

iOS SDK 集成指南

使用提示

本文是 mPush iOS SDK 标准集成指南。如有问题，请咨询技术支持QQ群：[372650575](#)

- 如果您想要快速地测试、感受下mPush的效果，请参考 [3 分钟快速 Demo](#)。
 - 如果您看到本文档，但还未下载iOS SDK，请访问[SDK下载](#)页面下载。
 - 本SDK只在真机环境下有效！
-

产品功能说明

mPush 是一个端到端的推送服务，使得服务器端消息能够及时地推送到终端用户手机上，让开发者积极地保持与用户的连接，从而提高用户活跃度、提高应用的留存率。

本 iOS SDK 方便开发者基于 mPush 来快捷地为 iOS App 增加推送功能。

主要特点

- 客户端集成简单，无需对APNs进行相关操作。
 - 可定制展示的界面。方便在接收到指定类型的通知后，打开展示界面。
 - 可注册多个推送代理，分别处理回调信息。
-

集成指南

1. 开发前准备

[注册mPush账号](#)

[创建应用](#)

2. 配置推送证书

[配置推送证书](#)

3. 导入SDK至工程并配置

[下载SDK导入项目](#)

[配置 MPushAppID 引入必要的框架](#)

4. 编写代码

[编写代码](#)

5. 测试确认

[测试确认](#)

6. 特别说明

[特别说明](#)

3 分钟快速 Demo（IOS）

- [第一步：创建mPush开发者帐号](#)
- [第二步：Portal上创建应用](#)
- [第三步：配置推送证书](#)
- [第四步：下载应用Demo](#)
- [第五步：使用xCode 打开 Demo 工程](#)
- [第六步：运行 Demo 这个应用](#)
- [第七步：Portal上推送通知](#)
- [第八步：手机上收到通知](#)
- [第九步：在DebugArea中查看收到的消息](#)

本文目的在于，指导新接触mPush的开发者，在短短几分钟时间内把mPush跑起来：

- 配置推送证书
- 安装Demo客户端到手机
- 在Portal上推送通知
- 客户端收到推送并显示

如有问题，请咨询技术支持QQ群：372650575

第一步：创建mPush开发者帐号

要创建mPush开发者帐号，请访问 [mPush官方网站](#)。

第二步：Portal上创建应用

帐号创建成功并登录mPush后，首先显示的是创建应用的界面。填上你的应用名称即可。

APP应用应用详情

应用名称

应用名称

应用分类

应用

接入应用

第三步：配置推送证书

此部分内容请参考[iOS 推送证书配置指南](#)。

第四步：下载应用Demo

应用创建成功，即进入“应用详情”显示界面。这个界面右上角，你可以看到“下载 SDK”按钮。

点击“下载 SDK”后，进入 SDK 下载界面，选择iOS项下载iOS SDK，你将下载到一个 .zip 压缩文件。此压缩文件中包含以下内容：

lib文件夹：包含头文件 PushManager.h，静态库文件 libmPushSDK.a。
demo文件夹：（一个完整的iOS示例项目，演示mPush基本用法，可参考。）

第五步：使用xCode 打开 Demo 工程

使用xCode打开Demo目录下 `mPushDemo.xcodeproj` 工程文件，打开示例Demo。

第六步：运行 Demo 这个应用

上步骤打开这个 iOS 项目后，打开项目配置文件‘Your project name’-Info.plist，更换 MPushAppID 的值为你第二步创建应用所分配的 APP ID，选择与第三步您所设置的推送证书对应的 Provisioning Profile，设置无误后，连接您的iOS设备至电脑，在xCode中选择您的设备后，你就可以马上 **Run** 这个项目，把这个 iOS 应用跑起来了。

做下一步动作之前，请确保你的手机已连接网络。

第七步：Portal上推送通知

登录mPush Portal选择第二步你创建的应用，点击页面左侧 **新建通知**，打开新建通知页面，你就可以根据需要创建不同类型的通知了，创建完成后，点击 **确认推送** 按钮即可。

创建 iOS 通知

内容

填写推送内容

还能输入70字

☐ 静默推送

点击通知后的操作 ?

☒ 打开应用

☐ 打开应用中的某页面

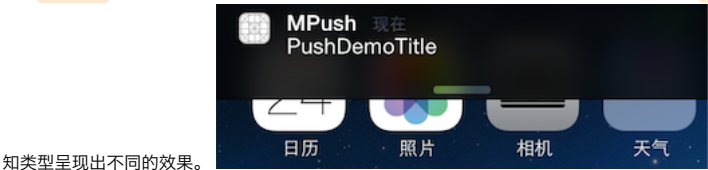
☐ 打开一个URL

☐ 自定义参数 ?

在上述步骤安装 mPush Demo 的手机上，你就可以收到推送的通知了。

第八步：手机上收到通知

点击 **确认推送** 按钮后，iOS 设备端即可收到推送得到的通知信息。如果在新建通知时定义了 **点击通知后的操作** 部分，客户端点击通知进入应用后将根据通



知类型呈现出不同的效果。

第九步：在DebugArea中查看收到的消息

```
2014-05-24 22:40:53.850 PushDemo[2399:60b] MPush-SDK-Log -> 最新消息获取成功.
2014-05-24 22:40:53.852 PushDemo[2399:60b] title : PushDemoTitle content :
PushDemoContent extention : {
}
```

“

更多详细使用方法，请参考[iOS SDK 集成指南](#)。

注册mPush账号

在使用mPush服务之前，需要前往mPush官方网站 <http://mpush.cn> 注册账号。填写好各项信息并确认无误后，点击注册即可：

注册新帐号

邮箱：	*	<input type="text" value="demo@email.com"/>	✓
密码：	*	<input type="password" value="....."/>	✓
确认密码：	*	<input type="password" value="....."/>	✓
您的姓名：	*	<input type="text" value="mPushDemo"/>	✓
公司/团队名称：	*	<input type="text" value="魔推"/>	✓
手机：	*	<input type="text" value="18888888888"/>	✓

☒ 我已阅读并接受 [《用户协议》](#)

注册新用户

注册成功后即自动跳转到应用管理控制后台。

创建应用

注册或登录成功后，需要创建新应用，以用来对接开发的产品：

APP应用应用详情

应用名称

mPushDemo

应用分类

应用

接入应用

创建成功后，页面会自动跳转至应用管理和设置页面。

“

建议：为了能够完全区分开发和生产环境，对于同一个应用来说，建议创建两个应用，一个为线上版本（生产环境），另一个为开发版本（开发环境）。这样能够很好的区分版本，对于开发测试能够完全和线上版本分离。但这并不是必须的。

配置推送证书

每一个iOS应用如果需要使用Push推送服务，都需要申请带有Push权限的证书。整个申请流程和mPush上的设置可参考：[iOS 推送证书配置指南](#)。

Apple推送区分开发证书和生产证书，开发者在申请的时候，需要注意。

“

小技巧：按照iOS推送证书配置指南最终拿到pem格式证书后，可选中pem证书文件，单击空格键，即可看到证书的基础信息。

- 推送开发证书样例：

```
Subject Name
  User ID commRocker.MPush
  Common Name Apple Development IOS Push Services: commRocker.MPush
  Organizational Unit 37M955KT24
  Country CN
```

- 推送生产证书样例：

```
Subject Name
  User ID commRocker.MPush
  Common Name Apple Production IOS Push Services: commRocker.MPush
  Organizational Unit 37M955KT24
  Country CN
```

“

从这些简单的信息中，可以确认证书对应的Bundle id（即上图中的User ID），Common Name可以看出当前证书的环境信息：Apple Development IOS Push Services 对应开发环境推送，Apple Production IOS Push Services 对应生产环境推送。

iOS 推送证书配置指南

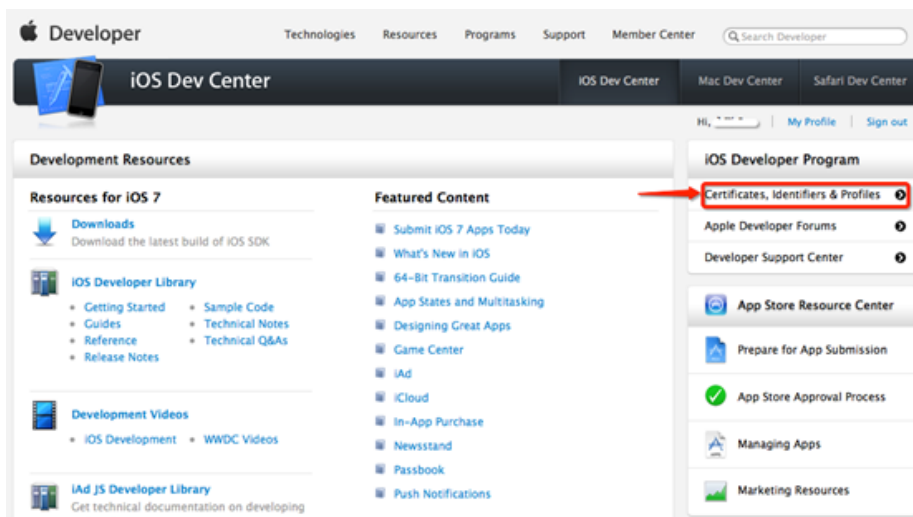
首先请确保您的Mac已经具备iOS应用开发环境以及苹果开发者账号。

- [第一步：创建应用 App ID](#)
- [第二步：配置开发者证书证书](#)
- [第三步：APNs p12证书导出](#)
- [第四步：APNs pem证书生成](#)
- [第五步：APNs pem证书上传](#)

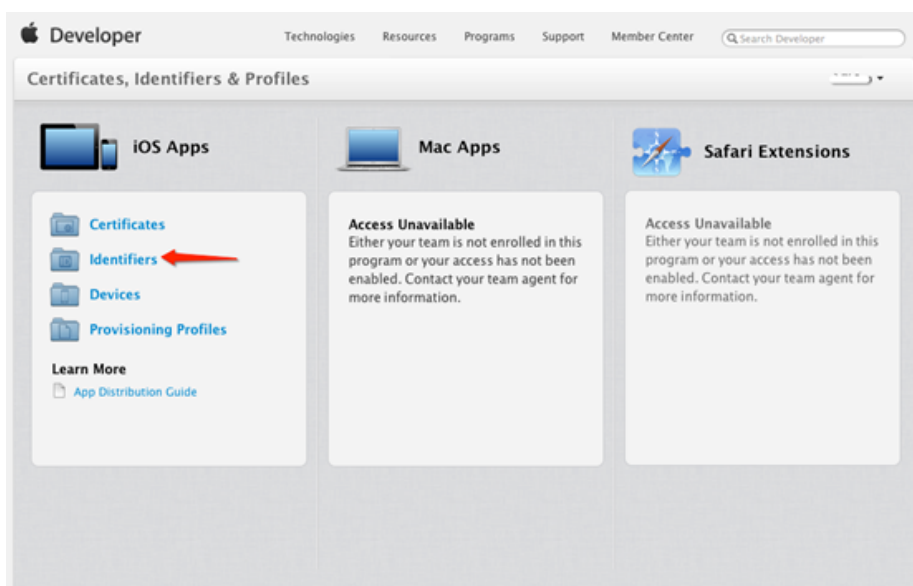
第一步：创建应用 App ID

如果已在苹果的开发者中心申请了具备Push Notification 的 APP ID，请直接跳转至[第二步：配置开发者证书证书](#)。

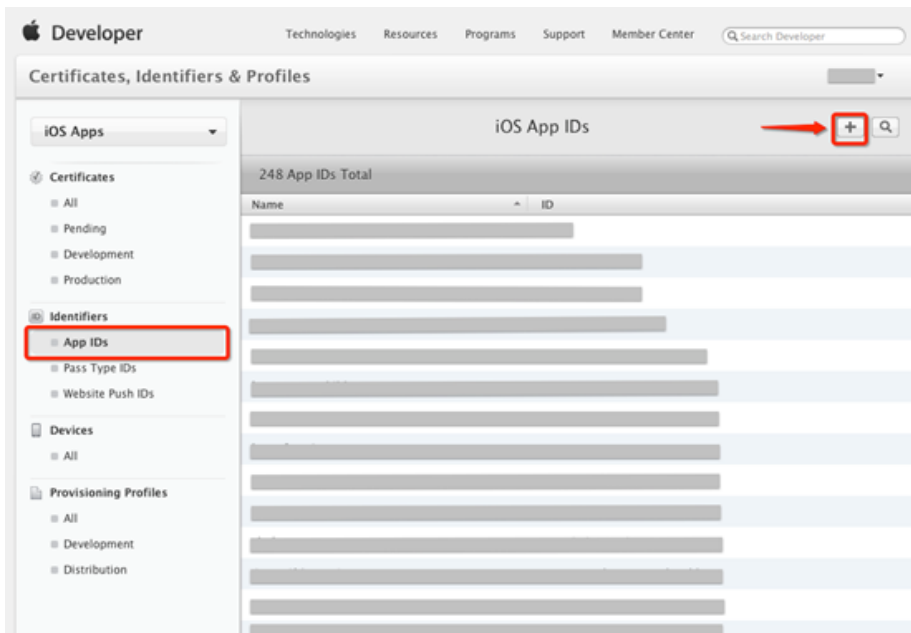
登录 [Apple iOS Dev Center](#)，点击如图所示位置进入证书配置页面。



打开配置页面后，点击左侧 **Identifiers** 菜单进入App ID 配置界面：



在App IDs内容页点击右上角 添加 按钮，如图：



进入APP ID 创建页面，选择 **Explicit App ID** 并填写 **Bundle ID**，特别注意要勾选 **Push Notifications** 选项，开启Push功能：

“

注意，这里的App ID必须是 **Explicit App ID**，否则无法使用APNs推送服务。

App ID Suffix

☒ **Explicit App ID**

If you plan to incorporate app services such as Game Center, In-App Purchase, Data Protection, and iCloud, or want a provisioning profile unique to a single app, you must register an explicit App ID for your app.

To create an explicit App ID, enter a unique string in the Bundle ID field. This string should match the Bundle ID of your app.

Bundle ID:

We recommend using a reverse-domain name style string (i.e., com.domainname.appname). It cannot contain an asterisk (*).

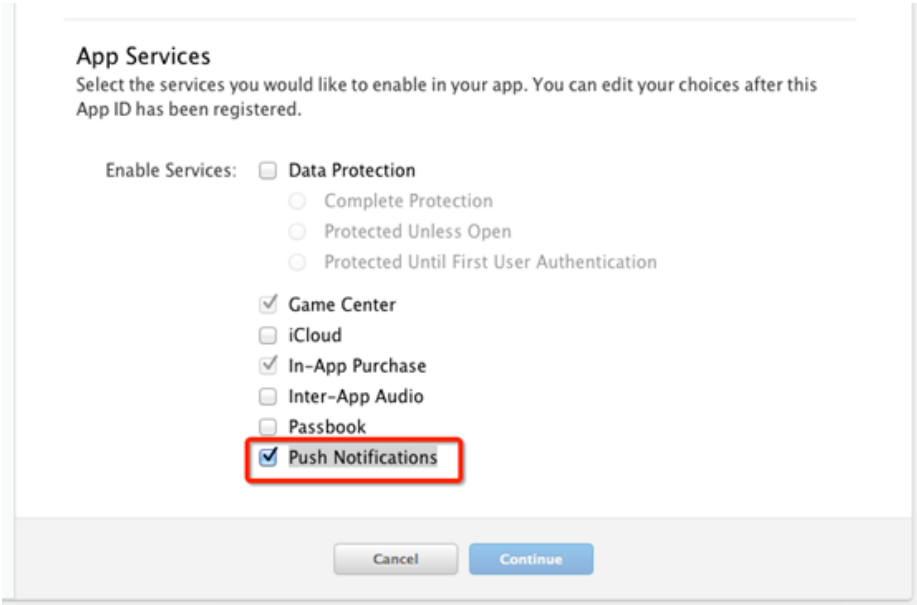
☐ **Wildcard App ID**

This allows you to use a single App ID to match multiple apps. To create a wildcard App ID, enter an asterisk (*) as the last digit in the Bundle ID field.

Bundle ID:

Example: com.domainname.*

填写 **Bundle ID**，格式必须和你项目中得Bundle ID一致。确认无误后，在页面的下方，即可看到 **Push Notification** 服务的选项变为可选状态：



App Services
Select the services you would like to enable in your app. You can edit your choices after this App ID has been registered.

Enable Services:

- ☐ Data Protection
 - ☐ Complete Protection
 - ☐ Protected Unless Open
 - ☐ Protected Until First User Authentication
- ☒ Game Center
- ☐ iCloud
- ☒ In-App Purchase
- ☐ Inter-App Audio
- ☐ Passbook
- ☒ Push Notifications

[Cancel](#) [Continue](#)

勾选 **Push Notification** 后，点击 **continue** 进入信息确认页面：



ID **Confirm your App ID.**

To complete the registration of this App ID, make sure your App ID information is correct, and click the submit button.

App ID Description:

Identifier:

Data Protection: ☐ Disabled

Game Center: ☒ **Enabled**

iCloud: ☐ Disabled

In-App Purchase: ☒ **Enabled**

Inter-App Audio: ☐ Disabled

Passbook: ☐ Disabled

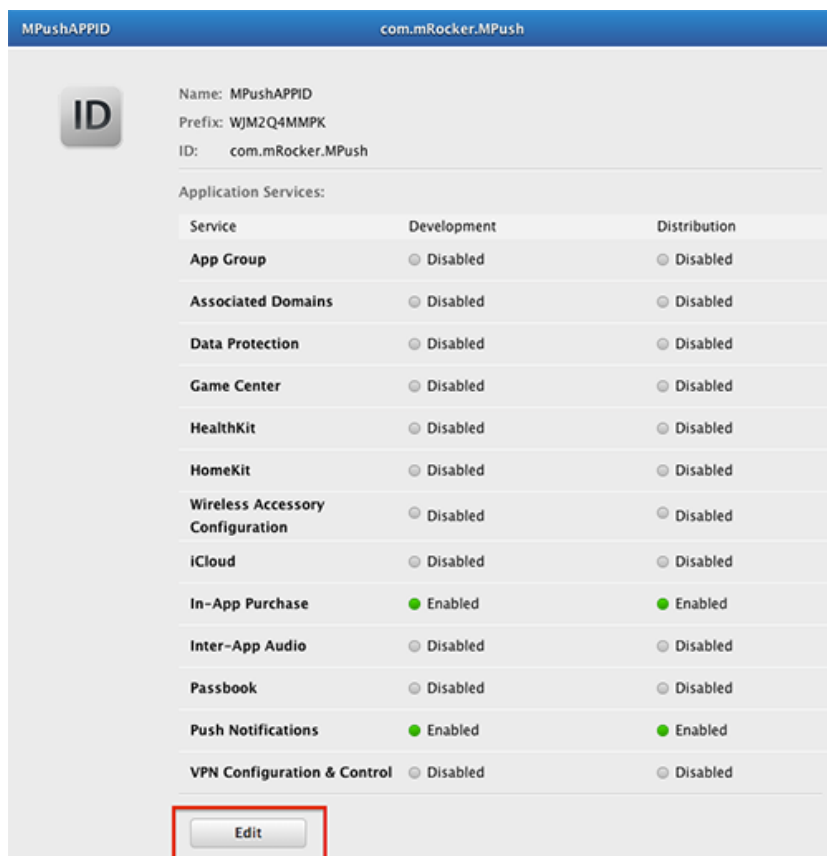
Push Notifications: ☒ **Enabled**

[Cancel](#) [Back](#) [Submit](#)

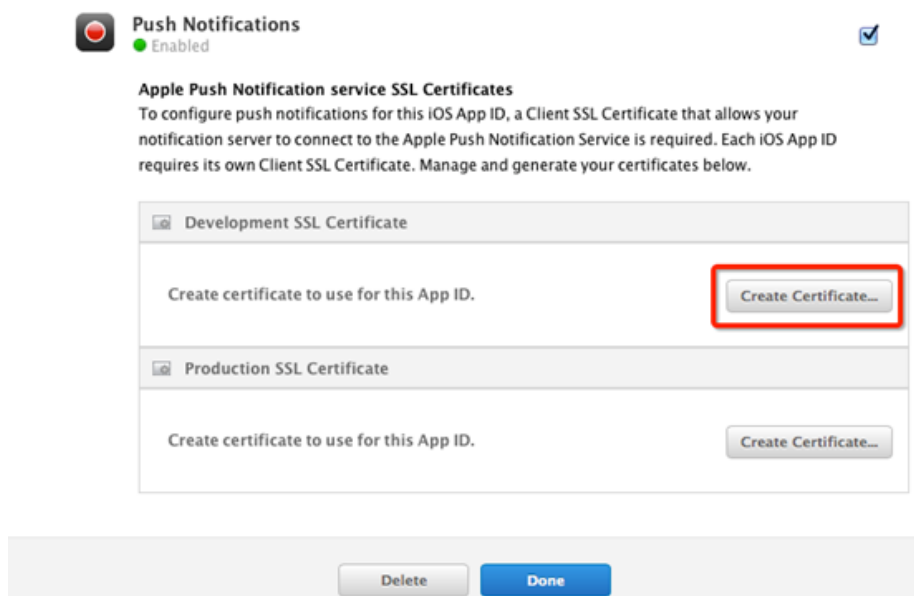
确认信息无误后点击 **submit** 按钮提交，完成App ID创建。

第二步：配置证书

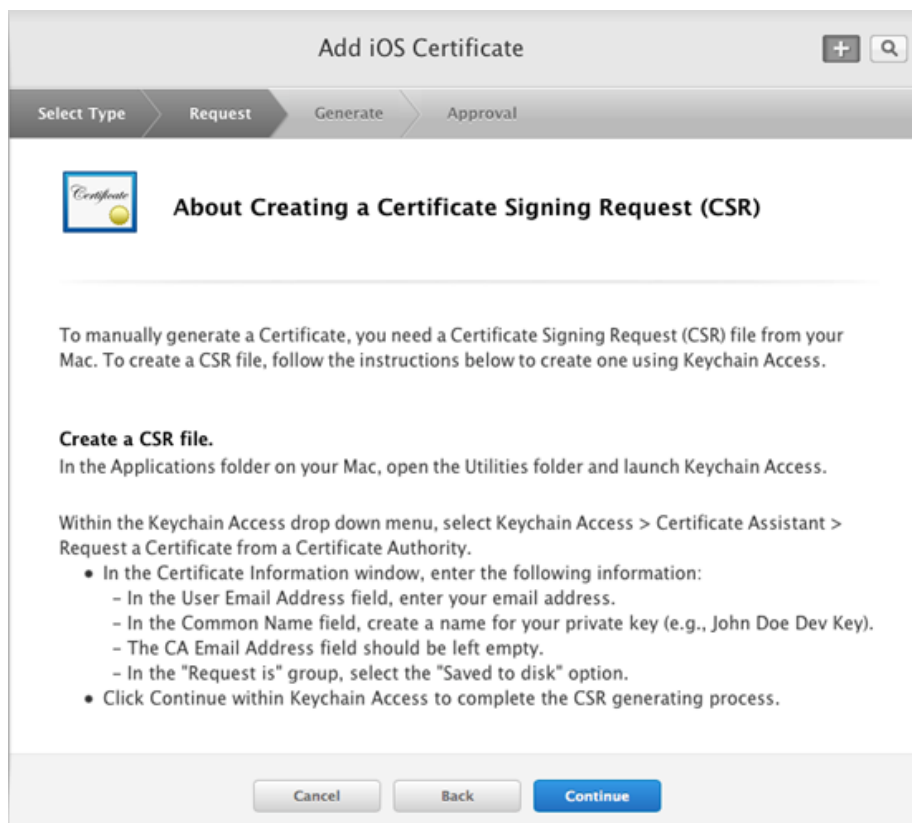
从 **App IDs** 列表中找到刚才创建的App ID，选中展开，点击 **Edit** 进入配置编辑页面：



在编辑页面下方可以看到 **Service** 项里有 **Push Notifications**内容项，可以创建 **开发证书** 和 **发布证书**，这里我们以开发证书为例，点击如图进入创建页面：



点击创建按钮进入，此页面告诉我们如何 **创建CSR文件**



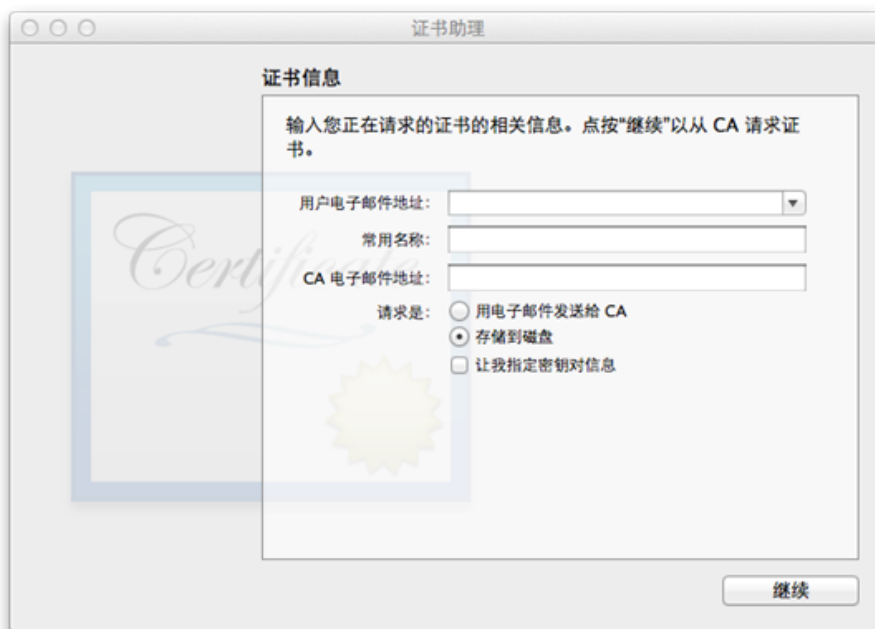
根据以上界面提示创建CSR文件，在应用程序里的 **其他** 中找到 **钥匙串访问**，打开：



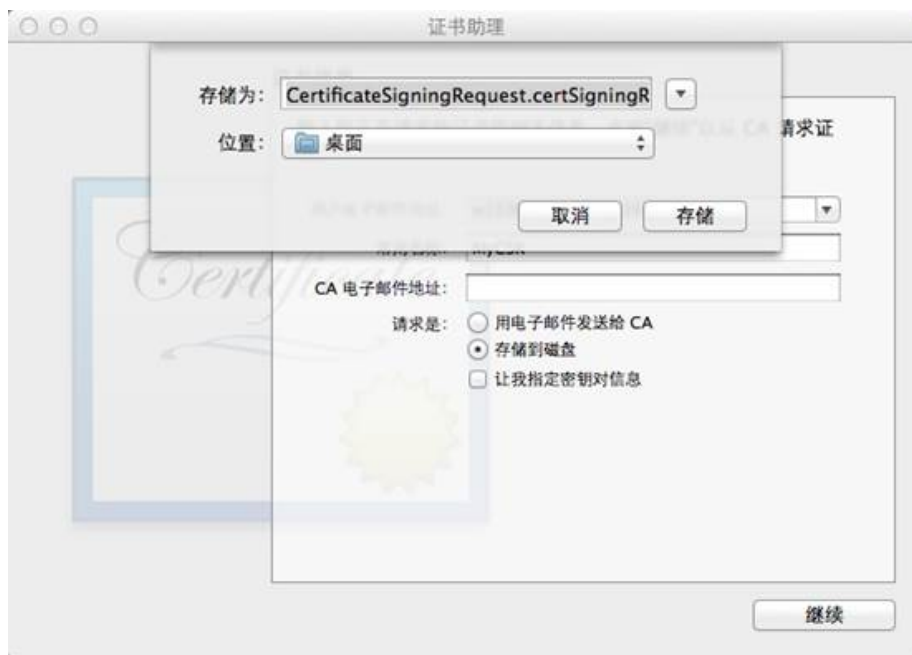
在菜单栏里选择 钥匙串访问 -> 证书助理 -> 从证书颁发机构请求证书，如图



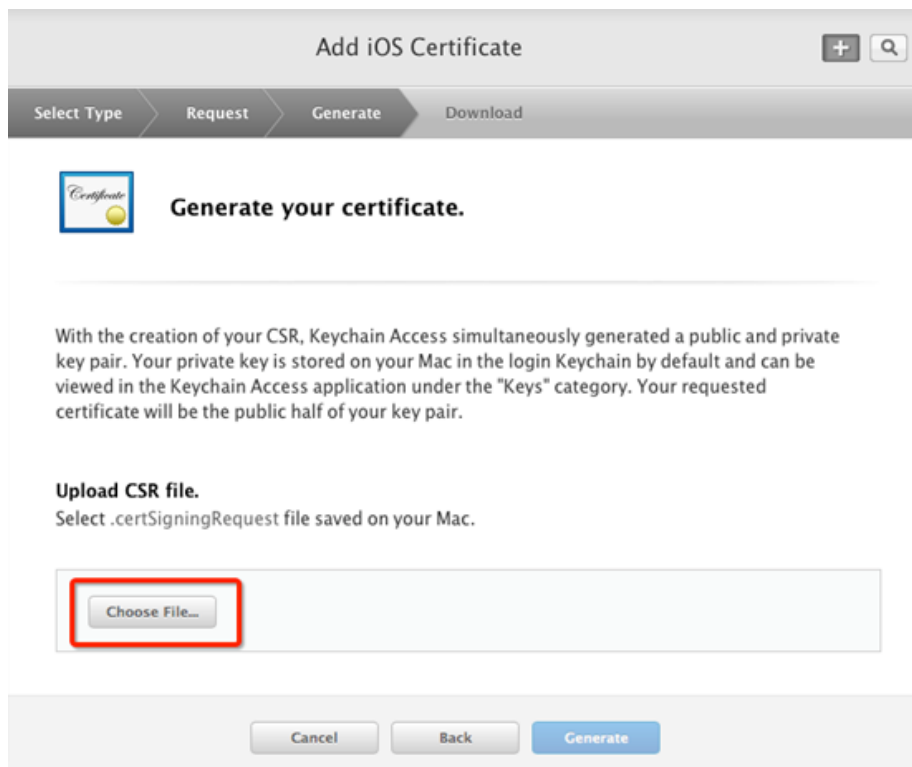
填入用户 电子邮件地址 和 常用名，选择 存储到磁盘，如图



点击 继续，选择保存位置，之后要上传这个文件



回到网页，iOS Developer Center中继续配置证书过程，点击 **continue** 进入如下页面：



选择 **choose File**，上传 刚才生成的CSR文件，之后选择 **Generate**，如图：

Add iOS Certificate

+


Q

Select Type

Request

Generate

Download




Generate your certificate.

With the creation of your CSR, Keychain Access simultaneously generated a public and private key pair. Your private key is stored on your Mac in the login Keychain by default and can be viewed in the Keychain Access application under the "Keys" category. Your requested certificate will be the public half of your key pair.

Upload CSR file.
Select .certSigningRequest file saved on your Mac.

Choose File...

 CertificateSigningRequest.certSigningRequest

Cancel

Back

Generate


成功后下载证书到本地:

Select Type


Request

Generate

Download

**Your certificate is ready.**

Download, Install and Backup
Download your certificate to your Mac, then double click the .cer file to install in Keychain Access. Make sure to save a backup copy of your private and public keys somewhere secure.



Name: Apple Development iOS Push Services: com.mRocker.MPush

Type: APNs Development iOS

Identifier ID: MPushAPPID

Expires: 八月 26, 2015

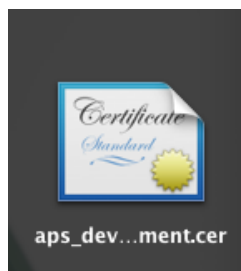
Download

Documentation
For more information on using and managing your certificates read:
[App Distribution Guide](#)

Add Another

Done

双击下载的cer文件会跳到钥匙串访问里面，在里面可以看到证书：



第三步：p12证书导出

在钥匙串访问中选中已经安装的推送证书，导出为个人信息交换文件（.p12格式）。

第四步：pem证书生成

将p12证书文件转换成pem证书文件。在终端里切换到p12文件保存的文件夹，运行以下命令转换：

```
openssl pkcs12 -in MyApnsCert.p12 -out MyApnsCert.pem -nodes
```

提示输入密码，就是导出p12文件时设置的导出密码。

输入完后回车完成。

第五步：pem证书上传

登录MPush portal，选中某个应用后，点击左侧的 **配置信息**，打开应用的配置界面，找到 **IOS专有选项**，点击 **上传证书**，选择本地的APNs开发证书或生产证书，确认修改即可。

证书上传完后，请注意配置 **部署环境**，以及 **确认修改**。

iOS特有选项： ?

☒ 开发状态 (Development)

? 开发证书： com.pogo.dq

选择文件

请上传.pem证书

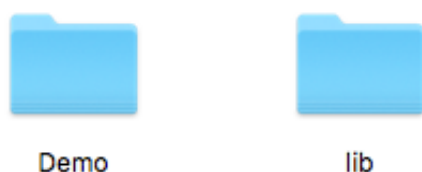
☐ 生产状态 (Production)

对APNs证书生成不太了解？请参考 [iOS证书设置指南](#)

下载SDK导入项目

从mPush网站下载中心页面，下载iOS SDK。

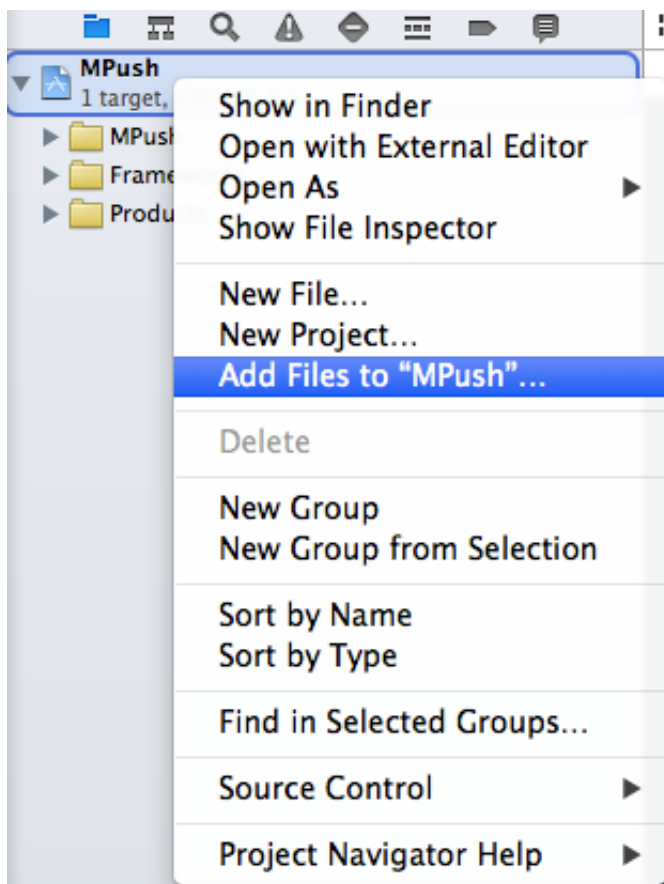
- lib文件夹：包含头文件 PushManager.h，静态库文件 libmPushSDK.a。
- demo文件夹：（一个完整的iOS示例项目，演示mPush基本用法，可参考。）



添加SDK到您的工程：

方法一：下载SDK压缩包，导入工程

- 1.解压缩mPush-sdk-v1.x.y.zip压缩包
- 2.复制lib文件夹到您的项目工程中，或者在项目工程名称处右键选择“Add files to 'Your project name'...", 将解压后的lib子文件夹添加到你的工程目录中。



方法二：使用CocoaPods 添加

- 1.在你项目的 Podfile 中添加以下代码：

```
pod 'mPushLib'
```

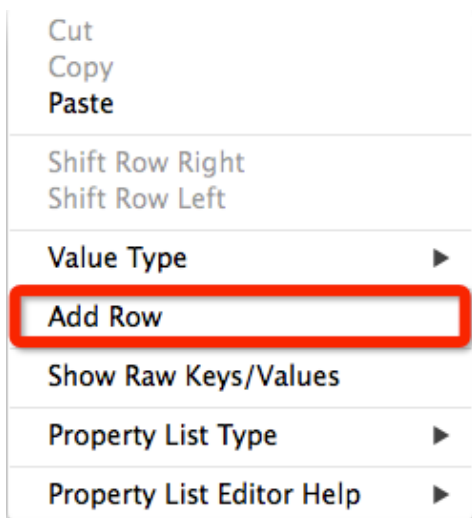
- 2.执行 pod 安装命令

```
$ pod install 或者 $ pod update
```

配置 MPushAppID 引入必要的框架

导入SDK后，首先需要配置MPushAppID：

- 在您的项目文件‘Your project name’-Info.plist中右键选择“Add Row”,添加一行。
- 命名新建行的key值为MPushAppID,类型为string,value为在Protal上创建应用后所分配的APP ID。
(此步骤为必选项，否则mPush将不会起作用！)



“

如果您的应用将支持 `iOS8.0` 以上的系统，请在您的项目文件 ‘Your project name’-`Info.plist` 中添加 `NSLocationWhenInUseUsageDescription` 缺省字段，类型为 `string`，值为您自定义的提示文字。例如：

<code>NSLocationAlwaysUsageDescription</code>	⬆ ⬇ ⬆	<code>String</code>	请允许使用以便更好的使用服务。
<code>NSLocationWhenInUseUsageDescription</code>	⬆ ⬇ ⬆	<code>String</code>	请允许使用以便更好的使用服务。

必要的框架（如已引入，则忽略）

```
CFNetwork.framework
CoreFoundation.framework
CoreTelephony.framework
SystemConfiguration.framework
CoreLocation.framework
Foundation.framework
UIKit.framework
```

启用mPush服务，并设置消息回调

在 `application: didFinishLaunchingWithOptions:` 中调

用 `startPushServicePushDelegate:tokenDelegate:`，注册mPush服务：

```
- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    // Override point for customization after application launch.
    .....
    /**
    开启推送服务并设置推送代理以及DeviceToken回调代理、自定义展示页面数据
    pushDelegate 推送代理，必须，参数对象必须实现onMessage:content:extention:方法
    deviceTokenDelegate DeviceToken获取代理，如果无需获取SDK中得DeviceToken，此项
    可为nil。
    /

    [PushManager startPushServicePushDelegate:self tokenDelegate:nil];

    return YES;
}
```

实现 `PushManagerDelegate` 协议，必须实现方法 `onMessage:content:extention:`：

```
/
代理回调方法

@param title    消息title内容
@param content  消息content内容
@param extention 参数字典,此字典包含用户自定义参数和应用状态参数。
                获取应用状态参数可通过 AppStateKey 来获取，状态类型参见
UIApplicationState

@return BOOL 当返回YES时，仅处理至当前事件处，后续事件将不再执行
            当返回NO时，按照事件链继续执行，直至返回YES或者所有事件执行完。
/

- (BOOL)onMessage:(NSString *)title content:(NSString *)content
extention:(NSDictionary *)extention{
    //在这里您可以处理收到的消息内容
    NSLog(@"title : %@ \n content : %@ \n extention : %@
\n",title,content,[extention description]);
    return YES;
}
```

测试确认

- 确认 MPushAppID（你在 mPush Portal 上创建应用后所分配给应用的APP ID）已经正确的写入‘Your project name’-Info.plist中，如未填写，请参考3. 配置 MPushAppID进行配置。
- 确认所使用的签名证书是具有推送权限的证书，如果不是，请参考iOS 推送证书配置指南进行配置。
- 如证书无误，确认项目的Bundle ID和推送证书配置时的Bundle ID是否一致。
- 确认程序启动时调用了startPushServicePushDelegate:tokenDelegate:接口。并实现了PushManagerDelegate协议方法。
- 确认测试设备已成功接入网络，且网络正常。
- 确认以上步骤无误后，启动应用程序，在 Portal 上向应用程序发送推送消息（详情请参考管理 Portal），如果 SDK 工作正常，客户端应可收到下发的通知，应有消息到达的日志信息。
- 如以上步骤确认无误，仍无法接收到推送消息，请咨询技术支持QQ群：372650575。

特别说明

关于IDFA的声明

“

如果您的应用本身不获取IDFA，只内嵌本SDK时，建议您提交至AppStore时按如下方式配置：

Advertising Identifier

Does this app use the Advertising Identifier (IDFA)?

The **Advertising Identifier (IDFA)** is a unique ID for each iOS device and is the only way to offer targeted ads. Users can choose to limit ad targeting on their iOS device.

If your app is using the Advertising Identifier, check your code—including any third-party code—before you submit it to make sure that your app uses the Advertising Identifier only for the purposes listed below and respects the Limit Ad Tracking setting. If you include third-party code in your app, you are responsible for the behavior of such code, so be sure to check with your third-party provider to confirm compliance with the usage limitations of the Advertising Identifier and the Limit Ad Tracking setting.

This app uses the Advertising Identifier to (select all that apply):

☐ Serve advertisements within the app

☒ Attribute this app installation to a previously served advertisement

☒ Attribute an action taken within this app to a previously served advertisement

If you think you have another acceptable use for the Advertising Identifier, [contact us](#).

Limit Ad Tracking setting in iOS

☒ I, Weiqiang Li, confirm that this app, and any third party that interfaces with this app, uses the Advertising Identifier checks and honors a user's Limit Ad Tracking setting in iOS and, when it is enabled by a user, this app does not use Advertising Identifier, and any information obtained through the use of the Advertising Identifier, in any way other than for "Limited Advertising Purposes" as defined in the [iOS Developer Program License Agreement](#).

☒ Yes

☐ No

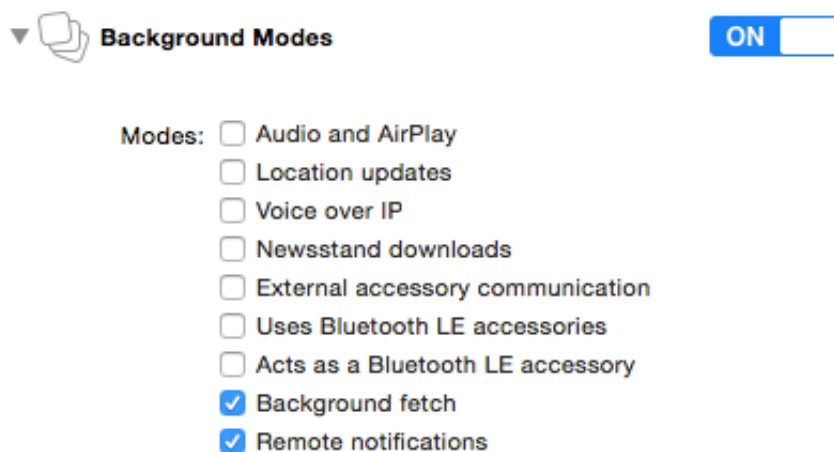
如果您应用本身获取IDFA，提交时请按应用本身情况配置。如果不幸被拒，您可尝试下述方式来解决：注意苹果拒绝上架邮件中，此原因以外的其他理由。

远程推送API

iOS SDK 支持基本的远程推送和静默推送两种模式。

“

静默推送需要做如下设置：



远程推送API

此方法必须在application: didFinishLaunchingWithOptions:中调用，及时注册mPush推送服务，代理可根据需要设置。

```
/**
 * 开启推送服务并设置相关参数
 *
 * @param pushDelegate 推送消息代理
 * @param tokenDelegate DeviceToken代理
 */
+ (void)startPushServicePushDelegate:(id <PushManagerDelegate>)pushDelegate
    tokenDelegate:(id
<DeviceTokenDelegate>)tokenDelegate;
```

自定义收到通知后的展示界面

当收到通知消息后，用户点击通知打开应用所要打开的界面设置。

```

/**
 * 定义打开应用内某个页面
 *
 * @param displayViews 需要展示的界面
 * @param viewAlias 针对每个界面指定的别名
 * @param methods 传递参数到界面的方法
 * @param displayModes 界面打开的方式，目前有两种方式可指定。PUSH 和 PRESENT，指
定时以字符串的方式指定，默认是PRESENT方式。
 */
+ (void)setDisplayViews:(NSArray *)displayViews
        viewAlias:(NSArray *)viewAlias
        methods:(NSArray *)methods
        displayModes:(NSArray *)displayModes;

```

定义收到远程通知后的操作是否使用SDK内部实现

当定义了收到通知后打开的界面步骤，SDK内部会自动打开所设置的界面，如果需要自定义打开，使用此方法禁止SDK实现。

```

/**
 * 定义收到远程通知后的操作是否使用SDK内部实现（默认SDK内部实现）。
 * 如果SDK内部实现，需要按照需求实
现'setDisplayViews:viewAlias:methods:displayModes'方法
 *
 * @param _inSDK 是否SDK内实现
 */
+ (void)setRemoteNotificationActionIMPInSDK:(BOOL)_inSDK;

```

是否开启调试日志

当开启调试日志的时候，xCode调试栏将打印部分日志，供查看。

```

/**
 * 是否开启调试日志，默认为不开启
 *
 * @param enabled YES 打开 NO 关闭
 */
+ (void)setDebugMode:(BOOL)enabled;

```

获取当前 mPush SDK 的版本号

```
/**
 * 获取当前 mPush SDK 的版本号
 */
+ (NSString *)getSDKVersion;
```

远程推送注意事项

当接收到远程推送时需要自己进行一些操作的时候,需要调用系统方法。

```
- (void)application:(UIApplication *)application
    didReceiveRemoteNotification:(NSDictionary *)userInfo
    fetchCompletionHandler:(void (^)(
    UIBackgroundFetchResult))completionHandler;
```

本地推送API

创建本地通知

根据所传入的参数创建一条计划任务，并为此任务分配一个标识。

```
/**
 * 本地推送，最多支持64个
 *
 * @param title      通知标题
 * @param content    通知内容
 * @param actionType 动作类型（定义接收到本地通知时，可执行的动作类型，可参见
LocalPushActionType 枚举）
 * @param actionValue 动作参数（定义接收到本地通知时，可执行的动作参数，根据
LocalPushActionType 设置）
 * @param extention  自定义参数集合
 * @param fireDate   通知发生时间
 *
 * @return SDK分配给本通知的唯一标识 key
 */
+ (NSString *)sendLocalPushMsg:(NSString *)title
                        content:(NSString *)content
            actionType:(LocalPushActionType)actionType
        actionValue:(NSString *)actionValue
        extentions:(NSDictionary *)extention
          fireDate:(NSDate *)fireDate;
```

取消指定标识的单个本地通知

根据标识取消掉一个计划任务。

```
/**
 * 取消指定标识的单个本地通知
 *
 * @param key 通知标识，为创建时所分配。
 */
+ (void)cancelLocalPushForKey:(NSString *)key;
```

取消所有的本地通知

一次性取消所有已创建的计划任务。

```
/**
 * 取消所有的本地通知
 */
+ (void)cancelAllLocalPush;
```

定义收到本地通知后的操作是否使用SDK内部实现

```
/**
 * 定义收到本地通知后的操作是否使用SDK内部实现（默认SDK内部实现）。
 * 如果SDK内部实现，需要按照需求实现'setDisplayViews:viewAlias:methods:displayModes'方法
 *
 * @param _inSDK 是否SDK内实现
 */
+ (void)setLocalNotificationActionIMPInSDK:(BOOL)_inSDK;
```

短信验证API

mPush 支持根据所指定的手机号，下发短信验证码功能。

```
/**
 * 获取验证码接口（调用此接口前，请确保手机号正常可用。）
 *
 * @param mobile      手机号
 * @param timeInterval 验证码有效时间，单位分钟，默认为3分钟
 * @param finishBlock 接口请求成功后的回调 成功是返回code，失败或者其他错误返回
errorMsg
 */
+ (void)authSms:(NSString *)mobile timeInterval:(NSTimeInterval)timeInterval
finished:(void (^)(NSString *code, NSString *errorMsg))finishBlock;
```

推送回调API

推送回调 API

```
/**
 * 收到消息后的代理回调方法（远程推送和本地推送的消息回调，均会执行此方法。）
 *
 * @param title 消息title内容
 * @param content 消息content内容
 * @param extention 扩展参数
 *
 * @return BOOL 当返回YES时，仅处理至当前事件处，后续事件将不再执行，当返回NO时，按照事件链继续执行，直至返回YES或者所有事件执行完。
 */
- (BOOL)onMessage:(NSString *)title
               content:(NSString *)content
          extention:(NSDictionary *)extention;
```

DeviceToken回调 API

```
/**
 * 获取DeviceToken，此DeviceToken为苹果原生DeviceToken 的32位hex字符串。
 * 如果需要原生格式的DeviceToken，可使用系统
 * didRegisterForRemoteNotificationsWithDeviceToken 方法获取
 *
 * @return deviceToken
 */
- (void)didReciveDeviceToken:(NSString *)deviceToken;
```

别名和标签API

别名 alias

上报用户的别名(例如用户昵称)，以便支持按别名推送消息。

```
/**
 * 为设备设置别名或移除别名,设置的前提条件是设备的DeviceToken已经获取成功。
 *
 * @param alias 别名名称, 当为nil或者@""时, 为移除别名, 否则为设置别名。重复调用会自动更新至最后一次调用时的别名。
 */
+ (void)setAlias:(NSString *)alias;
```

标签 tag

开发者可以针对不同的用户设置标签，以便根据标签进行指向性推送消息。

- 设置标签

```
/**
 * 优化的设置tag接口，建议使用（此接口不会重复上传相同的tag名称，推荐使用。）
 *
 * @param tags tag名称列表
 */
+ (void)ensureTags:(NSArray *)tags;
```

- 删除标签

```
/**
 * 删除设置过的tag，可以为多个。
 *
 * @param tags tag名称。
 */
+ (void)deleteTags:(NSArray *)tags;
```