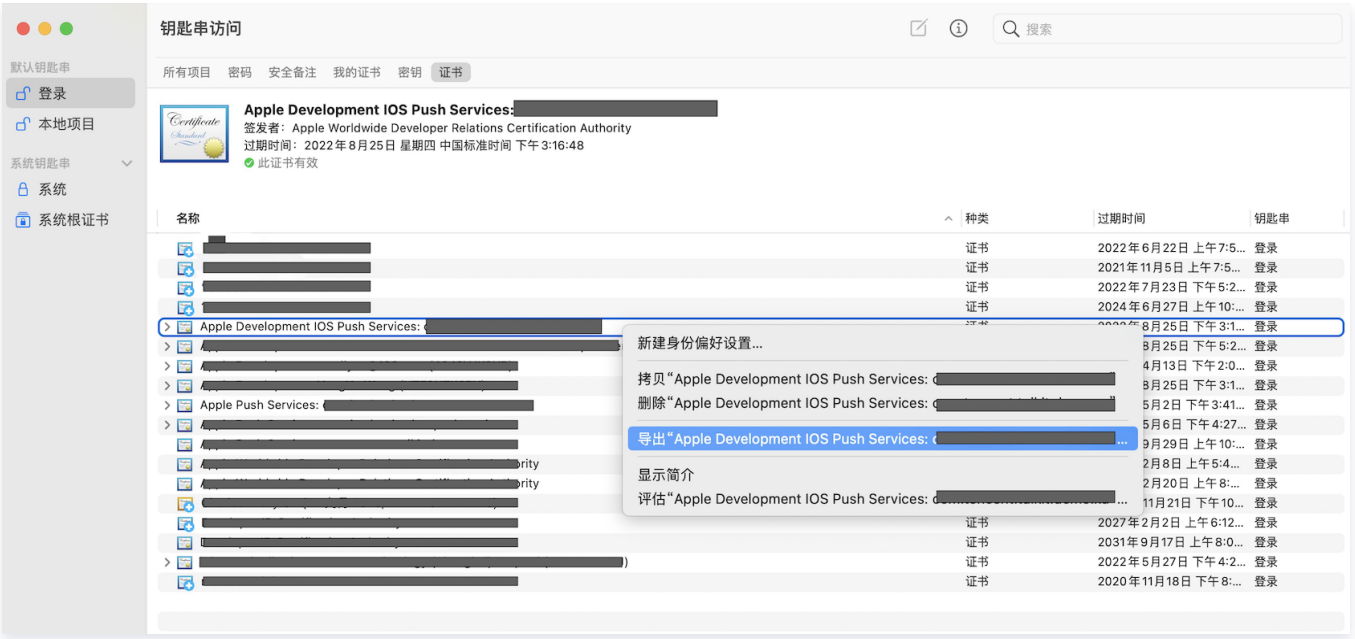
9. 再次按照上述步骤1 – 8，将生产环境的 `Production SSL Certificate` 下载到本地。

> **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。





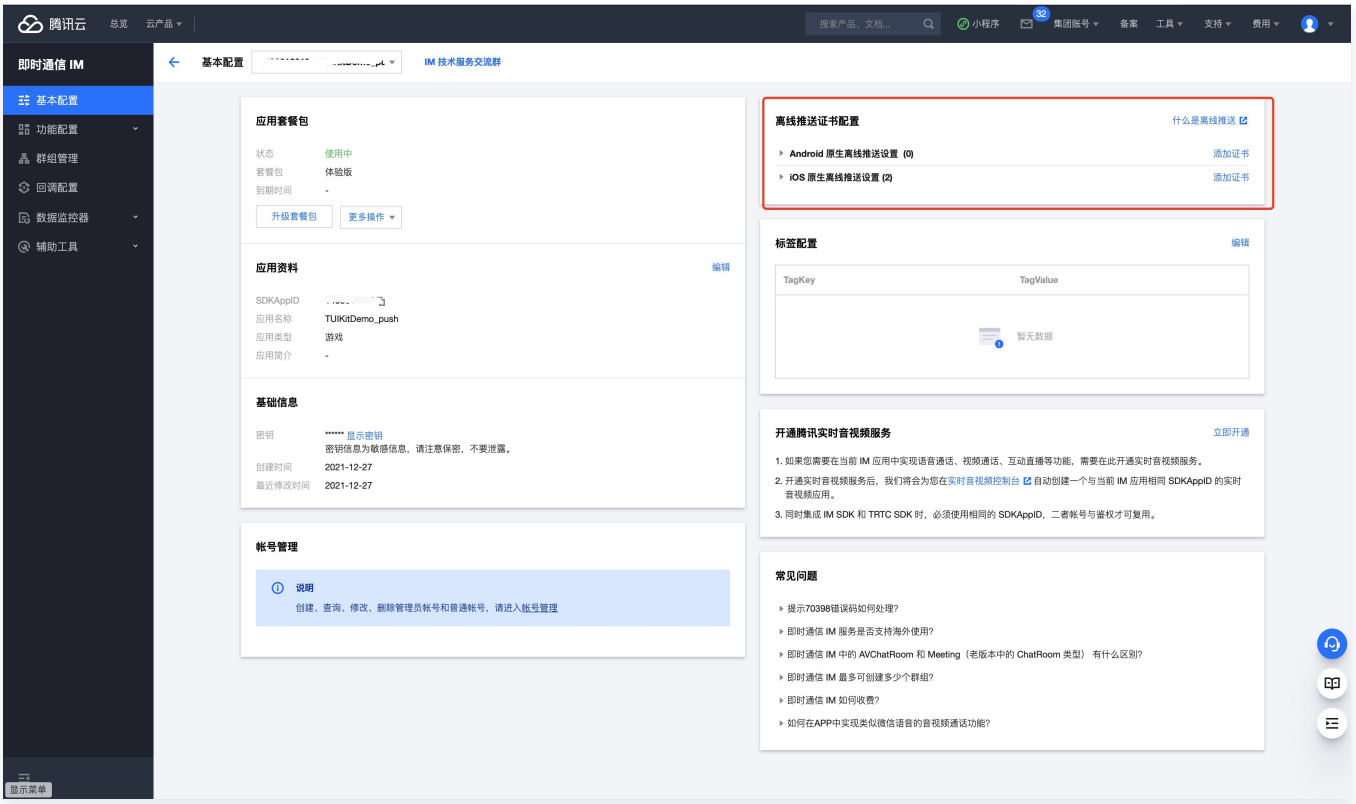
10. 双击打开下载的开发环境和生产环境的 `SSL Certificate`，系统会将其导入钥匙串中。
11. 打开钥匙串应用，在**登录 > 我的证书**，右键分别导出刚创建的开发环境（ `Apple Development IOS Push Service` ）和生产环境（ `Apple Push Services` ）的 `p12` 文件。



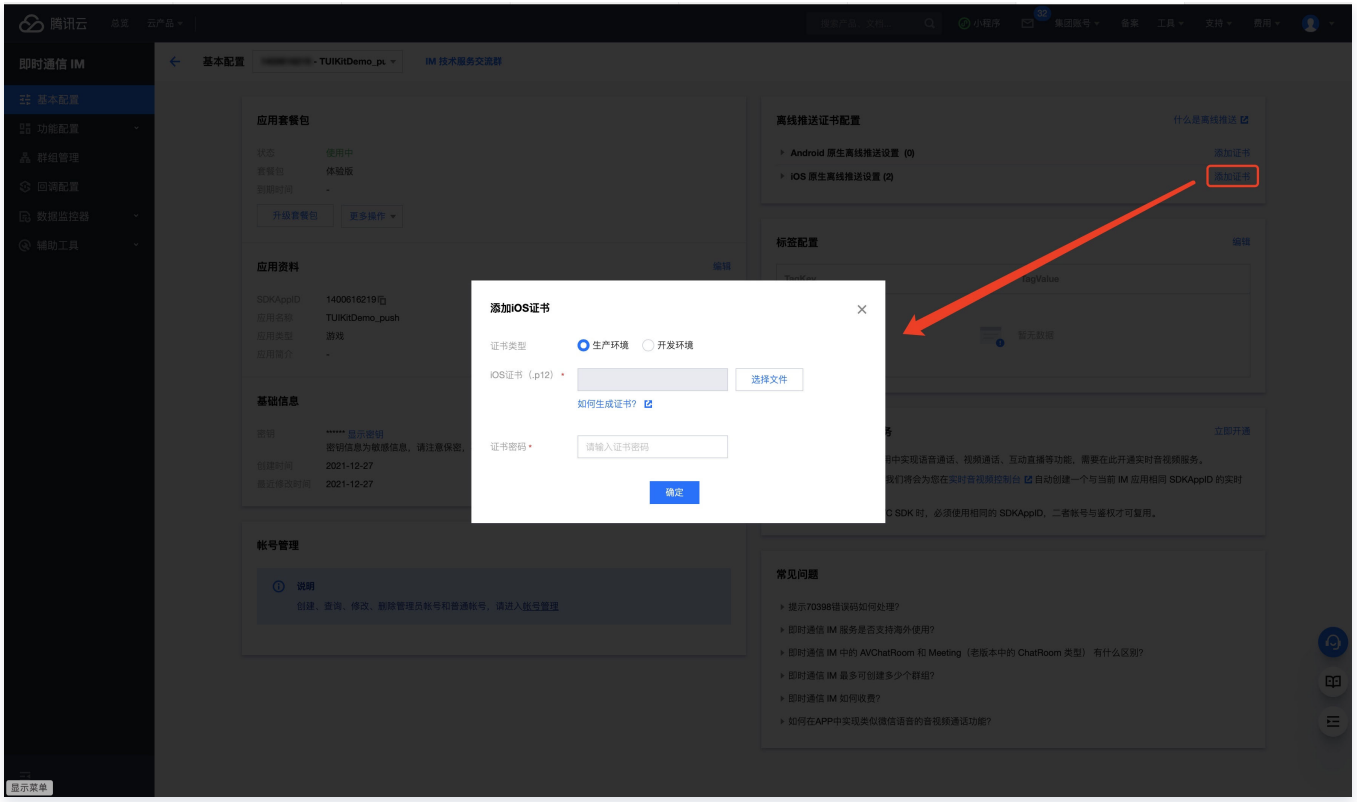
> **注意**
> 保存 `.p12` 文件时，请务必为其设置密码。

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。



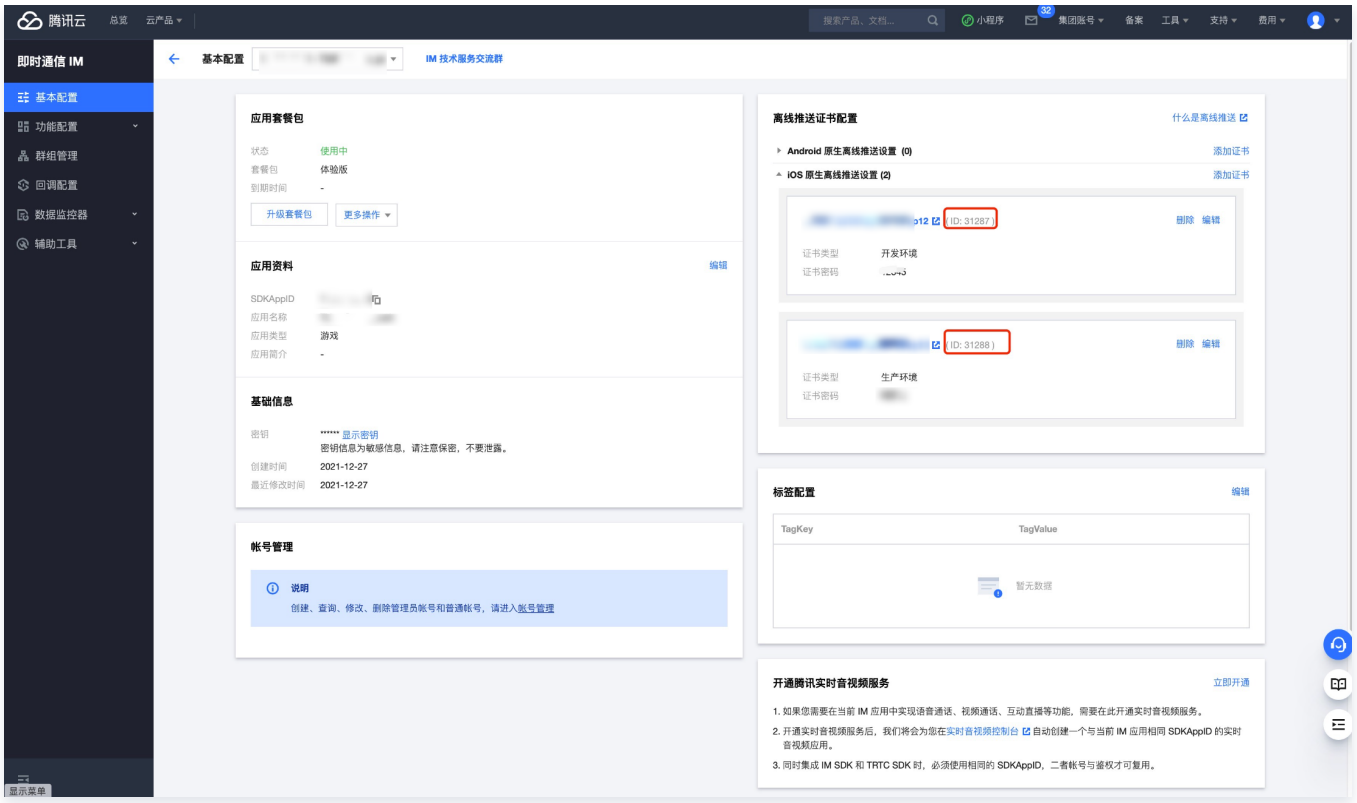
3. 单击 **iOS 原生离线推送设置**右侧的**添加证书**。
4. 选择证书类型，上传 iOS 证书（p.12），设置证书密码，单击**确认**。



说明：

- 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。

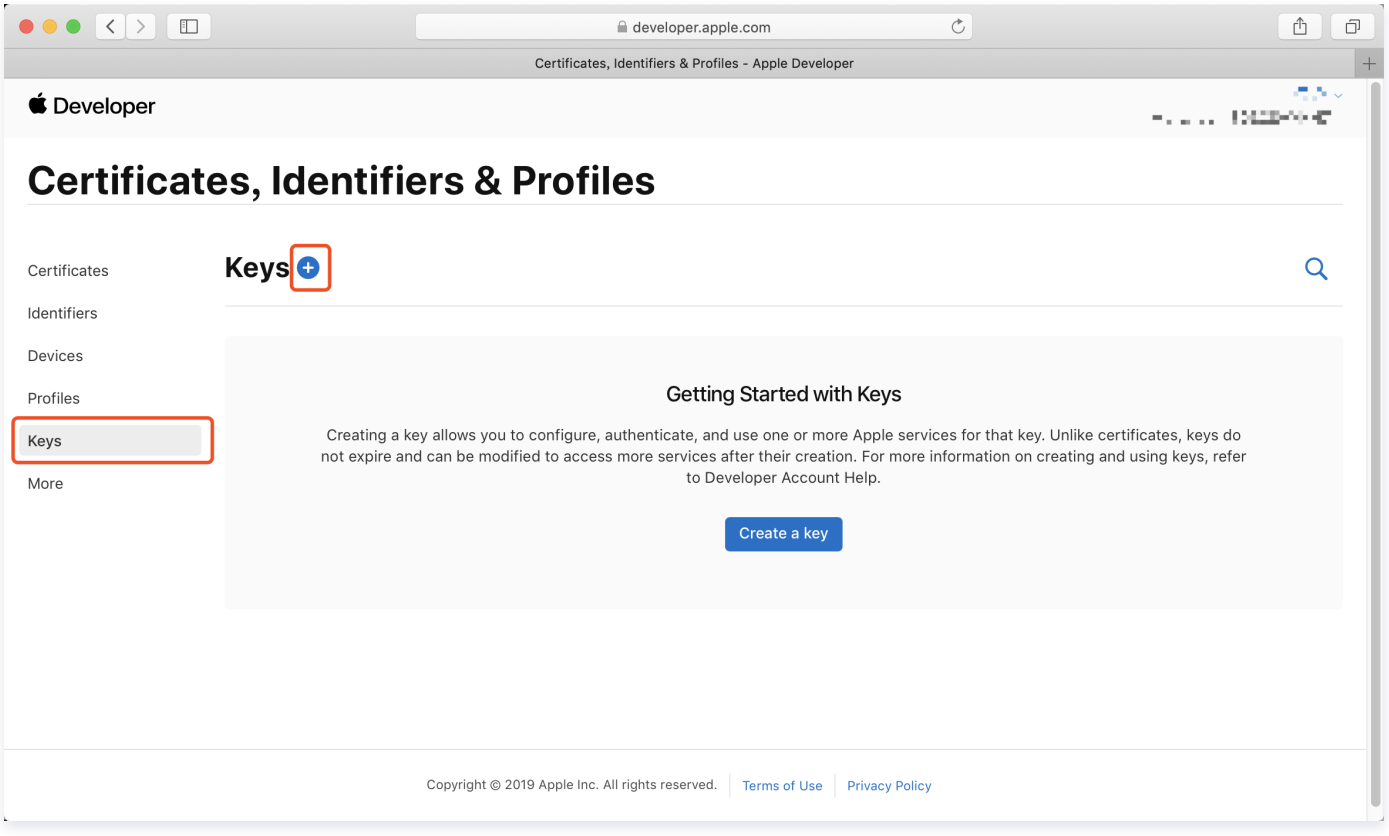


二、使用 p8 证书（支持灵动岛推送）

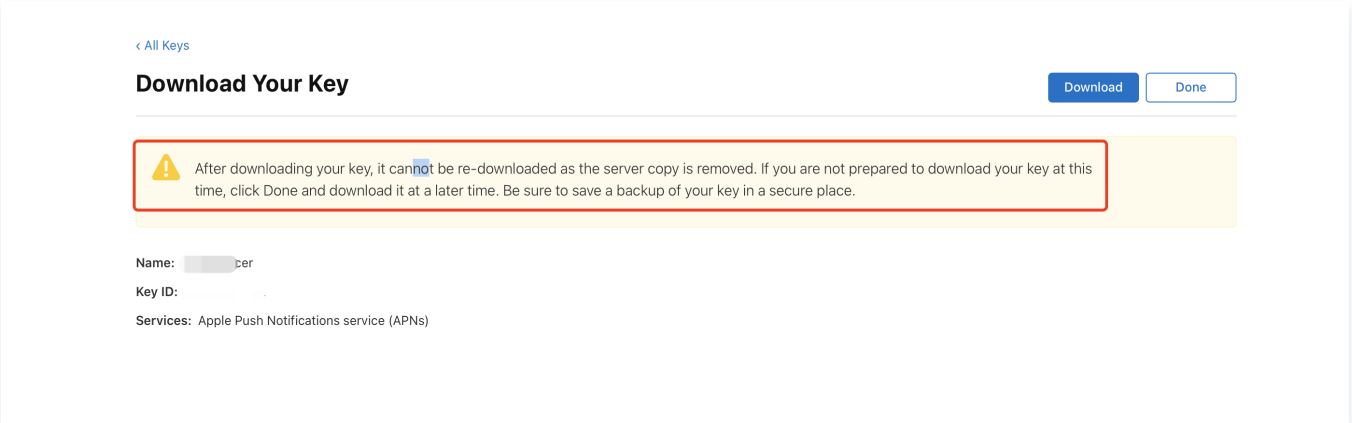
p8 证书：p8 证书没有到期日期，因此您无需担心证书过期。此外，使用 p8 证书可以简化证书管理，因为您可以使用一个p8 证书为多个应用程序提供推送通知服务。另外，p8 证书支持灵动岛推送。

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** 。

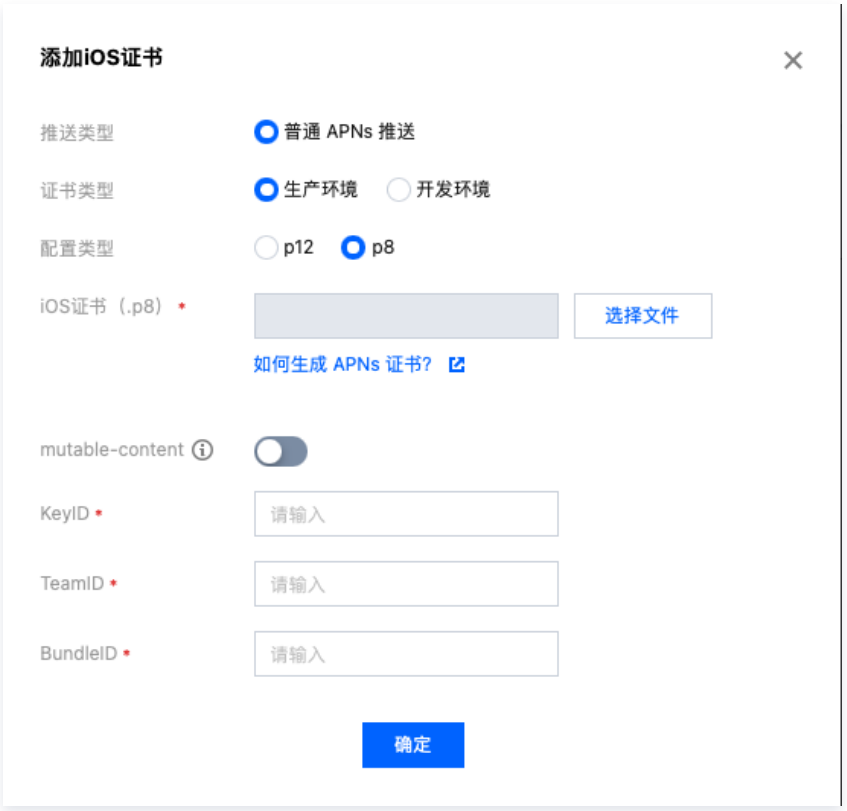


确认并生成密钥：在确认页面核对您的密钥信息，然后单击 **Register** 。接下来，您将看到一个页面提示您下载密钥。单击 **Download**，将生成的 .p8 文件保存到您的计算机上。

> **说明：**
> - 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



- **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。

Android

操作步骤

步骤1：注册应用到厂商推送平台

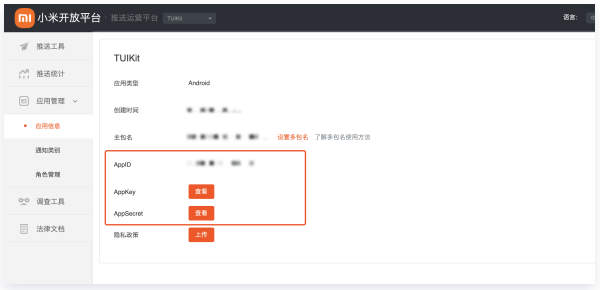
离线推送需要将您自己的应用注册到各个厂商的推送平台，得到 AppID 和 AppKey 等参数，来实现离线推送功能。目前国内支持的手机厂商有：小米、华为、荣耀、OPPO、VIVO、魅族，境外支持 Google FCM。

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。

> **说明：**
> 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

厂商推送平台	IM 控制台配置
	

华为

厂商推送平台	IM 控制台配置
	

> **说明：**
> Client ID 对应 AppID，Client Secret 对应 AppSecret。

OPPO

厂商推送平台	IM 控制台配置

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-vivo

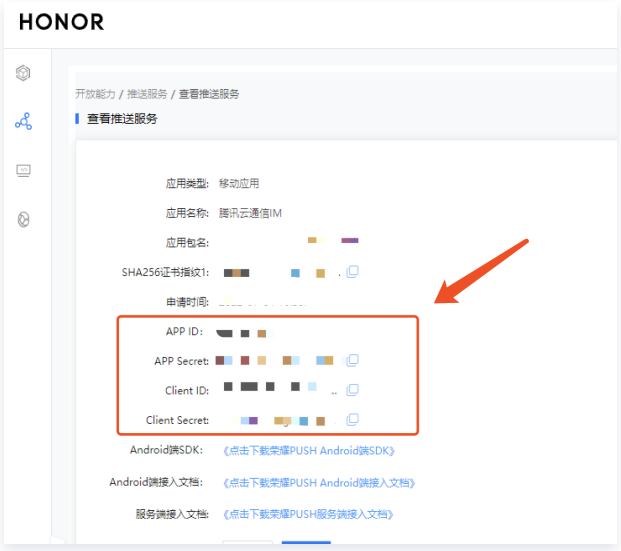
魅族

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

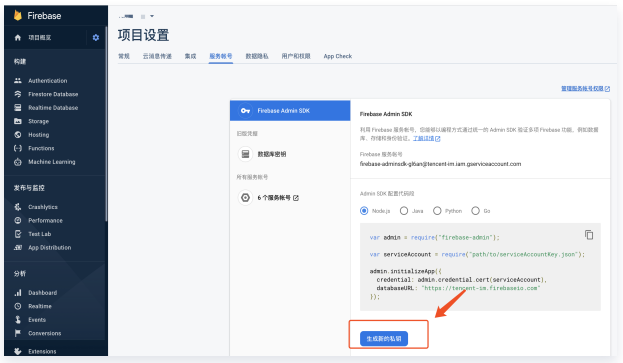
荣耀

厂商推送平台	IM 控制台配置



Google FCM

厂商推送平台	IM 控制台配置


Firebase
项目设置
Authentication
Firestore Database
Realtime Database
Storage
Hosting
Functions
Machine Learning
Crashlytics
Performance
Test Lab
App Distribution
Dashboard
Realtime
Events
Conversions
Extensions
Firebase Admin SDK
Firebase Admin SDK
数据库密钥
6 个服务帐号
Node.js
Java
Python
Go
生成新的私钥


添加Android证书
添加方式
上传证书
填写服务器密钥
上传证书
选择文件
如何生成谷歌（FCM）证书?
ChannelID
请输入ChannelID
确定

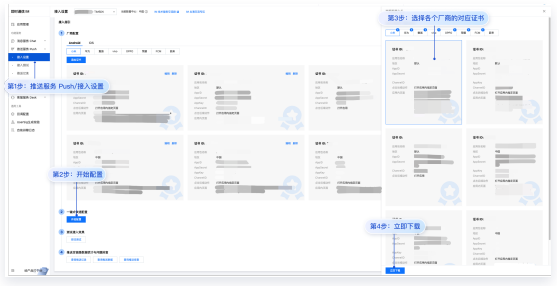
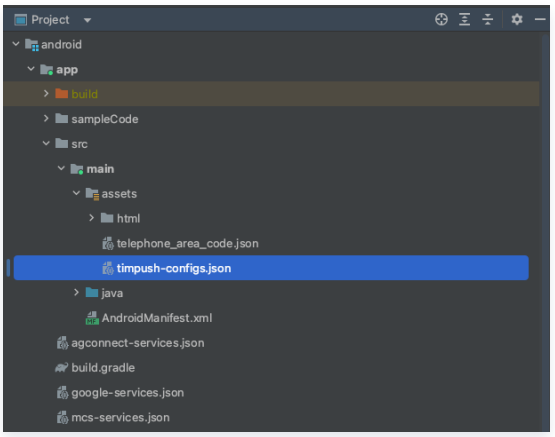
快速接入
Android

最近更新时间：2025-04-07 16:57:52

> **注意：**
> 如果您需要同时使用 Chat、CallKit 、RoomKit 、LiveKit 等产品，请参见 IM 快速接入方案 。

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

```
// 版本号“VERSION”请前往 更新日志 中获取配置。
// 推送主包必须要集成
implementation 'com.tencent.timpush:timpush:VERSION'
implementation 'com.tencent.liteav.tuikit:tuicore:VERSION'
// 按照需要集成对应厂商
implementation 'com.tencent.timpush:huawei:VERSION'
implementation 'com.tencent.timpush:xiaomi:VERSION'
implementation 'com.tencent.timpush:oppo:VERSION'
implementation 'com.tencent.timpush:vivo:VERSION'
implementation 'com.tencent.timpush:honor:VERSION'
implementation 'com.tencent.timpush:meizu:VERSION'
implementation 'com.tencent.timpush:fcm:VERSION'
```

- **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

```
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"
        ]
    }
}
```

方法2

```
// 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
```

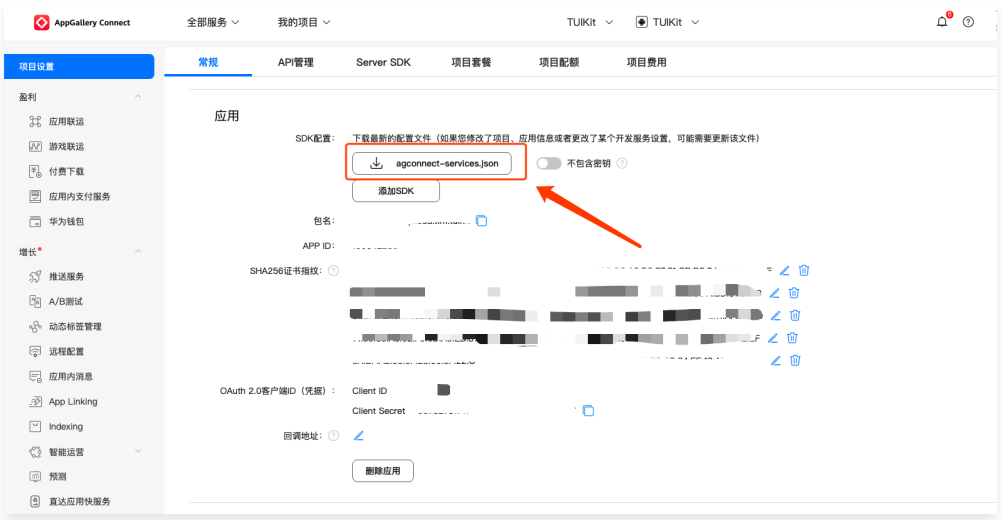
- **华为、荣耀和 Google FCM 适配**

  按照厂商方法，集成对应的 plugin 和 json 配置文件。

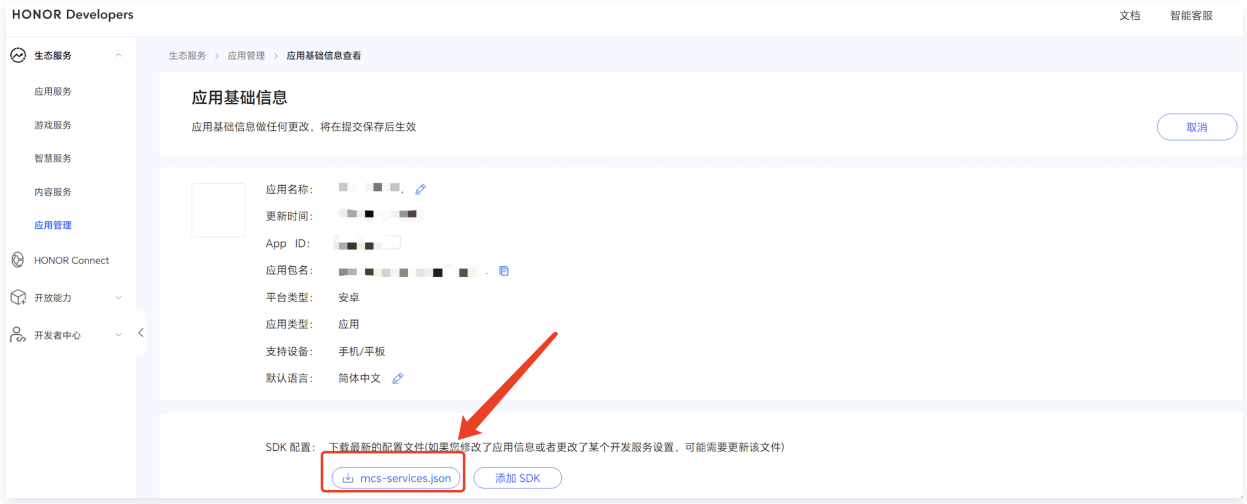
> ⚠ **注意：**
> 以下荣耀的适配仅 7.7.5283 及以上版本需要配置。

1.1 下载配置文件添加到工程根目录：

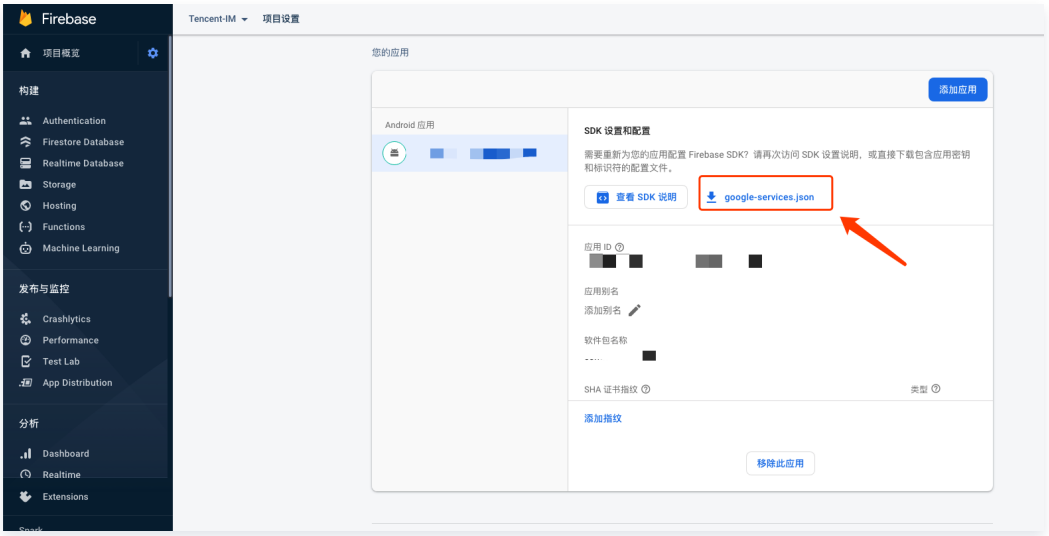
华为



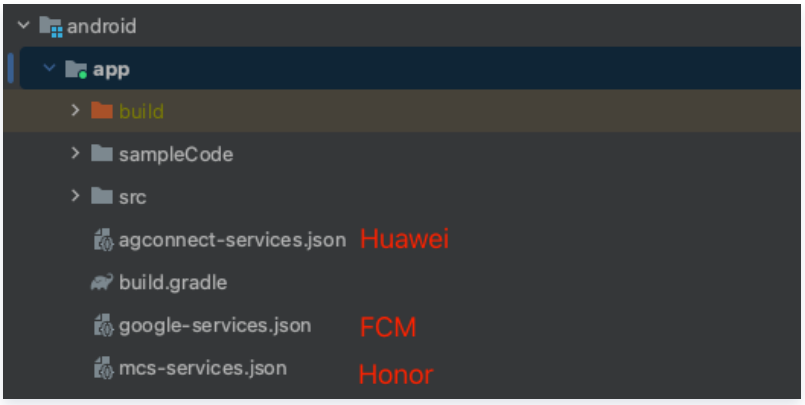
荣耀



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

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:1.6.0.300'
        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'}
    }
    }
}
```

Gradle 7.0 版本

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:1.6.0.300'
        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'}
    }
}
```

Gradle 7.0 以下版本

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:1.6.0.300'
        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：注册推送

调用接口推送注册成功后，就可以收到离线推送通知了。

```
TIMPushManager.getInstance().registerPush(context, 您的 sdkAppId, "客户端密钥", new TIMPushCallback()
{
```

```
    @Override
    public void onSuccess(Object data) {

    }

    @Override
    public void onError(int errCode, String errMsg, Object data) {

    }
});
```

步骤5：消息触达统计配置

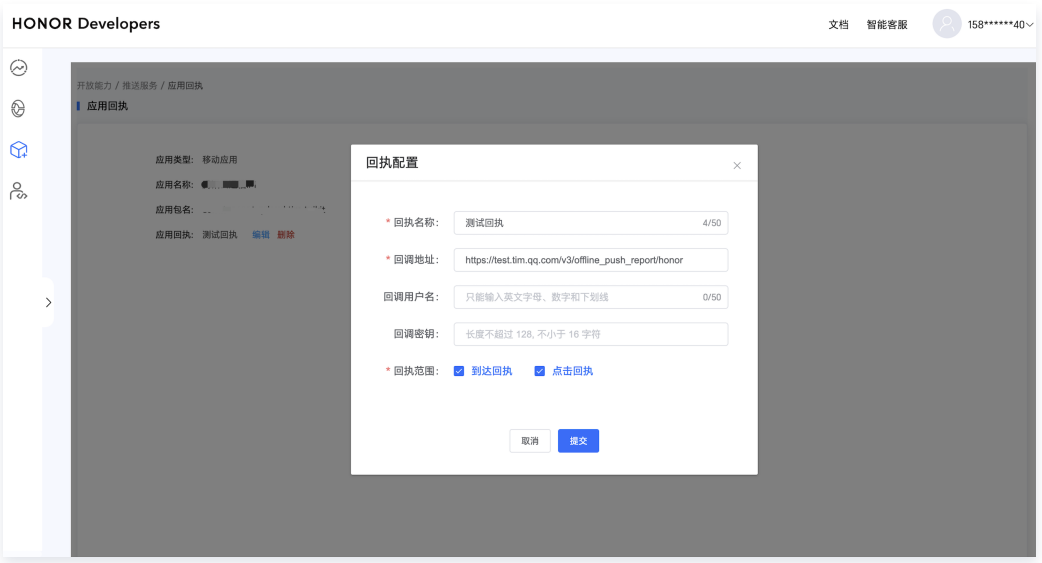
如果您需要统计触达数据，请按照如下完成配置：

华为

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 。

荣耀



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

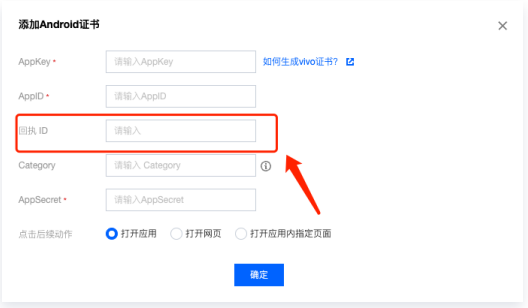
vivo

回调地址配置

回执地址：

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

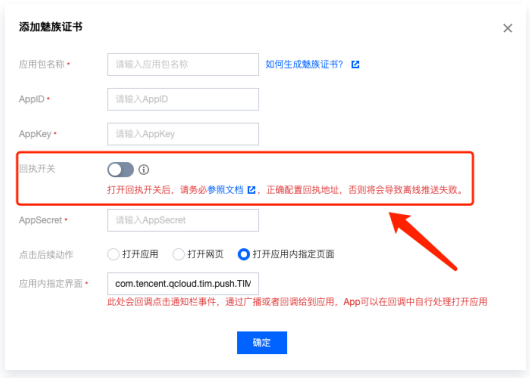
回执 ID 配置 IM 控制台



魅族

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。

> **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6：发送推送消息

接口详细说明可参见：REST API 接口-发起全员/标签推送 。

步骤7：解析离线推送消息

收到推送消息后点击通知栏，组件会回调该点击事件和透传离线消息。

自定义点击跳转实现

> **注意：**
> 1. 注册回调时机建议放在应用 Application 的 oncreate() 函数中。
> 2. 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**，请勿修改使用默认值。

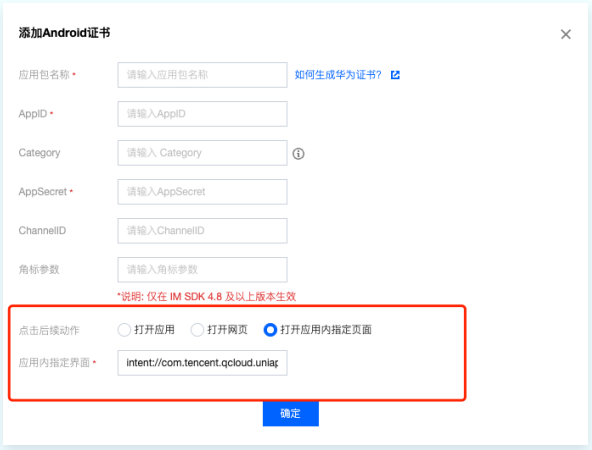

```
TIMPushManager.getInstance().addPushListener(new TIMPushListener() {
```

```
    @Override
    public void onNotificationClicked(String ext) {
        Log.d(TAG, "onNotificationClicked =" + ext);
        // 获取 ext 自定义跳转

    }
});
```

自定义点击跳转实现（旧方案）

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。

> **⚠ 注意：**
>
> 1. **注册回调时机建议放在应用 Application 的 oncreate() 函数中。**
> 2. **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, 请勿修改使用默认值。**



1. 回调方式如下：

```
TUICore.registerEvent(TUIConstants.TIMPush.EVENT_NOTIFY,
TUIConstants.TIMPush.EVENT_NOTIFY_NOTIFICATION, new ITUINotification() {
        @Override
        public void onNotifyEvent(String key, String subKey, Map<String, Object> param) {
            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.NOTIFICATION_EXT_KEY);
                        // 获取 ext 自定义跳转

                    }
                }
            }
        }
    });
```

2. 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, intentFilter);

//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_EXT_KEY);
            // 获取 ext 自定义跳转

        } 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 。

说明：

1. 厂商离线通道都有 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，离线推送会有数量和频次的限制，可能不会及时推送到设备。
2. 接入完成收不到推送，请先自助使用 排查工具 查看下具体原因。推送指标数据查看，请使用 数据统计 查询。
3. 全员/标签推送功能请参见：REST API 接口 – 发起全员/标签推送 。

iOS

最近更新时间：2025-02-13 17:54:12

> **注意：**
> 如果您需要同时使用 Chat、CallKit 、RoomKit 、LiveKit 等产品，请参考 IM 快速接入方案 。

步骤1：集成 TIMPush

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 frameworks
  use_frameworks!
  use_modular_headers!

  # Pods for Example
  pod 'TXIMSDK_Plus_iOS_XCFramework'
  # 版本号 “VERSION” 请前往 更新日志 中获取配置。
  pod 'TIMPush', 'VERSION'
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 `- businessID` 协议方法返回证书 ID 即可。

Objective-C

```
#pragma mark - TIMPush
- (int)businessID {
    //上一步控制台给的证书ID，如 1234567
    int  kBusinessID = 1234567;
    return kBusinessID;
}

- (NSString *)applicationGroupID {
    //AppGroup ID
    return kTIMPushAppGroupKey;
}

- (BOOL)onRemoteNotificationReceived:(NSString *)notice {
    // custom navigate
    return NO;
}
```

Swift

```
#pragma mark - TIMPush
//Swift 务必携带 @objc 关键字
@objc func businessID() -> Int32 {
    //上一步控制台给的证书ID
    return 0
}
@objc func applicationGroupID() -> String {
    //AppGroup ID
    return "group.com.yourcompony.pushkey"
}
@objc func onRemoteNotificationReceived(_ notice: String?) -> Bool {
    // custom navigate
    return false
}
```

步骤3：注册推送

调用接口推送注册成功后，就可以收到离线推送通知了。

Objective-C

```
const int sdkAppId = 您的 sdkAppId;
static const NSString *appKey = @"客户端密钥";

[TIMPushManager registerPush:sdkAppId appKey:appKey succ:^(NSData * _Nonnull deviceToken) {

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

}];
```

Swift

```
let sdkAppId: Int = 0
let appKey: String = "客户端密钥"

TIMPushManager.registerPush(Int32(sdkAppId), appKey: appKey, succ: { deviceToken in
    // 成功回调处理
}, fail: { code, desc in
    // 失败回调处理
})
```

> ⚠ **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。

步骤4：发送推送消息

详细使用指引可参见： REST API 接口-发起全员/标签推送 。

步骤5: 单击离线推送后自定义跳转

如果您需要自定义解析收到的远程推送，您可按照如下方法实现：

自定义点击跳转实现

> ⚠ **注意：**
> 注册回调时机建议放在应用 AppDelegate 的 didFinishLaunchingWithOptions 函数中。

Objective-C

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary
*)launchOptions {
    [TIMPushManager addPushListener:self];
    return YES;
}
#pragma mark - TIMPushListener
- (void)onNotificationClicked:(NSString *)ext {
        // 获取 ext 自定义跳转
}
```

Swift

```
func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[UIApplication.LaunchOptionsKey: Any]?) -> Bool {
    // Override point for customization after application launch.
    TIMPushManager.addPushListener(listener: self)
    return true
}

@objc func onNotificationClicked(_ ext: String) {
    //Clicked
}
@objc func onRecvPushMessage(_ message: TIMPushMessage) {
```

```
    //onRecvPushMessage
}
@objc func onRevokePushMessage(_ messageID: String) {
    //onRevokePushMessage
}
```

自定义点击跳转实现（旧方案）

您需要在 AppDelegate.m 文件中实现 `- onRemoteNotificationReceived` 方法。

Objective-C

```
#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 -
navigateToBuiltInChatViewController:groupID: 方法。
    return NO;
}
```

Swift

```
@objc func onRemoteNotificationReceived(_ notice: String) -> Bool {
    // - 如果返回 true， 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；
    // let ext = notice
    // let info = OfflinePushExtInfo.create(withExtString: ext)
    // return true

    // - 如果返回 false，TIMPush 将继续执行内置的 TUIKit 离线推送解析逻辑，继续回调 -
navigateToBuiltInChatViewController:groupID: 方法。
    return false
}
```

步骤6: 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：

Objective-C

```
@implementation NotificationService
- (void)didReceiveNotificationRequest:(UNNotificationRequest *)request withContentHandler:(void (^)
(UNNotificationContent * _Nonnull))contentHandler {
```

```
    //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。
    //格式为 group + [主bundleID]+ key
    //如 group.com.tencent.im.pushkey
    NSString * appGroupID = kTIMPushAppGroupKey;
    __weak typeof(self) weakSelf = self;
    [TIMPushManager handleNotificationServiceRequest:request appGroupID:appGroupID
callback:^(UNNotificationContent *content) {
        weakSelf.bestAttemptContent = [content mutableCopy];
        // Modify the notification content here...
        // self.bestAttemptContent.title = [NSString stringWithFormat:@"%@ [modified]",
self.bestAttemptContent.title];
        weakSelf.contentHandler(weakSelf.bestAttemptContent);
    }];
}
@end
```

Swift

```
override func didReceive(_ request: UNNotificationRequest, withContentHandler contentHandler:
@escaping (UNNotificationContent) -> Void) {
    self.contentHandler = contentHandler
    bestAttemptContent = (request.content.mutableCopy() as? UNMutableNotificationContent)
    //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。
    //推荐格式为 group + [主bundleID]
    //如 group.com.主bundleID.pushkey
    TIMPushManager.handleNotificationServiceRequest(request: request, appGroupID: "appGroupID") {
        [weak self] content  in
        if let bestAttemptContent = self?.bestAttemptContent {
            // Modify the notification content here...
            bestAttemptContent.title = "\(bestAttemptContent.title) [modified]"
            contentHandler(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

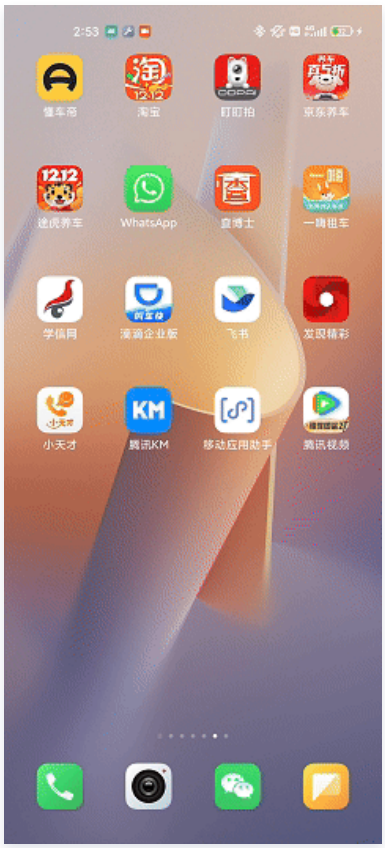
全员/标签推送支持发送特定内容，还可根据标签、属性，针对特定用户群体发送个性化内容，例如会员活动、区域通知等，助力拉新、转化、促活等各个阶段运营工作的有效进行，同时支持推送送达报表，自助推送故障排查工具，具体效果请参见 效果展示 。
更多详细内容建议查阅 全员/标签推送
恭喜您已经完成了推送插件的接入，需要提醒您：消息推送插件**试用或购买到期后，将自动停止提供推送服务（包括普通消息离线推送、全员推送等服务）**。为避免影响您业务正常使用，请提前 购买/续费 。

uni-app

最近更新时间：2025-05-27 16:59:01

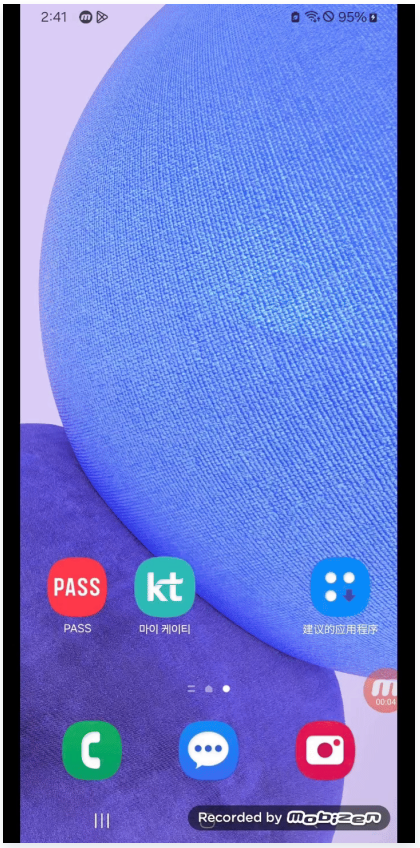
效果展示

小米 11 Pro（与 CallKit 融合）



iPhone 13

三星 Galaxy A23 海外版（Google FCM 推送）



集成 TencentCloud-Push

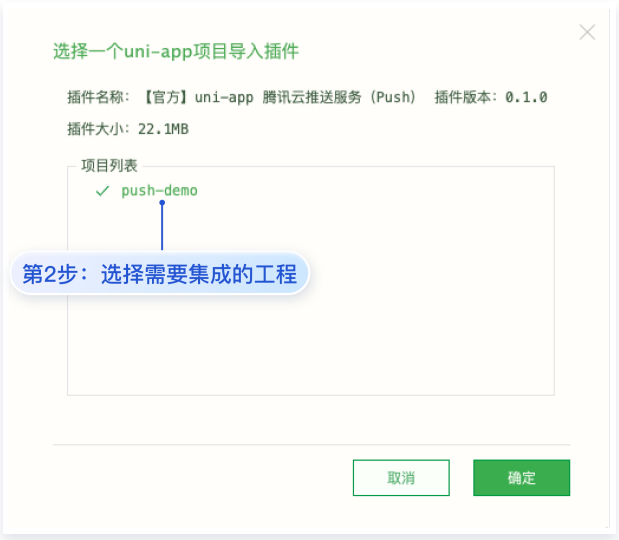
说明：

- HBuilderX 4.64 版本、HBuilderX 4.65 版本 有 bug，推荐：
  - HBuilderX 4.36 版本 ≤ 推荐版本 < HBuilderX 4.64 版本，并升级 uni-app 腾讯云推送服务（Push）到 1.1.0 或更高版本。
  - ≥ HBuilderX 4.66 版本，并升级 uni-app 腾讯云推送服务（Push）到 1.1.0 或更高版本。
- 如果您的项目需要支持 uni-app x，请使用 ≥ HBuilderX 4.55 版本，并升级 uni-app 腾讯云推送服务（Push）到 1.1.0 或更高版本。

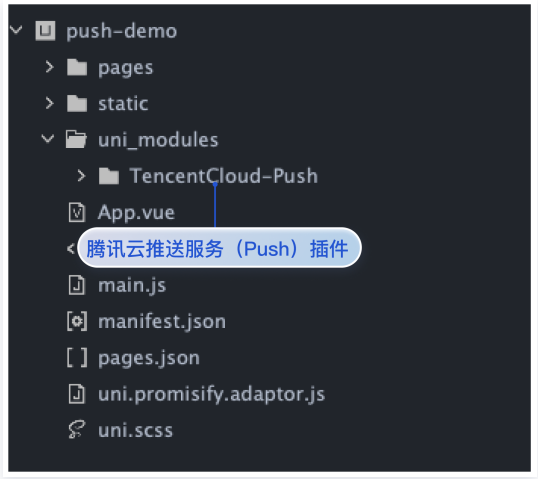
步骤1. 下载插件并导入 HBuilderX

1. 打开 uni-app 腾讯云推送服务（Push），单击**下载插件并导入HBuilderX**，将插件导入 HBuilderX 工程中。

2. 选择需要集成的工程并单击**确定**。



3. 集成后效果如下图所示：



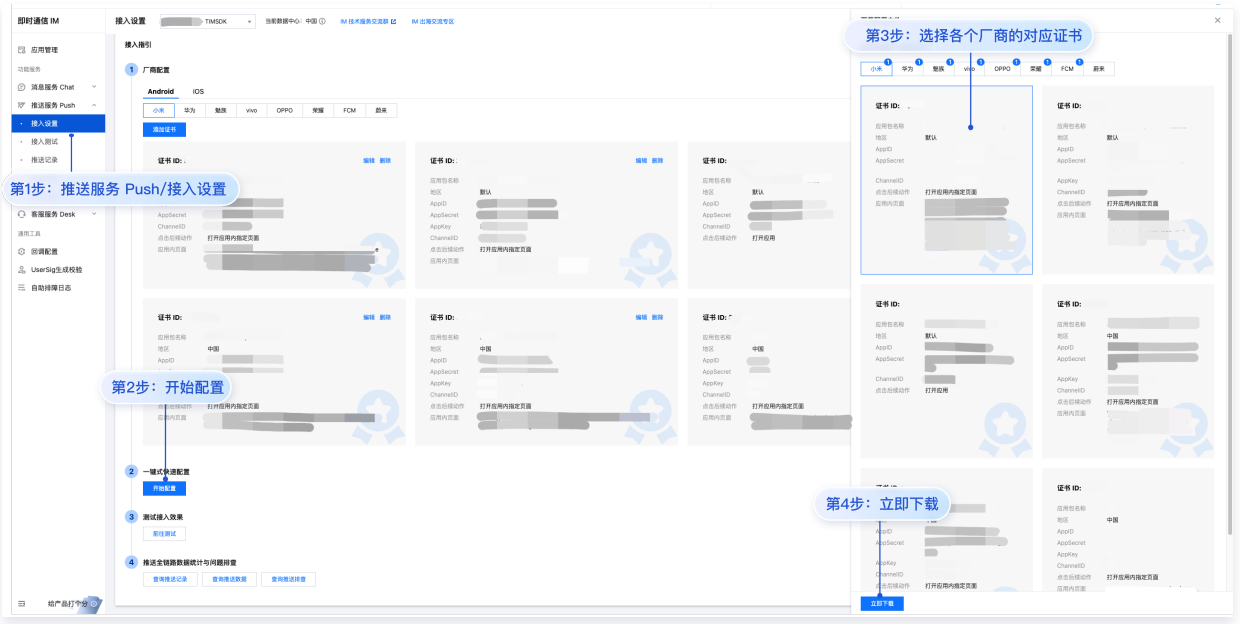
步骤2. 离线推送配置

> **说明：**

1. HBuilderX 4.36 发布了不向下兼容的更新，如果您使用的是 HBuilderX 4.36 或者更高版本，且需要 vivo/荣耀 的厂商推送，请升级推送版本到 1.1.0 或更高版本，并参考文档正确配置 `manifestPlaceholders.json` 和 `mcs-services.json` 。
2. 您需在 `nativeResources` 目录下进行推送配置。若项目根目录尚未创建该文件夹，请新建一个名为 `nativeResources` 的文件夹。
3. 确保您用 HBuilderX 打开的项目中 `nativeResources` 目录存在，且与 `uni_modules` 目录平级。

Android

1. 新建 `nativeResources/android/assets` 目录。
2. 配置 `timpush-configs.json` （在 推送服务 Push > 接入设置 > 一键式快速配置 下载），到 `nativeResources/android/assets/` 目录下。如图所示：

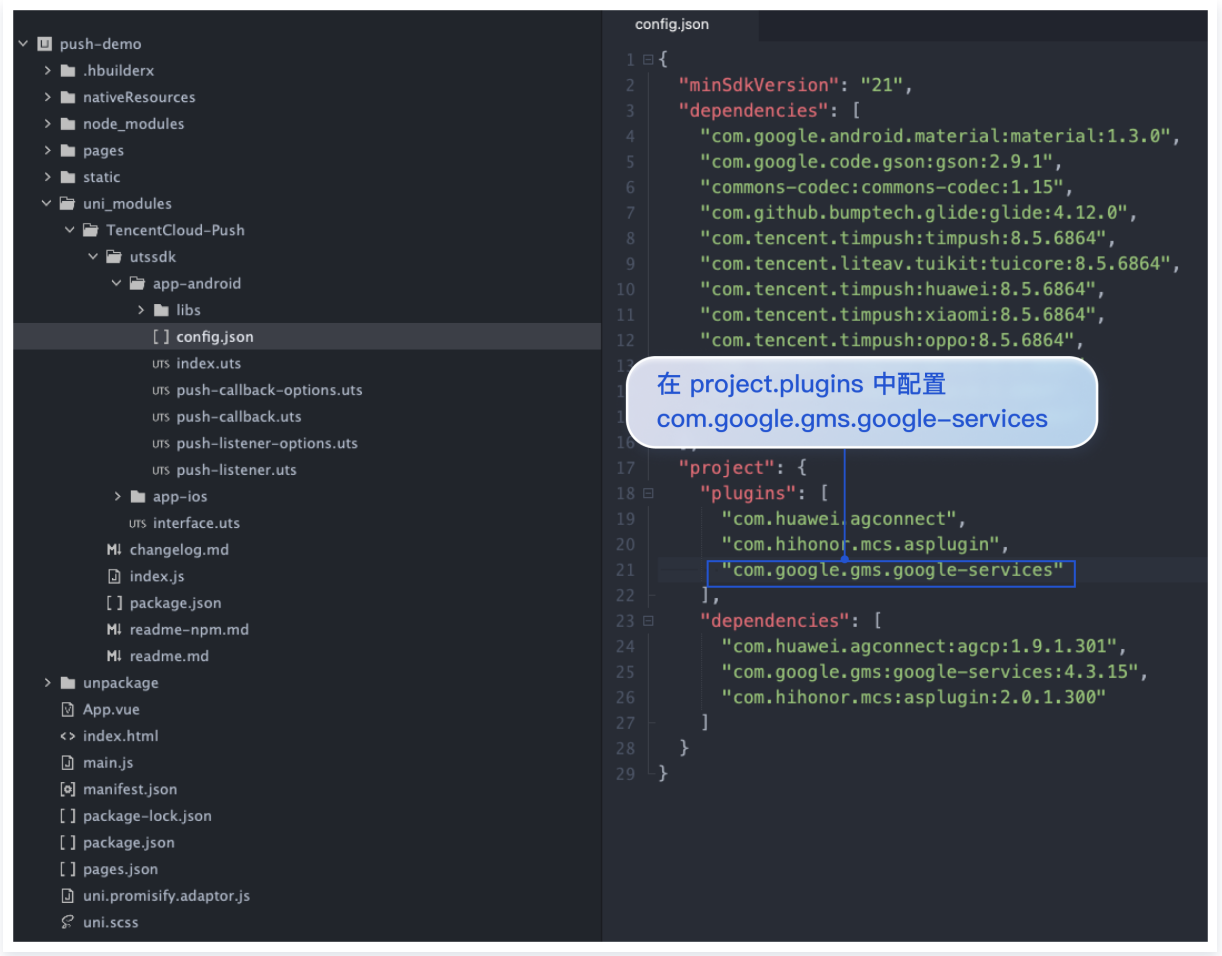


3. 华为、荣耀、vivo、FCM。

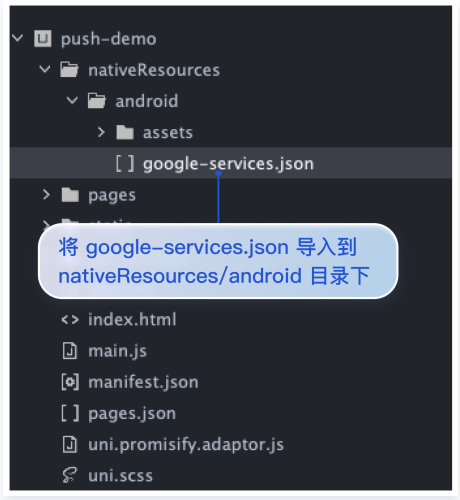
FCM

1. 配置 `com.google.gms.google-services` 到 `uni_modules/TencentCloud-Push/utssdk/app-android/config.json` 的 `project.plugins` 中，如下所示：

```
"project": {
    "plugins": [
        ...
        "com.google.gms.google-services"
    ],
    "dependencies": [
        "com.huawei.agconnect:agcp:1.9.1.301",
        "com.google.gms:google-services:4.3.15",
        "com.hihonor.mcs:asplugin:2.0.1.300"
    ]
}
```

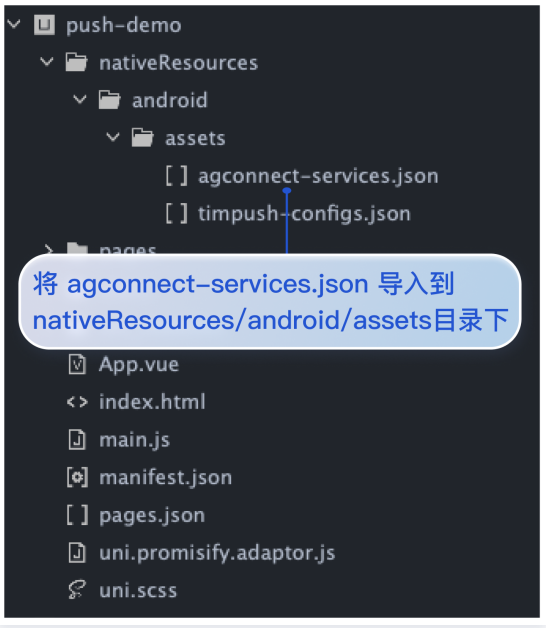


2. 配置 `google-services.json` 文件到 `nativeResources/android/` 目录下（注意！请勿配置到 `nativeResources/android/asstes` 目录下）。如图所示：



华为

配置 `agconnect-services.json` （此文件获取详见 厂商配置 > uniapp > 华为 > 步骤4：获取应用信息 ）到 `nativeResources/android/assets/` 目录下。如图所示：



荣耀

1. 编辑 `uni_modules/TencentCloud-Push/utssdk/app-android/config.json` 的 `dependencies` ，添加 `"com.tencent.timpush:honor:8.3.6498"` 。如图所示：

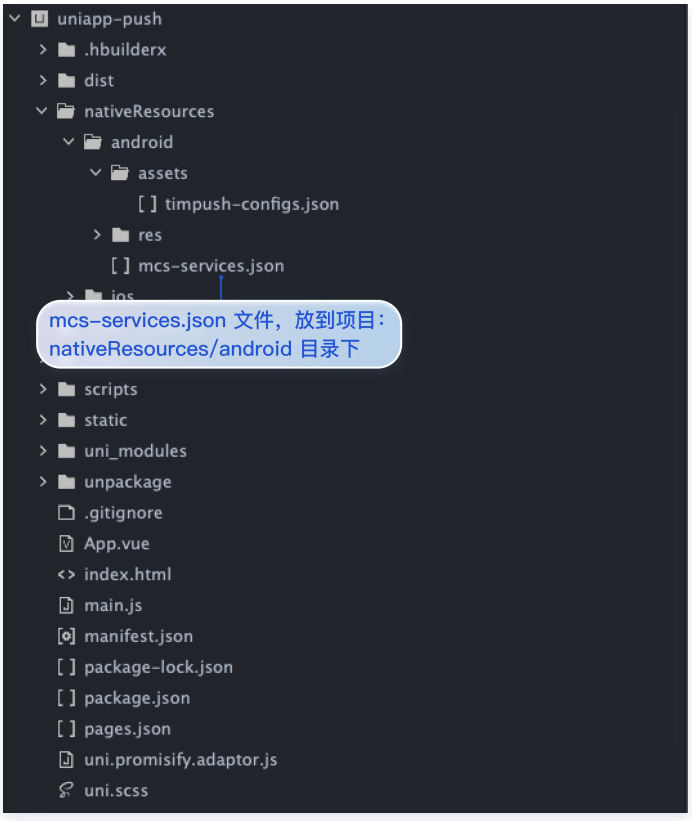
```
{
    ...
    "dependencies": [
        ...
        "com.tencent.timpush:honor:8.5.6864"
    ]
}
```

config.json

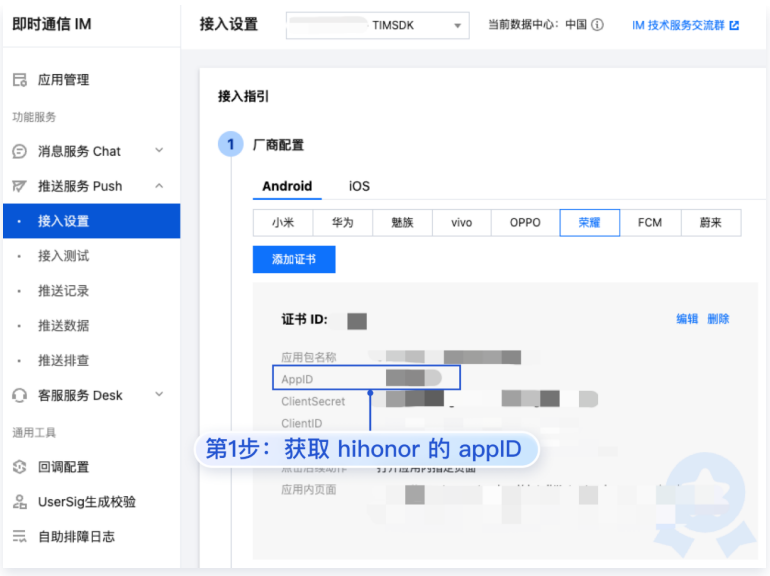
```
{
  "minSdkVersion": "21",
  "dependencies": [
    "com.google.android.material:material:1.3.0",
    "com.google.code.gson:gson:2.9.1",
    "commons-codec:commons-codec:1.15",
    "com.github.bumptech.glide:glide:4.12.0",
    "com.tencent.timpush:timpush:8.5.6864",
    "com.tencent.liteav.tuikit:tuicore:8.5.6864",
    "com.tencent.timpush:huawei:8.5.6864",
    "com.tencent.timpush:xiaomi:8.5.6864",
    "com.tencent.timpush:oppo:8.5.6864",
    "com.tencent.timpush:meizu:8.5.6864",
    "com.tencent.timpush:fcm:8.5.6864",
    "com.tencent.timpush:honor:8.5.6864"
  ],
  "project": {
    "plugins": [
      "com.huawei.agconnect",
      "com.hihonor.mcs.asplugin"
    ],
    "dependencies": [
      "com.huawei.agconnect:agcp:1.9.1.301",
      "com.google.gms:google-services:4.3.15",
      "com.hihonor.mcs:asplugin:2.0.1.300"
    ]
  }
}
```

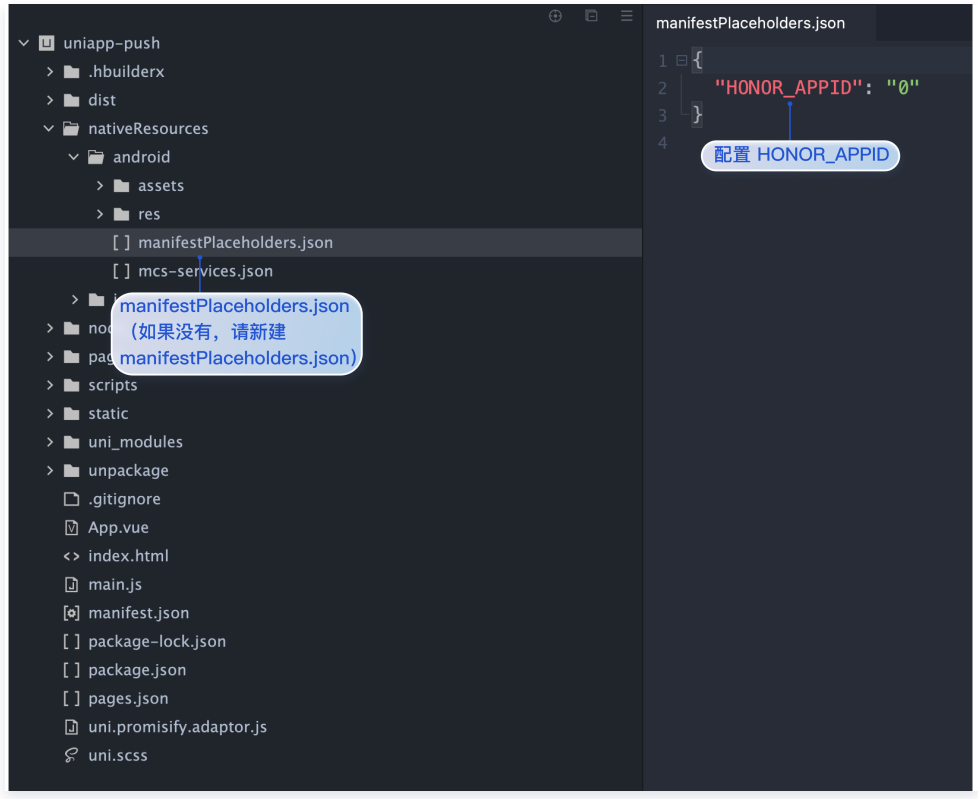
honor 的 push 依赖包

2. 配置 `mcs-services.json` 文件到 `nativeResources/android` （此文件获取详见 厂商配置 > uniapp > 荣耀 > 步骤3.2：进入应用详情， 绑定应用包名，下载 mcs-services.json 文件 ）目录下。如图所示：



3. 配置 `appID` 到 `nativeResources/android/manifestPlaceholders.json` 中的 `"HONOR_APPID"` 。






```
{
    "HONOR_APPID": ""
}
```

vivo

1. 编辑 `uni_modules/TencentCloud-Push/utssdk/app-android/config.json` 的 `dependencies`，添加 `"com.tencent.timpush:vivo:8.3.6498"`。如图所示：

```
{
    ...
```

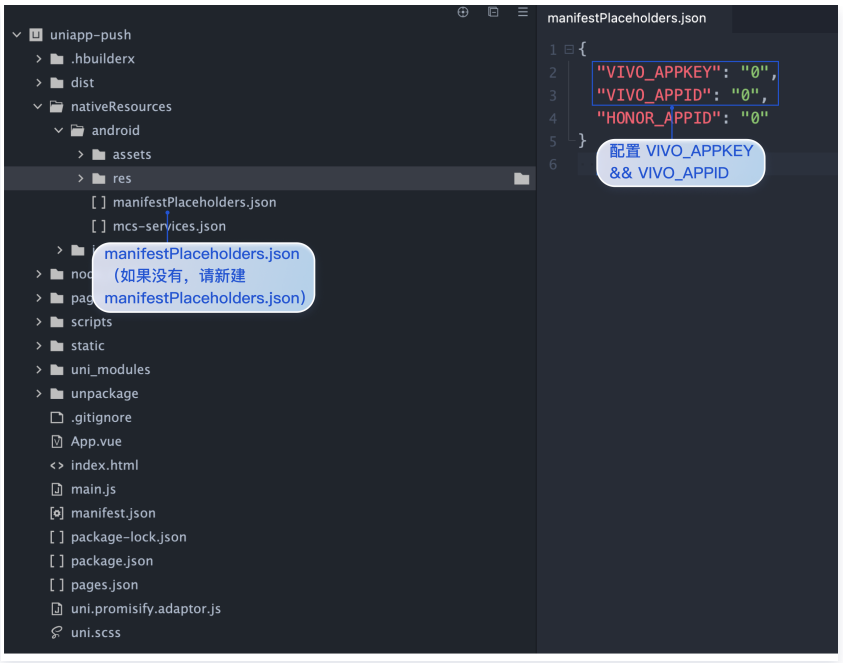
```
    "dependencies": [
        ...
        "com.tencent.timpush:vivo:8.5.6864"
    ]
}
```

config.json

```
{
  "minSdkVersion": "21",
  "dependencies": [
    "com.google.android.material:material:1.3.0",
    "com.google.code.gson:gson:2.9.1",
    "commons-codec:commons-codec:1.15",
    "com.github.bumptech.glide:glide:4.12.0",
    "com.tencent.timpush:timpush:8.5.6864",
    "com.tencent.liteav.tuikit:tuicore:8.5.6864",
    "com.tencent.timpush:huawei:8.5.6864",
    "com.tencent.timpush:xiaomi:8.5.6864",
    "com.tencent.timpush:oppo:8.5.6864",
    "com.tencent.timpush:meizu:8.5.6864",
    "com.tencent.timpush:fcm:8.5.6864",
    "com.tencent.timpush:vivo:8.5.6864"
  ],
  "project": {
    "plugins": [
      "com.huawei.agconnect",
      "com.hihonor.mcs.asplugin"
    ],
    "dependencies": [
      "com.huawei.agconnect:agcp:1.9.1.301",
      "com.google.gms:google-services:4.3.15",
      "com.hihonor.mcs:asplugin:2.0.1.300"
    ]
  }
}
```

vivo 的 push 依赖包

2. 配置 `appID` 和 `appKey` 到 `nativeResources/android/manifestPlaceholders.json` 中的 `VIVO_APPKEY` 和 `VIVO_APPID`。




```
{
    "VIVO_APPKEY": "",
    "VIVO_APPID": "",
}
```

iOS

1. 新建 `nativeResources/ios/Resources` 目录；
2. 在 `nativeResources/ios/Resources` 中新建 `timpush-configs.json` 文件；
3. 并将在 IM控制台 > 推送服务 Push > 接入设置 获取的证书ID，补充到 `timpush-configs.json` 文件中。如下所示：

```
{
    "businessID":"xxx"
}
```

步骤3. 引入并注册腾讯云推送服务（Push）

将 `SDKAppID` 和 `appKey` 替换为您在 IM 控制台 – 推送服务 Push – 接入设置页面 获取的应用的信息。如图所示：


```
// 集成 TencentCloud-Push
import * as Push from '@/uni_modules/TencentCloud-Push';
const SDKAppID = 0; // 您的 SDKAppID
const appKey = ''; // 客户端密钥

// 如果您需要与 Chat 的登录 userID 打通（即向此 userID 推送消息），请使用 setRegistrationID 接口
// Push.setRegistrationID(userID, () => {
    // console.log('setRegistrationID ok', userID);
// });

Push.registerPush(SDKAppID, appKey, (data) => {
        console.log('registerPush ok', data);
        Push.getRegistrationID((registrationID) => {
            console.log('getRegistrationID ok', registrationID);
        });
    }, (errCode, errMsg) => {
```

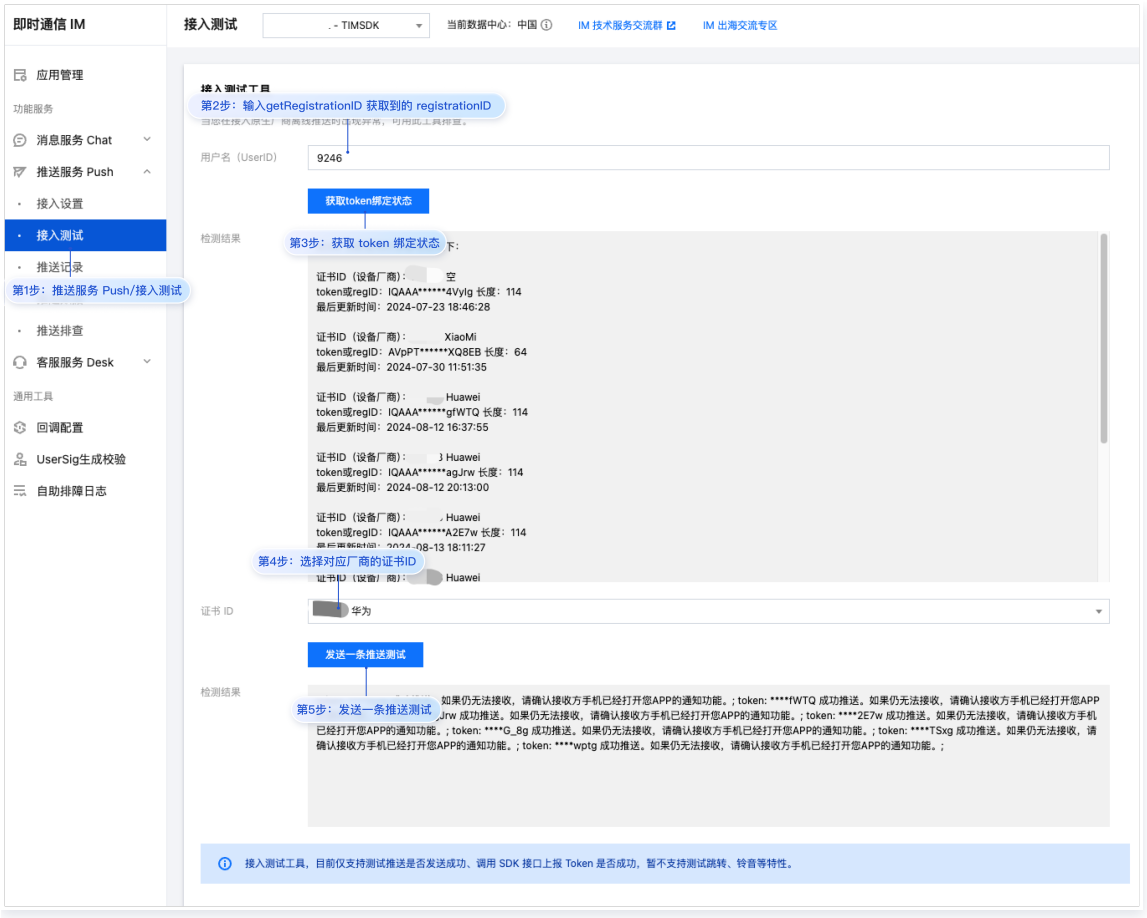
```
        console.error('registerPush failed', errCode, errMsg);
    }
);

// 监听通知栏点击事件，获取推送扩展信息
Push.addPushListener(Push.EVENT.NOTIFICATION_CLICKED, (res) => {
    // res 为推送扩展信息
    console.log('notification clicked', res);
});

// 监听在线推送
Push.addPushListener(Push.EVENT.MESSAGE_RECEIVED, (res) => {
    // res 为消息内容
    console.log('message received', res);
});

// 监听在线推送被撤回
Push.addPushListener(Push.EVENT.MESSAGE_REVOKED, (res) => {
    // res 为被撤回的消息 ID
    console.log('message revoked', res);
});
```

步骤4. 使用云端证书，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**，使用云端证书制作 Android 或 iOS 自定义调试基座。

步骤5. 体验您的第一次推送

在测试推送前，请务必打开通知和状态栏。进入 推送服务 Push > 接入测试 发送您的第一条推送。


即时通信 IM
接入测试
- TIMSDK
当前数据中心：中国
IM 技术服务交流群
IM 出海交流专区
应用管理
功能服务
消息服务 Chat
推送服务 Push
接入设置
接入测试
推送记录
推送排查
客服服务 Desk
通用工具
回调配置
UserSig生成校验
自助排障日志
第1步：推送服务 Push/接入测试
第2步：输入getRegistrationID 获取到的 registrationID
用户名（UserID）
9246
获取token绑定状态
检测结果
第3步：获取 token 绑定状态
证书ID（设备厂商）：空
token或regID：IQAAA******4Vylg 长度：114
最后更新时间：2024-07-23 18:46:28
证书ID（设备厂商）：XiaoMi
token或regID：AVpPT******XQ8EB 长度：64
最后更新时间：2024-07-30 11:51:35
证书ID（设备厂商）：Huawei
token或regID：IQAAA******gfWTQ 长度：114
最后更新时间：2024-08-12 16:37:55
证书ID（设备厂商）：Huawei
token或regID：IQAAA******agJrw 长度：114
最后更新时间：2024-08-12 20:13:00
证书ID（设备厂商）：Huawei
token或regID：IQAAA******A2E7w 长度：114
第4步：选择对应厂商的证书ID
证书 ID
华为
发送一条推送测试
检测结果
第5步：发送一条推送测试
如果仍无法接收，请确认接收方手机已经打开您APP的通知功能。; token: ****fWTQ 成功推送。如果仍无法接收，请确认接收方手机已经打开您APP
Jrw 成功推送。如果仍无法接收，请确认接收方手机已经打开您APP的通知功能。; token: ****2E7w 成功推送。如果仍无法接收，请确认接收方手机
已经打开您APP的通知功能。; token: ****G_8g 成功推送。如果仍无法接收，请确认接收方手机已经打开您APP的通知功能。; token: ****TSxg 成功推送。如果仍无法接收，请
确认接收方手机已经打开您APP的通知功能。; token: ****wptg 成功推送。如果仍无法接收，请确认接收方手机已经打开您APP的通知功能。;
接入测试工具，目前仅支持测试推送是否发送成功、调用 SDK 接口上报 Token 是否成功，暂不支持测试跳转、铃音等特性。


10:52
7月25日 周四 上午
云通信 IM · 现在
Tencent
test
您接收到的第一条推送

推送结果回调

开启推送服务后，推送结果可以通过配置基础回调的方式，将结果转发给 App 后台，详见：

- 普通推送结果回调
- 全员推送结果回调

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，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

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 。

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。

微信小程序多端框架

最近更新时间：2025-04-07 18:08:12

前置条件

1. 厂商配置

> **说明：**
> 在 微信开发者工具 中开启推送服务，默认支持在线推送。如果您需要使用离线推送，请您先完成厂商配置，具体操作方法可参见 厂商配置 - 微信小程序多端框架 。

2.微信开发者工具 版本 ⩾ 1.06.2410152 >>下载地址

集成腾讯云消息推送服务

步骤1: 推送插件配置

> **说明：**
> 在 微信开发者工具 中开启推送服务，默认支持在线推送。

Android

插件配置

1. 在可视化界面配置填写插件版本号，勾选开启腾讯云消息推送功能

查看 更新日志 获取插件最新版本号。

2. 设置 Android SDK 版本

推送服务所需的微信小程序多端框架 Android SDK 最低版本为 1.4.7，推荐使用 最新版本 。

iOS

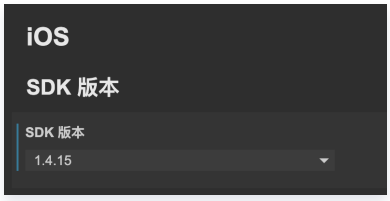
插件配置

1. 在可视化界面配置填写插件版本号，勾选开启腾讯云消息推送功能

查看 更新日志 获取插件最新版本号。

2. 设置 iOS SDK 版本

推送服务所需的微信小程序多端框架 iOS SDK 版本号为 `1.4.15` ，推荐使用 最新的 SDK 。



步骤2: 离线推送配置

> **说明：**
> 如果只使用在线推送，则无需进行厂商离线推送配置。

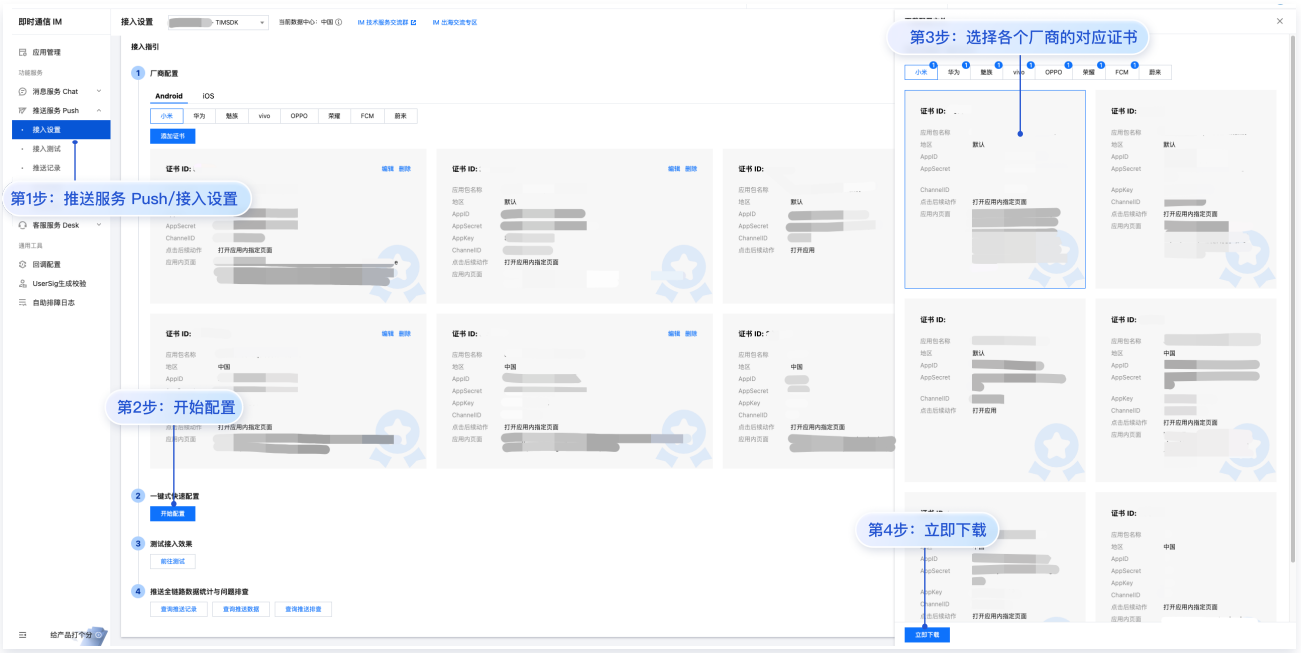
Android

离线推送配置

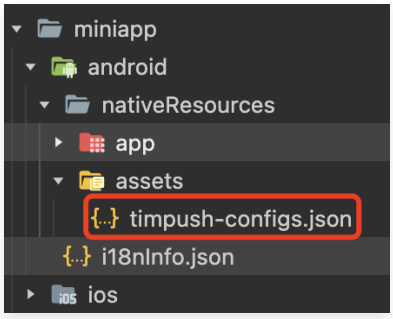
可参见微信小程序多端框架官方文档 Android 配置原生资源 。

1. 配置 timpush-configs.json

推送服务运行时需要从配置文件读取 appid、appkey 、证书 ID 等信息，需要把相应的 timpush-configs.json 文件放到 assets 文件夹打包到 apk 中；
timpush-configs.json 文件可以从 即时通信 IM 控制台 - 推送服务 Push 直接下载：



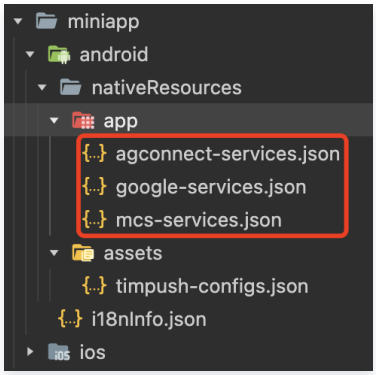
将 timpush-configs.json 放到原生资源目录的 assets 文件夹下：



2. 配置厂商资源文件

1. 华为、荣耀、谷歌 FCM 的厂商推送，需要读取各自的配置文件，要把相应的配置文件放在 app 目录下 。
各厂商的配置文件需要分别去厂商的推送控制台下载。

> **注意：**
> 不要修改厂商配置文件名称，否则厂商推送会集成失败。



3. 配置证书指纹（华为、荣耀）

华为、荣耀需要配置证书指纹才能正常收到离线推送。请参见 华为证书指纹配置 、荣耀证书指纹配置 。

4. 配置离线推送消息分类

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。需正确 配置推送消息分类 才能正常收到离线推送。

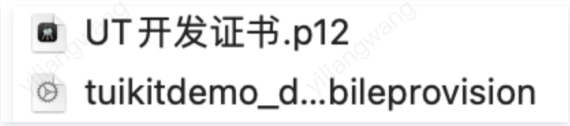
iOS

离线推送配置

1. 配置证书和描述文件

请参见 微信小程序多端框架配置文档 或 腾讯云推送厂商配置文档 以生成 iOS 应用运行时所需的证书和 Provisioning Profile，这些文件将在后续配置中使用。您需要准备以下文件：

- 开发证书.p12
- 主 App 描述文件.mobileprovision



2. Apple Developer 平台 Bundle ID 配置

请参见 Apple Developer 平台 Bundle ID 配置指引，配置时请勾选以下能力：

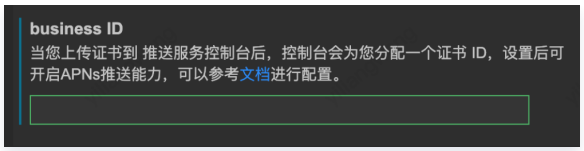
- Push Notification
- App Group (按需配置：字段配置可参见 生成 App GroupID)
- Donut 建议的权限：Access WiFi Information、Associated Domains、Hotspot、Wireless Accessory Configuration；如果使用苹果登录，还需开启 Sign In with Apple（更多其他权限开发者按需开启，并非越多越好哈）

3. 在 Donut 控制台里绑定移动应用的 Bundle ID

在 微信小程序多端应用控制台 里绑定移动应用的 `Bundle ID` ，微信开发者工具在真机运行 iOS 时需要明确 `Bundle ID` 。

4.苹果推送服务 (APNs) 证书 ID 配置：

当您需要接入 APNs 时开启厂商通道推送时， `businessID` **为必传项，**请先按照 证书配置文档 上传证书到推送控制台，之后控制台会为您分配此ID。



5. 运行前准备

当您点击 Donut 运行程序运行于真机时，会出现**提示框引导**您使用开发证书（.p12 文件）和主应用程序的描述文件。

> **说明：**
> 接入至此，您已经具备了在线推送和厂商推送的所有推送能力，已可满足绝大部分业务需求。
> 后续 NSE 的配置，推荐您完整测试完推送功能后按照业务需求接入。

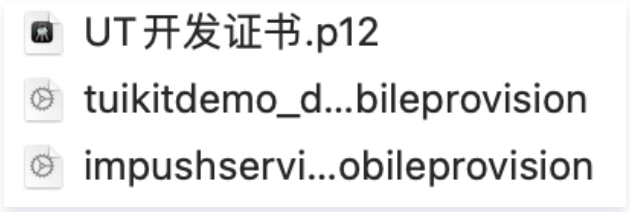
6. NSE 功能配置（可选）

当您希望利用 iOS 10 Service Extension 特性来设置 APNs 离线推送的通知图片或统计消息的到达率时，您需要在相应的 **project.miniapp.json** 文件中主动配置 NSE。同时，请确保在证书控制台中**启用 mutable-content** 选项，以便充分利用 iOS 10 Service Extension 的功能。
需要注意的是 NSE 是主 App 的插件，推送能力是主 App 带来的，NSE是推送服务扩展，因此常用 NSE 来做一些特性需求的。

> **注意：**
> 请再次参见 微信小程序多端框架配置文档 或 腾讯云推送厂商配置文档 以生成 NSE 的Provisioning Profile，您需要重复之前生成描述文件的步骤。区别是标识符不同，NSE 的标识符规则如下：
> Notification Service Extension (NSE) 的 Bundle ID 通常是主应用程序的 Bundle ID 后面加上扩展的标识符，生成描述文件的步骤和主 App 一致。例如：您的主 App 描述文件（ BundleID 为：`com.yourcompany.yourapp` ）那么您的 NSE 描述文件（ BundleID 为：`com.yourcompany.yourapp.extension` ）

您需要准备以下文件：

- **开发证书.p12 （之前步骤已经准备完毕）**
- **主 App 描述文件.mobileprovision（之前步骤已经准备完毕）**
- **Service Extension 的描述文件.mobileprovision（本次功能需要新增的描述文件）**



当您准备好以上文件后，可以按照描述填入相关信息：

> **说明：**
> **NSE相关配置详细描述如下：**

开启 NSE :借助 iOS 10 Service Extension 特性统计消息抵达率 及设置 APNs 离线推送的通知图片。
NSE BundleID : 通常是主应用程序的 Bundle ID 后面加上扩展的标识符，生成描述文件的步骤和主 App 一致。
NSE profilePath : 开发环境下的 NSE 的Profile的文件路径，推荐放置于 project.miniapp.json 当前目录。
NSE distributeProfilePath: 发布环境下 NSE 的Profile的文件路径，推荐放置于 project.miniapp.json 当前目录。
TIMPushAppGroupID , 当您需要使用 TIMPush 组件统计推送抵达率时，推荐您按照配置文档配置。
请您按照情况填写。

推送服务使用

集成 JS API

说明：
如何使用 npm 请查阅 官方文档，此处不再赘述。

1. 在 miniprogram 目录下执行如下命令安装 @tencentcloud/donut-push：

```
npm i @tencentcloud/donut-push
```

2. 在微信开发者工具中构建 npm（微信开发者工具 -> 工具 -> 构建 npm）

使用方法

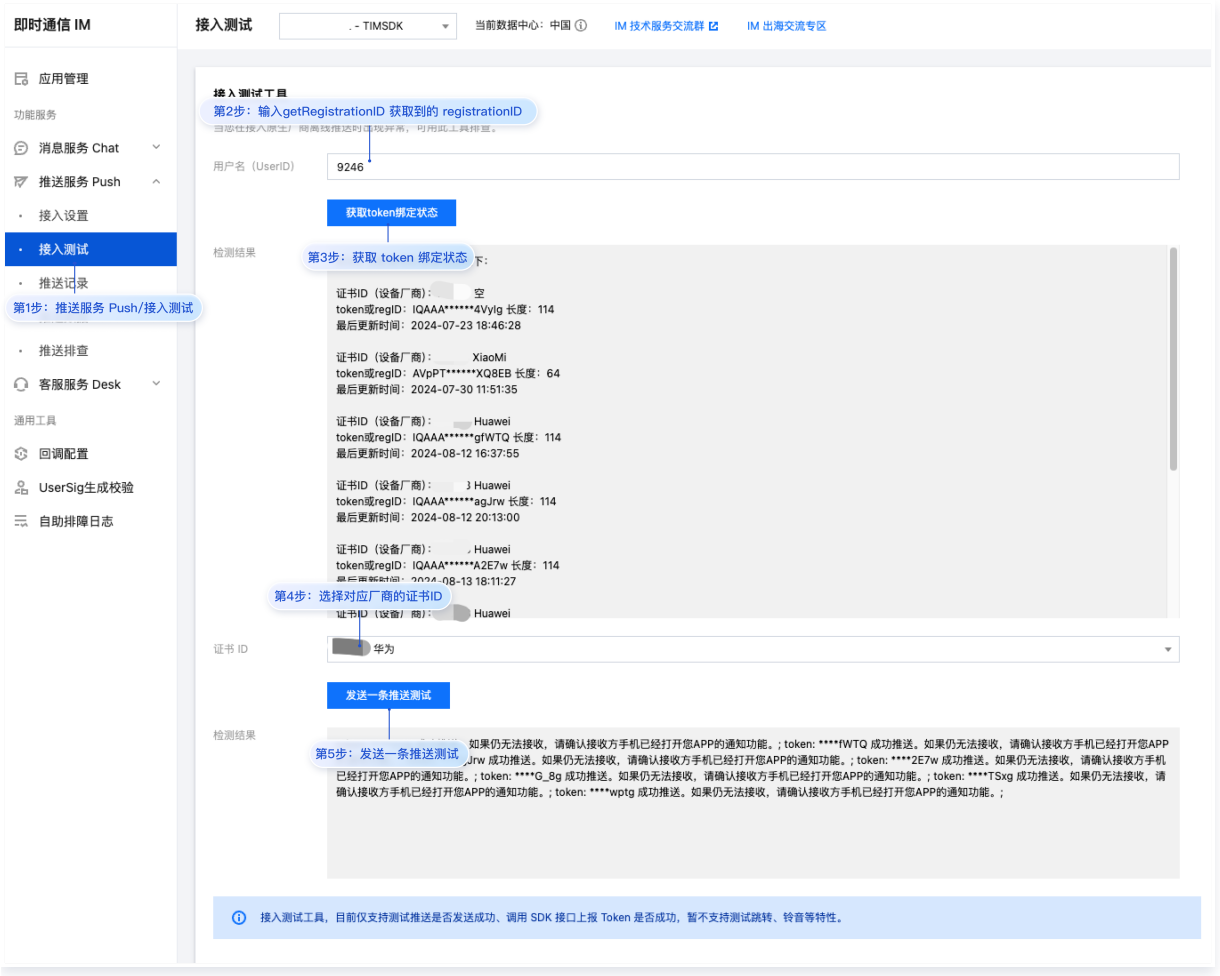
将 SDKAppID 和 appKey 替换为您在 IM 控制台 – 推送服务 Push – 接入设置页面 获取的应用的信息。如图所示：


```
import Push from "@tencentcloud/donut-push"
const sdkAppID = 0; // 您的 SDKAppID
const appKey = ''; // 客户端密钥
const registrationID = ""; //用户的 registrationID
const listener = (param) => {
  console.log('onEvent', JSON.stringify(param));
}
```

```
App({
  onLaunch: function () {
    // 请确保在调用 registerPush 方法之前设置好registrationID。
    // 如果您卸载并重新安装应用，registrationID 会发生改变。
    Push.setRegistrationID(registrationID)
      .then((res) => {
          console.info("setRegistrationID", JSON.stringify(res));
          return Push.registerPush(sdkAppID, appKey);
      }).then((res) => {
          console.info("registerPush", JSON.stringify(res));
          return Push.getRegistrationID();
      }).then((res) => {
          console.info("getRegistrationID", JSON.stringify(res));
      })
      .catch((res) => {
          console.error("registerPush failed", JSON.stringify(res));
    });
    // 监听在线推送
    Push.addPushListener(Push.EventName.NOTIFICATION_CLICKED, listener);
  }
})
```

体验您的第一次推送

在测试推送前，请务必打开通知和状态栏。进入 推送服务 Push > 接入测试 发送您的第一条推送。



推送结果回调

开启推送服务后，推送结果可以通过配置基础回调的方式，将结果转发给 App 后台，详细可参见：

- 普通推送结果回调
- 全员推送结果回调

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，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

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. **在开发者工具上清了缓存但是构建时还是用的原来的基座 App**

   需要修改 project.miniapp.json 文件才能重新构建远程基座 App，改一下应用的 version code 再进行构建即可。
2. **价格及购买相关问题，请参见 推送服务计费说明 。**
3. **iOS 在控制台上测试工具显示“上传 token 无效，请检查”**

   需要您检查下上传的 p12证书是否为推送证书、且 Bundle ID 应该是与主 App 一致。

技术咨询

点此进入腾讯云 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：2025-04-07 16:57:52

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

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：

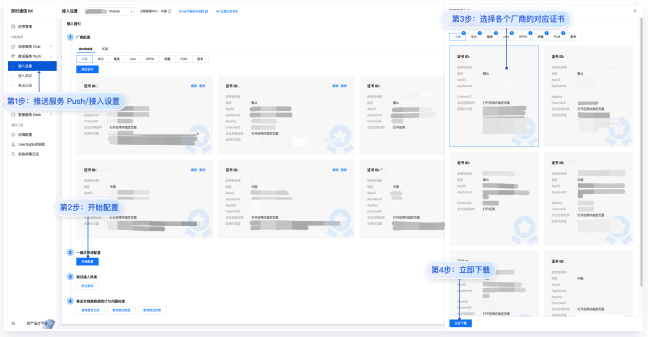
注册推送需要将此证书 ID（apnsCertificateID）传入：

```
TencentCloudChatPush().registerPush(apnsCertificateID: 您配置的证书 ID);
```

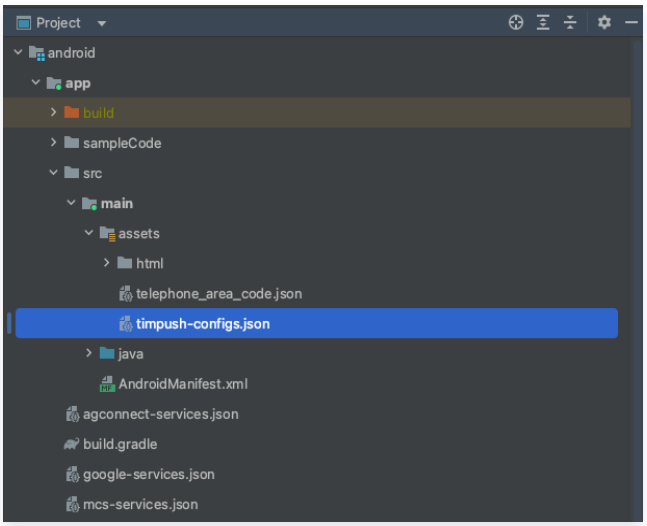
Android

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

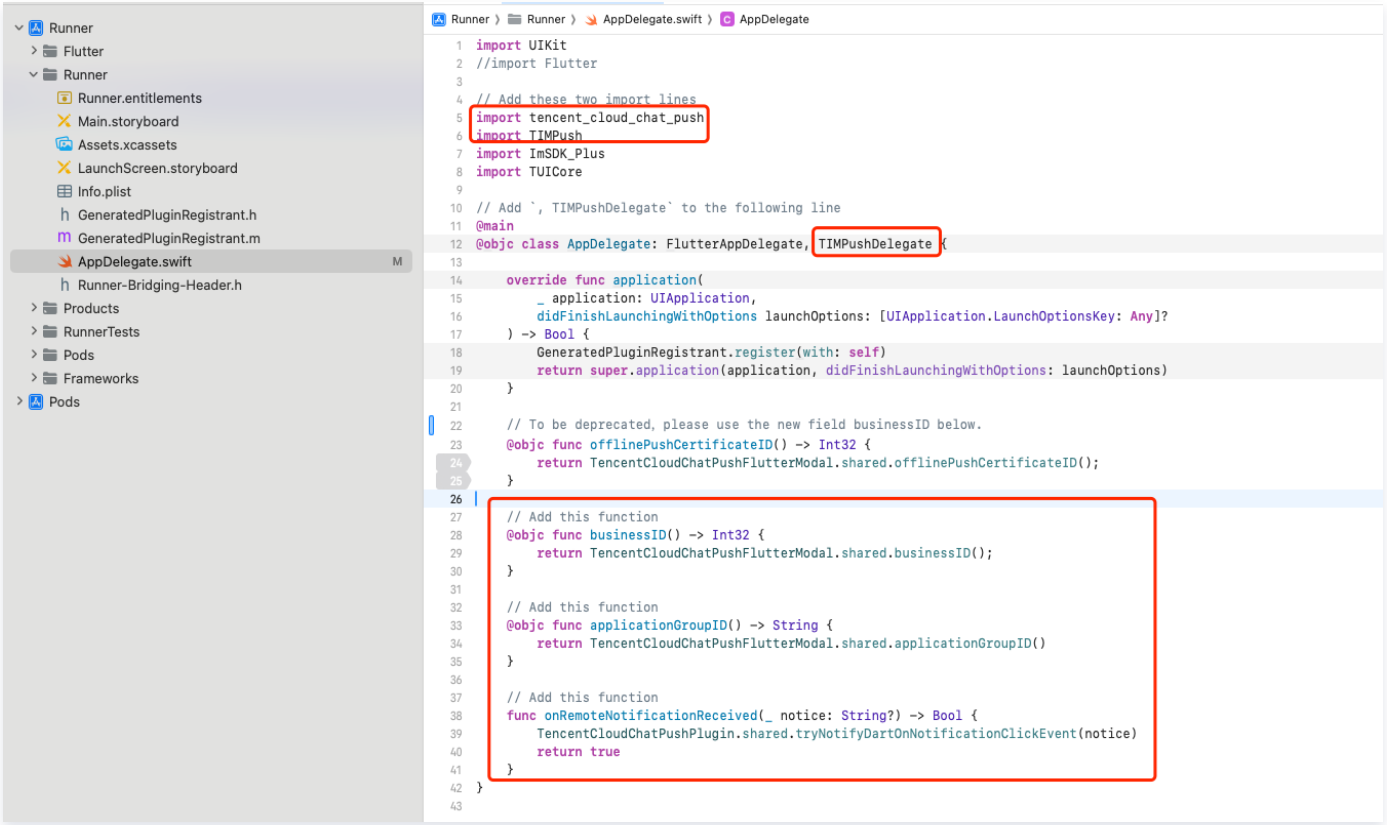


步骤3: 客户端代码配置

本步骤将需要编写若干原生代码，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

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。




```swift
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)
    }

    // To be deprecated, please use the new field businessID below.
    @objc func offlinePushCertificateID() -> Int32 {
        return TencentCloudChatPushFlutterModal.shared.offlinePushCertificateID();
    }

    // Add this function
    @objc func businessID() -> Int32 {
        return TencentCloudChatPushFlutterModal.shared.businessID();
    }
```

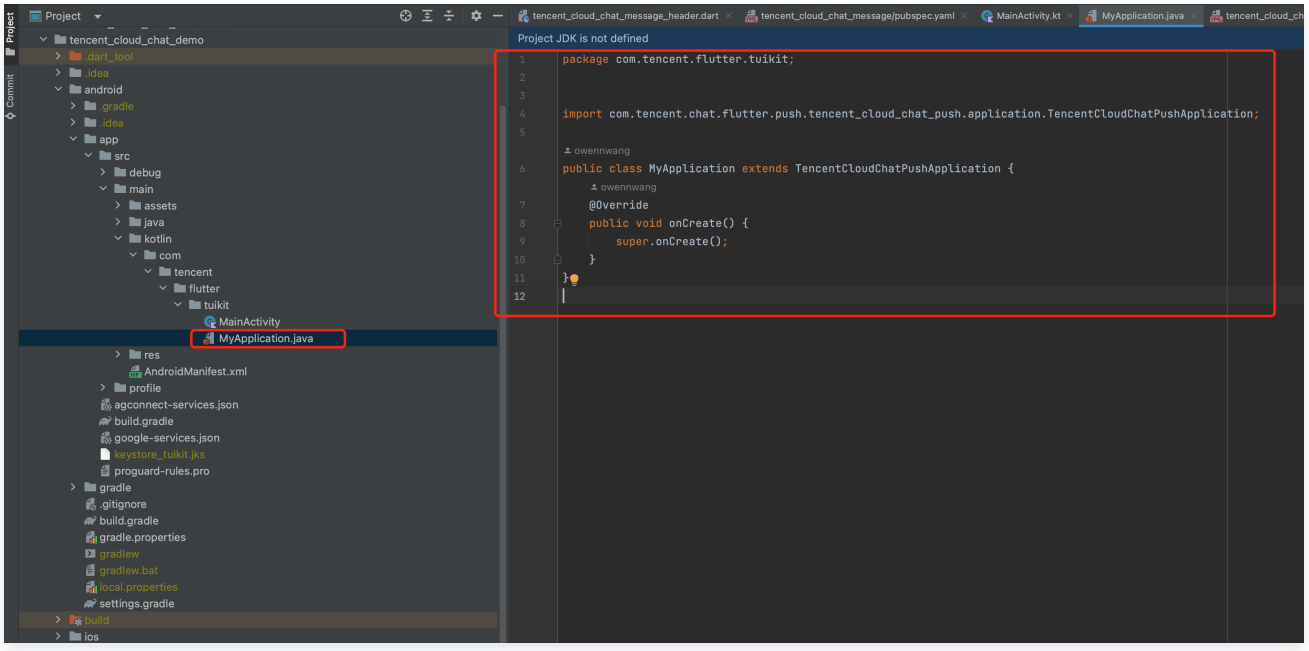
```
    // Add this function
    @objc func applicationGroupID() -> String {
        return TencentCloudChatPushFlutterModal.shared.applicationGroupID()
    }

    // Add this function
    @objc func onRemoteNotificationReceived(_ notice: String?) -> Bool {
        TencentCloudChatPushPlugin.shared.tryNotifyDartOnNotificationClickEvent(notice)
        return true
    }
}
```

注意：

1. iOS 控制台的证书 ID 请设置使用 businessID， offlinePushCertificateID 已废弃。
2. 也支持继续使用旧字段 offlinePushCertificateID，需要增加 @objc 关键字。

Android

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。



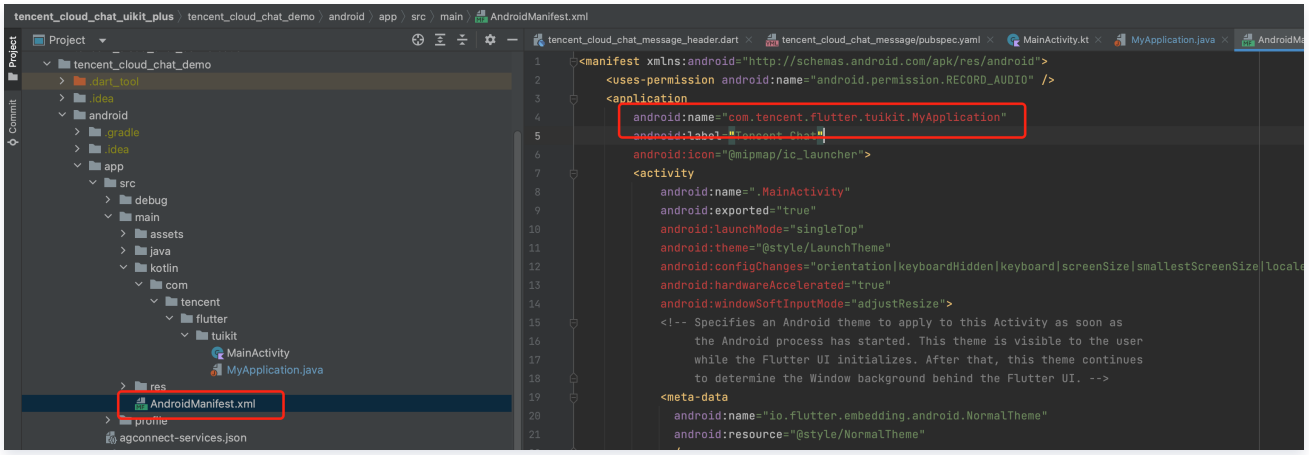
将下列代码粘贴到该文件中, 如上图所示：

```
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.TencentCloudChatPushApplicati
on;

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

iOS 端无需进行此步骤。

Android

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 {
    // 版本号“VERSION”请前往 更新日志 中获取配置。
    // Huawei
    implementation 'com.tencent.timpush:huawei:VERSION'
    // XiaoMi
    implementation 'com.tencent.timpush:xiaomi:VERSION'
    // OPPO
    implementation 'com.tencent.timpush:oppo:VERSION'
    // vivo
    implementation 'com.tencent.timpush:vivo:VERSION'
    // Honor
    implementation 'com.tencent.timpush:honor:VERSION'
    // Meizu
    implementation 'com.tencent.timpush:meizu:VERSION'
    // Google Firebase Cloud Messaging (Google FCM)
    implementation 'com.tencent.timpush:fcm:VERSION'
```

```
}
```

- **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

```
// 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"
        ]
    }
}
```

方法2

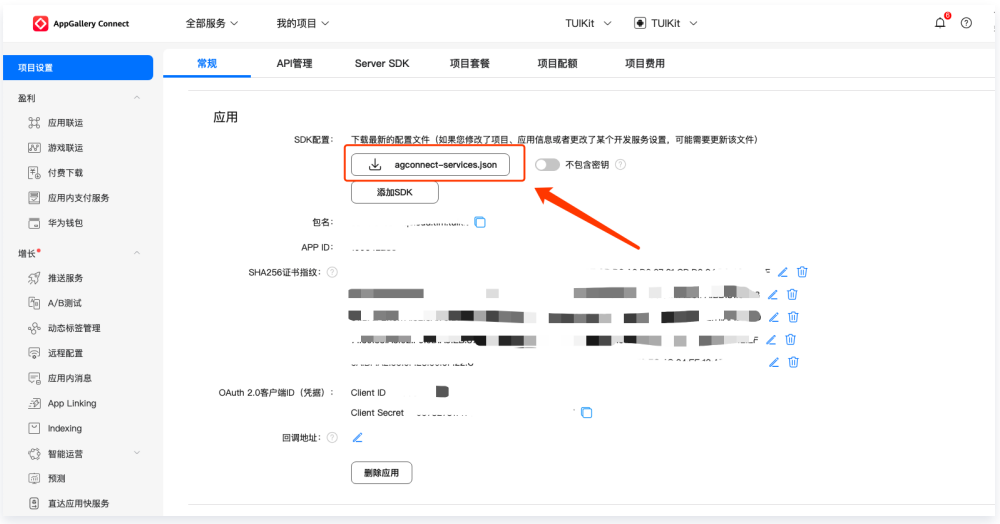
```
// android/app/src/main/AndroidManifest.xml

// 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。

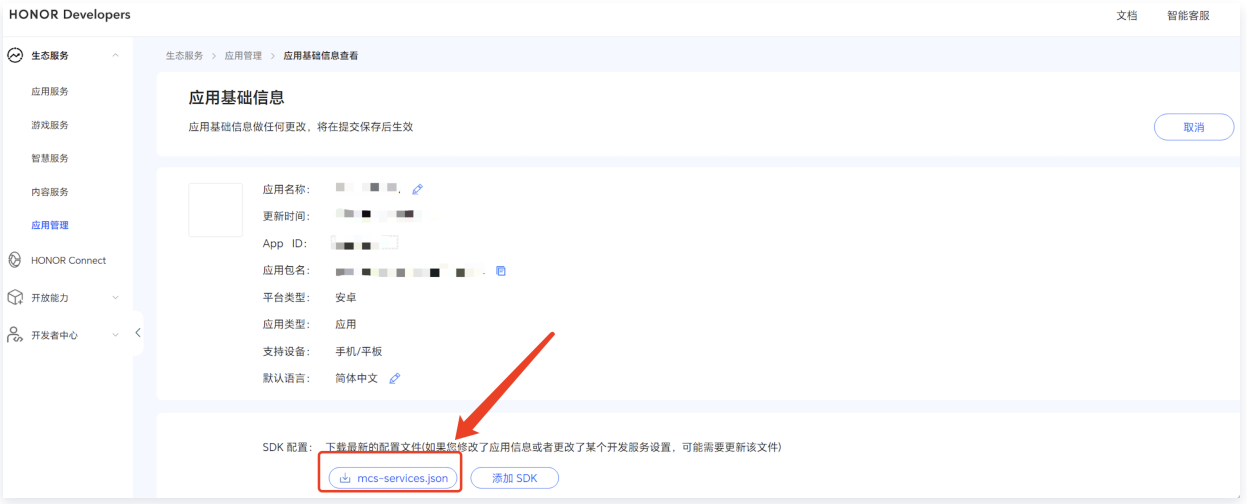
> ⚠ **注意：**
> 以下荣耀的适配仅 7.7.5283 及以上版本需要配置。

1.1 下载配置文件添加到工程根目录/Android/app。

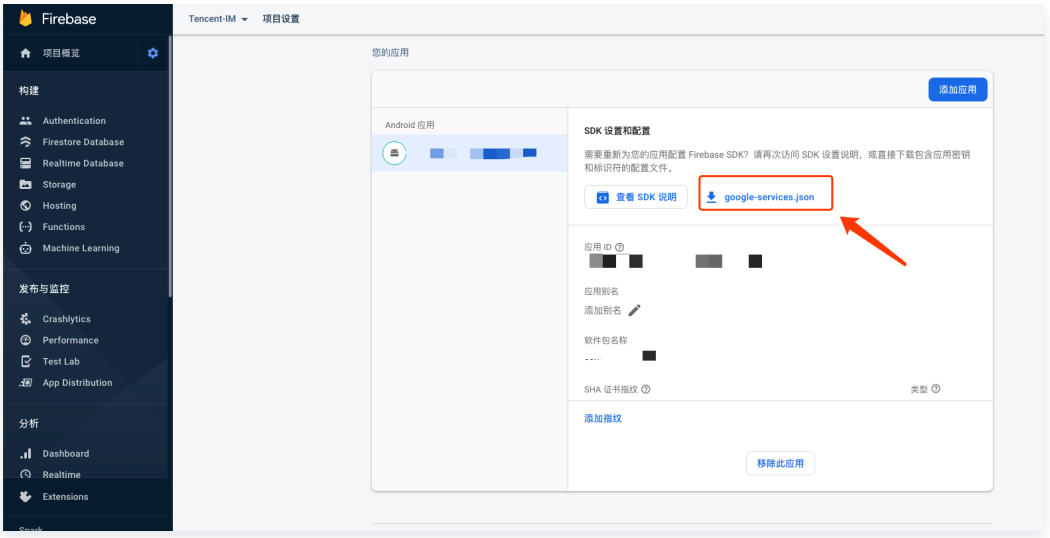
华为



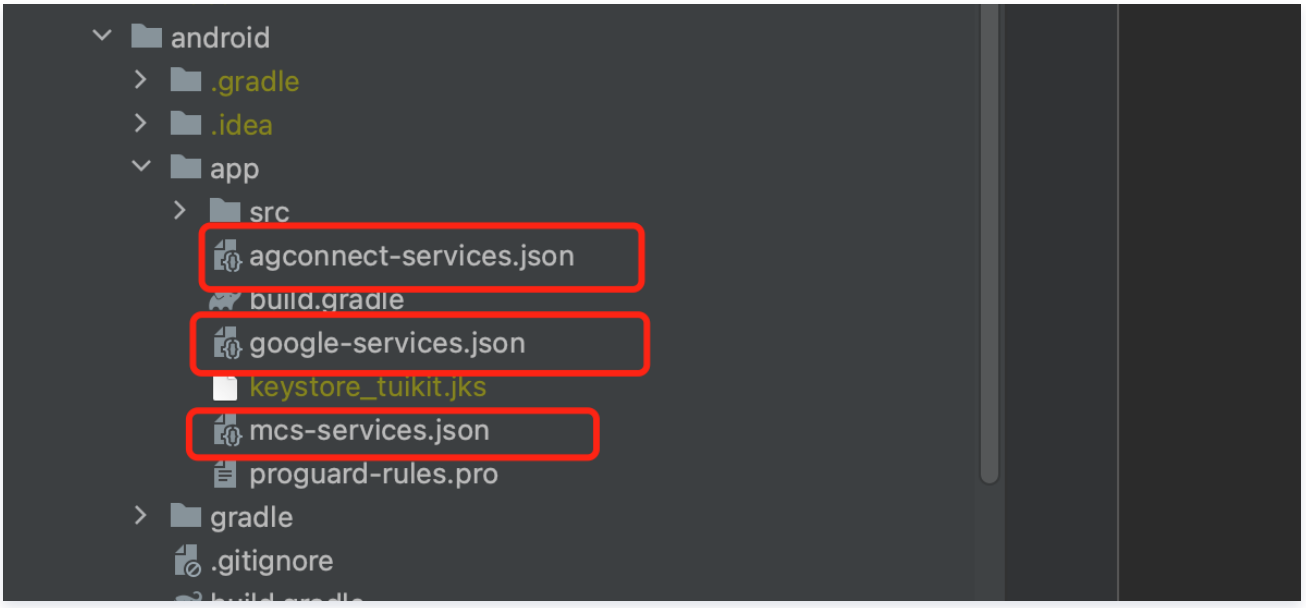
荣耀



Google FCM



操作路径



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

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:4.3.15'
    }
```

```
}
```

在项目级 settings.gradle 文件中 buildscript -> repositories 和 allprojects -> 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'}
    }
    }
}
```

Gradle 7.0 版本

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'}
    }
}
```

Gradle 7.0 以下版本

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'
```

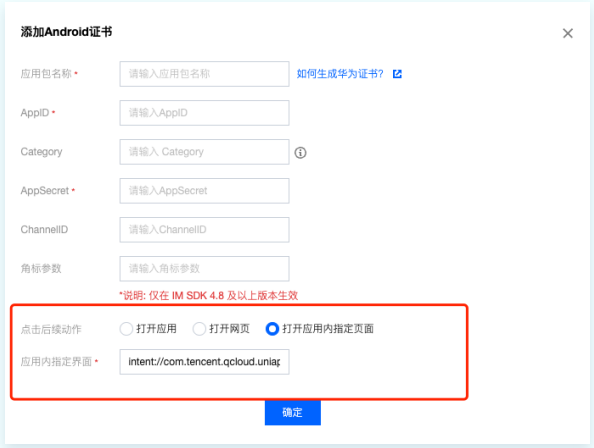
步骤5：处理消息点击回调，并解析参数

如果您需要自定义解析收到的远程推送，您可按照如下方法实现：

自定义点击跳转实现

注意：

1. 注册回调时机建议放在程序入口 main() 函数中。
2. 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**，请勿修改使用默认值。




```
TIMPushListener timPushListener = TIMPushListener(
      onNotificationClicked: (String ext) {
        debugPrint("ext: $ext");
        // 获取 ext 自定义跳转

      }
    );
tencentCloudChatPush.addPushListener(listener: timPushListener);
```

自定义点击跳转实现（旧方案）

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

请注意，不要在 Flutter 程序入口的 main 方法中调用。
调用 `TencentCloudChatPush().registerPush` **方法成功后，就可以收到离线推送通知了。**

```
TencentCloudChatPush().registerPush(
    onNotificationClicked: _onNotificationClicked,
    sdkAppId: 您的sdkAppId,
    appKey: "客户端密钥",
    apnsCertificateID: 您配置的证书 ID);
```

步骤7: 消息推送触达统计

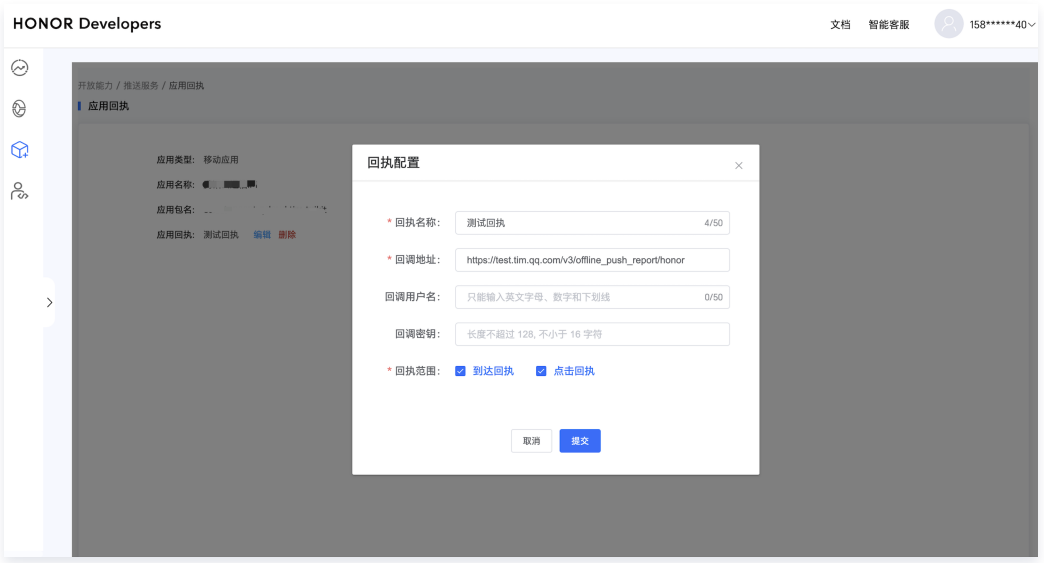
如果您需要统计触达数据，请按照如下完成配置：

华为

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。

荣耀



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

vivo

回调地址配置

回执地址：

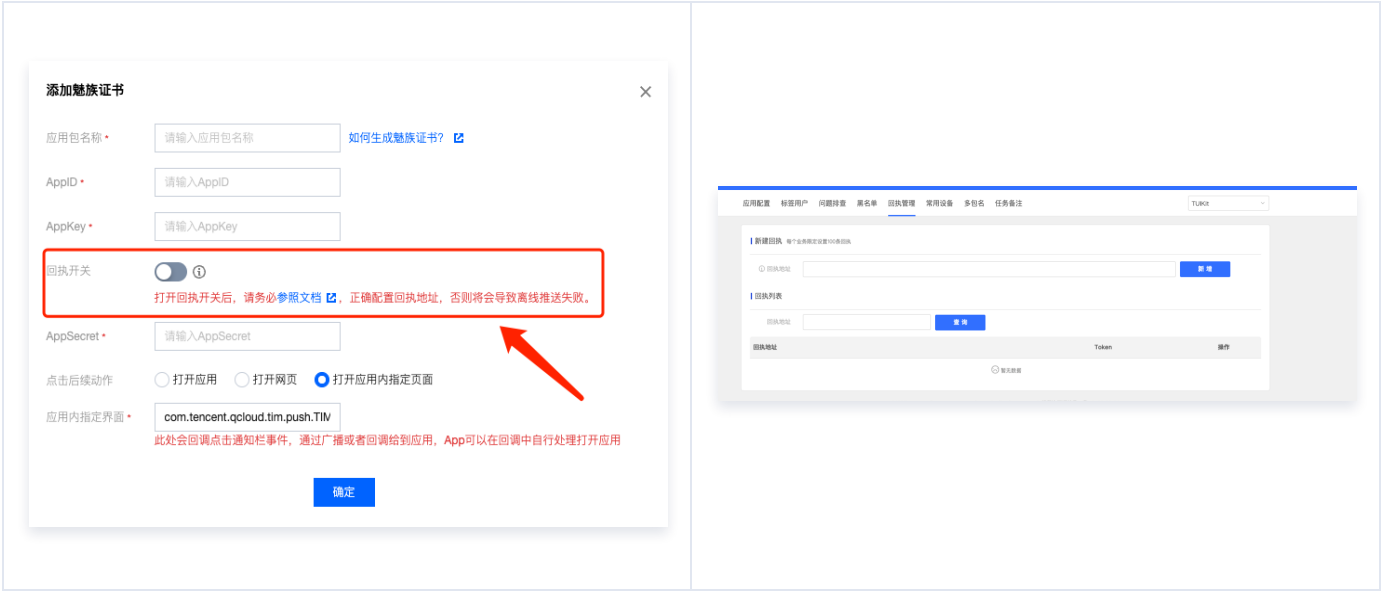
`https://api.im.qcloud.com/v3/offline_push_report/vivo`

回执 ID 配置 IM 控制台

魅族

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

1. iOS 端推送触达统计配置，请参见：
   - 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 ）。
   - 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
withContentHandler:(void (^)(UNNotificationContent * _Nonnull))contentHandler {
    //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。
    //格式为group + [主bundleID]+ key
    //如group.com.tencent.im.pushkey
    NSString * appGroupID = kTIMPushAppGroupKey;
    __weak typeof(self) weakSelf = self;
    [TIMPushManager handleNotificationServiceRequest:request appGroupID:appGroupID
callback:^(UNNotificationContent *content) {
        weakSelf.bestAttemptContent = [content mutableCopy];
        // Modify the notification content here...
        // self.bestAttemptContent.title = [NSString stringWithFormat:@"%@ [modified]",
self.bestAttemptContent.title];
        weakSelf.contentHandler(weakSelf.bestAttemptContent);
    }];
}
```

```
@end
```

注意：

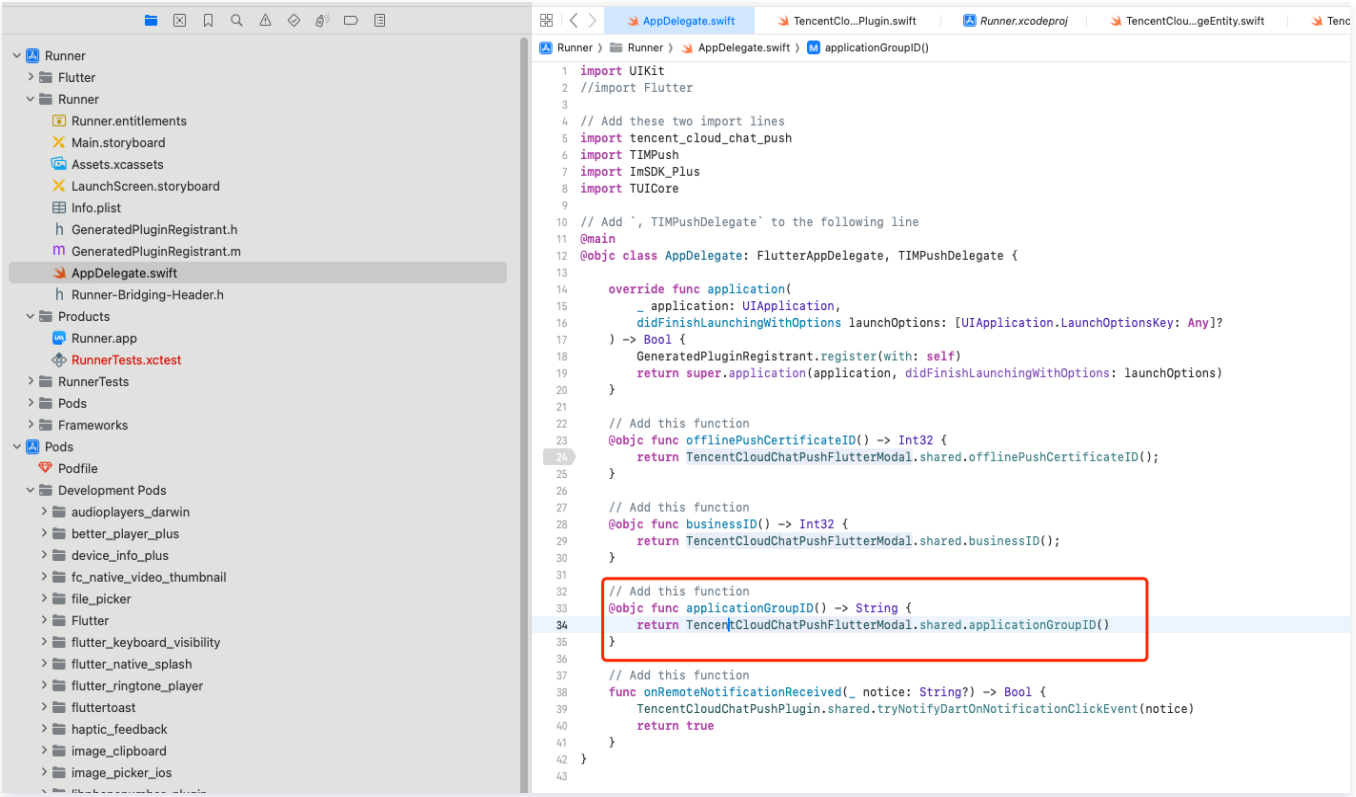
1. 上报推送触达数据，需要开启 mutable-content 开关来支持 iOS 10 的 Extension 功能。

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。

2. 调用 TencentCloudChatPush().registerPush 方法，需传入 App Group ID, 是标识当前主 APP 和 Extension 之间共享的 APP Group，需要在主 APP 的 Capability 中配置 App Groups 。

```
TencentCloudChatPush().registerPush(
    onNotificationClicked: _onNotificationClicked,
    sdkAppId: 您的sdkAppId,
    appKey: "客户端密钥",
    applicationGroupID: "您配置的 applicationGroupID");
```

3. 检查 `ios/Runner/AppDelegate.swift` 文件中是否有实现 applicationGroupID 方法。



其余支持厂商不需要配置，FCM 暂不支持推送统计功能。

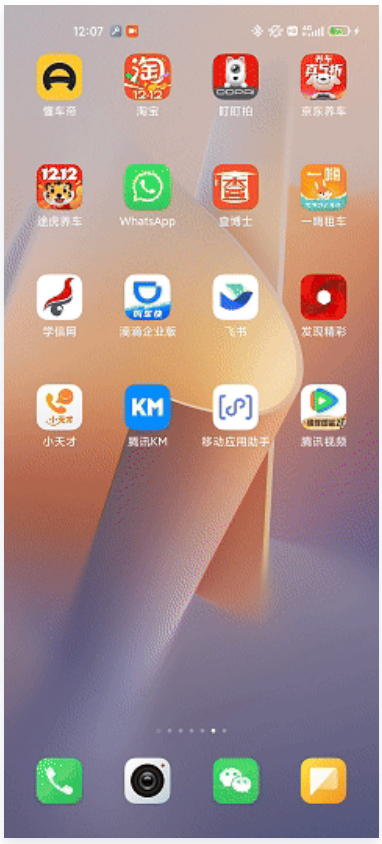
恭喜您已经完成了推送插件的接入，需要提醒您：推送插件**试用或购买到期后，将自动停止提供推送服务（包括普通消息离线推送、全员/标签推送等服务）**。为避免影响您业务正常使用，请提前购买/续费 。

React Native

最近更新时间：2025-04-07 15:53:52

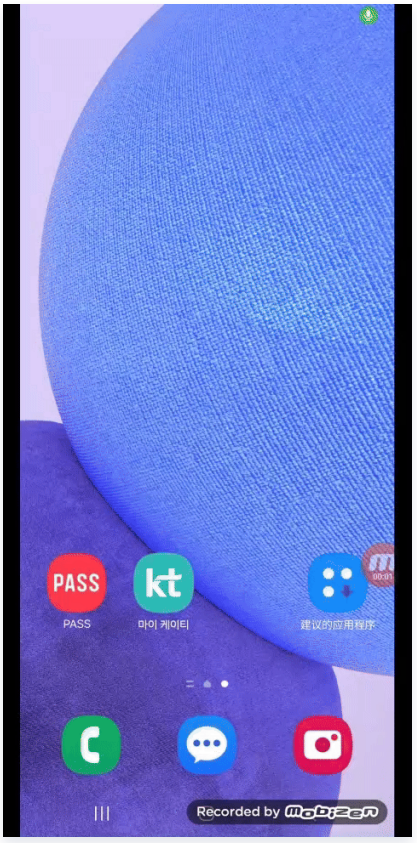
效果展示

小米 11 Pro



iPhone 13

三星 Galaxy A23 海外版（Google FCM 推送）



步骤1：创建一个 React Native 项目（已有项目可忽略此步骤）

```
npx @react-native-community/cli@latest init MyReactNativeApp --version 0.75.0
```

步骤2：进入 MyReactNativeApp 目录，集成 @tencentcloud/react-native-push

npm

```
npm install @tencentcloud/react-native-push --save
```

yarn

```
yarn add @tencentcloud/react-native-push
```

步骤3：注册推送

复制下面的代码到 `App.tsx`，并将 `SDKAppID` 和 `appKey` 替换为您的应用的信息。

说明：

`registerPush` 注册推送服务成功后，您可通过 `getRegistrationID` 获取推送 ID 标识，即 RegistrationID。您可以向指定的 RegistrationID 推送消息。

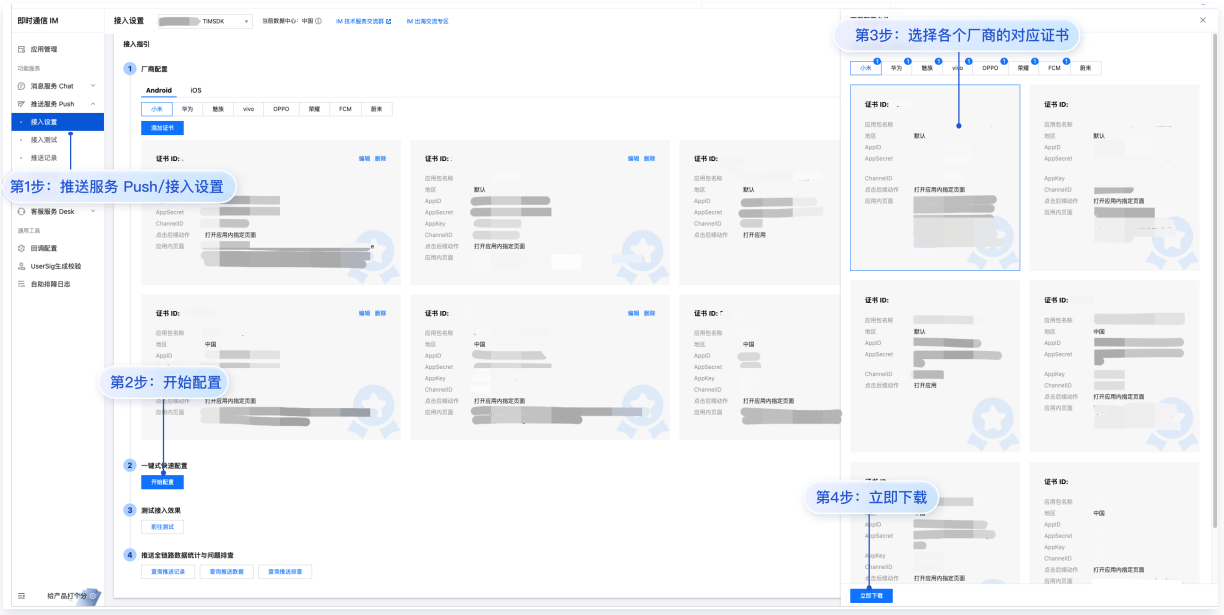

```
import Push from '@tencentcloud/react-native-push';

const SDKAppID = 0; // 您的 SDKAppID
const appKey = ''; // 客户端密钥

if (Push) {
  // 如果您需要与 Chat 的登录 userID 打通（即向此 userID 推送消息），请使用 setRegistrationID 接口
  // Push.setRegistrationID(userID, () => {
      // console.log('setRegistrationID ok', userID);
  // });

  Push.registerPush(SDKAppID, appKey, (data) => {
      console.log('registerPush ok', data);
      Push.getRegistrationID((registrationID) => {
        console.log('getRegistrationID ok', registrationID);
      });
    }, (errCode, errMsg) => {
      console.error('registerPush failed', errCode, errMsg);
    }
  );

  // 监听通知栏点击事件，获取推送扩展信息
  Push.addPushListener(Push.EVENT.NOTIFICATION_CLICKED, (res) => {
    // res 为推送扩展信息
    console.log('notification clicked', res);
  });

  // 监听在线推送
  Push.addPushListener(Push.EVENT.MESSAGE_RECEIVED, (res) => {
    // res 为消息内容
```

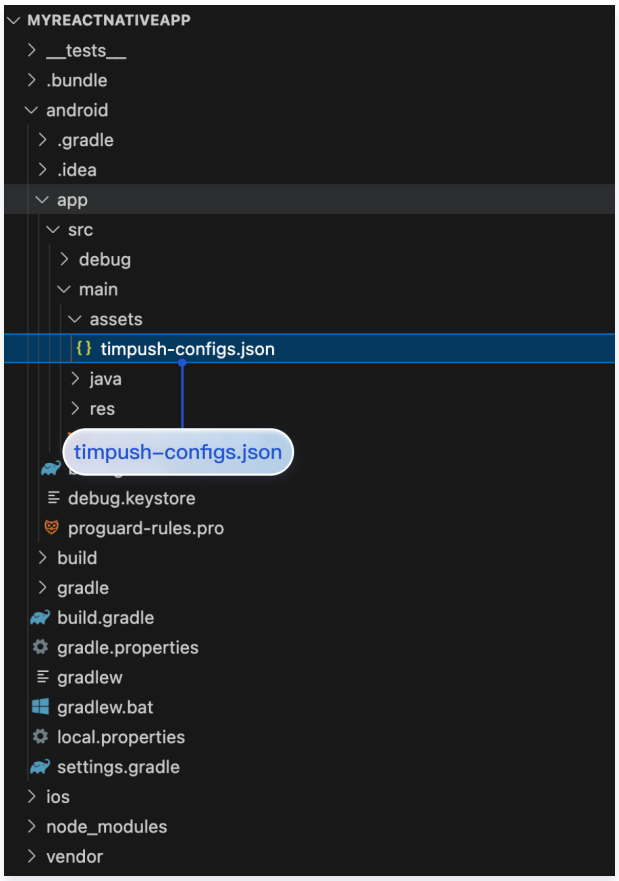
```
    console.log('message received', res);
  });

  // 监听在线推送被撤回
  Push.addPushListener(Push.EVENT.MESSAGE_REVOKED, (res) => {
    // res 为被撤回的消息 ID
    console.log('message revoked', res);
  });
}
```

步骤4：配置厂商推送

Android

1. 请您完成控制台厂商推送信息填写后，从控制台下载 `timpush-configs.json` 文件，并添加到项目的 `MyReactNativeApp/android/app/src/main/assets` 目录下，如果该目录不存在，请手动创建。如图所示：

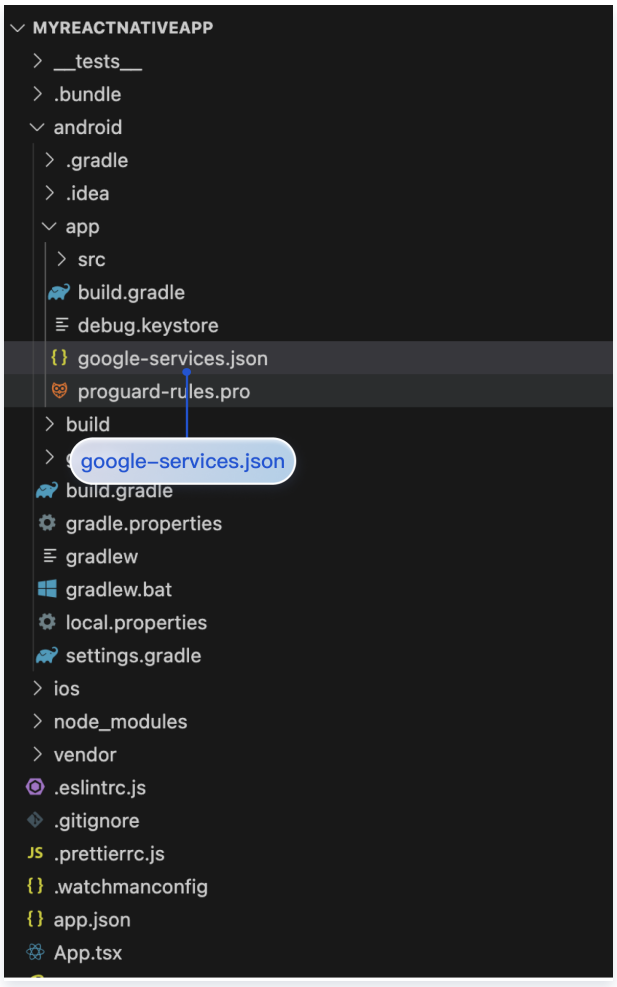




2. 华为、荣耀、vivo、FCM。

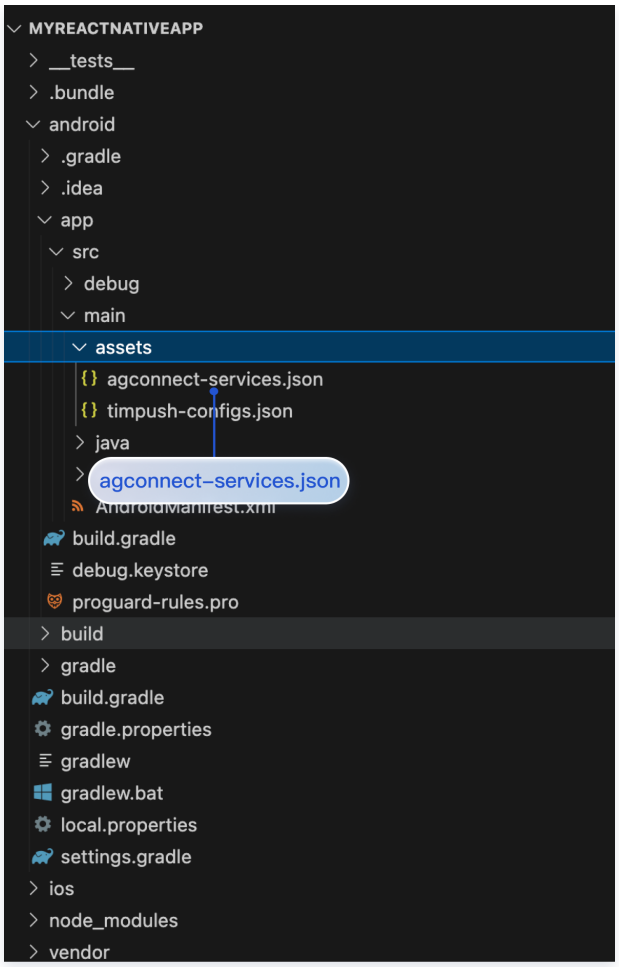
FCM

您需要支持 FCM 推送时，须配置 `google-services.json` 文件到 `MyReactNativeApp/android/app` 目录下（**请注意！不是 `MyReactNativeApp/android/app/src/main/assets` 目录**）。如图所示：



华为

您需要支持华为推送时，须配置 `agconnect-services.json` **文件到** `MyReactNativeApp/android/app/src/main/assets/` **目录下。如图所示：**

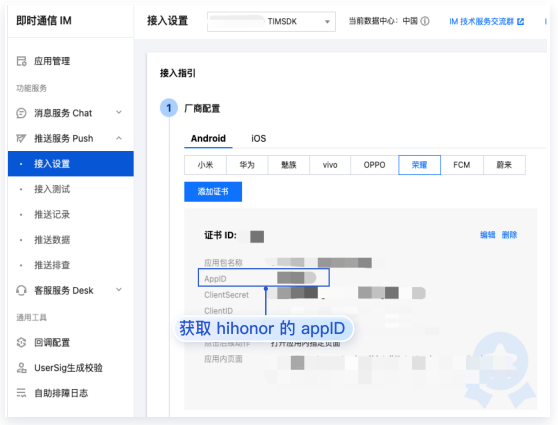


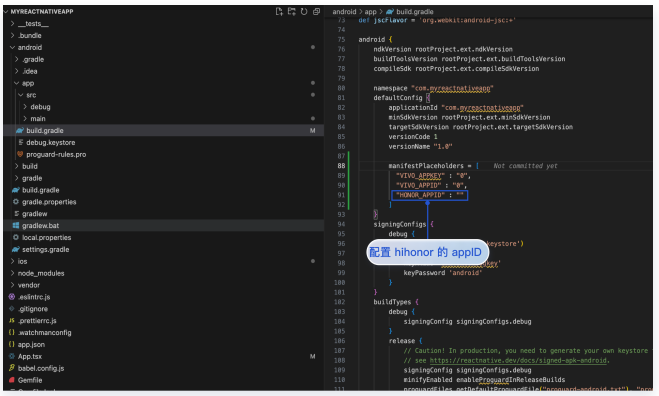
荣耀

1. 您需要支持荣耀推送时，须配置 `appID` 到 `MyReactNativeApp/android/app/build.gradle` 文件中。如图所示：

```
......
android {
    ......
    defaultConfig {
        ......
        manifestPlaceholders = [
          "HONOR_APPID" : ""
        ]
    }
}
```

获取荣耀的 appID　　配置荣耀的 appID





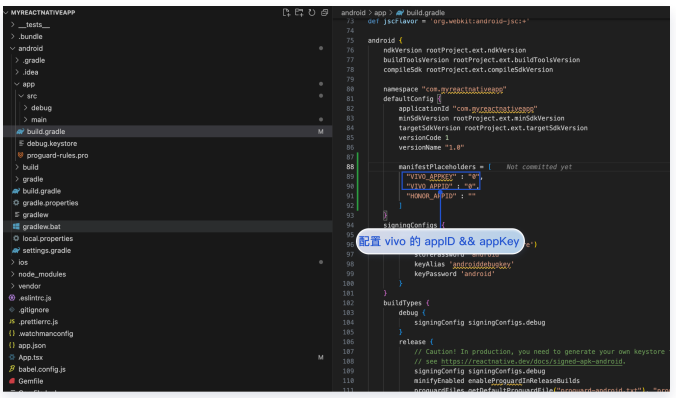
2. 从**荣耀开发者管理中心**下载 `mcs-services.json` **文件，并配置到** `MyReactNativeApp/android/app` **目录下（请注意！不是** `MyReactNativeApp/android/app/src/main/assets` **目录）。**

vivo

您需要支持 vivo 推送时，须配置 `appID` **和** `appKey` **到** `MyReactNativeApp/android/app/build.gradle` **文件中。如图所示：**

```
......
android {
    ......
    defaultConfig {
        ......
        manifestPlaceholders = [
          "VIVO_APPKEY" : "0",
          "VIVO_APPID" : "0",
        ]
    }
}
```

获取 vivo 的 appID && appKey	配置 vivo 的 appID && appKey

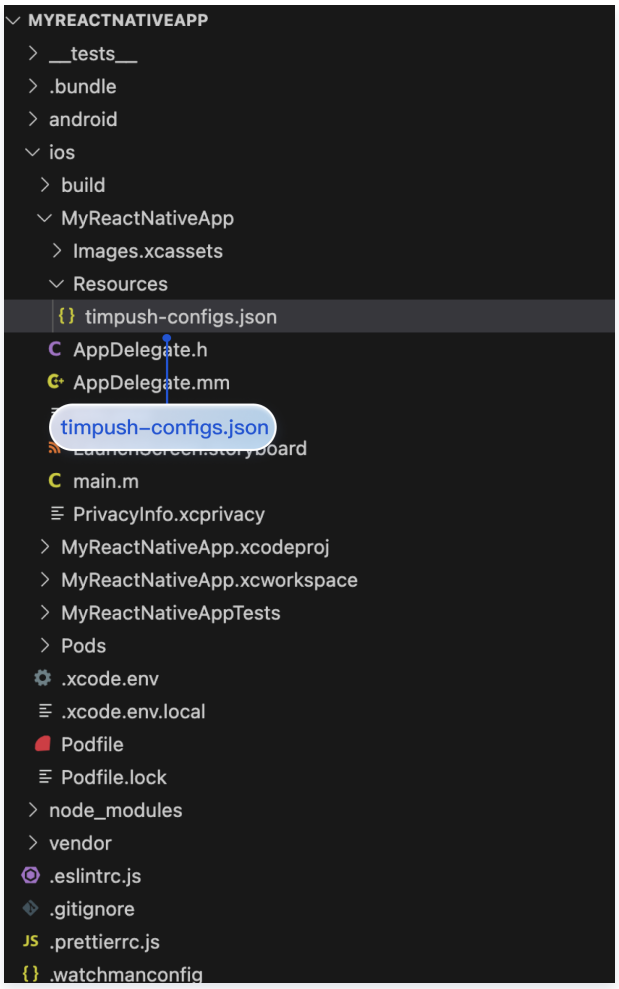


iOS

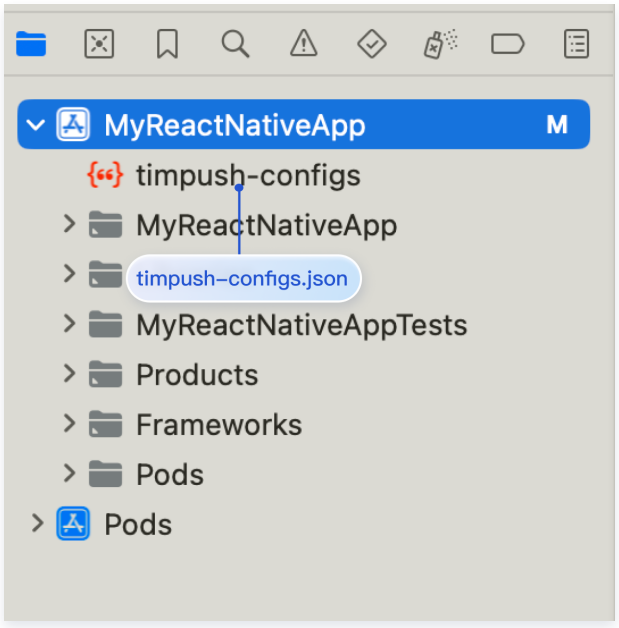
1. 请将您在厂商配置步骤中，获取到的 iOS APNs 推送证书，上传至 IM 控制台。IM 控制台会为您分配一个证书 ID，如图所示：

2. 在 `MyReactNativeApp/ios/MyReactNativeApp` 目录下，新建 `Resources` 文件夹，并新建 `timpush-configs.json` 文件。编辑 `timpush-configs.json`，填入控制台获取的证书 ID，如下所示：

```
{
  "businessID": "您的证书ID"
}
```



3. XCode 打开 MyReactNativeApp 项目，右键单击项目 > **Add Files to "MyReactNativeApp"**，将 `timpush-configs.json` 目录添加到工程。如图所示：



步骤5：配置 Native Modules 和相关依赖

Android

> **说明：**
> 请确保 `timpush-configs.json` 文件内的包名和 `MyReactNativeApp/android/app/build.gradle` 文件内的 `applicationId` 值一致，不一致则会导致离线推送不可用。

1. 使用 Android Studio 打开 `MyReactNativeApp/android` 目录。
2. 修改项目入口文件。

项目入口文件为：MainApplication.kt

```
...
import com.tencent.qcloud.rntimpush.TencentCloudPushApplication

// Replace Application with TencentCloudPushApplication
class MainApplication : TencentCloudPushApplication(), ReactApplication {
  ...
  //  add TencentCloudPushPackage to the list of packages returned in ReactNativeHost's
getPackages() method
  override fun getPackages(): List<ReactPackage> =
    PackageList(this).packages.apply {
        // Packages that cannot be autolinked yet can be added manually here, for example:
        // add(MyReactNativePackage())
    }
}
```

项目入口文件为：MainApplication.java

```
...
import com.tencent.qcloud.rntimpush.TencentCloudPushApplication;

// Replace Application with TencentCloudPushApplication
public class MainApplication extends TencentCloudPushApplication implements ReactApplication {
  ...
  // add TencentCloudPushPackage to the list of packages returned in ReactNativeHost's
getPackages() method
  @Override
  protected List<ReactPackage> getPackages() {
    List<ReactPackage> packages = new PackageList(this).getPackages();
    // Packages that cannot be autolinked yet can be added manually here, for example:
    // packages.add(new MyReactNativePackage());
    return packages;
  }
  ...
```

3. 编辑 `android/build.gradle` 文件，更新 `repositories`，`dependencies` 和 `allprojects`。

```
buildscript {
    ...
    repositories {
        ...
        google()
        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 {
        ...
        // 如果您创建的项目 com.android.tools.build:gradle 未带版本号，请设置为 8.5.0
        // classpath("com.android.tools.build:gradle:8.5.0")
        classpath 'com.google.gms:google-services:4.3.15'
        classpath 'com.huawei.agconnect:agcp:1.9.1.301'
        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' }
    }
}
...
```

4. 编辑 `android/app/build.gradle` 文件，按需配置厂商的推送包并应用插件。

```
...
// 如果您的 APP 需要 FCM 推送，请取消下行的注释
// apply plugin: 'com.google.gms.google-services'
// 如果您的 APP 需要华为推送，请取消下行的注释
// apply plugin: 'com.huawei.agconnect'
// 如果您的 APP 需要荣耀推送，请取消下行的注释
// apply plugin: 'com.hihonor.mcs.asplugin'
...
android {
    ...
    defaultConfig {
        ...
        manifestPlaceholders = [
            "VIVO_APPKEY" : "0", // 如果您的 APP 需要 vivo 推送，请配置 'VIVO_APPKEY' 和
'VIVO_APPID'
            "VIVO_APPID" : "0",
            "HONOR_APPID" : "" // 如果您的 APP 需要荣耀推送，请配置 'HONOR_APPID'
        ]
    }
}

dependencies {
  ...
  // 请您根据需要，引入下列全部或者部分厂商的推送包。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。
  implementation 'com.tencent.timpush:huawei:8.5.6864'
  implementation 'com.tencent.timpush:xiaomi:8.5.6864'
  implementation 'com.tencent.timpush:oppo:8.5.6864'
  implementation 'com.tencent.timpush:vivo:8.5.6864'
  implementation 'com.tencent.timpush:honor:8.5.6864'
  implementation 'com.tencent.timpush:meizu:8.5.6864'
  implementation 'com.tencent.timpush:fcm:8.5.6864'
```

```
}
```

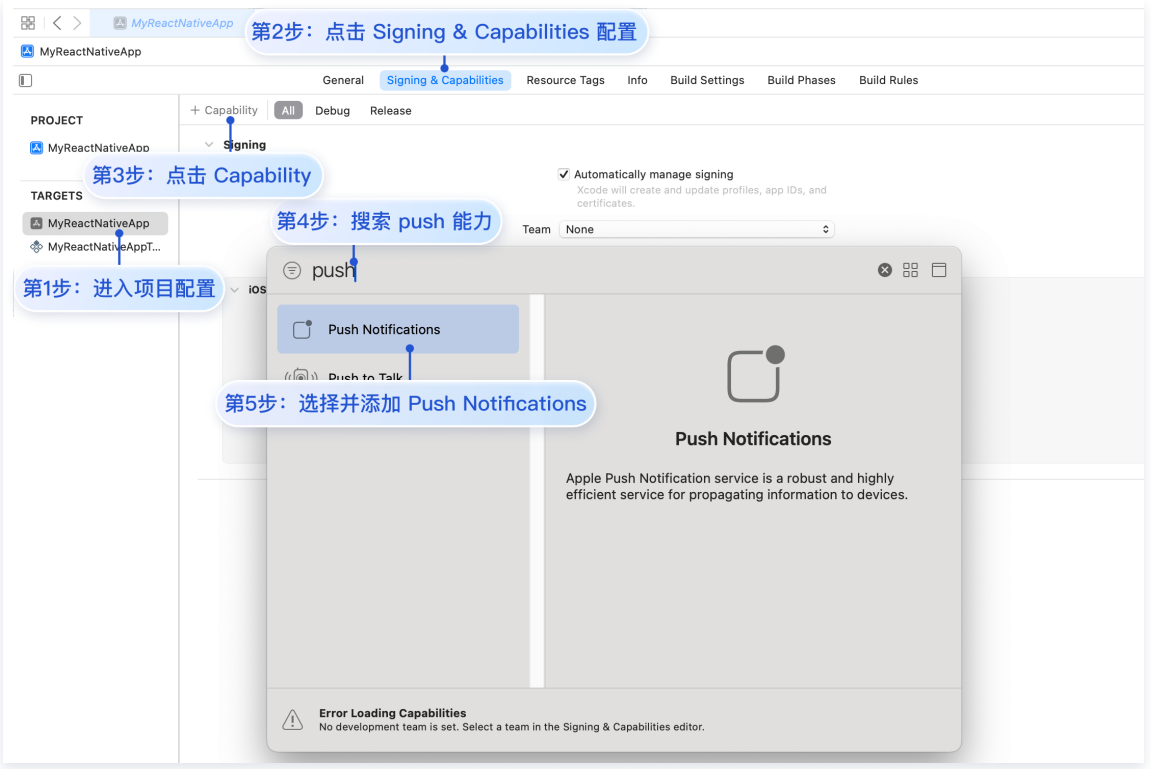
5. 以上操作都完成后，选择 **File > Sync Project with Gradle Files**。

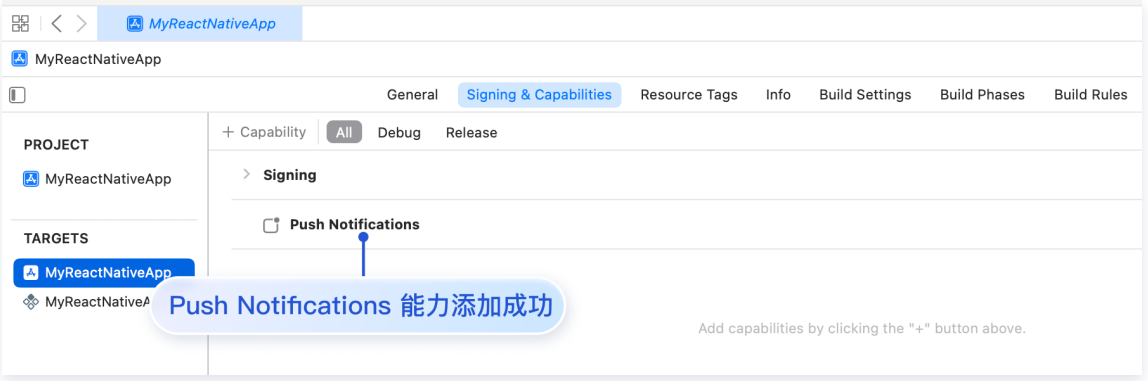
iOS

1. *使用 XCode 打开 MyReactNativeApp/ios/MyReactNativeApp.xcworkspace。*
2. 进入 `MyReactNativeApp/ios` 目录，安装 TIMPush。

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

3. 在 App 中启用推送通知功能。打开 Xcode 项目，在 **Project > Target > Capabilities** 页面选择并添加 **Push Notifications**。





步骤6：在真机上运行（测试前请务必打开手机通知权限，允许应用通知。）

从项目根目录开始，在命令提示符中运行以下命令，在设备上安装并启动您的应用程序：

Android

```
npm run android
```

iOS

```
npm run ios
```

步骤7：消息推送触达统计

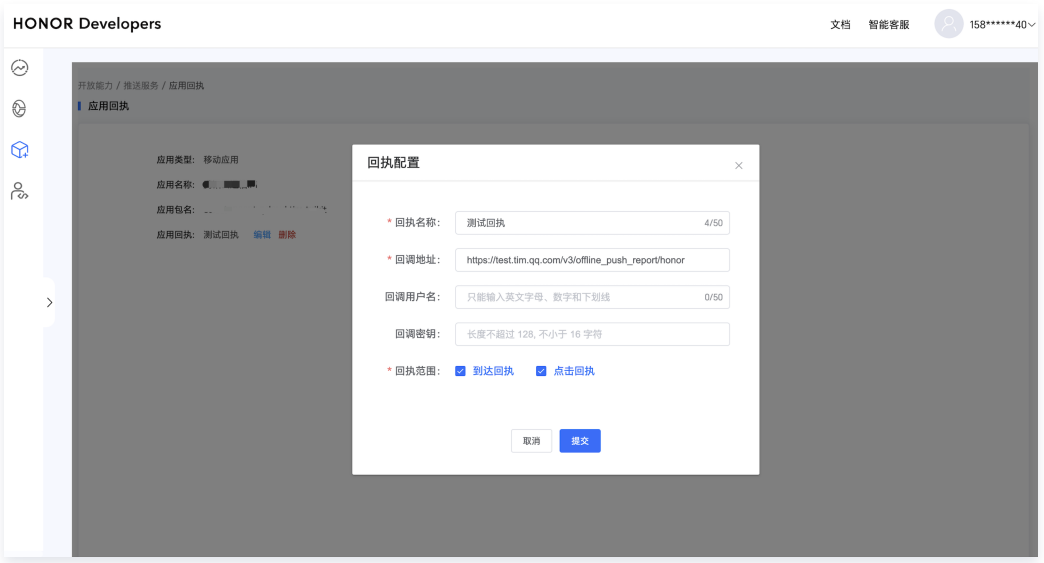
如果您需要统计触达数据，请按照如下完成配置：

华为

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

> **说明：**
> 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。

荣耀



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

vivo

回调地址配置

回执地址：

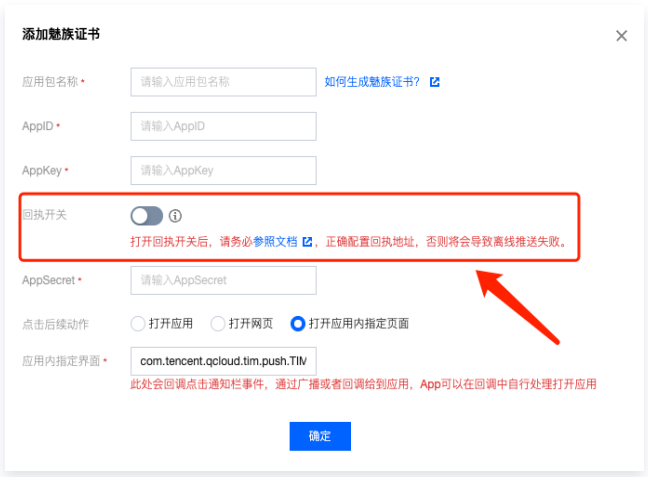
`https://api.im.qcloud.com/v3/offline_push_report/vivo`

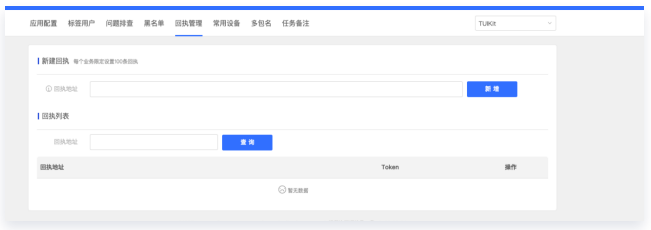
回执 ID 配置 IM 控制台

魅族

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`

> **说明：**
> 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

iOS 端推送触达统计配置，详细请参见 **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。

数据统计

最近更新时间：2025-05-16 17:08:22

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

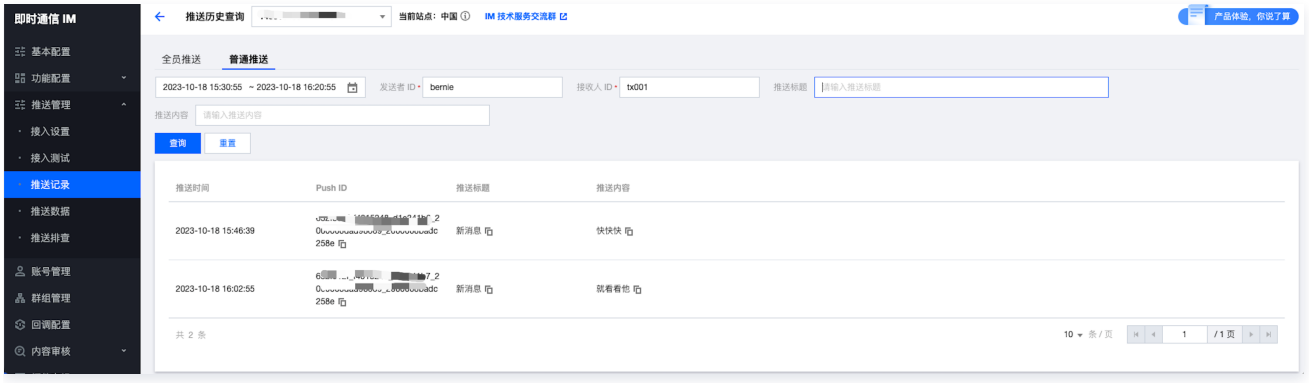
可以查询该应用下用户的所有推送记录数据，需要指定查询的时间窗口、发送者 ID 和接受者 ID，查询用户推送数据包括推送时间、Push ID、推送标题和推送内容；并支持按照推送标题或者推送内容搜索定位具体的推送记录。

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

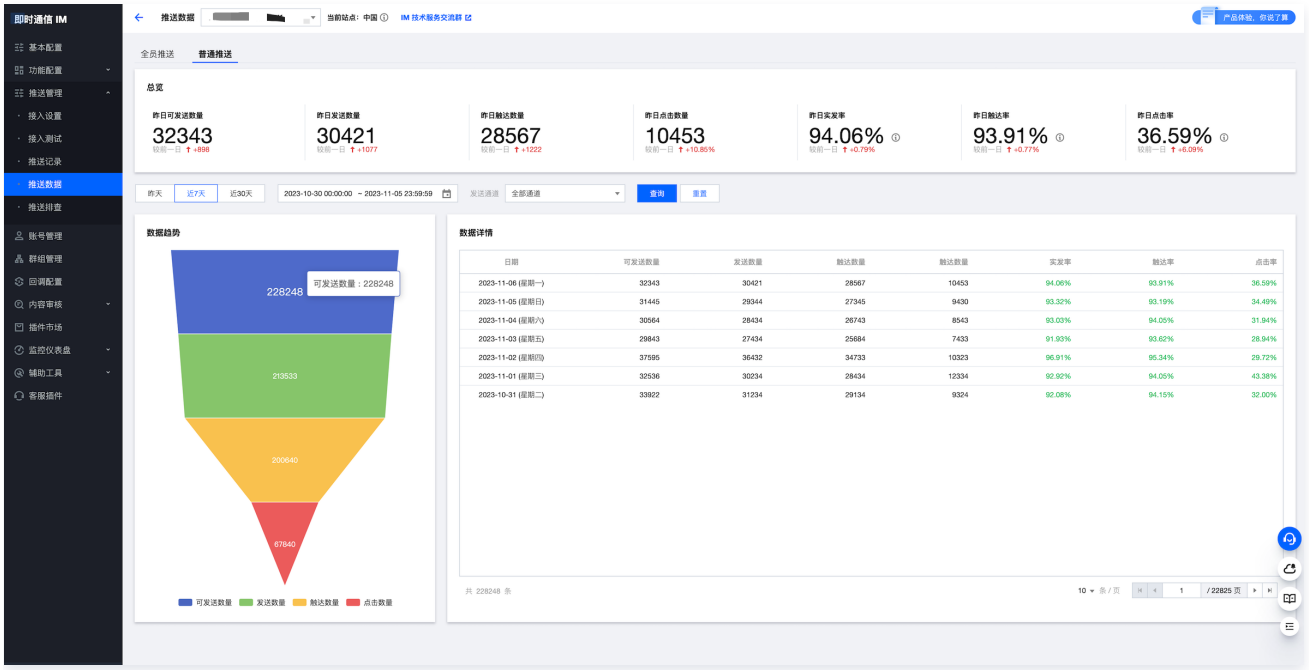
支持昨日、近7天、近30天以及指定时间窗口类型。

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

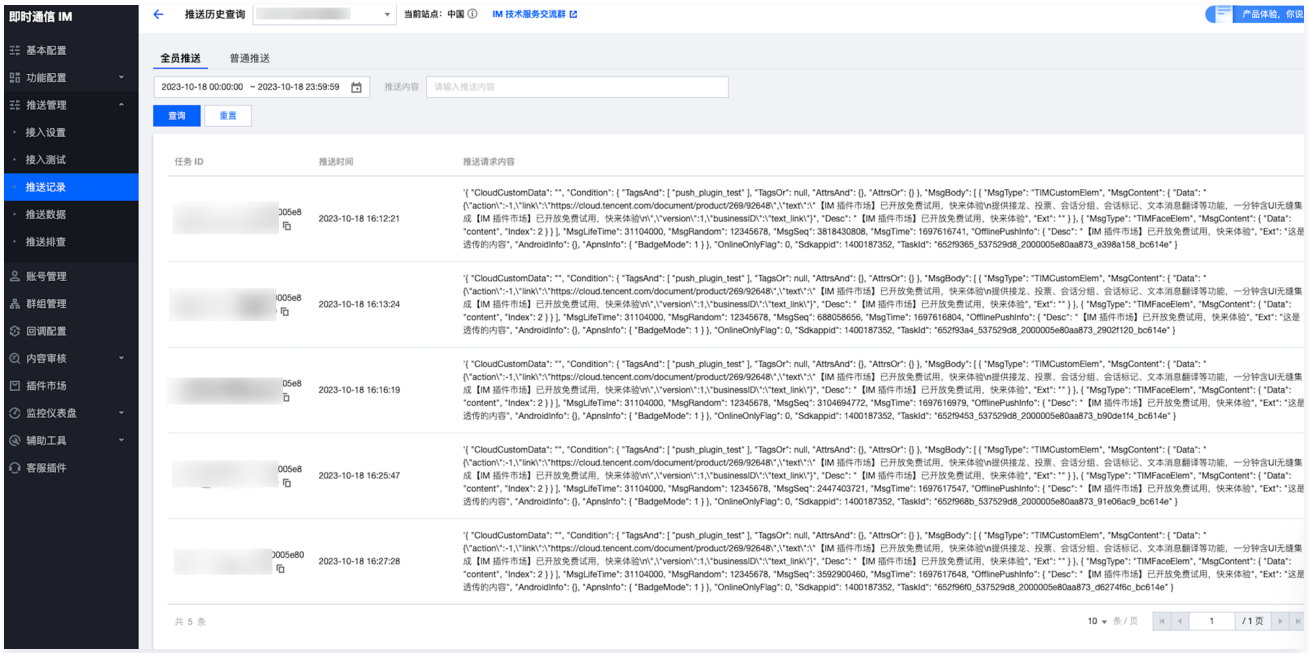
可以查询该应用下向用户发送的所有全员/标签推送记录数据，需要指定查询的时间窗口，查询用户推送数据包括推送时间、任务 ID、推送请求内容；并支持按照推送内容搜索定位具体的推送记录。

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

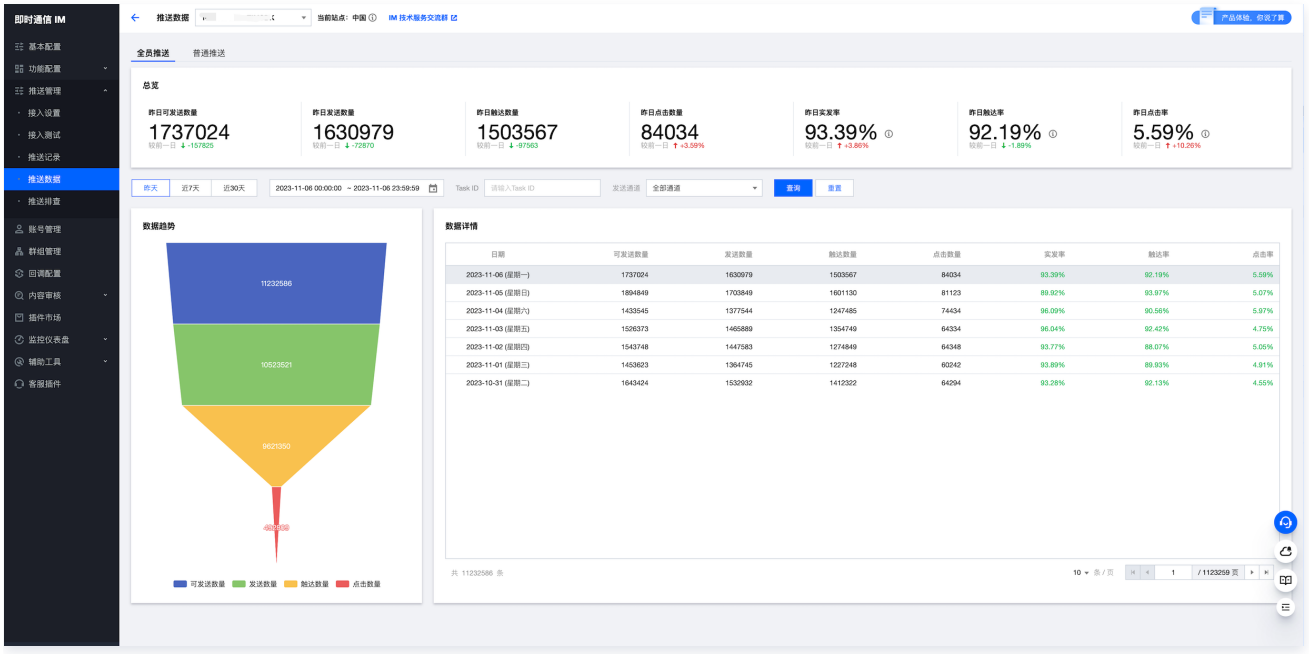
支持昨日、近7天、近30天以及指定时间窗口类型。

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。

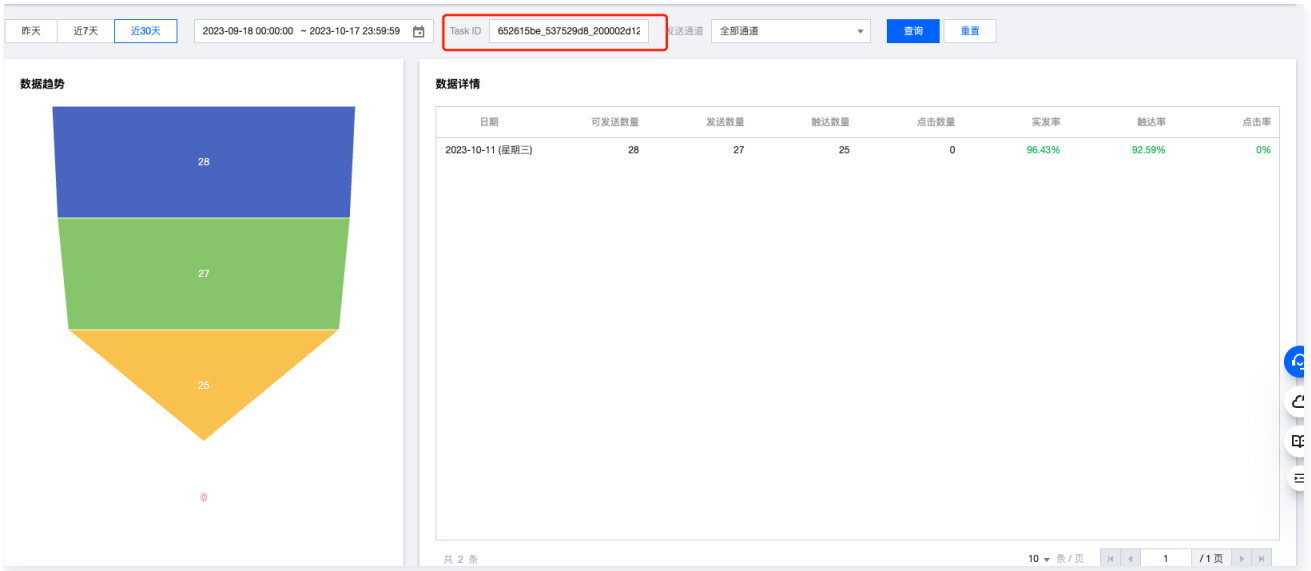
支持按照厂商维度分类查看

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	支持（需集成 TIMPush）
Apns	支持（需集成 TIMPush）	支持（需集成 TIMPush）

排查工具

最近更新时间：2024-10-10 17:39:31

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

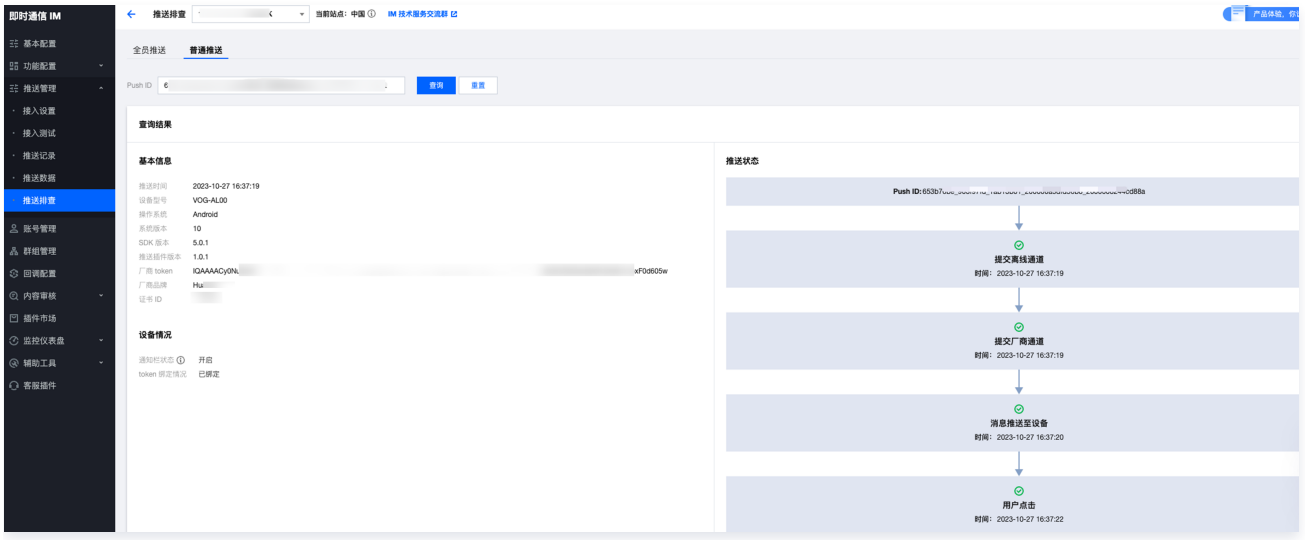
普通推送排查工具主要面向开发用户用来排查具体一条推送收不到问题。通过在推送记录中找到该条推送，复制本条推送的唯一 Push ID，来查询该条推送的详细推送链路详情。

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

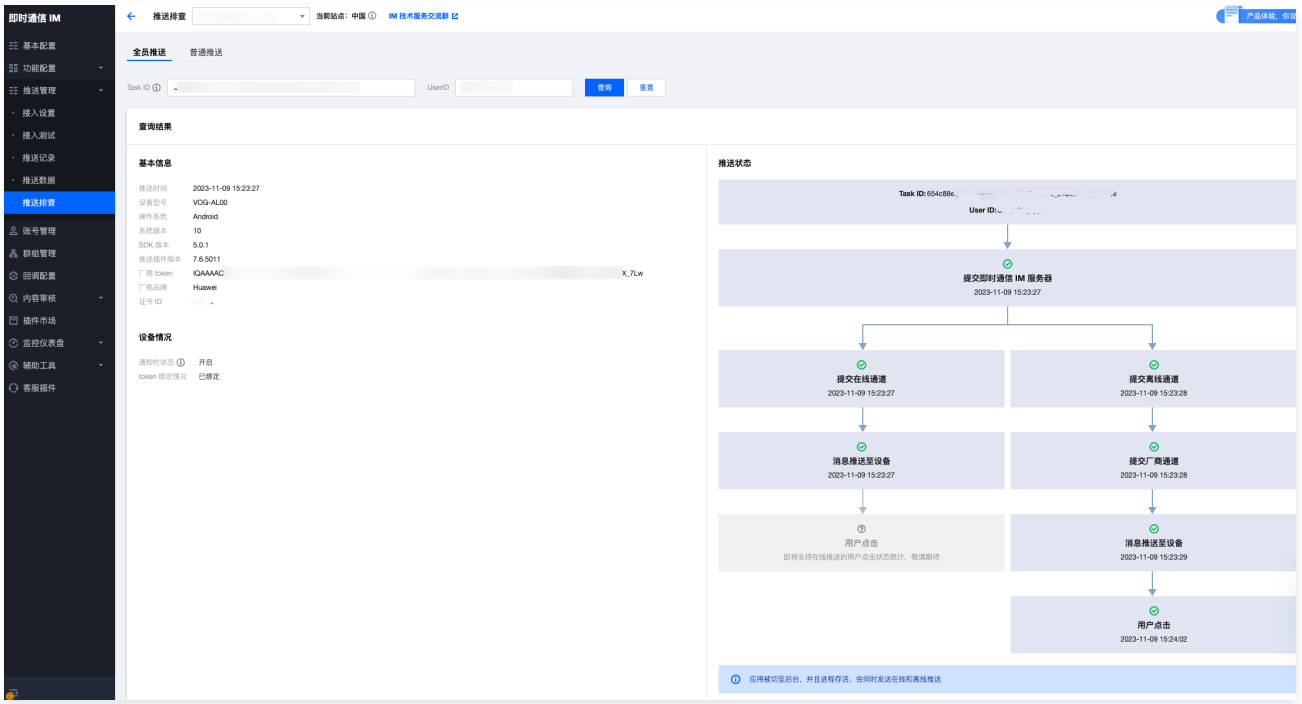
全员/标签推送排查工具主要面向管理员用户用来排查一次全员/标签推送的某个用户的接收情况。通过在推送记录中找到该条推送，复制本条推送的唯一任务 ID，输入接收方的 UserID，来查询该条推送到该 UserID 用户的详细推送链路详情。

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。



客户端 API
Android

最近更新时间：2024-12-02 12:11:12

TIMPush - 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

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（必须在 App 用户同意了隐私政策后，再调用该接口使用推送服务）。
unRegisterPush	反注册关闭推送服务。
setRegistrationID	RegistrationID 是推送接收设备的唯一标识 ID。默认情况下，注册推送服务成功时自动生成该 ID，同时也支持您自定义设置。您可根据 RegistrationID 向指定设备推送消息。需要注意的是，卸载并重新安装设备会更改 RegistrationID，因此需要在注册推送服务之前调用 setRegistrationID 接口。
getRegistrationID	在成功注册推送服务后，可通过调用 getRegistrationID 接口获取推送接收设备的唯一标识 ID，即 RegistrationID。您可根据 RegistrationID 向指定设备推送消息。

Push 全局监听接口

API	描述
addPushListener	添加 Push 监听器。
removePushListener	移除 Push 监听器。

自定义配置接口

API	描述
forceUseFCMPushChannel	指定设备离线推送使用 FCM 通道，需要在注册推送服务之前调用。
disablePostNotificationInForeground	关闭 App 在前台时弹出通知栏。

接口详情

静态 Public 成员函数

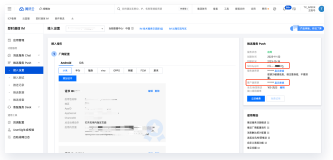
static TIMPushManager getInstance()：获取 TIMPushManager 管理器实例。

成员函数说明

abstract void registerPush(Context context, int sdkAppId, String appKey, TIMPushCallback callback)

注册推送服务，请正确传递 sdkAppId 和 appKey 两个参数，即可注册推送服务。
参数说明：

参数	描述	获取路径

sdkAppId	IM 控制台为您分配的应用 ID。	
appKey	IM 控制台为您分配的客户端密钥。	

abstract void unRegisterPush(TIMPushCallback callback)

反注册关闭推送服务。

abstract void setRegistrationID(String registrationID, TIMPushCallback callback)

设置注册推送服务使用的推送 ID 标识, 即 RegistrationID，需要在注册推送服务之前调用。
参数说明：

参数	描述
registrationID	设备的推送唯一标识 ID，卸载重装会改变。

abstract void getRegistrationID(TIMPushCallback callback)

注册推送服务成功后，获取推送 ID 标识，即 RegistrationID。

abstract void addPushListener(TIMPushListener listener)

添加 Push 监听器

abstract void removePushListener(TIMPushListener listener)

移除 Push 监听器

abstract void forceUseFCMPushChannel(boolean enable)

指定设备离线推送使用 FCM 通道，需要在注册推送服务之前调用。
参数说明：

参数	描述
enable	• true：使用 FCM 通道。 • false：使用本机通道。

abstract void disablePostNotificationInForeground(boolean disable)

关闭 App 在前台时弹出通知栏。推送 SDK 收到在线推送时，会自动向通知栏增加 Notification 提示，如果您想自己处理在线推送消息，可以调用该接口关闭自动弹通知栏提示的特性。
参数说明：

参数	描述
disable	• true：关闭 • false：开启

TIMPush – TIMPushListener

public abstract class TIMPushListener：Push 监听器类

接口概览

API	描述
onRecvPushMessage	收到 Push 消息。

onRevokePushMessage	收到 Push 消息撤回的通知。
onNotificationClicked	点击通知栏消息回调。

接口详情

成员函数说明

void onRecvPushMessage(TIMPushMessage message)

收到 Push 消息，message 消息。

void onRevokePushMessage(String messageID)

收到 Push 消息撤回的通知，messageID 消息唯一标识。

void onNotificationClicked(String ext)

点击通知栏消息回调。

> ⚠ **注意：**
> 控制台推送证书需要配置为"打开应用内指定界面"，并使用默认填充值生效。

iOS

最近更新时间：2024-12-02 18:25:42

TIMPush - TIMPushManager

@interface TIMPushManager : NSObject ： 推送插件接口类。

接口概览

注册/反注册推送服务接口

API	描述
registerPush	注册推送服务 (必须在 App 用户同意了隐私政策后，再调用该接口使用推送服务)。
unRegisterPush	反注册关闭推送服务。
setRegistrationID	RegistrationID 是推送接收设备的唯一标识 ID。默认情况下，注册推送服务成功时自动生成该 ID，同时也支持您自定义设置。您可根据 RegistrationID 向指定设备推送消息。需要注意的是，卸载并重新安装设备会更改 RegistrationID，因此需要在注册推送服务之前调用 setRegistrationID 接口。
getRegistrationID	在成功注册推送服务后，可通过调用 getRegistrationID 接口获取推送接收设备的唯一标识 ID，即 RegistrationID。您可根据 RegistrationID 向指定设备推送消息。

Push 全局监听接口

API	描述
addPushListener	添加 Push 监听器。
removePushListener	移除 Push 监听器。

自定义配置接口

设置应用前台是否展示推送

API	描述
disablePostNotificationInForeground	关闭 App 在前台时弹出通知栏。

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
handleNotificationServiceRequest:appGroupID: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。

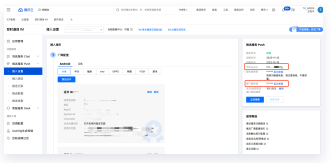
接口详情

函数说明

+ (void)registerPush:(int) sdkAppId appKey:(NSString *) appKey succ:(TIMPushSuccessCallback) successCallback fail:(TIMPushFailedCallback) failedCallback;

注册推送服务，请正确传递 sdkAppId 和 appKey 两个参数，即可注册推送服务。

- 参数说明：

参数	描述	获取路径
sdkAppId	IM 控制台为您分配的应用 ID。	
appKey	IM 控制台为您分配的客户端密钥。	

- 代码示例：

```
const int sdkAppId = 您的 sdkAppId;
static const NSString *appKey = @"客户端密钥";

[TIMPushManager registerPush:sdkAppId appKey:appKey succ:^(NSData * _Nonnull deviceToken) {
    //
 } fail:^(int code, NSString * _Nonnull desc) {
   //error
}];
```

+ (void)unRegisterPush:(TIMPushCallback) successCallback fail:(TIMPushFailedCallback) failedCallback;

反注册关闭推送服务

- 代码示例：

```
[TIMPushManager unRegisterPush:^{
     //success
   } fail:^(int code, NSString * _Nonnull desc) {
     //error
 }];
```

+ (void)setRegistrationID:(NSString *)registrationID callback: (TIMPushCallback) callback;

设置注册推送服务使用的推送 ID 标识, 即 RegistrationID，需要在注册推送服务之前调用。

- 参数说明：

参数	描述
registrationID	设备的推送标识 ID，卸载重装会改变。

+ (void)getRegistrationID:(TIMPushValueCallback) callback;

注册推送服务成功后，获取推送 ID 标识, 即 RegistrationID 。

+ (void)addPushListener:(id<TIMPushListener>)listener

添加 Push 监听器。

+ (void)removePushListener:(id<TIMPushListener>)listener

移除 Push 监听器。

+ (void)disablePostNotificationInForeground:(BOOL)disable;

关闭 App 在前台时弹出通知栏。推送 SDK 收到在线推送时，会自动向通知栏增加 Notification 提示，如果您想自己处理在线推送消息，可以调用该接口关闭自动弹通知栏提示的特性。

- 参数说明：

参数	描述
disable	• true：关闭 • false：开启

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

统计 TIMPush 的推送抵达率

1. 您需要在 AppDelegate.m 文件中实现 `- applicationGroupID` 方法，返回 App Group ID。
2. 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

- **示例代码：**

```
@implementation NotificationService

- (void)didReceiveNotificationRequest:(UNNotificationRequest *)request withContentHandler:(void
(^)(UNNotificationContent * _Nonnull))contentHandler {
    //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。
    //格式为group + [主bundleID]+ key
    //如group.com.tencent.im.pushkey
    NSString * appGroupID = kTIMPushAppGorupKey;
    __weak typeof(self) weakSelf = self;
    [TIMPushManager handleNotificationServiceRequest:request appGroupID:appGroupID
callback:^(UNNotificationContent *content) {
        weakSelf.bestAttemptContent = [content mutableCopy];
        // Modify the notification content here...
        // self.bestAttemptContent.title = [NSString stringWithFormat:@"%@ [modified]",
self.bestAttemptContent.title];
        weakSelf.contentHandler(weakSelf.bestAttemptContent);
    }];
}

@end
```

TIMPush - TIMPushListener

@protocol TIMPushListener <NSObject>： Push 监听器协议

接口概览

API	描述
onRecvPushMessage	收到 Push 消息
onRevokePushMessage	收到 Push 消息撤回的通知
onNotificationClicked	点击通知栏消息回调

接口详情

成员函数说明

- (void)onRecvPushMessage:(TIMPushMessage *)message;

收到 Push 消息，message 消息。

- (void)onRevokePushMessage:(NSString *)messageID;

收到 Push 消息撤回的通知，messageID 消息唯一标识。

- (void)onNotificationClicked:(NSString *)ext;

点击通知栏消息回调。

uni-app

最近更新时间：2024-11-14 14:59:32

接口概览

API	描述
`registerPush`	注册推送服务，（必须在 App 用户同意了隐私政策后，再调用该接口使用推送服务）。
`unRegisterPush`	反注册关闭推送服务。
`setRegistrationID`	设置推送设备标识 ID。 • RegistrationID 是推送接收设备的唯一标识 ID。默认情况下，注册推送服务成功时自动生成该 ID，同时也支持您自定义设置。**请注意！如果您调用此接口，请务必在 `registerPush` 前调用。** • 适用场景举例： 若您同时集成了消息服务（Chat）和推送服务（Push），在某个设备上，用户登录 Chat 的 userID 假设为 "user123"，如果您想指定向 "user123" 推送消息，则需要调用此接口设置设备标识 ID，如下： `Push.setRegistrationID("user123", () => {});`
`getRegistrationID`	在成功注册推送服务后，调用此接口可获取推送接收设备的唯一标识 ID，即 RegistrationID。
`getNotificationExtInfo`	收到离线推送时，点击通知栏拉起 App，调用此接口可获取推送扩展信息。
`addPushListener`	添加 Push 监听器。
`removePushListener`	移除 Push 监听器。
`disablePostNotificationInForeground`	应用在前台时，开/关通知栏通知（默认开）。
`createNotificationChannel`	创建客户端通知 channel。此接口可在 Android 平台上实现自定义铃音功能。

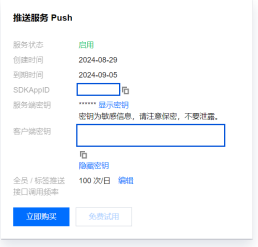
接口详情

注册推送服务

接口

```
registerPush(SDKAppID, appKey, onSuccess, onError)
```

参数说明：

参数	类型	说明	获取路径
SDKAppID	Number	推送服务 Push 的 SDKAppID。	IM 控制台 > 推送服务 Push > 接入设置页面 
appKey	String	推送服务 Push 的客户端密钥。	

onSuccess	Function	注册推送成功的回调。	-
onError	Function | undefined	注册推送失败的回调。	-

反注册关闭推送服务

接口

```
unRegisterPush(onSuccess, onError)
```

参数说明：

参数	类型	说明
onSuccess	Function	反注册推送成功的回调。
onError	Function | undefined	反注册推送失败的回调。

设置推送 ID 标识 RegistrationID

> **说明：**
> 1. 如果您调用此接口，请务必在 `registerPush` 前调用。

接口

```
setRegistrationID(registrationID, onSuccess)
```

参数说明：

参数	类型	说明
registrationID	String	自定义的推送 ID 标识。
onSuccess	Function	设置自定义推送 ID 标识成功后的回调。

获取推送 ID 标识 RegistrationID

> **说明：**
> 1. 若您调用过 `setRegistrationID` 接口设置标识 ID，此接口将返回您设置的标识 ID，否则返回由 Push SDK 生成的随机值。

接口

```
getRegistrationID(onSuccess)
```

参数说明：

参数	类型	说明
onSuccess	Function	获取推送 ID 标识成功的回调。

获取推送扩展信息

接口

```
getNotificationExtInfo(onSuccess)
```

参数说明：

参数	类型	说明
onSuccess	Function	获取点击透传的内容成功的回调。

添加 Push 监听器

接口

```
addPushListener(eventName: string, listener: (data: any) => void);
```

参数说明：

参数	类型	说明
eventName	String	推送事件类型。
listener	Function	推送事件处理方法。

移除 Push 监听器

接口

```
removePushListener(eventName: string, listener?: (data: any) => void);
```

参数说明：

参数	类型	说明
eventName	String	推送事件类型。
listener	Function | undefined	推送事件处理方法。

应用在前台时，开/关通知栏通知

接口

```
disablePostNotificationInForeground(disable: boolean);
```

参数说明：

参数	类型	说明
disable	boolean	应用在前台时，开/关通知栏通知，默认开： • true: 应用在前台时，关闭通知栏通知。 • false: 应用在前台时，开启通知栏通知。

创建客户端通知 channel

接口

```
createNotificationChannel(options: any, listener: (data: any) => void);
```

参数说明：

参数	类型	说明
options.channelID	String	自定义 channel 的 ID。 OPPO：控制台-> 接入配置 中 channelID。
options.channelName	String	自定义 channel 的名称。
options.channelDesc	String | undefined	自定义 channel 的描述。
options.channelSound	String | undefined	自定义 channel 的铃音，音频文件名，不带后缀，音频文件需要放到 `xxx/nativeResources/android/res/raw` 中。 例如： `options.channelSound = private_ring`，即设置 `xxx/nativeResources/android/res/raw/private_ring.mp3` 为自定义铃音
listener	Function	接口调用成功的回调函数。

微信小程序多端框架

最近更新时间：2025-06-12 11:53:32

接口概览

API	描述
setRegistrationID	设置注册推送服务使用的推送 ID 标识，即 RegistrationID，需要在注册推送服务之前调用。
getRegistrationID	注册推送服务成功后，获取推送 ID 标识，即 RegistrationID。
registerPush	注册推送服务 (必须在 App 用户同意了隐私政策，并且确定为 App 用户开始提供推送服务后，再调用该接口使用推送服务)。
unRegisterPush	反注册关闭推送服务。
addPushListener	订阅推送事件，如点击通知栏事件。
removePushListener	取消订阅推送事件。
disablePostNotificationInForeground	应用在前台时，开/关通知栏通知（默认开）。
createNotificationChannel	创建客户端通知 channel。此接口可实现自定义铃音功能。
setCustomBadgeNumber	设置应用角标数字（仅华为、iOS）

接口详情

注册推送服务

> **注意：**
> 必须在 App 用户同意了隐私政策，并且确定为 App 用户开始提供推送服务后，再调用该接口使用推送服务。否则可能因提前获取用户隐私导致上架失败。

接口

```
registerPush(sdkAppID: number, appKey: string): Promise<JSON>;
```

参数说明

参数	类型	说明	获取路径
sdkAppID	number	推送（Push）应用 ID	IM 控制台 > 推送服务 Push > 接入设置页面 
appKey	string	推送（Push）应用客户端密钥	

返回值说明

返回值	说明
Promise	成功返回 {"errCode":0,"errMsg":"success","data":{"token":"xxx"}}，失败返回 { errCode: number, errMsg: string }

反注册关闭推送服务。

接口

```
unRegisterPush(): Promise<JSON>;
```

返回值说明

返回值	说明
Promise	成功返回 {"errCode":0,"errMsg":"success"}，失败返回 { errCode: number, errMsg: string }

设置推送 ID 标识 RegistrationID

> **注意：**
> 需要在注册推送服务之前调用。

接口

```
setRegistrationID(registrationID: string): Promise<JSON>;
```

参数说明

参数	类型	必填	说明
registrationID	string	是	设备的推送标识 ID，卸载重装会改变

返回值说明

返回值	说明
Promise	成功返回 {"errCode":0,"errMsg":"success"}，失败返回 { errCode: number, errMsg: string }

获取推送 ID 标识 RegistrationID

> **说明：**
> 需要在注册推送服务成功之后调用。

接口

```
getRegistrationID(): Promise<JSON>;
```

返回值说明

返回值	说明
Promise	成功返回 {"errCode":0,"errMsg":"success","data":{"registrationID":"xxx"}}，失败返回 { errCode: number, errMsg: string }

订阅推送事件，如点击通知栏事件

接口

```
addPushListener(eventName: string, listener: (res: any) => void): void;
```

参数说明

参数	类型	必填	说明
eventName	string	是	推送事件类型，可查看 Push.EventName
listener	(res: any) => void	是	推送事件处理方法

取消订阅推送事件

接口

```
removePushListener(eventName: string, listener: (res: any) => void): void;
```

参数说明

参数	类型	必填	说明
eventName	string	是	推送事件类型，可查看 Push.EventName
listener	(res: any) => void	是	推送事件处理方法

应用在前台时，开/关通知栏通知

接口

```
disablePostNotificationInForeground(disable: boolean): void;
```

参数说明

参数	类型	必填	说明
disable	boolean	是	应用在前台时，开/关通知栏通知，默认开： • true: 应用在前台时，关闭通知栏通知。 • false: 应用在前台时，开启通知栏通知。

创建客户端通知 channel

> **说明：**
> 如果不同厂商的 ChannelID 各不相同，那么每个 Channel 都需要创建。

接口

```
createNotificationChannel(options: {
        channelID: string;
        channelName: string;
        channelDesc?: string;
        channelSound?: string;
    }): Promise<JSON>;
```

参数说明

参数	类型	必填	说明
options.channelID	String	是	自定义 channel 的 ID。
options.channelName	String	是	自定义 channel 的名称。
options.channelDesc	String	否	自定义 channel 的描述。
options.channelSound	String	否	自定义 channel 铃音的音频文件名，不带文件扩展名，音频文件需要放到 xxx/android/nativeResources/res/raw 中。 例如：options.channelSound = private_ring，即设置 xxx/android/nativeResources/res/raw/private_ring.mp3 为自定义铃音。

返回值说明

返回值	说明
Promise	成功返回 {"errCode":0,"errMsg":"success"}，失败返回 { errCode: number, errMsg: string }

设置应用角标数字

> **注意：**
> 1. 需要在注册推送服务之后调用。
> 2. 仅支持华为、iOS。

接口

```
setCustomBadgeNumber(badgeNumber: number): Promise<JSON>;
```

参数说明

参数	类型	必填	说明
badgeNumber	number	是	应用角标数字

返回值说明

返回值	说明
Promise	成功返回 {"errCode":0,"errMsg":"success"}，失败返回 { errCode: number, errMsg: string }

Flutter

最近更新时间：2024-12-23 16:51:43

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

API	描述
registerPush	注册推送服务，可选覆盖推送信息来自接口参数 json。
unRegisterPush	反注册推送服务。
setRegistrationID	RegistrationID 是推送接收设备的唯一标识 ID。默认情况下，注册推送服务成功时自动生成该 ID，同时也支持您自定义设置。您可根据 RegistrationID 向指定设备推送消息。需要注意的是，卸载并重新安装设备会更改 RegistrationID，因此需要在注册推送服务之前调用 setRegistrationID 接口。
getRegistrationID	在成功注册推送服务后，可通过调用 getRegistrationID 接口获取推送接收设备的唯一标识 ID，即RegistrationID。您可根据 RegistrationID 向指定设备推送消息。

Push 全局监听接口

API	描述
addPushListener	添加 Push 监听器。
removePushListener	移除 Push 监听器。

自定义配置接口接口

API	描述
forceUseFCMPushChannel	指定设备离线推送使用 FCM 通道，需要在注册推送服务之前调用。
disablePostNotificationInForeground	关闭 App 在前台时弹出通知栏。

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

registerPush

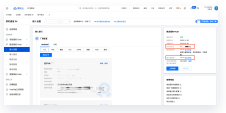
注册推送服务，IM 账号登录成功后调用。
示例代码：

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

```
    onNotificationClicked: _onNotificationClicked,
    sdkAppId: 您的sdkAppId,
    appKey: "客户端密钥",
    apnsCertificateID: 您配置的证书 ID);
```

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。
sdkAppId		int?	IM 控制台为您分配的应用 ID
appKey		String?	IM 控制台为您分配的客户端密钥
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。
示例代码：

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

setRegistrationID

设置注册离线推送服务使用的推送 ID 标识，即 RegistrationID，需要在注册推送服务之前调用。
参数说明：

参数	描述
registrationID	设备的推送标识 ID，卸载重装会改变。

示例代码：

```
TencentCloudChatPush().setRegistrationID(registrationID: registrationID);
```

getRegistrationID

注册离线推送服务成功后，获取推送 ID 标识，即 RegistrationID。
示例代码：

```
TencentCloudChatPush().getRegistrationID();
```

addPushListener

添加 Push 监听器
示例代码：

```
TIMPushListener timPushListener = TIMPushListener(
  onRecvPushMessage: (TimPushMessage message) {
      String messageLog = message.toLogString();
      debugPrint(
          "message: $messageLog");
      },

  onRevokePushMessage: (String messageId) {
    debugPrint(
        "message: $messageId");
  },

  onNotificationClicked: (String ext) {
    debugPrint(
        "ext: $ext");
  }
);
TencentCloudChatPush.addPushListener(listener: timPushListener);
```

removePushListener

移除 Push 监听器
示例代码：

```
TIMPushListener timPushListener = TIMPushListener(
  onRecvPushMessage: (TimPushMessage message) {
      String messageLog = message.toLogString();
      debugPrint(
          "message: $messageLog");
      },

  onRevokePushMessage: (String messageId) {
    debugPrint(
        "message: $messageId");
  },

  onNotificationClicked: (String ext) {
    debugPrint(
        "ext: $ext");
  }
);
TencentCloudChatPush.removePushListener(listener: timPushListener);
```

forceUseFCMPushChannel

指定设备离线推送使用 FCM 通道，需要在注册推送服务之前调用。

参数说明：

参数	描述
enable	• true：使用 FCM 通道。 • false：使用本机通道。

示例代码：

```
TencentCloudChatPush.forceUseFCMPushChannel(enable: true);
```

disablePostNotificationInForeground

关闭 App 在前台时弹出通知栏。推送 SDK 收到在线推送时，会自动向通知栏增加 Notification 提示，如果您想自己处理在线推送消息，可以调用该接口关闭自动弹通知栏提示的特性。
参数说明：

参数	描述
disable	• true：关闭 • false：开启

示例代码：

```
TencentCloudChatPush.disablePostNotificationInForeground(disable: true);
```

React Native

最近更新时间：2024-12-23 17:19:52

接口概览

API	描述
`registerPush`	注册推送服务，（必须在 App 用户同意了隐私政策后，再调用该接口使用推送服务）。
`unRegisterPush`	反注册关闭推送服务。
`setRegistrationID`	设置推送设备标识 ID。 • RegistrationID 是推送接收设备的唯一标识 ID。默认情况下，注册推送服务成功时自动生成该 ID，同时也支持您自定义设置。**请注意！如您调用此接口，请务必在 `registerPush` 前调用。** • 适用场景举例： 若您同时集成了消息服务（Chat）和推送服务（Push），在某个设备上，用户登录 Chat 的 userID 假设为 "user123"，如果您想指定向 "user123" 推送消息，则需要调用此接口设置设备标识 ID，如下： `Push.setRegistrationID("user123", () => {});`
`getRegistrationID`	在成功注册推送服务后，调用此接口可获取推送接收设备的唯一标识 ID，即 RegistrationID。
`getNotificationExtInfo`	收到离线推送时，点击通知栏拉起 App，调用此接口可获取推送扩展信息。
`addPushListener`	添加 Push 监听器。
`removePushListener`	移除 Push 监听器。
`disablePostNotificationInForeground`	应用在前台时，开/关通知栏通知（默认开）。
`createNotificationChannel`	创建客户端通知 channel。此接口可在 Android 平台上实现自定义铃音功能。

接口详情

注册推送服务

接口

```
registerPush(SDKAppID: number, appKey: string, onSuccess: (data: string) => void, onError?:
(errCode: number, errMsg: string) => void);
```

参数说明：

参数	类型	说明	获取路径
SDKAppID	Number	推送服务 Push 的 SDKAppID。	IM 控制台 > 推送服务 Push > 接入设置页面
appKey	String	推送服务 Push 的客户端密钥。	

onSuccess	Function	注册推送成功的回调。	
onError	Function | undefined	注册推送失败的回调。	

反注册关闭推送服务

接口

```
unRegisterPush(onSuccess: () => void, onError?: (errCode: number, errMsg: string) => void): void;
```

参数说明：

参数	类型	说明
onSuccess	Function	反注册推送成功的回调。
onError	Function | undefined	反注册推送失败的回调。

设置推送 ID 标识 RegistrationID

> **说明：**
> 需要在注册推送服务之前调用。

接口

```
setRegistrationID(registrationID: string,  onSuccess: () => void): void;
```

参数说明：

参数	类型	说明
registrationID	String	自定义的推送 ID 标识。
onSuccess	Function	设置自定义推送 ID 标识成功后的回调。

获取推送 ID 标识 RegistrationID

> **说明：**
> - 需要在注册推送服务成功之后调用。
> - 若您调用过 `setRegistrationID` 接口设置标识 ID，此接口将返回您设置的标识 ID，否则返回由 Push SDK 生成的随机值。

接口

```
getRegistrationID(onSuccess: (registrationID: string) => void): void;
```

参数说明：

参数	类型	说明
onSuccess	Function	获取推送 ID 标识成功的回调。

获取点击透传的内容

> **说明：**
> 收到离线推送时，点击通知栏拉起 App，调用此接口可获取推送扩展信息。

接口

```
getNotificationExtInfo(onSuccess: (extInfo: string) => void): void;
```

参数说明：

参数	类型	说明
onSuccess	Function	获取点击透传的内容成功的回调。

添加 Push 监听器。

接口

```
addPushListener(eventName: string, listener: (data: any) => void);
```

参数说明：

参数	类型	说明
eventName	String	推送事件类型。
listener	Function	推送事件处理方法。

移除 Push 监听器。

接口

```
removePushListener(eventName: string, listener?: (data: any) => void);
```

参数说明：

参数	类型	说明
eventName	String	推送事件类型。
listener	Function | undefined	推送事件处理方法。

应用在前台时，开/关通知栏通知。

接口

```
disablePostNotificationInForeground(disable: boolean);
```

参数说明：

参数	类型	说明
disable	boolean	应用在前台时，开/关通知栏通知，默认开： • true: 应用在前台时，关闭通知栏通知。 • false: 应用在前台时，开启通知栏通知。

创建客户端通知 channel。

接口

```
createNotificationChannel(options: any, onSuccess: (data: any) => void);
```

参数说明：

参数	类型	说明
options.channelID	String	自定义 channel 的 ID。 • OPPO：控制台-> 接入配置 中 channelID。
options.channelName	String	自定义 channel 的名称。
options.channelDesc	String | undefined	自定义 channel 的描述。
options.channelSound	String | undefined	自定义 channel 的铃音，音频文件名，不带后缀，音频文件需要放到 `MyReactNativeApp/android/app/src/main/res/raw` 中。 例如： `options.channelSound = private_ring`，即设置 `MyReactNativeApp/android/app/src/main/res/raw/private_ring.mp3` 为自定义铃音。
onSuccess	Function	接口调用成功的回调函数。

服务端 API
发起全员/标签推送

最近更新时间：2025-05-19 14:41:12

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 。

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。
- 支持在线通道，厂商通道（APNS、华为、荣耀、OPPO、vivo、小米、魅族、Google）。
- 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）。

> ⚠ **注意：**
> 本接口支持在线推送和离线推送，在线推送仅 SDK 版本 >= 8.2.6325支持。

接口调用说明

请求 URL 示例

```
https://xxxxxx/v4/timpush/push?
usersig=xxx&identifier=admin&sdkappid=88888888&random=99999999&contenttype=json
```

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` • 硅谷：`adminapiusa.im.qcloud.com` • 雅加达：`adminapiidn.im.qcloud.com`
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，更多详情请参见 App 管理员 。
sdkappid	创建应用时，即时通信控制台分配的 SdkAppid。
random	请输入随机的32位无符号整数，取值范围0 - 4294967295。
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20元/日。如需调整接口频率，请前往 IM 控制台 操作。

请求包示例

> ⚠ **注意：**
> 本接口支持在线推送和离线推送，在线推送仅 SDK 版本 >= 8.2.6325支持。

全员推送示例

- 管理员进行全员推送：

```
{
  "From_Account": "administrator",
  "MsgRandom": 3674128,
  "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"
  }
}
```

- 只推送在线用户（在线推送）：

```
{
  "From_Account": "administrator",
  "MsgRandom": 3674128,
  "OfflinePushInfo": {
      "PushFlag": 1,  // 0表示进行离线推送，1表示不进行离线推送
      "Ext": "{\"entity\":{\"key1\":\"value1\",\"key2\":\"value2\"}}"  // 透传字段，推送使用 JSON 格式字符串
  }
}
```

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": "administrator",
  "MsgRandom": 124032,
  "Condition": {
      "TagsAnd": ["股票A","股票B"]
  },
  "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"
  }
}
```

- 管理员给关注“股票A”或“股票B”的用户推送消息：

```
{
  "From_Account": "administrator",
  "MsgRandom": 124032,
  "Condition": {
      "TagsOr": ["股票A","股票B"]
  },
  "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": "administrator",
  "MsgRandom": 389475,
  "Condition": {
      "AttrsAnd": {
          "会员等级": "超白金会员",
          "city": "深圳"
      }
  },
  "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"
  }
}
```

- 管理员给在深圳的超白金用户推送消息：

```
{
  "From_Account": "administrator",
  "MsgRandom": 389475,
  "Condition": {
      "AttrsAnd": {
          "会员等级": "超白金用户",
          "city": "深圳"
      }
  },
  "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） **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	32位无符号整数随机数，取值范围0 - 4294967295 后台用于同一秒内的消息去重，**请确保是随机数。**
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 **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。 **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。 **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。**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。**属性推送和标签推送不可同时作为推送条件。**
OfflinePushInfo	Object	选填	推送信息配置，具体请参见 离线推送 OfflinePushInfo 说明 。

应答包体示例

```
{
    "ActionStatus": "OK",
    "ErrorInfo": "",
    "ErrorCode": 0,
    "TaskId": "667015d4_537529d8_2000005e80aa873_d03ac87_56f5e750"
}
```

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。
90005	JSON 格式请求包体中缺少 MsgRandom 字段或者 MsgRandom 字段不是 Integer 类型。
90009	请求需要 App 管理员权限。
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。
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）。
90056	全员推送的请求体过大，目前支持最大10K长度。
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

单发推送

最近更新时间：2025-06-30 21:24:41

功能说明

- 给定接收方账号列表进行推送，接收方账号列表在 [1, 500] 个之间。
- 支持在线通道，厂商通道（APNS、华为、荣耀、OPPO、vivo、小米、魅族、Google）。

> **注意：**
> 本接口支持在线推送和离线推送，在线推送仅 SDK 版本 >= 8.2.6325支持。

接口调用说明

请求 URL 示例

```
https://xxxxxx/v4/timpush/batch?
usersig=xxx&identifier=admin&sdkappid=88888888&random=99999999&contenttype=json
```

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` • 硅谷：`adminapiusa.im.qcloud.com` • 雅加达：`adminapiidn.im.qcloud.com`
v4/timpush/batch	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，更多详情请参见 App 管理员 。
sdkappid	创建应用时，即时通信控制台分配的 SdkAppid。
random	请输入随机的32位无符号整数，取值范围0 - 4294967295。
contenttype	固定值为：`json` 。

调用频率

30次/s

请求包示例

```
{
  "From_Account": "administrator",
  "To_Account": ["user1","user2"], // 数组大小范围在[1,500]之间
```

```
    "MsgRandom": 3674128,
    "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：**
> 本接口支持在线推送和离线推送，在线推送仅 SDK 版本 >= 8.2.6325支持。

请求包字段说明

字段	类型	属性	说明
From_Account	String	必填	发送方账号，支持 UserID 或 RegistrationID
To_Account	Array	必填	接收方账号列表，支持 UserID 或 RegistrationID
MsgRandom	Integer	必填	32位无符号整数随机数，取值范围0 – 4294967295 后台用于同一秒内的消息去重，**请确保是随机数**。
OfflinePushInfo	Object	必填	推送信息配置，具体请参见 离线推送 OfflinePushInfo 说明 。

应答包体示例

```
{
    "ActionStatus": "OK",
    "ErrorInfo": "",
    "ErrorCode": 0,
    "TaskId": "batch_667015d4_537529d8_2000005e80aa873_d03ac87_56f5e750"
}
```

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。
90009	请求需要 App 管理员权限。
90045	未开通全员/标签/单推推送功能。
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

推送撤回

最近更新时间：2025-05-19 14:41:12

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&random=99999999&contenttype=json
```

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` • 硅谷：`adminapiusa.im.qcloud.com` • 雅加达：`adminapiidn.im.qcloud.com`
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，更多详情请参见 App 管理员 。
sdkappid	创建应用时，即时通信控制台分配的 SdkAppid。
random	请输入随机的32位无符号整数，取值范围0 - 4294967295。
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小时内发送的推送taskId
```

```
    "OfflinePushInfo": {
        "Title": "撤回标题",
        "Desc": "对方撤回了一条消息",
        "Ext": "{\"entity\":{\"key1\":\"value1\",\"key2\":\"value2\"}}" // 透传字段，推送使用 JSON 格式
字符串
    }
}
```

⚠ 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	必填	离线推送信息配置，具体请参见 离线推送 OfflinePushInfo 说明 。

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 发起全员/标签推送 或 单发推送 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

- 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：2025-05-19 14:41:12

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=88888888&random=99999999&contenttype=json
```

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` • 硅谷：`adminapiusa.im.qcloud.com` • 雅加达：`adminapiidn.im.qcloud.com`
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，更多详情请参见 App 管理员 。
sdkappid	创建应用时即时通信控制台分配的 SdkAppid。
random	请输入随机的32位无符号整数，取值范围0 - 4294967295。
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": "country"
    },
    "AttrTypes": {
        "0": 0,
        "1": 0,
        "2": 1 // 数字键2对应属性country，改属性只能被类型为1账号设置（注册推送服务成功时自动生成该RegistrationID）
    }
}
```

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。
AttrTypes	Object	包含多个键对。每对键值对，表示第几个属性对应的账号类型。 • "0": 0表示第0号属性的账号类型是0（IM 账号类型）。 • "0": 1表示第0号属性的账号类型是1（注册推送服务成功时自动生成的 RegistrationID）。

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：2025-05-19 14:41:12

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=88888888&random=99999999&contenttype=json
```

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` • 硅谷：`adminapiusa.im.qcloud.com` • 雅加达：`adminapiidn.im.qcloud.com`
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，更多详情请参见 App 管理员 。
sdkappid	创建应用时即时通信控制台分配的 SdkAppid。
random	请输入随机的32位无符号整数，取值范围0 - 4294967295。
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"
    },
    "AttrTypes": {
        "0": 0, // 设置属性对应的账号类型，设置之后不支持修改。
        "1": 0,
```

```
        "2": 0
    }
}
```

请求包字段说明

AttrNames Object 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。

AttrTypes Object 说明：

字段	类型	属性	说明
数字键	String	选填	表示第几个属性（“0”到“9”之间），必需是 AttrNames 中包含的数字键。
属性名	Interger	选填	相应属性对应的账号类型，默认为 0。一个属性只能对应一种账号类型，详细参见 推送典型场景介绍 。 • 0 账号类型代表：注册推送服务成功时自动生成的 RegistrationID。 • 1 账号类型代表：用户登录 IM 账号传递 UserID。

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

获取用户属性

最近更新时间：2025-05-19 14:41:12

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&random=99999999&contenttype=json
```

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` • 硅谷：`adminapiusa.im.qcloud.com` • 雅加达：`adminapiidn.im.qcloud.com`
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，更多详情请参见 App 管理员 。
sdkappid	创建应用时即时通信控制台分配的 SdkAppid。
random	请输入随机的32位无符号整数，取值范围0 - 4294967295。
contenttype	固定值为：json。

调用频率限制

每秒100次。

请求包示例

```
{
    "To_Account": [
        "Mary",
        "xiaoming",
        "xiaohua"
    ]
}
```

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": "xiaoming",
            "Attrs": {
                "sex": "male",
                "city": "ShenZhen"
            }
        },
        {
            "To_Account": "xiaohua",
            "Attrs": {}
        }
    ],
     "ErrorList": [
        {
            "ErrorCode": 70107, // 账号不存在
            "To_Account": "Mary"
        }
    ]
}
```

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	Object Array	用户标签内容列表。
ErrorList	Object Array	设置属性有可能部分用户成功，部分失败。失败账号的错误信息列表。

UserAttrs 数组中 json Object 字段说明

字段	类型	说明
To_Account	String	用户账号。
Attrs	Object	属性内容，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。

ErrorList 数组中 json Object 字段说明

字段	类型	说明
To_Account	String	目标用户账号。
ErrorCode	Integer	错误码。
ErrorInfo	String	错误描述，有可能为空。

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：2025-05-19 14:41:12

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&random=99999999&contenttype=json
```

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` • 硅谷：`adminapiusa.im.qcloud.com` • 雅加达：`adminapiidn.im.qcloud.com`
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，更多详情请参见 App 管理员 。
sdkappid	创建应用时即时通信控制台分配的 SdkAppid。
random	请输入随机的32位无符号整数，取值范围0 - 4294967295
contenttype	固定值为：`json` 。

调用频率限制

每秒100次。

请求包示例

```
{
    "UserAttrs":
    [
        {
            "To_Account": "379C2F0D-290D-47AE-94D1-919058C39C77", // 注册推送服务成功时自动生成
RegistrationID
            "Attrs": {
                "sex": "female",
                "city": "NewYork"
            }
```

```
        },
        {
            "To_Account": "xiaoming",
            "Attrs": {
                "sex": "male",
                "city": "ShenZhen"
            }
        }
    ]
}
```

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

全部成功：

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

部分成功：

```
{
    "ActionStatus": "OK",
    "ErrorInfo": "",
    "ErrorCode": 0,
    "ErrorList": [
        {
            "ErrorCode": 90035, // 一种属性只能适用于一种账号类型
            "To_Account": "379C2F0D-290D-47AE-94D1-919058C39C77"
        }
    ]
}
```

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。
ErrorList	Object Array	设置属性有可能部分用户成功，部分失败。失败账号的错误信息列表。

ErrorList 数组中 json Object 字段说明

字段	类型	说明
To_Account	String	目标用户账号。
ErrorCode	Integer	错误码。
ErrorInfo	String	错误描述，有可能为空。

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。通常是属性不存在，或者属性对应的账号类型不匹配。请通过 获取应用属性名称 查看属性详情。
91000	服务内部错误，请重试。
90035	一种属性只能适用于一种账号类型。注册推送服务成功时自动生成 RegistrationID 与 IM 账号类型详细参见 推送典型场景介绍 。

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：2025-05-19 14:41:12

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=88888888&random=99999999&contenttype=json
```

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` • 硅谷：`adminapiusa.im.qcloud.com` • 雅加达：`adminapiidn.im.qcloud.com`
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，更多详情请参见 App 管理员 。
sdkappid	创建应用时即时通信控制台分配的 SdkAppid。
random	请输入随机的32位无符号整数，取值范围0 - 4294967295。
contenttype	固定值为：`json` 。

调用频率限制

每秒100次。

请求包示例

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

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

请求包字段说明

字段	类型	属性	说明
UserAttrs	Array	必填	用户属性的数组，单个用户属性由 To_Account 和 Attrs 组成。
To_Account	String	必填	目标用户账号。
Attrs	Array	必填	属性名称集合，注意这里只需要给出属性名即可，不是键值对。

应答包体示例

全部成功：

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

部分成功：

```
{
    "ActionStatus": "OK",
    "ErrorInfo": "",
    "ErrorCode": 0,
    "ErrorList": [
        {
            "ErrorCode": 70107,
            "To_Account": "Mary"
        }
    ]
}
```

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。
ErrorList	Object Array	设置属性有可能部分用户成功，部分失败。失败账号的错误信息列表。

ErrorList 数组中 json Object 字段说明

字段	类型	说明
To_Account	String	目标用户账号。
ErrorCode	Integer	错误码。
ErrorInfo	String	错误描述，有可能为空。

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：2025-05-19 14:41:12

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&random=99999999&contenttype=json
```

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` • 硅谷：`adminapiusa.im.qcloud.com` • 雅加达：`adminapiidn.im.qcloud.com`
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，更多详情请参见 App 管理员 。
sdkappid	创建应用时即时通信控制台分配的 SdkAppid。
random	请输入随机的32位无符号整数，取值范围0 - 4294967295。
contenttype	固定值为：`json` 。

调用频率限制

每秒100次。

请求包示例

```
{
    "To_Account": [
        "xiaoming",
        "xiaohong",
        "Mary"
    ]
}
```

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": "xiaoming",
            "Tags": ["黄金会员", "周卡用户"]
        },
        {
            "To_Account": "xiaohong",
            "Tags": ["白金会员", "月卡用户"]
        }
    ],
    "ErrorList": [
        {
            "ErrorCode": 70107,
            "To_Account": "Mary"
        }
    ]
}
```

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	Object Array	用户标签内容列表。
ErrorList	Object Array	设置属性有可能部分用户成功，部分失败。失败账号的错误信息列表。

UserTags 数组中 json Object 字段说明

字段	类型	说明
To_Account	String	用户账号。
Tags	Array	标签数组，单个标签最大长度不超过50字节，每一个标签都是一个 string 类型的字符串。

ErrorList 数组中 json Object 字段说明

字段	类型	说明
To_Account	String	目标用户账号。

ErrorCode	Integer	错误码。
ErrorInfo	String	错误描述，有可能为空。

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：2025-05-19 14:41:12

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&random=99999999&contenttype=json
```

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` • 硅谷：`adminapiusa.im.qcloud.com` • 雅加达：`adminapiidn.im.qcloud.com`
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，更多详情请参见 App 管理员 。
sdkappid	创建应用时即时通信控制台分配的 SdkAppid。
random	请输入随机的32位无符号整数，取值范围0 - 4294967295。
contenttype	固定值为：`json` 。

调用频率限制

每秒100次。

请求包示例

```
{
    "UserTags": [
```

```
        {
            "To_Account": "xiaoming",
             "Tags": ["黄金会员", "周卡用户"]
        },
        {
            "To_Account": "379C2F0D-290D-47AE-94D1-919058C39C77", // 注册推送服务成功时自动生成
RegistrationID
            "Tags": ["白金会员", "周卡用户"]
        }
    ]
}
```

请求包字段说明

字段	类型	属性	说明
UserTags	Object Array	必填	账号标签信息。

UserTags 数组中 json Object 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	标签数组，单个标签最大长度不超过50字节，每一个标签都是一个 String 类型的字符串。

应答包体示例

全部成功：

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

部分成功：

```
{
    "ActionStatus": "OK",
    "ErrorInfo": "",
    "ErrorCode": 0,
    "ErrorList": [
        {
            "ErrorCode": 90035, // 一个标签只能适用于一种账号类型
            "To_Account": "379C2F0D-290D-47AE-94D1-919058C39C77"
        }
    ]
}
```

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。
ErrorList	Object Array	设置属性有可能部分用户成功，部分失败。失败账号的错误信息列表。

ErrorList 数组中 json Object 字段说明

字段	类型	说明
To_Account	String	目标用户账号。
ErrorCode	Integer	错误码。
ErrorInfo	String	错误描述，有可能为空。

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。
90035	一个标签只能适用于一种账号类型。注册推送服务成功时自动生成 RegistrationID 与 IM 账号类型详细参见 推送典型场景介绍 。

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

删除用户标签

最近更新时间：2025-05-19 14:41:12

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&random=99999999&contenttype=json
```

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` • 硅谷：`adminapiusa.im.qcloud.com` • 雅加达：`adminapiidn.im.qcloud.com`
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，更多详情请参见 App 管理员 。
sdkappid	创建应用时即时通信控制台分配的 SdkAppid。
random	请输入随机的32位无符号整数，取值范围0 - 4294967295。
contenttype	固定值为：`json` 。

调用频率限制

每秒100次。

请求包示例

```
{
    "UserTags": [
        {
            "To_Account": "xiaoming",
            "Tags": ["黄金会员", "周卡用户"]
        },
        {
            "To_Account": "xiaohong",
            "Tags": ["白金会员", "月卡用户"]
        }
```

```
    ]
}
```

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	标签数组，单个标签最大长度不超过50字节，每一个标签都是一个 String 类型的字符串。

应答包体示例

全部成功：

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

部分成功：

```
{
    "ActionStatus": "OK",
    "ErrorInfo": "",
    "ErrorCode": 0,
    "ErrorList": [
        {
            "ErrorCode": 70107,
            "To_Account": "xiaohong"
        }
    ]
}
```

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。
ErrorList	Object Array	设置属性有可能部分用户成功，部分失败。失败账号的错误信息列表。

ErrorList 数组中 json Object 字段说明

字段	类型	说明
To_Account	String	目标用户账号。
ErrorCode	Integer	错误码。

ErrorInfo	String	错误描述，有可能为空。

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：2025-05-19 14:41:12

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=88888888&random=99999999&contenttype=json
```

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` • 硅谷：`adminapiusa.im.qcloud.com` • 雅加达：`adminapiidn.im.qcloud.com`
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，更多详情请参见 App 管理员 。
sdkappid	创建应用时即时通信控制台分配的 SdkAppid。
random	请输入随机的32位无符号整数，取值范围0 - 4294967295。
contenttype	固定值为：`json` 。

调用频率限制

每秒100次。

请求包示例

```
{
    "To_Account": [
        "xiaoming",
        "xiaohong"
    ]
}
```

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

全部成功：

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

部分成功：

```
{
    "ActionStatus": "OK",
    "ErrorInfo": "",
    "ErrorCode": 0,
    "ErrorList": [
        {
            "ErrorCode": 70107,
            "To_Account": "xiaohong"
        }
    ]
}
```

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。
ErrorList	Object Array	设置属性有可能部分用户成功，部分失败。失败账号的错误信息列表。

ErrorList 数组中 json Object 字段说明

字段	类型	说明
To_Account	String	目标用户账号。
ErrorCode	Integer	错误码。
ErrorInfo	String	错误描述，有可能为空。

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

- 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

推送回调
全员/标签/单推回调

最近更新时间：2024-10-18 15:41:01

功能说明

开启推送插件后，推送结果可以通过配置基础回调的方式，将结果转发给 App 后台。

注意事项

- 要启用回调，必须配置回调 URL，并打开本回调对应的开关，配置方法详见 第三方回调配置 文档。
- 回调的方向是即时通信 IM 后台向 App 后台发起 HTTP POST 请求。
- App 后台在收到回调请求之后，务必校验请求 URL 中的参数 SDKAppID 是否是自己的 SDKAppID。
- 其他安全相关事宜请参见 第三方回调简介：安全考虑 文档。

接口说明

请求 URL 示例

以下示例中 App 配置的回调 URL 为 `https://www.example.com`
示例：

```
https://www.example.com?SdkAppid=$SDKAppID&CallbackCommand=$CallbackCommand&contenttype=json
```

请求参数说明

字段	说明
https	• 请求协议为 HTTPS • 请求方式为 POST
www.example.com	回调 URL
SdkAppid	创建应用时在即时通信 IM 控制台分配的 SDKAppID
CallbackCommand	固定为：Push.AllMemberPush
contenttype	请求包体固定为 JSON

请求包示例

```
{
    "Events": [    // events数组长度范围为1～100
      {
        "CallbackCommand":"Push.AllMemberPush",
        "EventType": 1,                                                    // 事件类型,EventType=1表示离线
推送
        "TaskId": "657bf434_537529d8_2000005e80aa873_2780d131_bc614e", // 全员/标签/单推推送 TaskId
        "TaskTime": 1557481127,                                            // 全员推送任务发起时间戳，单位为
秒
        "EventTime": 1557481128,                                           // 事件发生时间戳，单位为秒
        "To_Account": "user2",                                             // 接受者
        "PushPlatform": 1,                                                 // 推送厂商
        "PushStage": 1,                                                    // 推送阶段
        "ErrCode": 0,                                                      // 推送事件结果
```

```
        "ErrInfo": "OK"                                                    // 推送事件结果描述，可能为空
      },
      {
        "CallbackCommand":"Push.AllMemberPush",
        "EventType": 2,                                                    // 事件类型,EventType=2表示在线
推送
        "TaskId": "657bf434_537529d8_2000005e80aa873_2780d131_9",          // 全员/标签/单推推送 TaskId
        "TaskTime": 1557481127,                                            // 全员推送任务发起时间戳，单位为
秒
        "EventTime": 1557481129,                                           // 事件发生时间戳，单位为秒
        "To_Account": "user3",                                             // 接受者
        "PushPlatform": 0,                                                 // 推送厂商
        "PushStage": 1,                                                    // 推送阶段
        "ErrCode": 0,                                                      // 推送事件结果
        "ErrInfo": "OK"                                                    // 推送事件结果描述，可能为空
      },
      ....
    ]
}
```

请求包字段说明

字段	类型	说明
Events	Array [Event Object]	批量回调内容，最多包含100个回调事件（Event Object）的数据

Event Object结构

字段	类型	说明
CallbackCommand	String	回调命令
EventType	Integer	事件类型： • EventType = 1表示离线推送 • EventType = 2表示在线推送
TaskId	String	全员推送发送时返回的任务 ID
TaskTime	Integer	全员推送任务发起时间戳，单位为秒
EventTime	Integer	事件发生时间戳，单位为秒
To_Account	String	消息接收者 UserID
PushPlatform	Integer	推送厂商（在线推送 EventType = 2不区分厂商，默认返回0）： • PushPlatform = 0表示未知厂商 • PushPlatform = 1表示 Apple APNS 推送 • PushPlatform = 2表示小米推送 • PushPlatform = 3表示华为推送 • PushPlatform = 4表示 Google FCM 推送 • PushPlatform = 5表示魅族推送 • PushPlatform = 6表示 OPPO 推送 • PushPlatform = 7表示 vivo 推送 • PushPlatform = 8表示荣耀推送
PushStage	Integer	推送阶段： • PushStage = 1表示推送发送

		• PushStage = 2表示推送触达 • PushStage = 3表示推送点击
ErrCode	Integer	推送事件结果： • ErrCode = 0表示成功 • ErrCode 非0表示失败
ErrInfo	String	推送事件结果描述，可能为空

应答包示例

```
{
    "ActionStatus": "OK",
    "ErrorInfo": "",
    "ErrorCode": 0 // 0为回调成功，1为回调出错
}
```

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	错误说明

其他推送回调

最近更新时间：2024-10-22 14:55:11

功能说明

开启推送插件后，推送结果可以通过配置基础回调的方式，将结果转发给 App 后台。

注意事项

- 要启用回调，必须配置回调 URL，并打开本回调对应的开关，配置方法详见 第三方回调配置 文档。
- 回调的方向是即时通信 IM 后台向 App 后台发起 HTTP POST 请求。
- App 后台在收到回调请求之后，务必校验请求 URL 中的参数 SDKAppID 是否是自己的 SDKAppID。
- 其他安全相关事宜请参见 第三方回调简介：安全考虑 文档。

接口说明

请求 URL 示例

以下示例中 App 配置的回调 URL 为 `https://www.example.com`
示例：

```
https://www.example.com?SdkAppid=$SDKAppID&CallbackCommand=$CallbackCommand&contenttype=json
```

请求参数说明

字段	说明
https	• 请求协议为 HTTPS • 请求方式为 POST
www.example.com	回调 URL
SdkAppid	创建应用时在即时通信 IM 控制台分配的 SDKAppID
CallbackCommand	固定为：Push.OfflinePush
contenttype	请求包体固定为 JSON

请求包示例

```
{
    "Events": [    // events数组长度范围为1～100
      {
        "CallbackCommand":"Push.OfflinePush",
        "EventType": 1,                          // 事件类型，EventType=1表示单聊
        "EventTime": 1557481127,                 // 事件发生时间戳， 单位为秒
        "From_Account": "user1",                 // 发送者
        "To_Account": "user2",                   // 接受者
        "PushPlatform": 1,                       // 推送厂商
        "PushStage": 1,                          // 推送阶段
        "MsgKey": "48374_2837546_1557481126",    // 单聊消息的唯一标识
        "MsgSeq": 48374,                         // 消息序列号
        "MsgRandom": 2837546,                    // 消息随机数
        "MsgTime": 1557481126,                   // 消息的发送时间戳，单位为秒
        "PushID": "67120c46_6a4086c0_9bf5fb50_200002d1089b322_2000005d8d46421", // 推送唯一标识
        "ErrCode": 0,                            // 推送事件结果
        "ErrInfo": "OK"                          // 推送事件结果描述，可能为空
```

```
    },
    {
      "CallbackCommand":"Push.OfflinePush",
      "EventType": 2,                          // 事件类型,EventType=2表示群聊
      "EventTime": 1557481127,                 // 事件发生时间戳，单位为秒
      "From_Account": "user2",                 // 发送者
      "To_Account": "user3",                   // 接受者
      "PushPlatform": 1,                       // 推送厂商
      "PushStage": 1,                          // 推送阶段
      "GroupID": "@TGS#12DEVUDHQ",             // 群聊的群组 ID
      "MsgSeq": 48375,                         // 消息序列号
      "MsgRandom": 2837546,                    // 消息随机数
      "MsgTime": 1557481126,                   // 消息的发送时间戳，单位为秒
      "PushID": "6710c631_85_392fcbcff_200000b2f2c6820_200002c869bfb81", // 推送唯一标识
      "ErrCode": 0,                            // 推送事件结果
      "ErrInfo": "OK"                          // 推送事件结果描述，可能为空
    },
    ....
  ]
}
```

请求包字段说明

字段	类型	说明
Events	Array [Event Object]	批量回调内容，最多包含100个回调事件（Event Object）的数据

Event Object结构

字段	类型	说明
CallbackCommand	String	回调命令
EventType	Integer	事件类型： • EventType = 1表示单聊 • EventType = 2表示群聊
EventTime	Integer	事件发生时间戳, 单位为秒
From_Account	String	消息发送者 UserID
To_Account	String	消息接收者 UserID
PushPlatform	Integer	推送厂商： • PushPlatform = 0表示未知厂商 • PushPlatform = 1表示 Apple APNS 推送 • PushPlatform = 2表示小米推送 • PushPlatform = 3表示华为推送 • PushPlatform = 4表示 Google FCM 推送 • PushPlatform = 5表示魅族推送 • PushPlatform = 6表示 OPPO 推送 • PushPlatform = 7表示 vivo 推送 • PushPlatform = 8表示荣耀推送

PushStage	Integer	推送阶段： • PushStage = 1表示推送发送阶段 • PushStage = 2表示推送触达阶段 • PushStage = 3表示推送点击阶段
MsgKey	String	单聊消息的唯一标识，仅在 EventType = 1时有效。群聊时该字段为空
GroupID	String	群聊的群组 ID，仅在 EventType = 2时有效。单聊时该字段为空
MsgSeq	Integer	消息序列号，用于标记该条消息（32位无符号整数）
MsgRandom	Integer	消息随机数，用于标记该条消息（32位无符号整数）。群聊时该字段为空
MsgTime	Integer	消息的发送时间戳，单位为秒
PushID	String	推送唯一标识，可以用PushID在IM控制台查询推送下发情况
ErrCode	Integer	推送事件结果： • ErrCode = 0表示成功 • ErrCode 非0表示失败
ErrInfo	String	推送事件结果描述，可能为空

应答包示例

```
{
    "ActionStatus": "OK",
    "ErrorInfo": "",
    "ErrorCode": 0 // 0为回调成功，1为回调出错
}
```

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	错误说明

高级功能
自定义角标

最近更新时间：2025-06-12 11:53:32

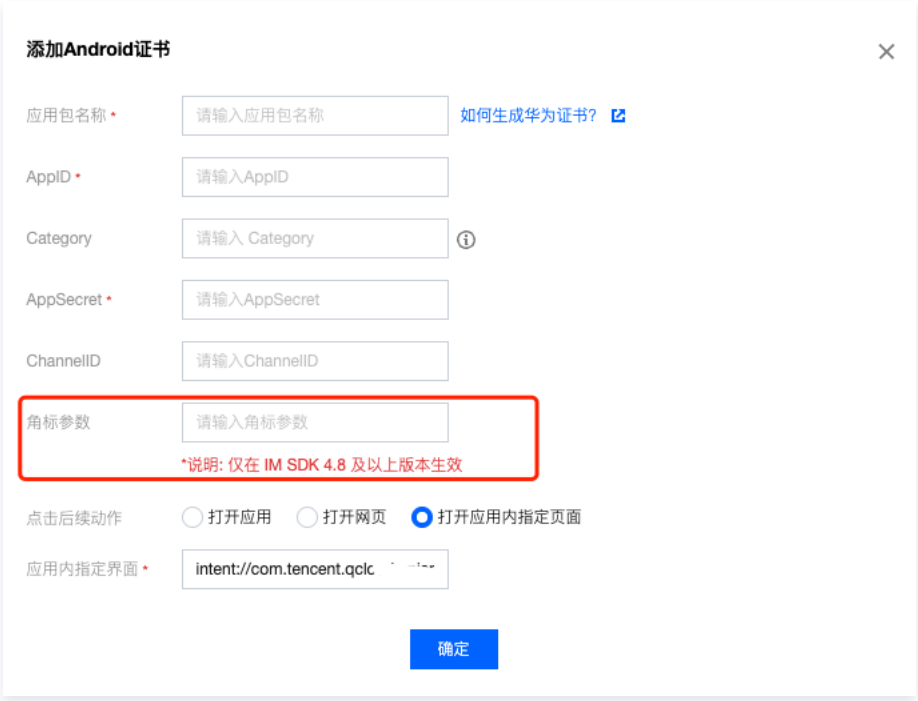
Android

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:(NSDictionary
*)launchOptions {
    // 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;
}
```

Flutter

请参见 Android 和 iOS 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。

uni-app

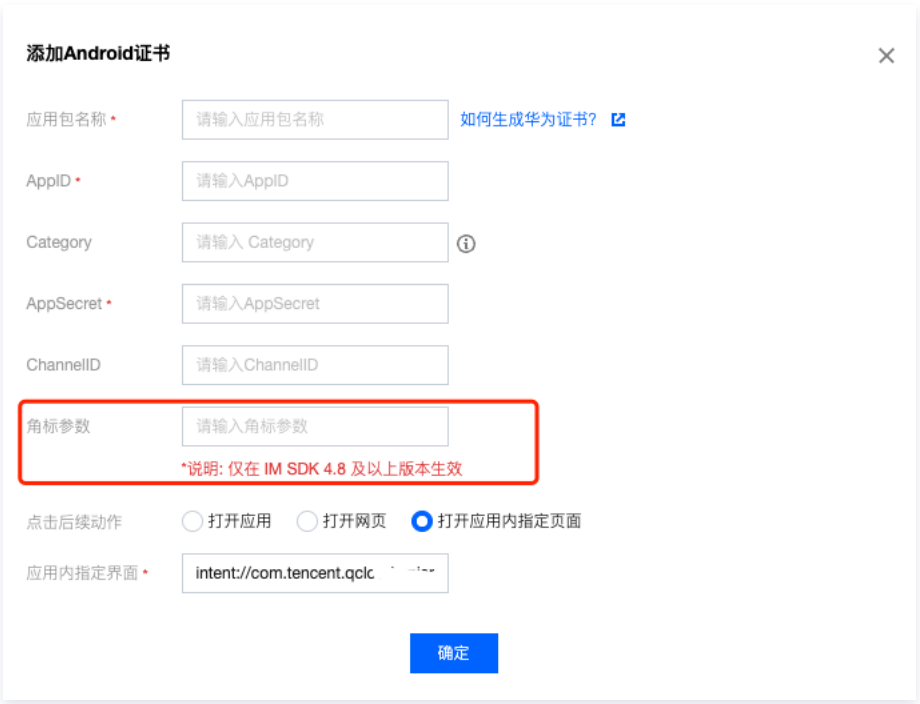
支持厂商

华为。

配置方法

步骤1. 控制台配置华为角标参数为应用的启动类。

> **说明：**
> uniapp 应用的启动类为 `io.dcloud.PandoraEntry` 。



步骤2. 监听 Chat SDK 未读总数变化自行设置角标数量。

1. 通过 TOTAL_UNREAD_MESSAGE_COUNT_UPDATED 监听 Chat SDK 会话未读总数更新。
2. 通过 `plus.runtime.setBadgeNumber` 设置角标数量。

```
let onTotalUnreadMessageCountUpdated = function(event) {
  const unreadCount = event.data; // 当前会话的未读总数
  plus.runtime.setBadgeNumber(unreadCount); // 设置角标数量
};
chat.on(TencentCloudChat.EVENT.TOTAL_UNREAD_MESSAGE_COUNT_UPDATED,
onTotalUnreadMessageCountUpdated);
```

React Native

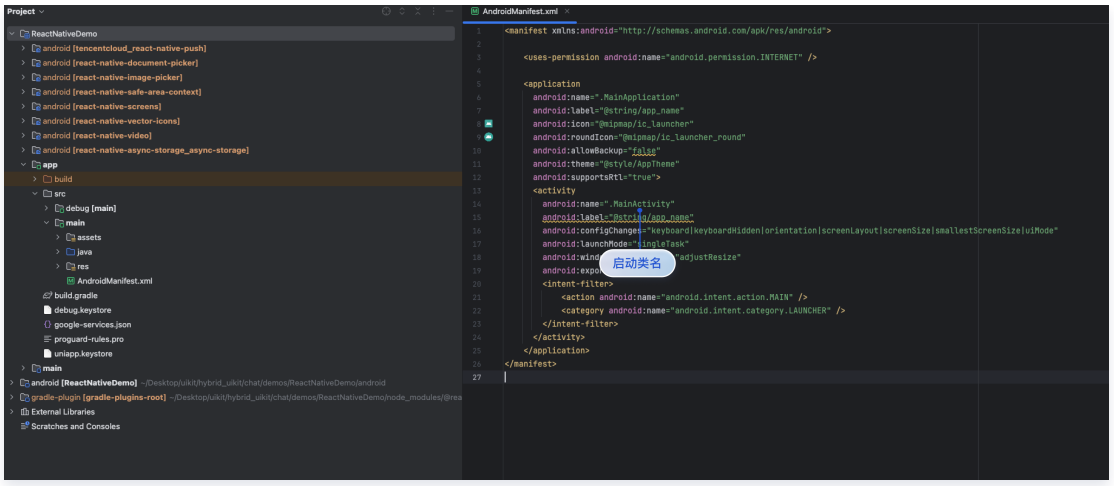
支持厂商

华为。

获取应用启动类

从 MyReactNativeApp/android/app/src/main/AndroidManifest.xml 中获取启动类名。如图所示：

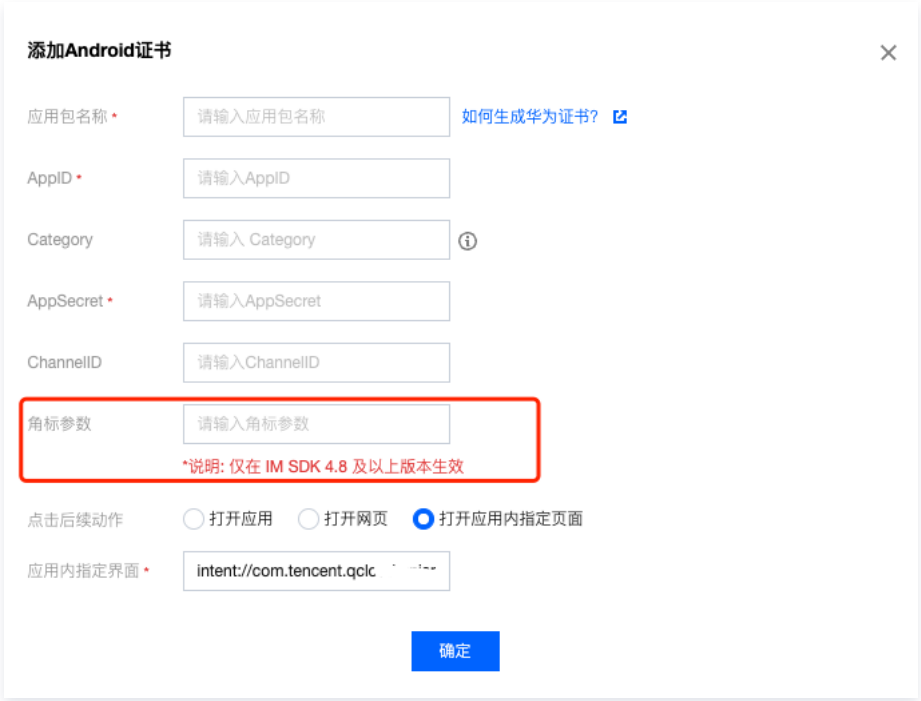
> **说明：**
> 应用的启动类 = “包名” + 启动类名。



包名	“com.tencent.qcloud.tim.demo”
启动类名	“MainActivity”
应用的启动类	“com.tencent.qcloud.tim.demo.MainActivity ”

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.MainActivity ”。组件会自动解析和更新角标，反之不会更新角标。



微信小程序多端框架

支持厂商：

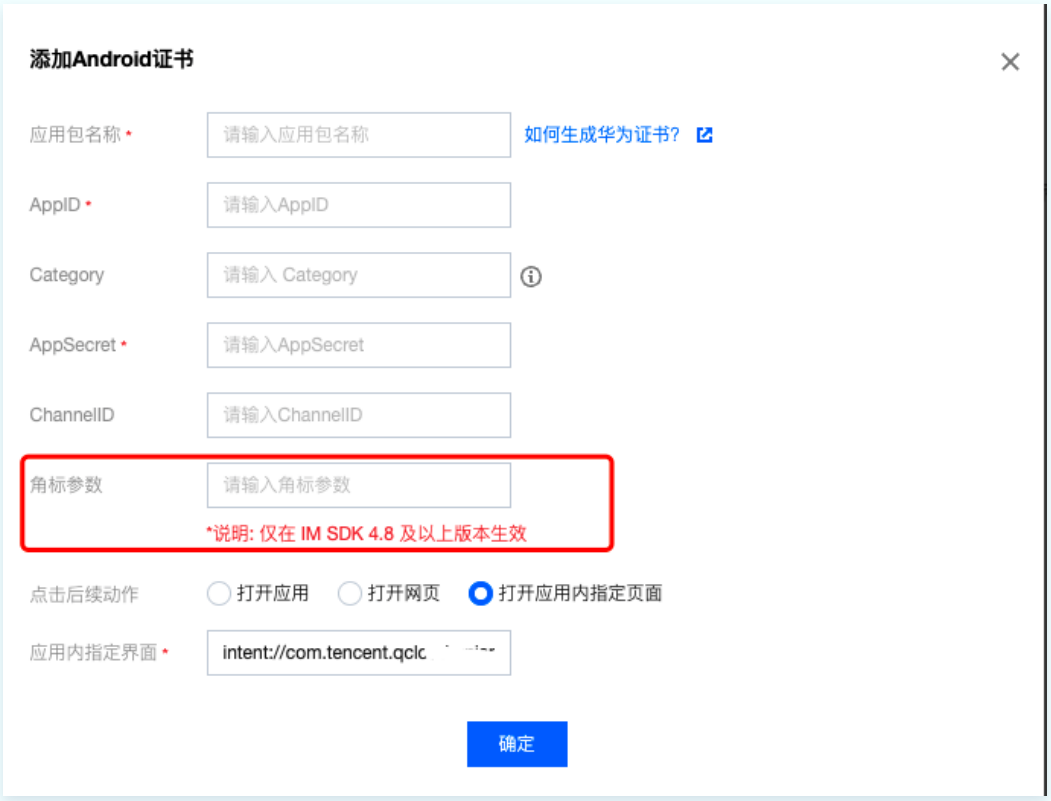
华为、iOS。

自定义角标步骤：

1. 调用 registerPush 注册推送服务。
2. 监听业务角标的变化，将变化数通过接口 setCustomBadgeNumber 同步给 Push。
3. 应用退后台或者离线后，Push 会根据离线消息自动更新角标数。
4. 再次打开应用回到前台，业务也可以继续通过该接口清理或者更新角标数。

注意：

1. 每次打开应用回到前台，都应该设置自定义角标，否则在 iOS 设备上，角标数会在之前自定义的基础上加 1 。
2. 在华为设备上实现角标功能时，需要在 IM 控制台额外配置角标参数。该参数需指定应用的主入口 Activity 类路径，格式为：[应用包名].ui.MainActivity。以包名com.tencent.weauth为例，对应的角标参数应配置为：com.tencent.weauth.ui.MainActivity。



自定义铃音

最近更新时间：2025-04-15 16:21:42

> ⚠ **注意：**
>
> 1. 离线推送消息铃音不设置，默认跟随设备的系统通知设置，以华为为例，详细参见：“手机设置 > 通知 > 应用的通知管理 > 通知铃声”。
> 2. 自定义铃声需要按照厂商平台逐一设置支持，详细参见以下方法总结：不同厂商不同平台自定义铃音方式有区别，Android 8.0 支持厂商还需要通过 channel 来设置，响铃时长和铃音资源时长有关。

Android

Android 8.0 以下系统

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. 发送消息指定生效自定义铃音。

restAPI

请参考 restAPI 接口，比如 单发推送接口，字段示例如下：

```
{
    // ...
    "OfflinePushInfo": {
        "AndroidInfo": {
            "Sound": "shake",  // 不带后缀
        },
        "ApnsInfo": {
            "Sound": "apns.mp3",
        }
    }
}
```

SDK API

若集成了 IM 相关产品，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 ? null
: userID, isGroup ? groupID : null,
 V2TIMMessage.V2TIM_PRIORITY_DEFAULT, false, v2TIMOfflinePushInfo, new
V2TIMSendCallback<V2TIMMessage>() {
@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。

Android 8.0 及以上系统

华为 和 ANPS

华为、APNS 可调用上面接口来设置离线推送铃音提示。

OPPO

1. 将定制的铃音资源文件放在工程资源的 raw 目录下，然后通过以下方式创建通知 channel。

```
// 自定义创建示例
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    NotificationManager nm = (NotificationManager)
context.getSystemService(Context.NOTIFICATION_SERVICE);
    NotificationChannel notificationChannel =
            new NotificationChannel("channelId", "channelName",
NotificationManager.IMPORTANCE_HIGH);
    notificationChannel.enableLights(true);
    notificationChannel.enableVibration(true);
    notificationChannel.setShowBadge(true);
    notificationChannel.setLockscreenVisibility(Notification.VISIBILITY_PUBLIC);

    // "android.resource://包名/raw/private_ring"
    notificationChannel.setSound(Uri.parse("sound"), null);
    nm.createNotificationChannel(notificationChannel);
}
```

2. 使用创建的 channel。
   - 创建私信通道。
   - 发送消息指定 channelID 生效自定义铃音，接口设置参考如下，控制台设置见证书的 channelID 字段，两者设置一个即可。

restAPI

请参考 restAPI 接口，比如 单发推送接口，字段示例如下：

```
{
   // ...
   "OfflinePushInfo": {
       "AndroidInfo": {
```

```
            "OPPOChannelID": "test_OPPO_channel_id",
        }
    }
}
```

SDK API

若集成了 IM 相关产品，请您发送消息调用接口 setAndroidOPPOChannelID 。

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。

2. 发送消息指定 channelID 生效自定义铃音，接口设置参考如下，控制台设置见证书的 channelID 字段，两者设置一个即可。

restAPI

请参考 restAPI 接口，比如 单发推送接口，字段示例如下：

```
{
    // ...
    "OfflinePushInfo": {
        "AndroidInfo": {
            "XiaoMiChannelID": "test_XiaoMi_channel_id",
        }
    }
}
```

SDK API

若集成了 IM 相关产品，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 ? null
: userID, isGroup ? groupID : null,
 V2TIMMessage.V2TIM_PRIORITY_DEFAULT, false, v2TIMOfflinePushInfo, new
V2TIMSendCallback<V2TIMMessage>() {
@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 将定制的铃音资源文件放在工程资源的 raw 目录下，然后通过以下方式创建通知 channel。

```
// 自定义创建示例
if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
    NotificationManager nm = (NotificationManager)
context.getSystemService(context.NOTIFICATION_SERVICE);
    NotificationChannel notificationChannel =
            new NotificationChannel("channelId", "channelName",
NotificationManager.IMPORTANCE_HIGH);
    notificationChannel.enableLights(true);
    notificationChannel.enableVibration(true);
    notificationChannel.setShowBadge(true);
    notificationChannel.setLockscreenVisibility(Notification.VISIBILITY_PUBLIC);

    // "android.resource://包名/raw/private_ring"
    notificationChannel.setSound(Uri.parse("sound"), null);
    nm.createNotificationChannel(notificationChannel);
}
```

2 发送消息指定自定义铃音的 channelID，

restAPI

请参考 restAPI 接口，比如 单发推送接口，字段示例如下：

```
{
```

```
    // ...
    "OfflinePushInfo": {
        "AndroidInfo": {
            "GoogleChannelID": "test_Google_channel_id",
        }
    }
}
```

SDK API

若集成了 IM 相关产品，详情请参见 setAndroidFCMChannelID。

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

String msgID = V2TIMManager.getMessageManager().sendMessage(v2TIMMessage, isGroup ? null
: userID, isGroup ? groupID : null,
 V2TIMMessage.V2TIM_PRIORITY_DEFAULT, false, v2TIMOfflinePushInfo, new
V2TIMSendCallback<V2TIMMessage>() {
@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 ChannnelID 仅支持证书模式。**

iOS

1. 请在发送消息的时候设置 OfflinePushInfo 的 iOSSound 字段， iOSSound 传语音文件名。

restAPI

请参考 restAPI 接口，比如 单发推送接口，字段示例如下：

```
{
    // ...
    "OfflinePushInfo": {
        "ApnsInfo": {
            "Sound": "apns.mp3",
        }
    }
}
```

SDK API

若集成了 IM 相关产品，请您发送消息参考：

```
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
priority:V2TIM_PRIORITY_DEFAULT onlineUserOnly:NO offlinePushInfo:pushInfo progress:nil
succ:^{

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

}];
```

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。
- iOS 自定义铃音长度不能超过 30s。

2. 请在发送消息的时候设置 OfflinePushInfo 的 `AndroidSound` 字段， `AndroidSound` 传语音文件名。

restAPI

请参考 restAPI 接口，比如 单发推送接口，字段示例如下：

```
{
   "OfflinePushInfo": {
       "AndroidInfo": {
           "Sound": "shake",  // 不带后缀
           "OPPOChannelID": "test_OPPO_channel_id",
           "XiaoMiChannelID": "test_XiaoMi_channel_id",
           "OPPOChannelID": "test_OPPO_channel_id",
           "GoogleChannelID": "test_Google_channel_id"
       },
       "ApnsInfo": {
           "Sound": "apns.mp3"
       }
   }
}
```

SDK API

若集成了 IM 相关产品，请您发送消息参考：

```
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
priority:V2TIM_PRIORITY_DEFAULT onlineUserOnly:NO offlinePushInfo:pushInfo progress:nil
succ:^{

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

}];
```

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。

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

restAPI

请参考 restAPI 接口，比如 单发推送接口，字段示例如下：

```
{
    // ...
    "OfflinePushInfo": {
        "AndroidInfo": {
            "Sound": "shake",  // 不带后缀
            "OPPOChannelID": "test_OPPO_channel_id",
            "XiaoMiChannelID": "test_XiaoMi_channel_id",
            "OPPOChannelID": "test_OPPO_channel_id",
            "GoogleChannelID": "test_Google_channel_id"
        },
        "ApnsInfo": {
            "Sound": "apns.mp3"
        }
    }
}
```

SDK API

若集成了 IM 相关产品，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 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。

uni-app

> **说明：**
> - 接收端需集成 uni-app 腾讯云推送服务（Push）。
> - 发送端 @tencentcloud/chat ⩾ 3.3.2 。
> - 支持华为、小米、OPPO、FCM 和 APNS。

接收端

Android

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

设置私信通道和自定义铃音（ OPPO、FCM 需要设置）

> **说明：**
> 1. uni-app 腾讯云推送服务（Push）≥ 0.5.0。
> 2. OPPO、FCM 使用自定义铃音需要先设置私信通道和自定义铃音。

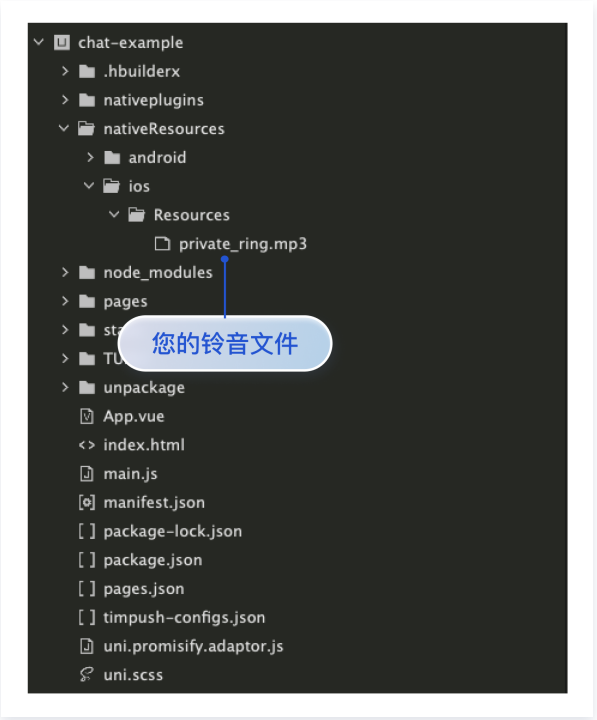
完上述步骤后，您可以调用 push.createNotificationChannel 设置 OPPO 、FCM 私信通道和自定义铃音。
在 `App.vue` 中设置 OPPO、FCM 私信通道。如图所示：

```
import * as Push from '@/uni_modules/TencentCloud-Push';
Push.createNotificationChannel({
    channelID: '', // 自定义 channel 的 ID。OPPO 为控制台配置的 channelID
    channelName: '自定义channel',// 自定义channel 名称
    channelDesc: '这只是描述', // 自定义 channel 描述
    channelSound: 'private_ring'  // 自定义铃音的名称且不需要后缀名
}, () => {
    console.log('Push | createNotificationChannel ok');
})
```

iOS

> **说明：**
> - iOS 自定义铃音，uniapp 必须为正式包。
> - iOS 自定义铃音长度不能超过 30s。

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



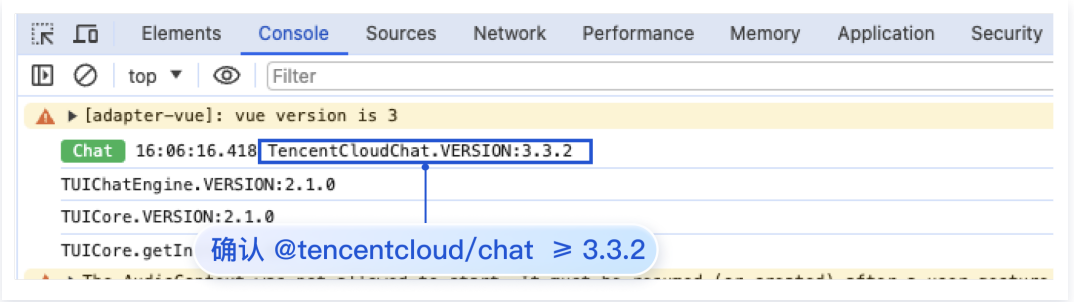
发送端

1. 升级 @tencentcloud/chat 到最新版本。

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

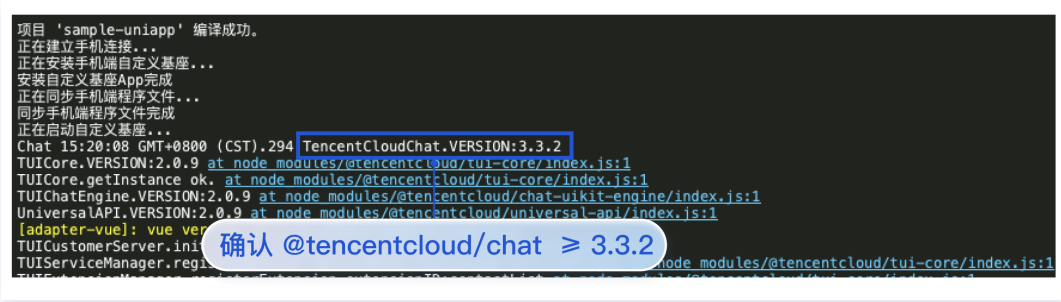
web

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：



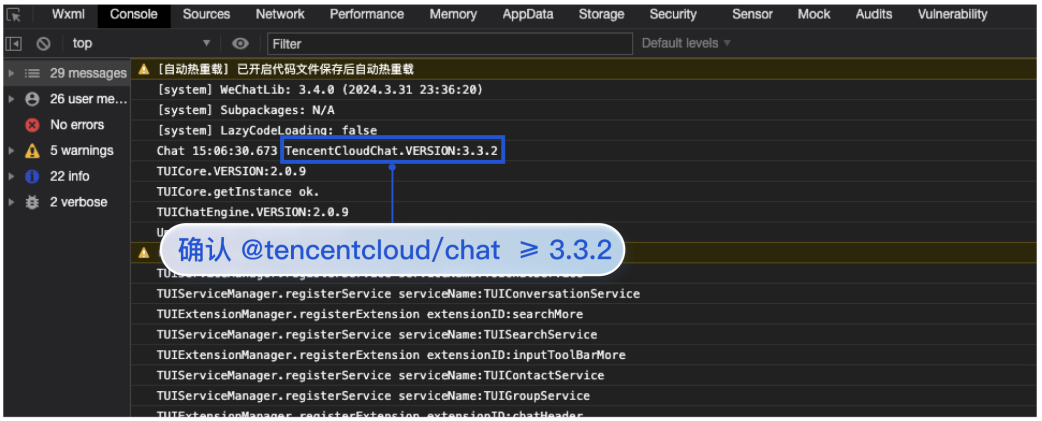
uniapp

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



小程序

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。

说明：

- 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

说明：

- 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 的情况下）。接入侧可自定义离线推送的标题及内容
    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 XiaoMiChannelID
          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 FCMChannelID。
          OPPOChannelID: '', // OPPO手机必须设置 OPPOChannelID
        },
        apnsInfo: { // APNs 推送配置
       // apnsInfo.sound = TUIChatEngine.TYPES.IOS_OFFLINE_PUSH_NO_SOUND，表示接收时不会播放声音。
       // apnsInfo.sound = TUIChatEngine.TYPES.IOS_OFFLINE_PUSH_DEFAULT_SOUND，表示接收时播放系统声音。
          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

无 UI 集成

> **说明：**
>
> - 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 的情况下）。接入侧可自定义离线推送的标题及内容
    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 XiaoMiChannelID
          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 FCMChannelID。
          OPPOChannelID: '', // OPPO手机必须设置 OPPOChannelID
        },
        apnsInfo: { // APNs 推送配置
       // apnsInfo.sound = TencentCloudChat.TYPES.IOS_OFFLINE_PUSH_NO_SOUND，表示接收时不会播放声音。
```

```
      // apnsInfo.sound = TencentCloudChat.TYPES.IOS_OFFLINE_PUSH_DEFAULT_SOUND，表示接收时播放
系统声音。
        sound: 'private_ring.mp3', // iOS 自定义铃音
      }
    }
});
```

参考文档：Chat SDK - sendMessage 文档

Callkit 集成

> **说明：**
> - androidSound 为铃声文件名，不需要后缀名。
> - iOSSound为铃声文件名，需要后缀名。

```
const TUICallKit = uni.requireNativePlugin('TencentCloud-TUICallKit');
const options = {
  userID: '',
  callMediaType: 1, // 语音通话(callMediaType = 1)、视频通话(callMediaType = 2)
  callParams: {
    // 如果接收方不在线，进行离线推送（在接收方 App 退后台或者进程被 kill 的情况下）。接入侧可自定义离
线推送的标题及内容
    offlinePushInfo: {
      title: 'test-title',
      description: 'you have a test call',
      androidSound: 'private_ring', // Andorid 离线推送的自定义铃音
      iOSSound: 'private_ring.mp3', // iOS 离线推送的自定义铃音
    },
  },
};
TUICallKit.call(options, (res) => {
  if (res.code === 0) {
    console.log('call success');
  } else {
    console.log(`call failed, error message = ${res.msg}`);
  }
});
```

参考文档：CallKit 配置 offlinePushInfo

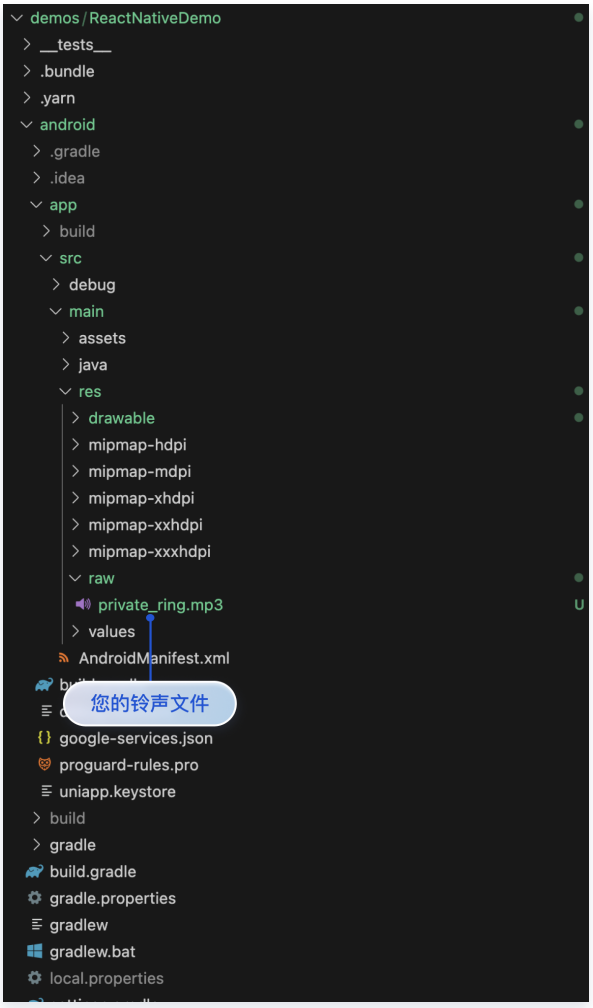
React Native

> **说明：**
> - 接收端需集成@tencentcloud/react-native-push 。
> - 发送端 @tencentcloud/chat ⩾ 3.3.2 。
> - 支持华为、小米和 APNS。

接收端

Android

定制的铃音资源文件，添加到项目 MyReactNativeApp/android/app/src/main/res/raw 目录下，如果该目录不存在，请手动创建。如下图所示：

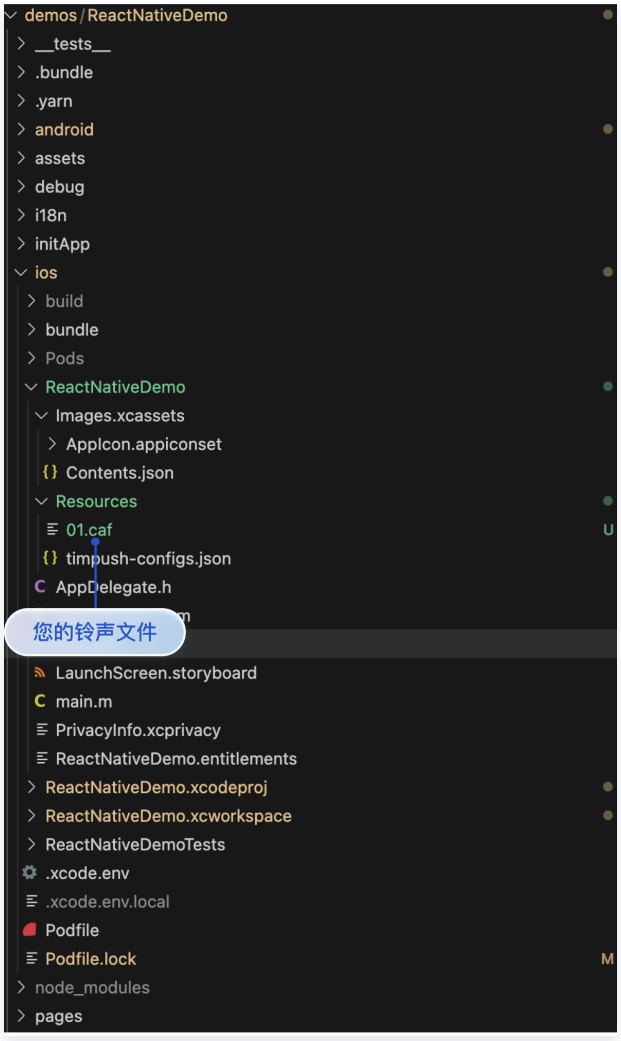


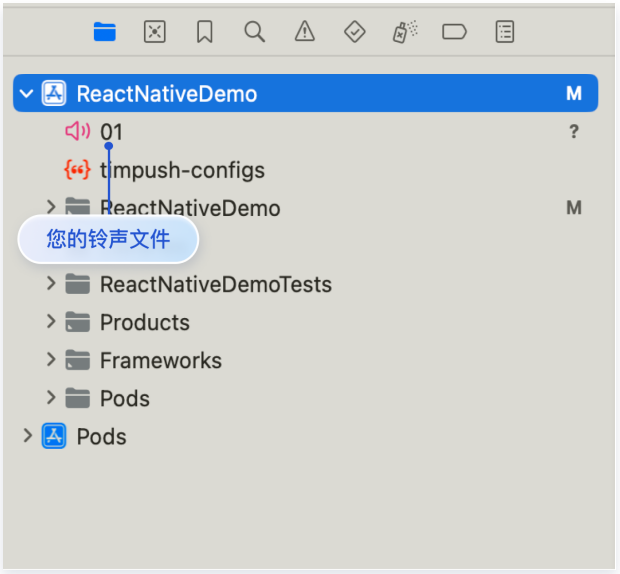
iOS

> **说明：**
> - iOS 自定义铃音长度不能超过 30s。

定制的铃音资源文件，添加到项目到 MyReactNativeApp/ios/MyReactNativeApp/Resources 目录下，并用 XCode 打开 MyReactNativeApp 项目，右键点击项目 > Add Files to "MyReactNativeApp"，将铃音文件添加到工程。如下图所示：

项目	Xcode





发送端

发送消息，设置 offlinePushInfo 自定义铃音的相关参数。

说明：

- 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 。
- 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。

```
// 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 的情况下）。接入侧可自定义离线推送的标题及内容
  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 XiaoMiChannelID
```

```
        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
FCMChannelID。
        OPPOChannelID: '', // OPPO手机必须设置 OPPOChannelID
      },
      apnsInfo: { // APNs 推送配置
     // apnsInfo.sound = TencentCloudChat.TYPES.IOS_OFFLINE_PUSH_NO_SOUND，表示接收时不会播放声音。
     // apnsInfo.sound = TencentCloudChat.TYPES.IOS_OFFLINE_PUSH_DEFAULT_SOUND，表示接收时播放系统
声音。
        sound: 'private_ring.mp3', // iOS 自定义铃音
      }
    }
});
```

参考文档：Chat SDK - sendMessage 文档

微信小程序多端框架

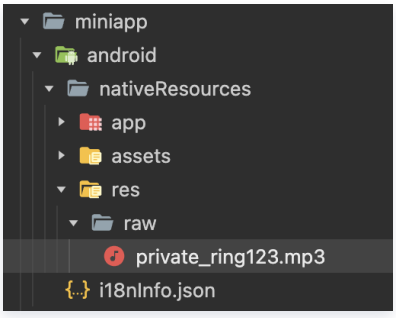
客户端配置

Android

> **说明：**
> 配置原生资源请参考文档。

1、添加自定义铃音文件

把铃音文件添加到项目 miniapp/android/nativeResources/res/raw 目录下，如下图所示：



2、创建客户端通知 channel

在 App 启动时调用 Push.createNotificationChannel 方法设置通知 channel。

```
import Push from "@tencentcloud/donut-push"

App({
  onLaunch: function () {
    Push.createNotificationChannel({
          channelID: "",  // 自定义 channel 的 ID。
          channelName: "",  // 自定义channel 名称
          channelDesc: "",  // 自定义 channel 描述
          channelSound: ""  // 自定义铃音文件的名称且不需要文件扩展名
        }).then((ret) => {
          console.info("createNotificationChannel success", ret);
```

```
        }).catch((ret) => {
          console.error("createNotificationChannel failed", ret);
        });
    }
})
```

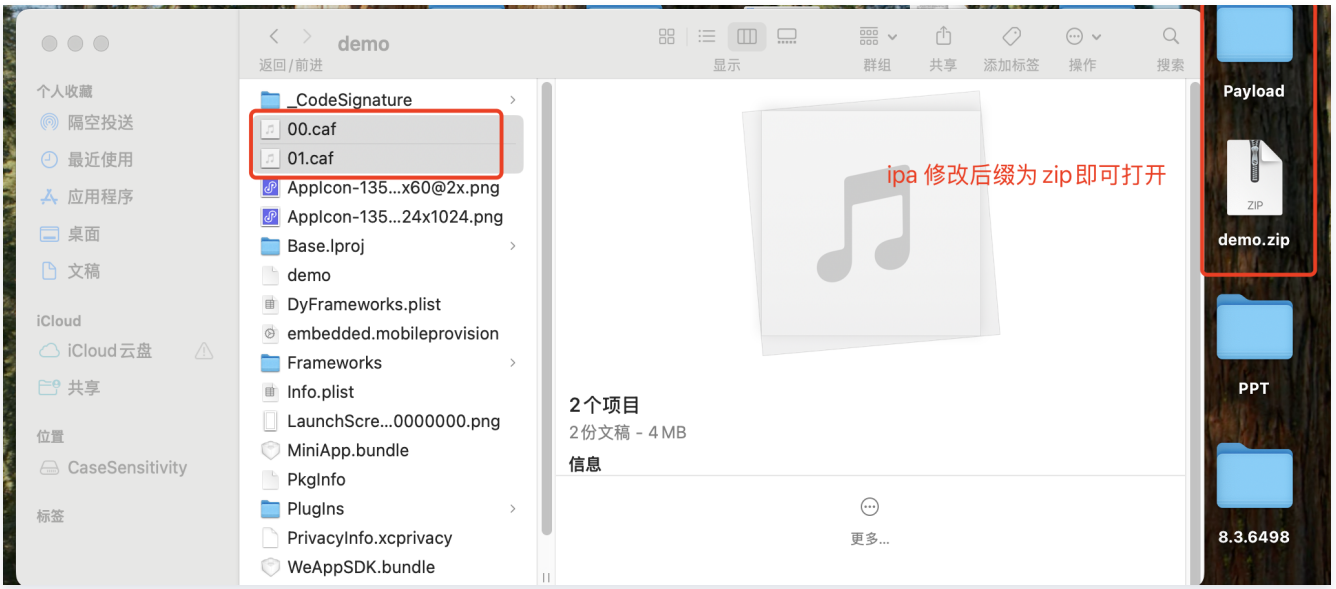
iOS

> **说明：**
> - iOS 自定义铃音长度不能超过 30s。
> - 微信开发者工具版本号 >= 1.06.2412042 。

1、添加自定义铃音文件

把铃音文件添加到项目目录下，如下图所示：

2、可通过构建检查您的 IPA 内是否包含此铃声，注意铃声长度不能超过30秒



服务端 REST API

请求包示例

```
{
  "From_Account": "administrator",
  "To_Account": ["100480"],
  "MsgRandom": 3674128,
  "OfflinePushInfo": {
      "PushFlag": 0,
      "Title": "离线推送标题1adad1",
      "Desc": "离线推送内容11adasd1",
      "Ext":"{\"aa\":12123}",
      "AndroidInfo": {
           "Sound": "private_ring123",  // 跟客户端创建的 channel 中的铃声文件名称一致
           "OPPOChannelID": "test_oppo_channel_id",
           "XiaoMiChannelID": "test_xiaomi_channel_id",
           "GoogleChannelID": "test_google_channel_id"
       },
       "ApnsInfo": {
            "Sound": "01.caf"// 跟接收端 ipa 包内的铃声文件名称一致
       }
    }
}
```

常见问题

配置完成后，收到推送但是手机没有铃声?

请先检查是否开启手机系统铃声，再检查 App 的通知权限中是否打开铃声开关。

部分手机离线推送自定义铃音设置无效?

目前支持离线推送自定义铃音的厂商有：华为、OPPO、小米、FCM、APNS。

自定义小图标

最近更新时间：2024-09-19 15:15:22

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

在应用主工程的清单文件配置生效：

华为

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

Google FCM

```
<!-- [START fcm_default_icon] -->
<!-- Set custom default icon. This is used when no icon is set for incoming notification
messages.
     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 is set
for the incoming
     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。

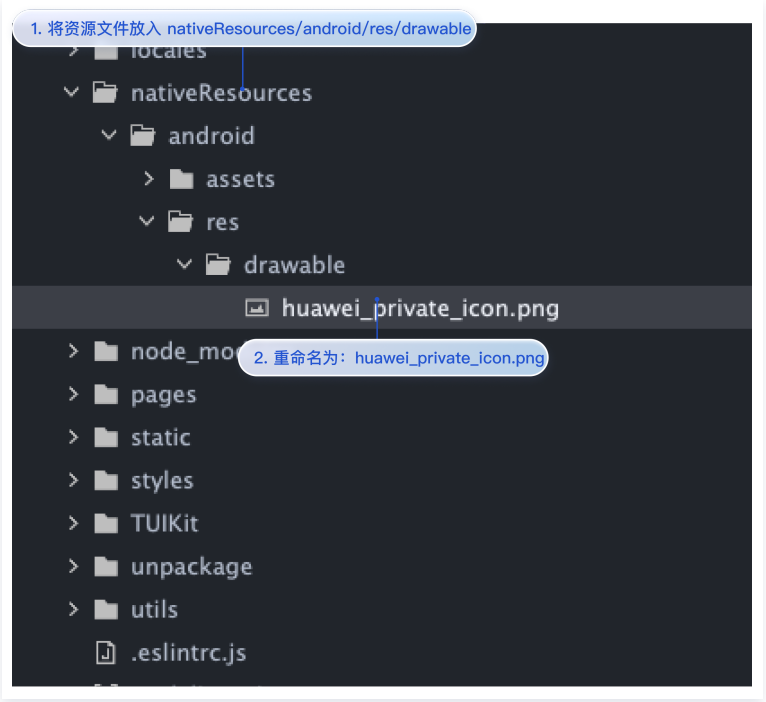
iOS

不支持自定义，默认使用 App 图标。

uni-app

说明：
仅**华为**支持设置，其他厂商不支持，默认使用 App 图标。

将小图标放到 `nativeResources/android/res/drawable` 文件夹中，将资源文件重命名为 `huawei_private_icon.png` 即可。如图所示：

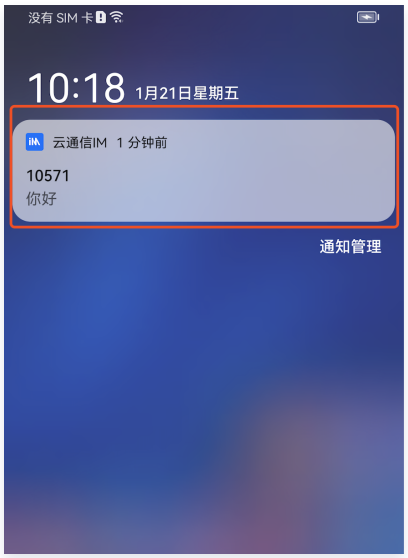


自定义点击跳转

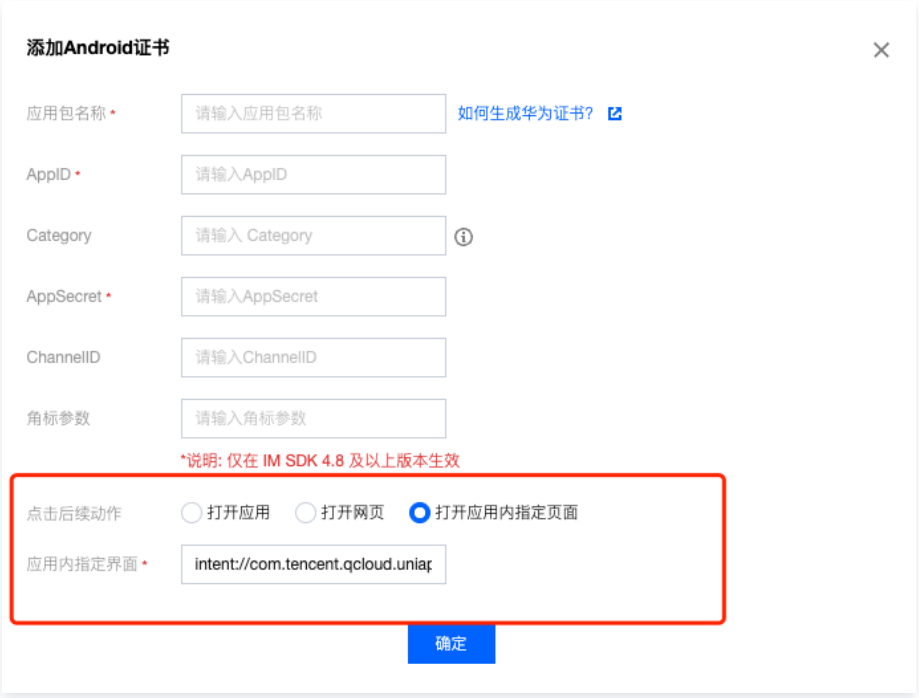
最近更新时间：2025-04-07 16:57:52

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. 收到推送消息后点击通知栏，组件会回调该点击事件和透传离线消息。

自定义点击跳转实现

> **注意：**
> 注册回调时机建议放在应用 Application 的 oncreate() 函数中。

```
TIMPushManager.getInstance().addPushListener(new TIMPushListener() {
    @Override
    public void onNotificationClicked(String ext) {
        Log.d(TAG, "onNotificationClicked =" + ext);
        // 获取 ext 自定义跳转

    }
});
```

自定义点击跳转实现（旧方案）

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。

> ⚠ **注意：**
> 注册回调时机建议放在应用 Application 的 oncreate() 函数中。

1. 回调方式如下：

```
TUICore.registerEvent(TUIConstants.TIMPush.EVENT_NOTIFY,
TUIConstants.TIMPush.EVENT_NOTIFY_NOTIFICATION, new ITUINotification() {
        @Override
        public void onNotifyEvent(String key, String subKey, Map<String, Object> param) {
            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.NOTIFICATION_EXT_KEY);
                        // 获取 ext 自定义跳转

                    }
                }
            }
        }
    });
```

2. 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, intentFilter);

//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_EXT_KEY);
            // 获取 ext 自定义跳转

        } else {
            Log.e(TAG, "onReceive ext is null");
        }
    }
}
```

iOS

如果您需要自定义解析收到的远程推送，您可按照如下方法实现：

自定义点击跳转实现

> **注意：**
> 注册回调时机建议放在应用 AppDelegate 的 didFinishLaunchingWithOptions 函数中。

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:
(NSDictionary *)launchOptions {
    [TIMPushManager addPushListener:self];
    return YES;
}
#pragma mark - TIMPushListener
- (void)onNotificationClicked:(NSString *)ext {
        // 获取 ext 自定义跳转
}
```

自定义点击跳转实现（旧方案）

您需要在 AppDelegate.m 文件中实现 `- onRemoteNotificationReceived` 方法。

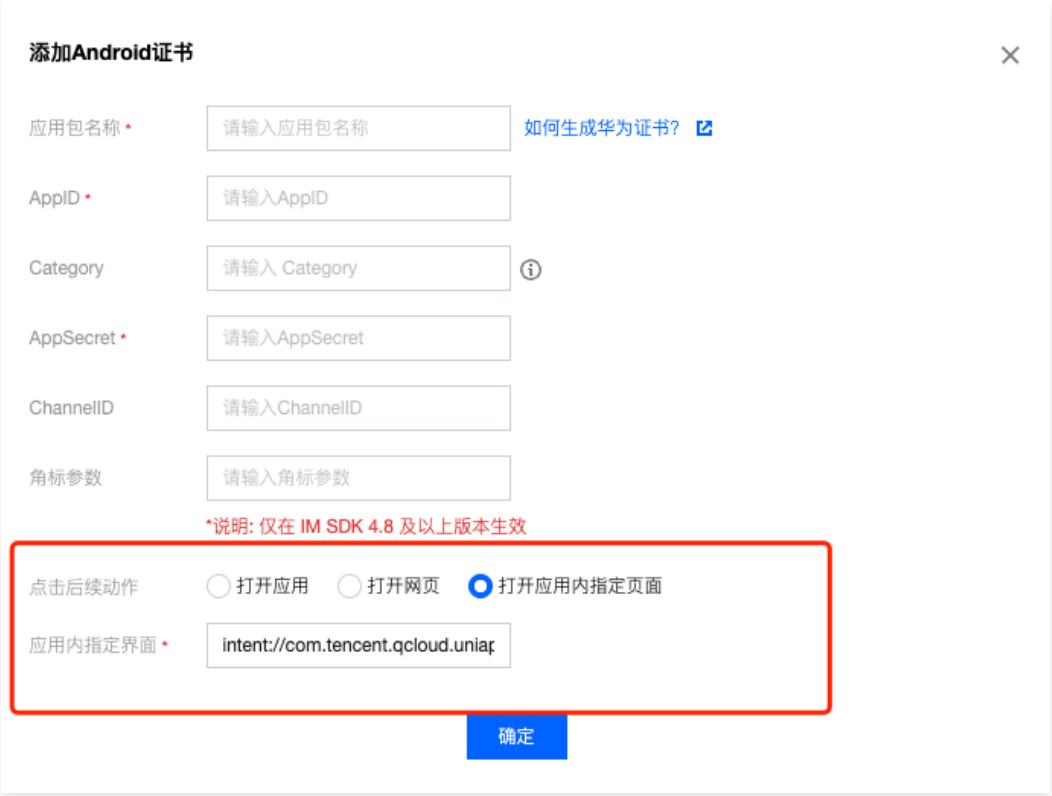
```
#pragma mark - TIMPush

- (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 -
navigateToBuiltInChatViewController:groupID: 方法。
    return NO;
}
```

Flutter

步骤1: 厂商配置

参见 厂商配置 > Flutter，完成厂商配置。

> **说明:**
> 应用内指定界面，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

如果您需要自定义解析收到的远程推送，您可按照如下方法实现：

自定义点击跳转实现

> **注意:**
> 注册回调时机建议放在程序入口 main() 函数中。

```
TIMPushListener timPushListener = TIMPushListener(
      onNotificationClicked: (String ext) {
        debugPrint("ext: $ext");
        // 获取 ext 自定义跳转

      }
    );
```

```
tencentCloudChatPush.addPushListener(listener: timPushListener);
```

自定义点击跳转实现（旧方案）

1、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.
  }
}
```

2、请注意，不要在 Flutter 程序入口的 main 方法中调用。
调用 `TencentCloudChatPush().registerPush` 方法成功后，就可以收到离线推送通知了。

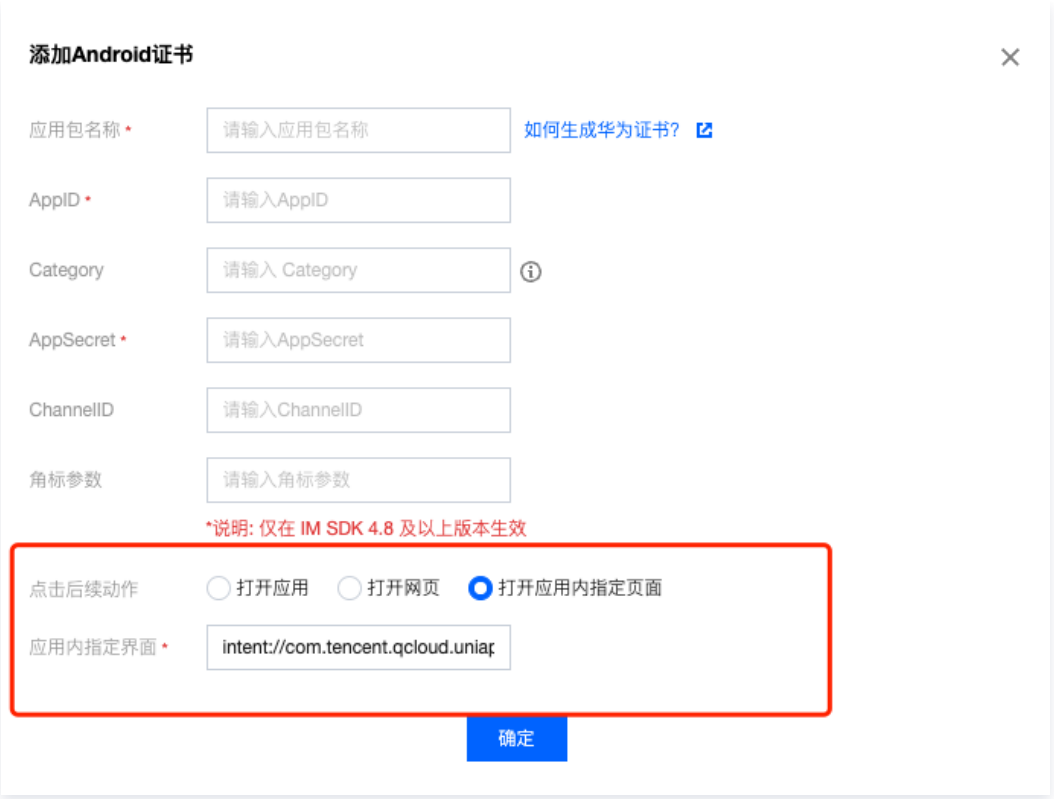
```
TencentCloudChatPush().registerPush(
    onNotificationClicked: _onNotificationClicked,
    sdkAppId: 您的sdkAppId,
    appKey: "客户端密钥",
    apnsCertificateID: 您配置的证书 ID);
```

uniapp

步骤1: 厂商配置

参见 厂商配置 > uniapp，完成厂商配置。

> **说明:**
> 应用内指定界面，请使用默认配置。



步骤2: 获取离线推送扩展信息，进行自定义跳转。

在 `App.vue` 的 `onShow` 回调中，当设备收到离线推送，拉起 App 时，业务侧可通过 getNotificationExtInfo 接口获取推送扩展信息。

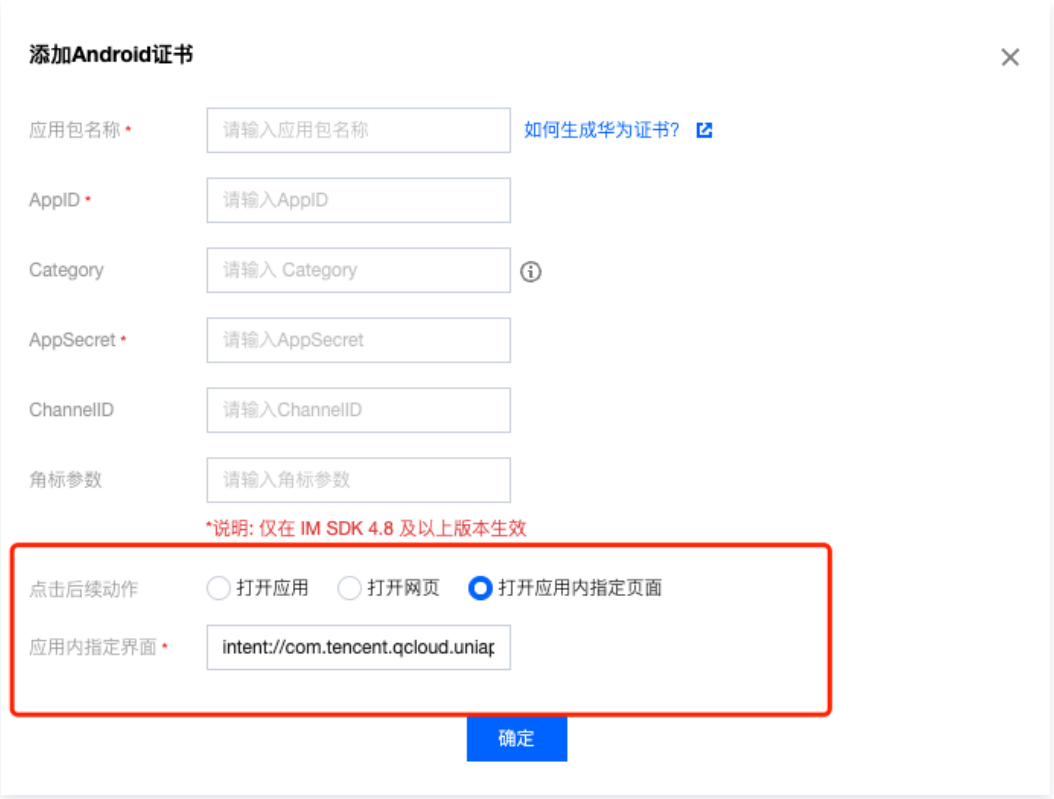
```
export default {
    onLaunch: function() {
    },
    onShow: function() {
        console.log('App Show')
        Push.getNotificationExtInfo((extInfo) => {
            console.log('getNotificationExtInfo ok', extInfo);
            // 在此处进行自定义跳转。
        })
    },
    onHide: function() {
        console.log('App Hide')
    }
}
```

React Native

步骤1: 厂商配置

参见 厂商配置 > React Native，完成厂商配置。

> **说明:**
> 应用内指定界面，请使用默认配置。



步骤2: 获取离线推送扩展信息，进行自定义跳转。

在 `App.tsx` 内监听 `AppState` 变化，当设备收到离线推送，拉起 App 时，业务侧可通过 getNotificationExtInfo 接口获取推送扩展信息。

```
import { AppState } from 'react-native';
const App = () => {
  useEffect(() => {
    const handleAppStateChange = (nextAppState: string) => {
      if (nextAppState === 'active') {
        console.log('App.tsx | AppState active');
        if (Push) {
          Push.getNotificationExtInfo((extInfo: string) => {
            console.log('App.tsx | getNotificationExtInfo ok', extInfo);
            // 在此处进行自定义跳转。
          });
        }
      }
    };
    AppState.addEventListener('change', handleAppStateChange);
  }, []);
};
```

微信小程序多端框架

步骤1: 厂商配置

参见 厂商配置 > 微信小程序多端框架，完成厂商配置。

> **注意：**
> 应用内指定界面配置项，请使用默认配置，否则会收不到点击通知事件。

步骤2: 注册点击通知事件监听，进行自定义跳转。

在小程序启动后，注册点击通知事件监听，用户点击通知后会收到点击通知事件，可根据扩展信息进行页面跳转。

```
const listener = (param) => {
  console.log('onEvent', JSON.stringify(param));
}
App({
    onLaunch: function () {
        console.log('App Show')
        Push.addPushListener(Push.EventName.NOTIFICATION_CLICKED, listener);
    }
}
```

推送消息分类

最近更新时间：2025-06-19 10:45:22

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

荣耀推送服务将对推送消息进行分类管理，将根据应用类型、消息内容和消息发送场景，将推送消息分成服务通讯和资讯营销两大类别。消息通知类型将会默认归为资讯营销类消息，资讯营销消息有每日推送数量上限。您可进行自分类权益的申请，自行对消息进行分类。

自分类推送定制方法

- 申请自分类权益。
- 推送消息携带 importance 字段，详情请参见 setAndroidHonorImportance，控制台设置见证书编辑 importance 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。

> **说明：**
> 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

根据推送消息的内容，将通知分类为通讯与服务和内容与营销两个大类别，推送效果和策略不同。其中通讯与服务是针对用户有一定关注度，且希望能及时接收的信息，通讯与服务消息类型需要邮件申请，内容与营销推送数量有限制。说明：如之前已开通私信通道权限，请参见“旧规则”。

新规则

自分类推送定制方法

- 通道权限申请。
- 推送消息携带 category 字段，详见 setAndroidOPPOCategory，控制台设置见证书编辑 Category 字段，两者设置一个即可。
- 如果您的应用为二级分类中属于聊天交友、电话短信、办公类的应用，且申请消息类型为“ IM 类消息”，还可以申请通知栏消息提醒等级设置，详见 setAndroidOPPONotifyLevel，具体请参见 强提醒申请。

具体请参见 消息分类标准。

旧规则（仅适用存量用户）

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

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，　　对收到此类消息　预期，关注程度较低，限制每　每设备推送数量。私信消息，　　对收到此类消息有预期，或需要及时知道的消息，如果错过可能会导致不良影响，推送数量无限制。

自分类推送定制方法

推送消息携带消息分类字段，详情请参见 setAndroidMeizuNotifyType，控制台设置参见证书编辑 “消息分类” 字段，两者设置一个即可。
具体请参见 推送消息新规。

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。

推送典型场景介绍

最近更新时间：2025-04-07 16:57:52

场景1：独立使用 Push 服务

仅集成 Push 产品，请参见 快速跑通 和 厂商通道 指引，完成配置接入流程即可。

场景2：接入了 IM 服务 ，还需要 Push 服务

集成使用 Chat、 CallKit 、 RoomKit 、 LiveKit 等产品，需要 Push 服务实现离线消息可触达，请参见 聊天互动 – 厂商配置 和 快速接入 指引，完成配置接入流程即可。

场景3：需要支持设备级免登录推送，IM 和 Push 混用情况

集成使用 Chat、 CallKit 、 RoomKit 、 LiveKit 等产品，用户未登录 IM 账号前，可以实现设备级免登录推送；用户登录 IM 账号后，也可以指定 userID 推送。

1. 请先参见独立 Push 服务文档，完成 快速跑通 和 厂商通道 接入流程，即可支持设备级免登录推送，设备推送 ID 见 getRegistrationID 。
2. 登录 IM 账号后请参见 registerPush 完成推送注册，即可实现指定 userID 推送。
   - 如果您使用 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，不需要再手动调用。
   - 如果您没有使用 TUILogin 提供的接口，请在 IM 登录成功后调用该接口，并将 appKey 参数设置为 null。

> **说明：**
> 1. RegistrationID 是推送接收设备的唯一标识 ID，注册推送服务成功时自动生成该 ID，卸载重装会改变；
> 2. 请不要通过 getRegistrationID 获取的 RegistrationID 做 IM 的账号登录，这样会打破 registrationID 的真实意义从而出现功能异常；
> 3. 前台推送开关详见 disablePostNotificationInForeground 。

实现 LiveActivity (灵动岛) 功能

最近更新时间：2025-04-07 16:57:52

限制说明

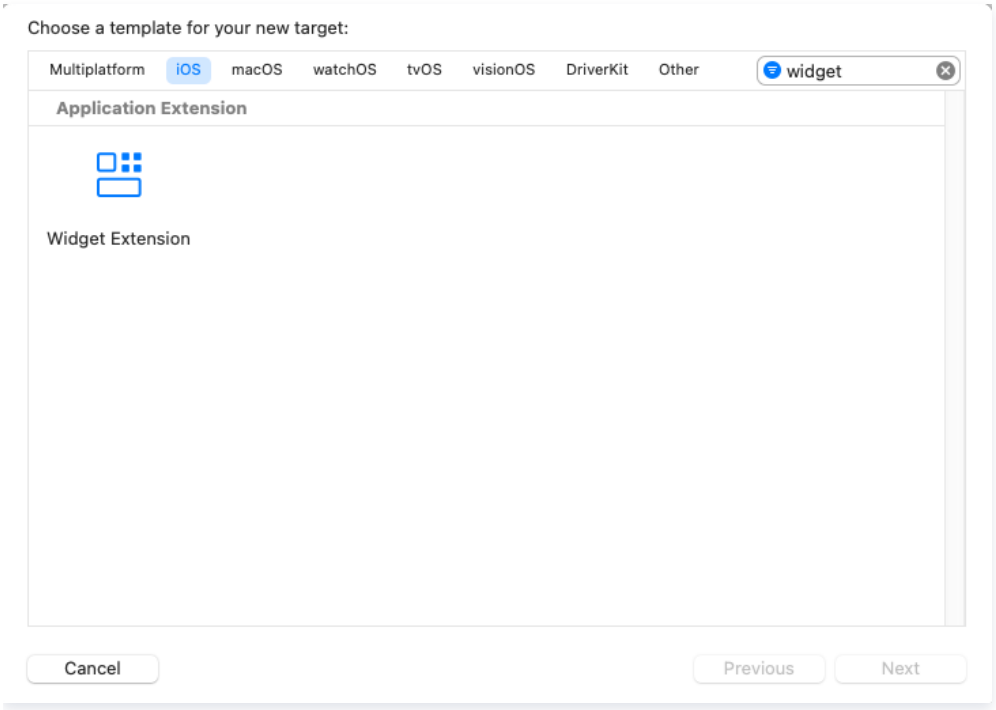
- iOS 16.1 及以上版本支持。
- 灵动上岛最多保留八小时，在锁定屏幕上最多保留十二小时。
- 远端操作需要用户配置 p8 证书。
- 设备只支持 iPhone，并且是有“药丸屏”的 iPhone14Pro 和 14Pro Max 上。

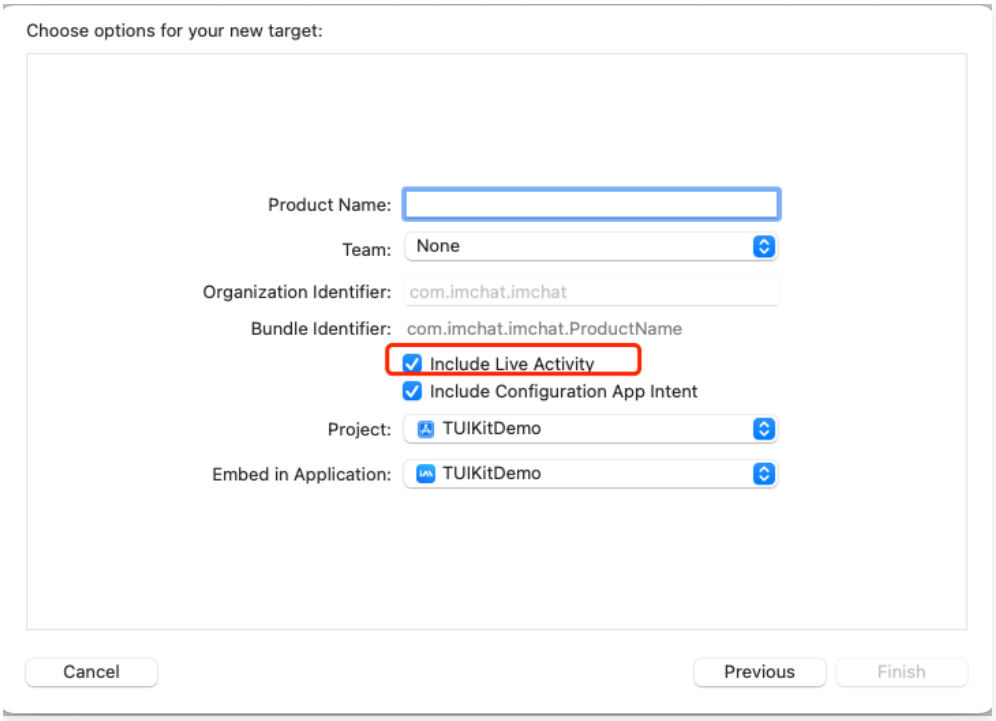
接入指引

步骤1：增加 Live Activities 支持配置

需要在主程序的 Info.plist 中添加键值：Supports Live Activities 为 YES。

步骤2：创建 WidgetExtension，如果项目中已经存在，则跳过该步骤。





步骤3：代码实现

1. 实现 Activity Attributes

以下实现需要按照自己的业务实现对应的数据模型，其中 ContentState 里为可以动态更新的数据，activityID 建议定义为 LiveActivity 的唯一标识 ID。

```
import Foundation
import ActivityKit

struct LiveActivityAttributes: ActivityAttributes {
    public struct ContentState: Codable, Hashable {
        // Dynamic stateful properties about your activity go here!
        var text: String
        var pauseTime: Date?
        var endTime: Date?
    }

    // Fixed non-changing properties about your activity go here!
    // custom activityID when creating a LiveActivity
    var activityID: String
}
```

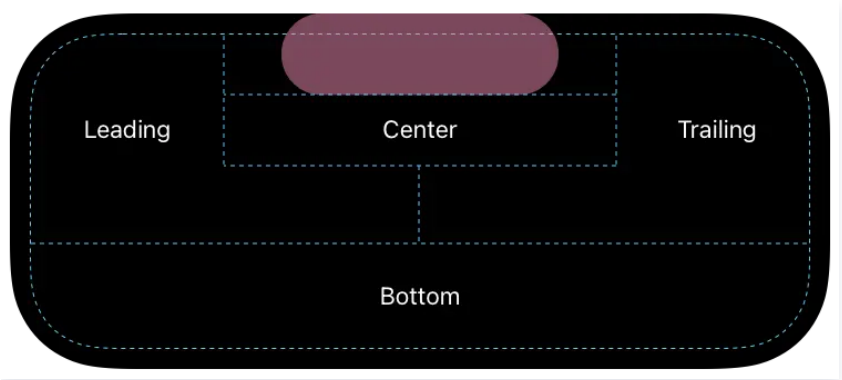
2. 创建 UI

具体 UI 请按照自己的业务编写，包含了锁屏 UI 和灵动岛 UI 的定义：

```
import ActivityKit
import WidgetKit
import SwiftUI

struct LiveActivityLiveActivity: Widget {
    var body: some WidgetConfiguration {
        ActivityConfiguration(for: LiveActivityAttributes.self) { context in
            // Lock screen/banner UI goes here
```

```
            VStack {
                Text("Hello \(context.state.text)")
            }
            .activityBackgroundTint(Color.cyan)
            .activitySystemActionForegroundColor(Color.black)

        } dynamicIsland: { context in
            DynamicIsland {
                // Expanded UI goes here.  Compose the expanded UI through
                // various regions, like leading/trailing/center/bottom
                DynamicIslandExpandedRegion(.leading) {
                    Text("Leading \(context.attributes.activityID)\(context.state.text)")
                }
                DynamicIslandExpandedRegion(.trailing) {
                    Text("Trailing \(context.state.text)")
                }
                DynamicIslandExpandedRegion(.center) {
                    Text("Center \(context.state.text)")
                }
                DynamicIslandExpandedRegion(.bottom) {
                    Text("Bottom \(context.state.text)")
                    // more content
                }
            } compactLeading: {
                Text("CL \(context.state.text)")
            } compactTrailing: {
                Text("CT \(context.state.text)")
            } minimal: {
                Text("CB \(context.state.text)")
            }
            .widgetURL(URL(string: "https://cloud.tencent.com/document/product/269/100621"))
            .keylineTint(Color.red)
        }
    }
}
```

灵动岛 UI 布局定义如下：



3. 客户端操作，添加启动、更新和停止逻辑

```
// start
let activity = try Activity.request(
            attributes: adventure,
            content: .init(state: initialState, staleDate: nil),
            pushType: .token
```

```
        )
// update
await activity.update(
    ActivityContent<AdventureAttributes.ContentState>(
        state: contentState,
        staleDate: Date.now + 15,
        relevanceScore: alert ? 100 : 50
    ),
    alertConfiguration: alertConfig
)
// end
await activity.end(ActivityContent(state: finalContent, staleDate: nil), dismissalPolicy:
dismissalPolicy)
```

4. 远端上报配置和更新操作

- **监听 token 更新和上报**

```
Task {
    for await pushToken in activity.pushTokenUpdates {
        let pushTokenString = pushToken.hexadecimalString
        Logger().debug("New push token: \(pushTokenString)")

        try await self.setLiveActivity(activityID:activity.attributes.activityID,
pushToken: pushToken)
    }
}


func setLiveActivity(activityID: String, pushToken: Data) async throws {
    var _apnsConfig = ImSDK_Plus.V2TIMLiveActivityConfig()
    _apnsConfig.businessID = xxxx
    _apnsConfig.token = pushToken
    _apnsConfig.activityID = activityID
    os_log("%@", type: .debug, "setLiveActivity activityID: \(activityID)\ntoken:\
(pushToken.hexadecimalString)")
        ImSDK_Plus.V2TIMManager.sharedInstance().setLiveActivity(_apnsConfig, succ: {
        print("setLiveActivity succ")
    }, fail: {code, desc in
        print("setLiveActivity fail, \(code), \(desc)")
    })
}
```

- **上报清理 LiveActivity**

```
func clearActivity(activityID: String) async throws {
    os_log("clearActivity ID: \(activityID)")
    ImSDK_Plus.V2TIMManager.sharedInstance().setLiveActivity(nil, succ: {
        print("clearActivity succ")
    }, fail: {code, desc in
        print("clearActivity fail, \(code), \(desc)")
    })
}
```

- **通过服务端更新和停止 iOS 实时活动。**

> ⚠ **注意：**
> 1. LiveActivity 推送限制 p8 证书，必须配置 P8 证书。
> 2. 如果接收方没有上报对应 activityID，将自动降级为普通通知推送。

ApnsInfo.LiveActivity 对应的 JSON Object格式说明

字段名称	类型	选项	字段说明
LaId	string	必填	需要推送的实时活动标识，对应客户端 activityID。 长度不超过64字节。
Event	string	必填	更新：“update”，结束："end"。
ContentState	JSON Object	必填	• 自定义的 key：value 的 object。需与客户端 SDK 值匹配。 • 对应 apns 官方的文档：Starting and updating Live Activities with ActivityKit push notifications | Apple Developer Documentation
DismissalDate	Integer	选填	event 为 end 时，锁屏实时活动结束展示的 uinx 时间。 不填默认为当前时间，锁屏会马上结束。

请求示例：

示例中省略或未列出参数，请参考文档：单发单聊，批发单聊，offlinePushInfo 相关。

```
{
  "MsgBody": [...] // 这里同 MsgBody 相关描述
  "OfflinePushInfo": {
      "PushFlag": 0,
      "Title": "离线推送标题",
      "Desc": "离线推送内容",
      "Ext": "{\"entity\":{\"k1\":\"v1\",\"k2\":\"v2\"}}",   // 透传字段，推送使用json格式字符串
      "ApnsInfo": {
          "LiveActivity": {
              "LaId": "timpush",
              "Event": "update", // update LA
              "ContentState": {
                  "k1": v1,
                  "k2": v2,
                  ...
               }
           }
        },
        "AndroidInfo": {
            ... // AndroidInfo相关参考官网文档
        }
  }
}
```

```
{
  "MsgBody": [...] // 这里同 MsgBody 相关描述
  "OfflinePushInfo": {
      "PushFlag": 0,
      "Title": "离线推送标题",
      "Desc": "离线推送内容",
      "Ext": "{\"entity\":{\"k1\":\"v1\",\"k2\":\"v2\"}}",   // 透传字段，推送使用json格式字符串
      "ApnsInfo": {
```

```
            "LiveActivity": {
                "LaId": "timpush",
                "Event": "end", // end LA
                "ContentState": {
                    "k1": v1,
                    "k2": v2,
                    ...
                 },
                "DismissalDate": 1739502750
             }
        },
        "AndroidInfo": {
            ... // AndroidInfo相关参考官网文档
        }
   }
}
```

更新日志
Android

最近更新时间：2025-06-05 10:52:32

8.6.7019 @2025.05.29

- 支持推送消息适配设备系统语言。
- 支持魅族消息分类。
- 优化申请通知权限逻辑。
- 解决部分设备懒加载 provider 导致初始化失败问题。
- 解决 token 获取可能存在的线程并发问题。
- 优化 C 端 OPPO 强提醒默认值。

8.5.6864 @2025.03.31

- 支持指定设备 ID 推送能力。
- 支持海外 FCM 通道自动适配功能。
- 优化多进程代码逻辑。
- 升级 OPPO 推送包。
- 优化混合模式下反注册逻辑。
- 优化数据库文件损坏异常问题。
- 优化 Push 错误码。

8.4.6667 @2025.01.16

- 优化反注册推送逻辑。
- 支持 Honor 离线消息分类。
- 增加 Push 消息离线存储功能。
- 优化漫游消息离线处理逻辑。

8.3.6498 @2024.11.27

- 支持 OPPO 推送消息分类新规。
- 增加通知栏点击事件监听方法 onNotificationClicked。
- 支持前台在线通道自定义铃音。
- 解决数据库并发问题。
- 解决登录并发问题。
- 解决前台推送收到推送直接跳转界面问题。
- 解决小米多次调用失败问题，升级小米推送包。
- 优化 registerPush 逻辑支持登录后带 appKey 参数调用。
- 华为机型识别优化。
- 混用模式默认关闭前台推送。

8.2.6325 @2024.09.29

- 增加不落地推送消息特性支持功能。
- FCM 支持点击通知栏自定义跳转功能。
- 优化注册推送前的日志打印功能。

8.1.6906 @2024.09.06

- 解决 Push 用户登录类型错误问题。
- 优化 FCM data 空消息弹窗问题。
- 优化全平台类型设置逻辑。
- 增加 Push 全局回调接口。

8.1.6902 @2024.08.20

- 新增前台推送配置开关。
- 新增 Android 13 以上通知权限申请逻辑。
- 优化小图标逻辑，默认使用应用图标。

8.1.6107 @2024.08.06

- 新增免登录推送功能。
- 智能探测可用通道策略默认关闭，增加配置开关。
- 解决自动探测任务轮转问题。
- 优化混淆设置。

8.0.6897 @2024.07.15

- 新增智能探测可用通道策略。
- 新增推送注册超时保护机制。
- 解决小图标设置问题。
- 解决跳转选项配置为首页拉不起应用问题。
- 优化机型识别逻辑。
- 提升代码稳定性优化。

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。

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。

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。

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。

7.6.5011 @2023.11.13

- 发布推送插件。

iOS

最近更新时间：2025-06-05 10:52:32

8.6.7019 @2025.05.29

- 支持推送消息适配设备系统语言。
- 优化 addPushListener 线程安全逻辑。

8.5.6864 @2025.03.31

- 支持指定设备 ID 推送能力。
- 支持 iOS LiveActivity 远端推送功能。
- 优化 iOS 前后台状态并发问题。
- 优化混合模式下反注册逻辑。
- 优化 iOS 关闭通知栏回调问题。
- 优化 Push 错误码。

8.4.6667 @2025.01.16

- 优化反注册推送逻辑。
- 支持 iOS 15+ 特性 Interruption level。
- 支持 iOS 7+ 特性 Background Remote Notification。
- 增加 Push 消息离线存储功能。
- 优化漫游消息离线处理逻辑。

8.3.6498 @2024.11.27

- 增加通知栏点击事件监听方法 onNotificationClicked。
- 支持前台在线通道自定义铃音。
- 解决登录并发问题。
- iOS 接口适配 Swift。
- 优化 registerPush 逻辑支持登录后带 appKey 参数调用。
- 混用模式默认关闭前台推送。

8.2.6325 @2024.09.29

- 增加不落地推送消息特性支持功能。
- 解决 iOS Donut 平台冷启动点击通知栏不回调的问题。

8.1.6906 @2024.09.06

- 解决 Push 用户登录类型错误问题。
- 解决 APNs 因为代理失败收不到推送问题。
- 优化 APNs 离线透传消息 Ext 为空不回调点击事件问题。
- 解决 APNs 前台状态解析通知异常回调问题。
- 优化全平台类型设置逻辑。
- 增加 Push 全局回调接口。

8.1.6902 @2024.08.20

- 新增前台推送配置开关。

8.1.6107 @2024.08.06

- 新增免登录推送功能。

8.0.6897 @2024.07.15

- 提升代码稳定性优化。

7.9.5668 @2024.04.09

- 增加 OfflinePushExtInfo 支持透传推送特性。
- UniApp 支持触达上报，支持杀进程回调点击事件。

7.8.5483 @2024.02.02

- iOS 支持最低版本更新为 10.0及以上。

7.7.5283 @2023.12.28

- 解决 iOS 托管 UNUserNotificationCenterDelegate 后，部分函数没有回调的问题。

7.7.5282 @2023.12.18

- 优化 registerPush 手动调用接口逻辑。

7.6.5011 @2023.11.13

- 发布推送插件。

Flutter

最近更新时间：2025-06-05 10:52:33

8.6.7019 @2025.05.29

- 支持推送消息适配设备系统语言。
- 支持魅族消息分类。
- 优化申请通知权限逻辑。
- 解决部分设备懒加载 provider 导致初始化失败问题。
- 解决 token 获取可能存在的线程并发问题。
- 优化 C 端 OPPO 强提醒默认值。
- 优化 iOS addPushListener 线程安全逻辑。
- 解决推送注册提示未登录问题。
- 解决 iOS 杀进程后点击回调两次问题。

8.5.6864 @2025.03.31

- 支持指定设备 ID 推送能力。
- 支持海外 FCM 通道自动适配功能。
- 优化混合模式下反注册逻辑。
- 优化 iOS 前后台状态并发问题。
- 优化 iOS 关闭通知栏回调问题。
- 优化 Push 错误码。

8.4.6667+3 @2025.02.24

- 优化消息接收逻辑，解决可能引发 OOM 问题。

8.4.6667 @2025.01.16

- 优化反注册推送逻辑。
- 支持 honor 离线消息分类。
- 增加 Push 消息离线存储功能。
- 优化漫游消息离线处理逻辑。

8.3.6498+2 @2024.12.27

- 优化 FCM VOIP 逻辑。
- 优化 registerPush 接口不强依赖堆栈校验逻辑。

8.3.6498+1 @2024.11.27

- 支持 OPPO 推送消息分类新规。
- 增加通知栏点击事件监听方法 onNotificationClicked。
- 支持前台在线通道自定义铃音。
- 解决数据库并发问题。
- 解决登录并发问题。
- 解决前台推送收到推送直接跳转界面问题。
- 解决小米多次调用失败问题，升级小米推送包。
- 优化 registerPush 逻辑支持登录后带 appKey 参数调用。
- 华为机型识别优化。
- 混用模式默认关闭前台推送。

- 废弃部分接口，建议您停止使用并切换其他接口。
- 废弃 iOS 推送证书 ID 设置字段 offlinePushCertificateID，请使用 businessID。

8.2.6325 @2024.09.29

- 增加不落地推送消息特性支持功能。
- FCM 支持点击通知栏自定义跳转功能。
- 优化注册推送前的日志打印功能。
- 解决 iOS Donut 平台冷启动点击通知栏不回调的问题。

8.1.6907 @2024.09.06

- 解决 Push 用户登录类型错误问题。
- 解决 APNs 因为代理失败收不到推送问题。
- 优化 APNs 离线透传消息 Ext 为空不回调点击事件问题。
- 解决 APNs 前台状态解析通知异常回调问题。
- 优化 FCM data 空消息弹窗问题。
- 优化全平台类型设置逻辑。
- 增加 Push 全局回调接口。

8.1.6107 @2024.08.06

- 新增免登录推送功能。
- 智能探测可用通道策略默认关闭，增加配置开关。
- 解决自动探测任务轮转问题。
- 优化混淆设置。

React Native

最近更新时间：2025-01-17 10:42:22

1.0.0 @2024-12-4

- 新增接口 `disablePostNotificationInForeground`，应用在前台时，开/关通知栏通知（默认开）。
- 新增接口 `createNotificationChannel`，支持 OPPO/FCM 设置自定义通知 Channel。
- 新增通知栏点击事件 `Push.EVENT.NOTIFICATION_CLICKED`。
- 优化接入体验。
- 支持自定义铃音功能。

0.3.1 @2024-10-15

- 更新 README 中 android 接入流程。

0.2.0 @2024-10-11

- 新增接口 `addPushListener/removePushListener`，支持获取在线推送消息，支持推送消息撤回通知。

0.1.0 @2024-09-10

- For React Native，基于腾讯云推送服务（Push），支持 iOS 和 Android 推送，同时适配各大厂商推送。

uni-app

最近更新时间：2025-01-17 10:42:22

1.1.0 @2024-12-11

- 大幅减小插件包体积，优化产品体验。
- 兼容 HBuilderX 4.36 的 Breaking changes。如果您需要 vivo/荣耀 的厂商推送，请参考 文档，正确配置 `manifestPlaceholders.json` 和 `mcs-services.json`。

1.0.0 @2024-11-29

- 优化和 TencentCloud-TUICallKit 插件 融合时的产品体验。
- 新增点击通知栏事件 NOTIFICATION_CLICKED，支持获取推送扩展信息。
- 在线通道支持自定义铃音功能。

0.5.1 @2024-11-07

- 优化和 @tencentcloud/chat-uikit-uniapp 融合时的产品体验。
- 优化和 TencentCloud-TUICallKit 插件 融合时的产品体验。
- 新增接口 disablePostNotificationInForeground，此接口可实现应用在前台时，开/关通知栏通知（默认开）。
- 新增接口 createNotificationChannel，支持 FCM/OPPO 自定义铃音。

0.4.0 @2024-10-17

- 支持与 TencentCloud-TUICallKit 插件 融合打包。

0.3.0 @2024-10-12

- 新增接口 addPushListener/removePushListener，支持获取在线推送消息，支持推送消息撤回通知。

0.2.0 @2024-09-18

- 支持 FCM
- 支持 hihonor

0.1.0 @2024-09-10

- 使用 uts 开发，基于腾讯云推送服务（Push），支持 iOS 和 Android 推送，同时适配各大厂商推送。

微信小程序多端框架

最近更新时间：2025-06-12 11:53:32

8.6.1 @2025.06.09

- 支持自定义应用角标（仅 iOS、华为）。

8.5.1 @2025.03.31

- 升级 OPPO 推送包。
- 优化数据库文件损坏异常问题。

8.4.1 @2025.01.16

- 优化反注册推送逻辑。
- 优化漫游消息离线处理逻辑。

8.3.1 @2024.12.02

- 新增在线消息监听。
- 支持开关在线通知栏通知。
- 支持创建客户端通知 Channel，支持自定义铃音。
- 修复多次调用 JSAPI 接口出错的问题。
- 通知栏点击事件名称由 ON_NOTIFICATION_CLICKED 修改为 NOTIFICATION_CLICKED。

8.2.3 @2024.10.14

Android：

- 修复荣耀、vivo 厂商配置问题。

8.2.2 @2024.10.14

- 支持 JSAPI @tencentcloud/donut-push。

错误码

最近更新时间：2025-04-11 17:41:22

服务端错误码

说明：

- 除非发生网络错误（例如 502 错误），否则推送服务 REST API 的 HTTP 返回码均为200。
- 请通过应答包体中的 ErrorCode、ErrorInfo 确认错误码，并对照下表查看。
- 公共错误码（60000 到 79999）请参见 错误码 文档。

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。
90005	JSON 格式请求包体中缺少 MsgRandom 字段或者 MsgRandom 字段不是 Integer 类型。
90009	请求需要 App 管理员权限。
90018	请求的账号数量超过限制。
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。
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/单推推送功能。
90047	推送次数超过当天限额（默认为100次）。
90049	撤回 TaskId 不合法，无推送记录。通过 发起全员/标签推送 或 单发推送 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。
90056	全员推送的请求体过大，目前支持最大10K长度。
91000	服务内部错误，请重试。

客户端错误码

说明：

更多客户端错误码请参见 错误码 文档。

错误码	含义说明
800001	注册推送服务，appKey 参数不合法。
800002	注册推送服务，sdkAppId 参数不合法。
800003	初始化 SDK 失败。
800004	长链接建立失败。
800005	本机通道注册推送失败。
800006	本机通道注册推送失败后，尝试 fcm 通道注册失败。
800007	探测所有通道失败。
800008	注册推送服务超时。
800009	注册推送 token 为空。
800010	给 SDK 设置 token 失败。
800011	关闭自动注册失败。
800012	关闭推送权限请求失败。
800013	创建通知 channel 失败。
800014	设置推送配置失败。
800015	实验性接口调用失败。
800016	用户拒绝推送获取弹窗权限。
6014	IM 账号未登录，请先登录后再注册推送。
9523	请检查网络是否正常。
70016	公钥不存在，UserSig 验证失败，请确认是否使用了正确的 appKey。

厂商错误码

说明：

- 此处仅列出各厂商服务端常见错误码作为参考，如有疑问，请自行咨询对应厂商。
- 各厂商推送服务错误码文档链接：小米 / 华为 / OPPO / vivo / 魅族 / 荣耀 / Google FCM / APNs 。

小米

错误码	含义说明
-1	未知错误。
0	成功。
1	内部错误。
10001	系统错误。
10002	系统繁忙。
10003	远程服务错误。
10008	参数错误，请参考 API 文档。
10012	非法请求。
10016	缺失必选参数。
10017	参数值非法。
10027	应用的 API 调用太频繁。
10029	不合法的设备。 发起 app 相关操作时，例如注册、取消注册时，找不到对应的设备。
10030	应用获取失效 regid 太频繁。
10031	应用在黑名单中，禁止发送消息。
10032	应用获取失效 alias 太频繁。
10033	应用在黑名单中，禁止发 feedback 请求。
10034	应用当日发送消息数量过多。
10035	没有更多的失效 alias 可供拉取。
10036	应用操作被禁止。
10037	请求过期。
10038	应用禁止访问统计和 trace 数据。
10039	应用获取消息统计和 trace 太过频繁。
20607	DB 错误。
20209	不合法的主题。
20215	订阅主题失败。
20216	退订主题失败。
20301	发送消息失败。
20315	删除广播消息失败。
21301	认证失败。

21302	token 认证失败。
21303	被限制的请求。
21305	缺少必要的参数。
22000	非法应用。
22006	应用程序 ID 不合法。
22007	应用程序 Key 不合法。
22022	应用程序 package name 不合法。
22102	发送应用通知消息失败。
26003	消息 Push 内部调用失败。
26004	广播消息发送太频繁。
26005	account 设置太频繁而被禁止。
26006	发送需要审核，请及时审核。
27001	channel 相关信息不匹配。
65003	未找到 device，通常是设备不在线导致。
65006	别名设置太频繁而被禁止。
65007	别名在黑名单中。
65008	别名长度太长。
65009	消息内容太长。
65010	主题长度太长。
65011	未提供参数。
65012	别名为空。
65013	主题为空。
65014	键值对数目太多。
65015	键值对总长度太长。
65016	User account 为空。
65017	User account 长度太长。
65028	未查询到相应消息。
65029	远程服务异常。
65030	构建 JSON 时出现异常。
65035	主题数量太多。
65036	Callback 参数太长。
65037	Callback url 太长。
65038	Callback url 非法。

65040	注销失败。
65041	ImeiMd5 为空。
65042	ImeiMd5 不合法。
66006	注册失败。这里指安卓设备注册异常，包括设备注册和应用注册。
66007	regId 非法，指 regId 格式不正确。
66008	不合法的请求。
66108	发送的参数信息不合法。
66109	regSecret 无效。
66300	未知的命令。
66301	执行命令出错。
66303	小米 ID 为空。
66304	小米 ID 太长。
66305	不合法的小米 ID。 小米 ID 的格式不正确。
66306	关闭或打开 push 失败。
66307	删除定时任务 job 时传入的消息 ID 无效。
66308	打分服务请求参数违法。
70011	输入为空。
70012	JobKey 格式不合法。
200001	推送数量超过当日限额。 限制说明请参见 每日推送数量限制 。
200002	推送 QPS 超过限额。 限制说明请参见 推送速率 QPS 限制 。

华为

错误码	含义说明
-5	获取 Token 任务失败。
502	请求连接异常，常见于网络状况不稳定。
503	流量控制。
6003	指纹证书配置不一致。
6004	接口鉴权时，权限不存在。
80100003	消息结构体错误。
80100016	消息体中带有敏感词汇。

80300002	下发消息给指定的用户（Token）报无权限。
80300007	指定的 Token 无效。
907122036	没有开通推送权益。
907122046	Push 不可服务。
907122047	通用错误码。
907122054	Push SDK 自动初始化中，请稍后重试操作。
907122069	不支持子用户操作。
907135000	传入的参数错误。
907135003	SDK 连接 HMS Core（APK）失败。
907135700	调用网关查询应用 scope 失败。
907135701	OpenGW 没有配置 Scope 。

OPPO

错误码	含义说明
-2	服务器流量控制。
-1	服务不可用，此时请开发者稍候再试。
0	成功，表明接口调用成功。
11	不合法的 AuthToken。
12	HTTP 方法不正确。
13	应用调用次数超限，包含调用频率超限。
14	无效的 AppKey 参数。
15	缺少 AppKey 参数。
16	sign 校验不通过，无效签名。
17	缺少签名参数。
18	缺少时间戳参数。
19	非法的时间戳参数。
20	不存在的方法名。
21	缺少方法名参数。
22	缺少版本参数。
23	非法的版本参数，用户传入的版本号格式错误，必需为数字格式。
24	不支持的版本号，用户传入的版本号没有被提供。
25	编码错误，一般是用户做 http 请求的时候没有用 UTF-8 编码请求造成的。

26	IP 黑名单。
27	没有此功能的权限，拒绝访问。
28	应用不可用。
29	缺少 Auth Token 参数。
30	该应用没有 API 推送的权限。
31	数据不存在。
32	数据重复。
33	消息条数超过日限额。
34	上传图片超过日限额。
40	缺少必选参数，API 文档中设置为必选的参数是必传的，请仔细核对文档。
41	参数错误，一般是用户传入参数非法引起的，请仔细检查入参格式、范围是否一一对应。
51	图片无效，一般是图片格式、图片分辨率、图片大小不符合格式及图片未上传等，请仔细检查图片格式及上传文件方式。
55	应用访问频率限制。
59	无备用链接跳转权限，拒绝访问。
67	分类错误（包含强提醒所有异常）。

vivo

错误码	含义说明
0	请求成功。
10000	权限认证失败。
10006	别名长度超过 70 个字符。
10043	该应用已关闭 push 通道。
10045	应用审核中不可发送正式消息。
10050	alias 和 regId 不能都为空。
10051	classification 暂不支持该消息类型。
10054	notifyType 不合法。
10055	title 不能为空。
10056	title 长度不能超过40个字符。
10057	content 不能为空。
10058	content 长度不能超过100个字符。
10059	timeToLive 不合法。

10060	skipType 不合法。
10061	skipType = 2，skipContent 不能为空。
10062	skipType = 2，skipContent 不能超过2048个字符。
10063	skipType = 3，skipContent 不能为空。
10064	skipType = 3，skipContent 不能超过2048个字符。
10065	networkType 不合法。
10067	自定义 key 和 value 键值对不合法。
10068	skipType = 4，skipContent 不能为空。
10069	skipType = 4，skipContent 不能超过2048个字符。
10070	运营消息发送量总量超出限制。
10071	超出发送时间允许范围。
10072	推送速度过快，请稍后再试。
10073	系统消息发送量总量超出限制。
10082	系统消息开关未打开。
10092	profileId 长度超过限制。
10094	鉴权码与请求体对应的 AppID 不一致。
10095	notifyId 非法。
10096	category 错误。
10097	category 与 classification 不对应。
10103	推送内容含敏感信息。
10104	请发送正式消息。
10150	aliases 和 regIds 不能都为空。
10151	taskId 不能为空。
10152	taskId 不合法。
10153	regIds 个数不在指定范围[2-1000]。
10154	aliases 个数不在指定范围[2-1000]。
10155	消息不存在或已过期。
10200	AppID 不能为空。
10201	appKey 不能为空。
10202	appKey 不合法。
10203	timestamp 不能为空。
10204	sign 不能为空。
10205	AppID 不存在。

10206	sign 不正确。
10207	timestamp 不合法。
10252	批量发送消息体超出限制。
10255	全量推送接口未开放。
10301	alias 长度不能超过70个字符
10302	regId 不合法 。
10304	extra 包含不支持的 key。
10305	extra callback 长度超限。
10306	extra callback.param 长度超限。
10307	alias 不合法。
10311	该设备当前无法推送。
10352	requestId 不能为空。
10353	requestId 长度不能超过 64 个字符。
10471	taskIds 数量不能超过 100 个。
10472	taskIds 格式错误。
10473	taskIds 为空。
10600	name 参数不合法。
10601	标签名字不能为空。
10602	标签描述不能超过 300 字符。
10603	oldName 参数不能为空。
10604	newName 参数不能为空。
10605	oldName 参数不合法。
10606	newName 参数不合法。
10608	type 参数不能为空。
10609	ids 参数不能为空。
10610	标签名已经存在。
10611	标签分类名已经存在。
10612	标签组合名已经存在。
10613	group 参数不合法。
10614	ids 数量不能超过1000。
10615	tag 数量不能超过100。
10616	type 参数不合法。
10700	userids 不能为空。

10701	UserID 个数超过限制。
10703	超过上限，稍后再试。
10704	requestId 长度不能超过 64 个字符。
10706	userType 错误。
10800	registration_tokens 个数不在指定范围。
10801	notification 不能为空。
10802	original_source_name 不能为空。
10803	original_source_name 长度非法。
10804	original_source_ip 不能为空。
10806	click_action 非法。
10807	url 长度超过限制。
10808	intent 长度超过限制。

魅族

错误码	含义说明
200	正常。
500	其他异常。
1001	系统错误。
1003	服务器忙。
1005	参数错误，请参考 API 文档。
1006	签名认证失败。
110000	AppID 不合法。
110001	appKey 不合法。
110004	参数不能为空。
110009	应用被加入黑名单。
110010	应用推送速率过快。
110053	透传超过限制。
201	没有权限，服务器主动拒绝。
501	推送消息失败（db_error）。
513	推送消息失败。
519	推送消息失败服务过载。
520	消息折叠（1分钟内同一设备同一应用消息收到多次，默认5次）。

110002	pushId 失效(pushId 未订阅)。
110003	pushId 非法。
110005	alias 失效(alias 未订阅或者消息开关关闭)。
110010	pushId 失效(消息开关关闭)。

荣耀

错误码	含义说明
200	成功。
400	参数错误。
403	鉴权不通过。
404	找不到服务。
500	服务内部错误。
502	请求连接异常，常见于网络状况不稳定。
503	请求频繁，请重试。
80100000	部分Token发送成功，返回的 failTokens 为发送失败的 Token。
80100003	消息结构体错误。
80300006	Token 重复。
80300007	所有 Push Token 都是无效的。
80300008	消息体大小超过系统设置的默认值（4096 Bytes）。
80300010	每次推送 Push Token 数量不能超过 1000。
80200020	回执不匹配。
10001	timestamp 为空。
10207	timestamp 不合法。
10300	Token 为空。
10205	appId 为空。

Google FCM

错误码	含义说明
400	INVALID_ARGUMENT，可能的原因包括注册无效、软件包名称无效、消息过大、数据键无效、TTL 无效或其他无效参数。
404	UNREGISTERED， 出现此错误的原因可能是缺少注册令牌，或令牌未注册。

403	SENDER_ID_MISMATCH， 一个注册令牌与一组特定的发送者关联。当客户端应用注册 FCM 时，必须指定允许哪些发送者发送消息。在向客户端应用发送消息时，您应使用这些发送者 ID 之一。如果您切换为其他发送者，则现有的注册令牌将不起作用。
429	QUOTA_EXCEEDED， 此错误可能是由超出消息率配额、超出设备消息率配额或超出主题消息率配额导致的。
503	UNAVAILABLE，服务器无法及时处理请求。
500	INTERNAL，服务器在尝试处理请求时遇到错误。

APNs

错误码	含义说明
200	成功。
403	BadCertificate， 证书无效。
	BadCertificateEnvironment， 客户端证书环境错误。
	ExpiredProviderToken，提供商令牌已过时，应生成新的令牌。
	InvalidProviderToken，提供商令牌无效，或者无法验证令牌签名。
410	ExpiredToken， 设备令牌已过期。

常见问题

最近更新时间：2025-06-12 14:56:42

Android

收不到离线推送怎么排查?

1.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。

2.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?

1.点击跳转配置检查

详细参见 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。

2.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。

3.推送链路排查

使用 排查工具 查看推送全链路推送详情。

海外推送指南

1. 海外离线推送支持的厂商有 FCM、华为、OPPO 和 APNS。
2. FCM 通道必备充分条件是：设备可以访问海外网站、设备有安装支持 GMS 服务、成功集成 TIMPush。注意：国内设备大部分没有安装 GMS 服务，无法支持 FCM 推送。

FCM 推送问题

1. 如果想让优先走 FCM 通道，请使用接口 forceUseFCMPushChannel 。
2. FCM 推送消息可能会拉起应用进程，请不要在应用启动处，即 application 的 onCreate() 声明周期内做登录、数据上报等不适合高频并发的逻辑。

解决接入冲突问题

接入产生冲突的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 。

其他问题

支持向三星、中兴、传音、坚果、海信、索尼等手机推送吗?

在线推送支持所有机型，包括三星、中兴、传音、坚果、海信、索尼等。

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.** { *; }
```

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，调用 unRegisterPush 即可。

收不到推送，且后台报错 bad devicetoken。

Apple 的 deviceToken 与当前编译环境有关，请检查：

1. 接入流程中 配置推送参数 的 businessID 是否是对应环境的证书 ID。
2. 控制台创建证书检查：
   - 生产环境证书：上传的证书类型为 Apple Push Notification service SSL (Sandbox & Production)，并在Archive 出Release 包后进行测试。注意：不可在Xcode 测试。
   - 开发环境证书：您需要 Archive 出 release 包后进行测试，创建选择正确证书即可。

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

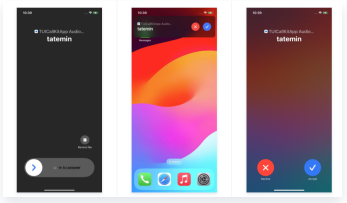
在 IM 控制台单击编辑对应证书，更新下 p12 证书即可。需要注意的是：请勿删除证书重新创建，重新创建证书 ID 会变化，需要发版本支持更新。

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 。

APNS、VOIP 和 LiveActivity 区别

特性	APNs	VoIP	LiveActivity
使用场景	发送通知提醒用户	音视频通话	显示实时更新的动态内容
生命周期	跟随通知栏	跟随通知栏	持续到事件结束
SDK 集成	集成 TIMPush	集成 CallKit 和 TUICallKitVoIPExtension	集成 ActivityKit 和 IMSDK
使用方式	申请 APNs 推送证书，端上上报 token，后台校验和触发	申请 VoIP 推送证书，端上上报 token，后台校验和触发	• 应用可主动更新 • 申请推送 P8 证书，端上上报 token，后台校验和触发
推送 token	设备级别	设备级别	与特定实时活动相对应，一台手机可同时创建多个实时活动
限制	通知内容有限	需要 VoIP 证书和 iOS Callkit	iOS 16 以上支持

实际效果			

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 。

关于 Push SDK 包体积

如果您要求集成包体积更低，并且希望单独使用 Push 推送服务不集成 IM SDK，您可以 点击进入交流群 直接咨询。

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。 • 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。 • 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。
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。