

iOS SDK 集成指南

- [使用提示](#)
- [产品功能说明](#)
 - [主要功能](#)
 - [主要特点](#)
 - [版本](#)
 - [集成压缩包内容](#)
 - [开发环境](#)
- [SDK集成步骤](#)
 - [1、在JPush Portal上创建应用](#)
 - [2、导入API开发包到应用程序项目](#)
 - [3、必要的框架](#)
 - [4、Build Settings](#)
 - [5、创建并配置PushConfig.plist文件](#)
 - [5、添加代码](#)
 - [API](#)
 - [调用代码](#)
 - [监听通知](#)
- [高级功能](#)
- [技术支持](#)

使用提示

本文匹配的 SDK版本：r1.2.5 以后。

产品功能说明

极光推送（JPush）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。极光推送客户端支持 Android, iOS 两个平台。

本 iOS SDK 方便开发者基于 JPush 来快捷地为 iOS App 增加推送功能，减少集成 APNs 需要的工作量、开发复杂度。

主要功能

- 为 JPush Server 上报 Device Token，免除开发者管理 Device Token 的麻烦
- 应用运行时，应用内 JPush 长连接可以持续地收到推送消息

主要特点

- 集成简单
- iOS SDK 集成后，服务器端向 iOS 设备推送简单方便

版本

- 当前提供的 SDK 使用 OpenUDID 的方案来确定设备的编号，符合 App Store 的上架规定。原 JPush SDK UDID 版本已停止更新。

集成压缩包内容

包名为JPush-iOS-SDK-[版本号]

- lib文件夹：包含头文件 APService.h，静态库文件 libPushSDK.a 和 libPushSDK-Simulator.a，支持的iOS版本为4.3以上版本。（ 请注意：模拟器不能实现APNS，提供libPushSDK-Simulator.a，只是为了避免SDK在模拟器环境下报错，方便开发者使用模拟器调试自己的功能 ）
- pdf文件：开发指南
- demo文件夹：示例

开发环境

- 建议使用 XCode 4.5 或以上版本

SDK集成步骤

1、在JPush Portal上创建应用

- 在[JPush的管理Portal上](#) 上传证书并创建应用。如果对APNs证书不太了解 请参考 [iOS 证书设置指南](#)

创建应用

应用名称 ?

应用图标 ? Choose File no file selected

Android

应用包名 ? 一旦指定，不可更改

开发环境用于调试 Apple

iOS 开发证书 Choose File no file selected 后缀为 .p12 的证书文件

开发证书密码 ? 密码与证书导出时设置的一致。

iOS 生产证书 Choose File no file selected

生产证书密码 ?

生产环境用于Ad-Hoc版本或者生产发布

对 APNs 证书生成不太了解？请参考 [iOS 证书设置指南](#)。

创建我的应用

- 创建成功后自动生成 AppKey 用以标识该应用。



2、导入API开发包到应用程序项目

- 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APService.h、libPushSDK.a 和 libPushSDK-Simulator.a文件，如果不需要在模拟器中调试，可以丢弃libPushSDK-Simulator.a文件）添加到你的工程目录中。

3、必要的框架

- CFNetwork.framework
- CoreFoundation.framework
- CoreTelephony.framework
- SystemConfiguration.framework
- CoreGraphics.framework
- Foundation.framework
- UIKit.framework

4、Build Settings

- 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)/[文件夹名称]”即可。

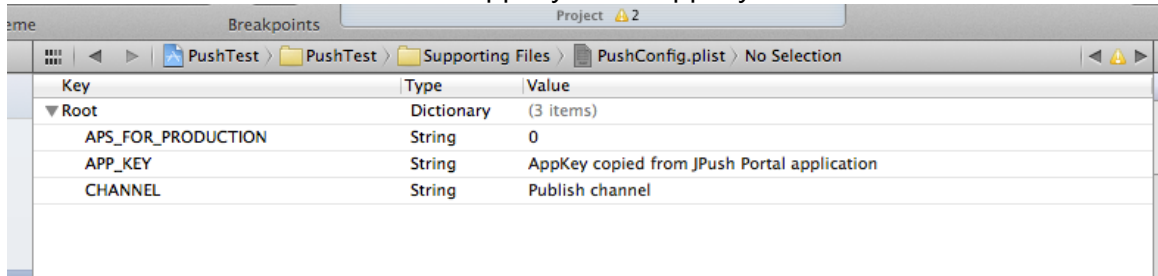
5、创建并配置PushConfig.plist文件

在你的工程中创建一个新的Property

List文件，并将其命名为PushConfig.plist，填入Portal为你的应用提供的APP_KEY等参数。

```
{
    "APS_FOR_PRODUCTION" = "0";
    "CHANNEL" = "Publish channel";
    "APP_KEY" = "AppKey copied from JPush Portal application";
}
```

- CHANNEL
 - 指明应用程序包的下载渠道，为方便分渠道统计。根据你的需求自行定义即可。
- APP_KEY
 - 在[管理Portal上创建应用](#)时自动生成的（AppKey）用以标识该应用。请确保应用内配置的 AppKey 与第1步在 Portal 上创建应用时生成的 AppKey 一致，AppKey 可以在应用详情中查询。



Key	Type	Value
Root	Dictionary	(3 items)
APS_FOR_PRODUCTION	String	0
APP_KEY	String	AppKey copied from JPush Portal application
CHANNEL	String	Publish channel

- APS_FOR_PRODUCTION
 - 1.3.1版本新增，表示应用是否采用生产证书发布(Ad_Hoc 或 APP Store)，0 (默认值)表示采用的是开发者证书，1 表示采用生产证书发布应用。请注意此处配置与 Web Portal 应用环境设置匹配。
- 在1.2.2或之前版本的配置文件中，有 TEST_MODE 这个键，新版的SDK不再使用，可以将它删除。

5、添加代码

API

APIs 主要集中在 APService 接口类里。

```
@interface APService : NSObject

// init Push
+ (void)setupWithOptions:(NSDictionary *)launchingOption;

// register notification type
+ (void)registerForRemoteNotificationTypes:(UIRemoteNotificationType)types;

// upload device token
+ (void)registerDeviceToken:(NSData *)deviceToken;

// handle notification recieved
+ (void)handleRemoteNotification:(NSDictionary *)remoteInfo;

@end
```

调用代码

监听系统事件，相应地调用 JPush SDK 提供的 API 来实现功能。

以下 3 个事件监听与调用 JPush SDK API 都是必须的。请直接复制如下代码块里，注释为 "Required" 的行，到你的应用程序代理类里相应的监听方法里。

```

- (BOOL)application:(UIApplication *)application
didFinishLaunchingWithOptions:(NSDictionary *)launchOptions
{
    self.window = [[[UIWindow alloc] initWithFrame:[[UIScreen mainScreen]
bounds]] autorelease];
    self.window.backgroundColor = [UIColor whiteColor];
    [self.window makeKeyAndVisible];

    // Required
    [APService
registerForRemoteNotificationTypes:(UIRemoteNotificationTypeBadge |
UIRemoteNotificationTypeSound |
UIRemoteNotificationTypeAlert)];
    // Required
    [APService setupWithOptions:launchOptions];

    return YES;
}

- (void)application:(UIApplication *)application
didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken {

    // Required
    [APService registerDeviceToken:deviceToken];
}

- (void)application:(UIApplication *)application
didReceiveRemoteNotification:(NSDictionary *)userInfo {

    // Required
    [APService handleRemoteNotification:userInfo];
}

```

监听通知

API里面提供了下面 5 种类型的通知：

```

extern NSString * const kAPNetworkDidSetupNotification;    // 建立连接
extern NSString * const kAPNetworkDidCloseNotification;    // 关闭连接
extern NSString * const kAPNetworkDidRegisterNotification; // 注册成功
extern NSString * const kAPNetworkDidLoginNotification;    // 登录成功
extern NSString * const kAPNetworkDidReceiveMessageNotification; // 收到消息(非APNS)

```

其中，kAPNetworkDidReceiveMessageNotification通知是有传递数据的，可以通过NSNotification中的userInfo

方法获取，包括标题、内容、内容类型、扩展信息等

高级功能

请参考：

[标签与别名API](#)

技术支持

邮件联系：support@jpush.cn

问答社区：<http://www.jpush.cn/qa/>

技术QQ群：178098500