# 极光推送文档

## SDK说明

### 适用版本

本文匹配的 SDK版本：r2.1.5 以后。  
[查看最近更新](http://docs.jiguang.cn/jpush/updates)了解最新的SDK更新情况。  
使用Xcode 6及以上版本可以使用新版Push SDK，Xcode 5环境下需要运行旧版本SDK(1.7.4)

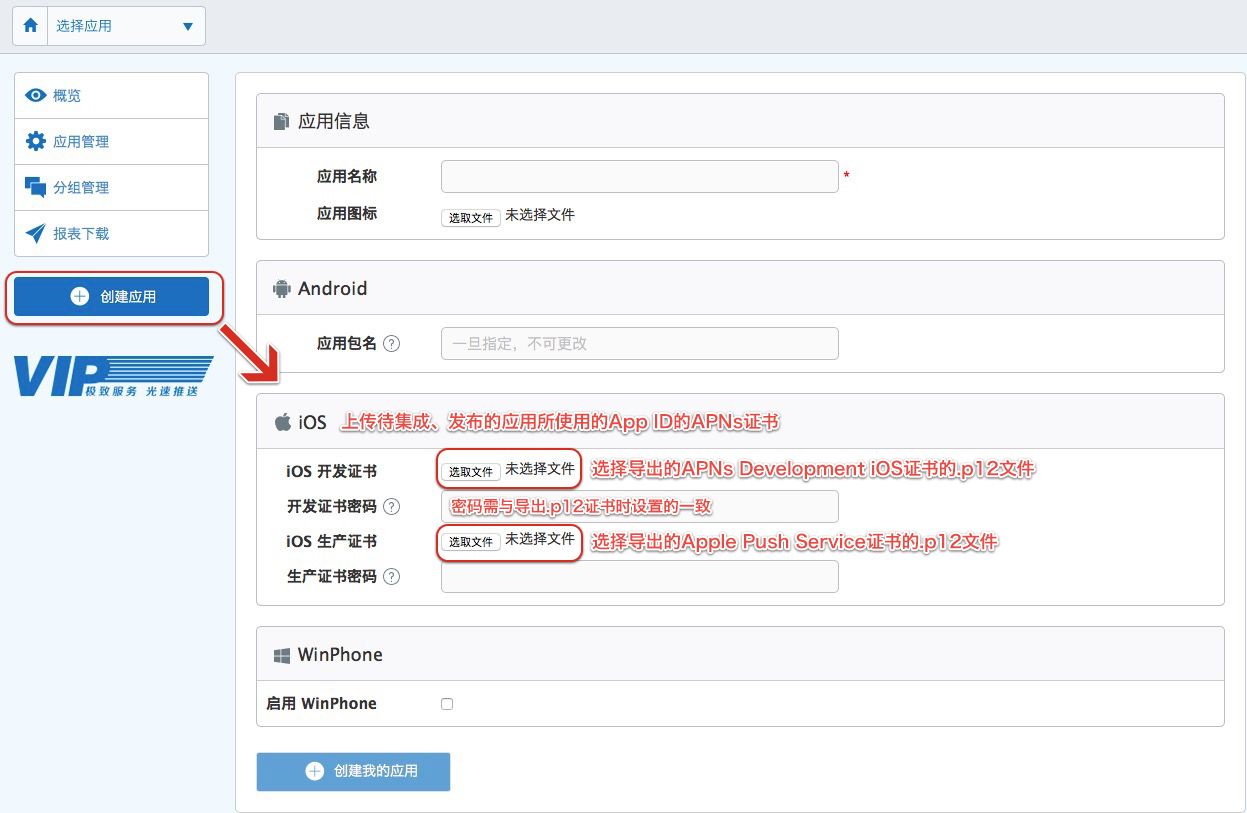
### 资源文件

包名为JPush-iOS-SDK-{版本号}

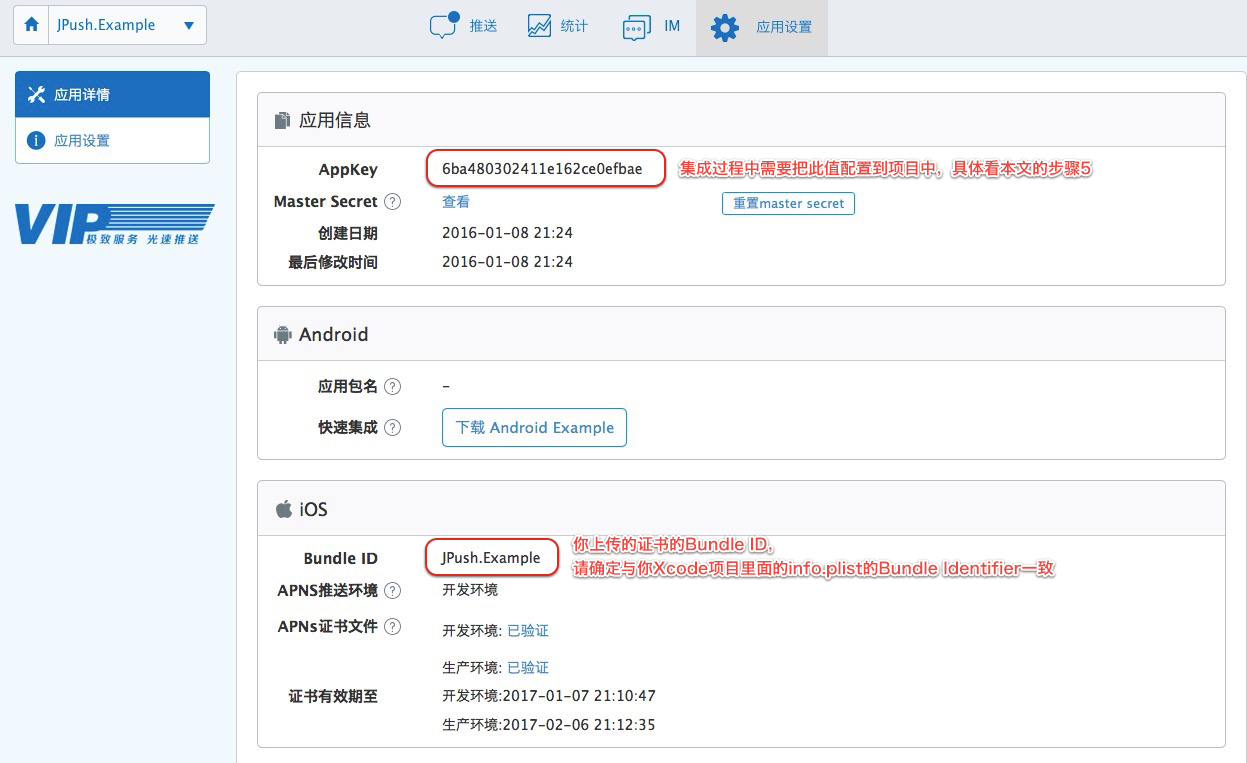
* lib文件夹：包含头文件 JPUSHService.h，静态库文件jpush-ios-x.x.x.a ，支持的iOS版本为 6.0 及以上版本。（请注意：模拟器不支持APNs）
* pdf文件：集成指南
* demo文件夹：示例

## 创建应用

* 在 JPush的管理Portal 上创建应用并上传APNs证书。如果对APNs证书不太了解 请参考： [iOS 证书设置指南](http://docs.jiguang.cn/jpush/client/iOS/ios_cer_guide)



* 创建成功后自动生成 AppKey 用以标识该应用。



## 配置工程

### 导入SDK

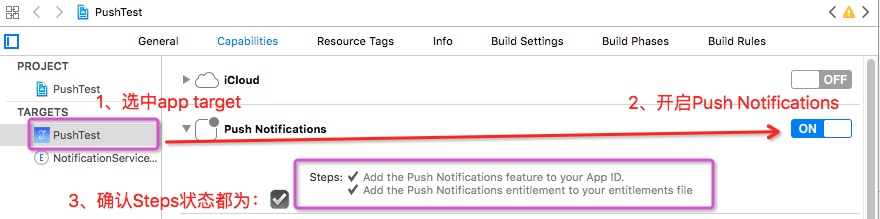
* 将SDK包解压，在Xcode中选择“Add files to 'Your project name'...”，将解压后的lib子文件夹（包含JPUSHService.h、jpush-ios-x.x.x.a）添加到你的工程目录中。
* 添加Framework
  + CFNetwork.framework
  + CoreFoundation.framework
  + CoreTelephony.framework
  + SystemConfiguration.framework
  + CoreGraphics.framework
  + Foundation.framework
  + UIKit.framework
  + Security.framework
  + Xcode7需要的是libz.tbd；Xcode7以下版本是libz.dylib
  + Adsupport.framework (获取IDFA需要；如果不使用IDFA，请不要添加)
  + UserNotifications.framework(Xcode8及以上)

### Build Settings

如果你的工程需要支持小于7.0的iOS系统，请到Build Settings 关闭 bitCode 选项，否则将无法正常编译通过。

* 设置 Search Paths 下的 User Header Search Paths 和 Library Search Paths，比如SDK文件夹（默认为lib）与工程文件在同一级目录下，则都设置为"$(SRCROOT)/{静态库所在文件夹名称}"即可。

### Capabilities

如使用Xcode8及以上环境开发，请开启Application Target的Capabilities->Push Notifications选项，如图：  


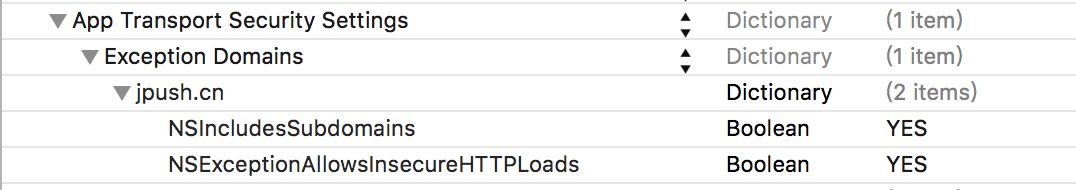
### 允许Xcode7支持Http传输方法

如果您使用的是2.1.9以后的版本则不需要配置此步骤  
如果用的是Xcode7或更新版本，需要在App项目的plist手动配置下key和值以支持http传输:

**选择1：根据域名配置**

* 在项目的info.plist中添加一个Key：NSAppTransportSecurity，类型为字典类型。
* 然后给它添加一个NSExceptionDomains，类型为字典类型；
* 把需要的支持的域添加給NSExceptionDomains。其中jpush.cn作为Key，类型为字典类型。
* 每个域下面需要设置2个属性：NSIncludesSubdomains、NSExceptionAllowsInsecureHTTPLoads。 两个属性均为Boolean类型，值分别为YES、YES。

如图：



**选择2：全局配置**

<key>NSAppTransportSecurity</key>

<dict>

<key>NSAllowsArbitraryLoads</key><true/>

</dict>

## 添加代码

2.1.0版本开始,API类名为JPUSHService，不再使用原先的APService。

### 初始化代码

导入#import "APNSManager.h"

请将以下代码添加到  
 - (BOOL)application:(UIApplication \*)application didFinishLaunchingWithOptions:(NSDictionary \*)launchOptions {

[APNSManager APNSAllMeansLaunchOptions:launchOptions APPKey:@"faa402af95f50d88ca500dfc" Production:NO];

[self NotificationCallBack];//推送监听

return YES;

}

注册APNs成功并上报DeviceToken

请在AppDelegate.m实现该回调方法并添加回调方法中的代码

- (void)application:(UIApplication \*)application didRegisterForRemoteNotificationsWithDeviceToken:(NSData \*)deviceToken { /// Required - 注册 DeviceToken

[APNSManager registerSuccedDeviceToken:deviceToken];

}

### 实现注册APNs失败接口（可选）

- (void)application:(UIApplication \*)application didFailToRegisterForRemoteNotificationsWithError:(NSError \*)error { //Optional NSLog(@"did Fail To Register For Remote Notifications With Error: %@", error);

}

### 添加处理APNs通知回调方法

请在AppDelegate.m实现该回调方法并添加回调方法中的代码

#pragma mark- JPUSHRegisterDelegate // iOS 10 Support

- (void)jpushNotificationCenter:(UNUserNotificationCenter \*)center willPresentNotification:(UNNotification \*)notification withCompletionHandler:(void (^)(NSInteger))completionHandler {

// Required

NSDictionary \* userInfo = notification.request.content.userInfo;

if([notification.request.trigger isKindOfClass:[UNPushNotificationTrigger class]]) {

[JPUSHService handleRemoteNotification:userInfo];

}

completionHandler(UNNotificationPresentationOptionAlert); // 需要执行这个方法，选择是否提醒用户，有Badge、Sound、Alert三种类型可以选择设置

}

// iOS 10 Support

- (void)jpushNotificationCenter:(UNUserNotificationCenter \*)center didReceiveNotificationResponse:(UNNotificationResponse \*)response withCompletionHandler:(void (^)())completionHandler {

// Required

NSDictionary \* userInfo = response.notification.request.content.userInfo;

if([response.notification.request.trigger isKindOfClass:[UNPushNotificationTrigger class]]) {

[JPUSHService handleRemoteNotification:userInfo];

}

completionHandler(); // 系统要求执行这个方法

}

### 添加处理JPush自定义消息回调方法

如需使用JPush的自定义消息功能，请参考[文档](http://docs.jiguang.cn/jpush/client/iOS/ios_api/#message)来实现自定义消息的处理回调方法。

## 成功运行

真机调试该项目，如果控制台输出以下日志则代表您已经集成成功。

2016-08-19 17:12:12.745823 219b28[1443:286814] | JPUSH | I - [JPUSHLogin] ----- login result ----- uid:5460310207 registrationID:171976fa8a8620a14a4

如果调试运行中遇到问题请参考：[iOS SDK 调试指南](http://docs.jiguang.cn/jpush/client/iOS/ios_debug_guide)

## 高级功能

### 关于IDFA

r2.1.5版本增加一个上传IDFA字符串的接口

+ (void)setupWithOption:(NSDictionary \*)launchingOption

appKey:(NSString \*)appKey

channel:(NSString \*)channel

apsForProduction:(BOOL)isProduction;

+ (void)setupWithOption:(NSDictionary \*)launchingOption

appKey:(NSString \*)appKey

channel:(NSString \*)channel

apsForProduction:(BOOL)isProduction

advertisingIdentifier:(NSString \*)advertisingId;

### JPush SDK 相关事件监听

建议开发者加上API里面提供的以下类型的通知：

extern NSString \*const kJPFNetworkIsConnectingNotification; // 正在连接中

extern NSString \* const kJPFNetworkDidSetupNotification; // 建立连接

extern NSString \* const kJPFNetworkDidCloseNotification; // 关闭连接

extern NSString \* const kJPFNetworkDidRegisterNotification; // 注册成功

extern NSString \*const kJPFNetworkFailedRegisterNotification; //注册失败

extern NSString \* const kJPFNetworkDidLoginNotification; // 登录成功

温馨提示：

Registration id 需要添加注册kJPFNetworkDidLoginNotification通知的方法里获取，也可以调用[registrationIDCompletionHandler:]方法，通过completionHandler获取

extern NSString \* const kJPFNetworkDidReceiveMessageNotification; // 收到自定义消息(非APNs)

其中，kJPFNetworkDidReceiveMessageNotification传递的数据可以通过NSNotification中的userInfo方法获取，包括标题、内容、extras信息等

-(void)NotificationCallBack

{

// 正在连接中

[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(JPushDidOnConnection:) name:kJPFNetworkDidSetupNotification object:nil];

// 建立连接

[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(JPushDidSetup:) name:kJPFNetworkDidSetupNotification object:nil];

// 关闭连接

[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(JPushDidClose:) name:kJPFNetworkDidCloseNotification object:nil];

// 注册成功

[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(JPushDidRegister:) name:kJPFNetworkDidRegisterNotification object:nil];

// 注册失败

[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(JPushDidRegisterFail:) name:kJPFNetworkDidRegisterNotification object:nil];

// 登录成功

[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(JPushDidLogin:) name:kJPFNetworkDidLoginNotification object:nil];

// 收到自定义消息

[[NSNotificationCenter defaultCenter] addObserver:self selector:@selector(JPushDidReceiveMessage:) name:kJPFNetworkDidReceiveMessageNotification object:nil];

}

#pragma mark - 极光推送监听

- (void)JPushDidOnConnection:(NSNotification \*)noti

{

NSLog(@"正在连接中");

}

- (void)JPushDidSetup:(NSNotification \*)noti

{

NSLog(@"建立连接");

}

- (void)JPushDidClose:(NSNotification \*)noti

{

NSLog(@"关闭连接");

}

- (void)JPushDidRegister:(NSNotification \*)noti

{

NSLog(@"注册成功");

}

-(void)JPushDidRegisterFail:(NSNotification \*)noti

{

NSLog(@"注册失败");

}

- (void)JPushDidLogin:(NSNotification \*)noti

{

NSLog(@"登录成功：");

}

- (void)JPushDidReceiveMessage:(NSNotification \*)noti

{

// 收到推送消息

NSLog(@"收到推送消息:%@", noti.userInfo);

}