

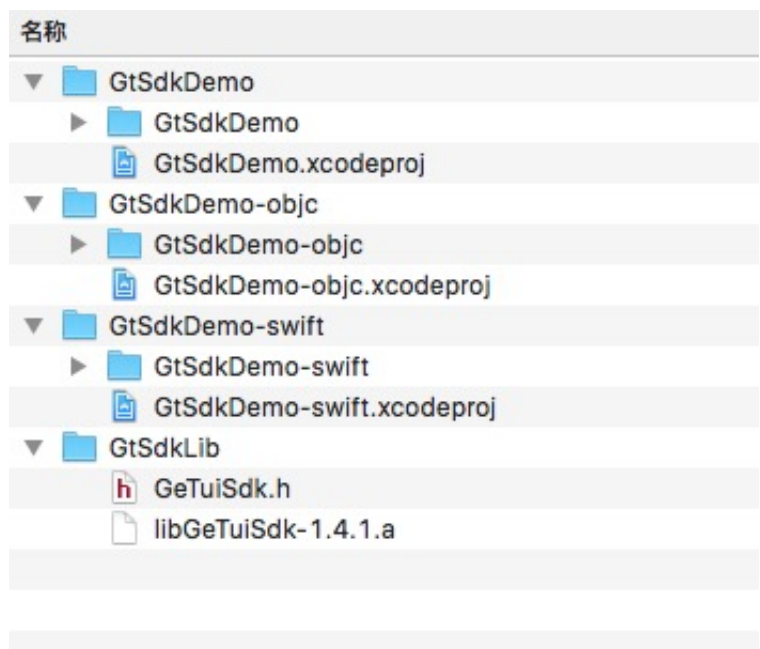
# iOS SDK 集成步骤

## 1.概述

### 1.1、集成压缩包内容：

我们提供的一个SDK开发工具包，包含了iOS SDK的全部所需资源，解压缩后的文件目录结构如图：

- GtSdkDemo：SDK演示Demo，能更好的展示个推SDK功能点。
- GtSdkDemo-objc：objc集成Demo，方便objc开发者集成个推SDK。
- GtSdkDemo-swift：swift集成Demo，方便swift开发者集成个推SDK。
- GtSdkLib：个推SDK资源文件，包含集成SDK所需的静态库和头文件。



注意：

libGeTuiSdk-{version}.a （version为具体的sdk版本号）使用libo工具将 支持i386、x86\_64、arm64、armv7、armv7s的代码打包到了一起，所以这个库将同时支持simulator和device，支持 IOS版本为7.0及以上。

## 2.项目设置

### 2.1、添加https支持

由于iOS9改用更安全的https，为了能够在iOS9中正常使用GeTuiSdk，请在"Info.plist"中进行如下配置，否则影响SDK的使用。

```

<key>NSAppTransportSecurity</key>
<dict>
<key>NSAllowsArbitraryLoads</key>
<true/>
</dict>

```

Key	Type	Value
▼ Information Property List	Dictionary	(17 items)
Localization native development re...	String	en
Executable file	String	\$(EXECUTABLE_NAME)
Bundle identifier	String	\$(PRODUCT_BUNDLE_IDENTIFIER)
InfoDictionary version	String	6.0
Bundle name	String	\$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.4.0
Bundle creator OS Type code	String	????
Bundle version	String	1.4.0
Application requires iPhone enviro...	Boolean	YES
▼ App Transport Security Settings	Dictionary	(1 item)
Allow Arbitrary Loads	Boolean	YES
NSLocationAlwaysUsageDescription	String	请求定位Always
NSLocationWhenInUseUsageDesc...	String	请求定位InUse
► Required background modes	Array	(5 items)
► Required device capabilities	Array	(1 item)
► Supported interface orientations	Array	(1 item)
aps-environment	String	development

## 2.2、个推SDK头文件和.a库设置

将GtSdkLib目录拷贝到项目工程目录下，导入GtSdkLib目录所有的头文件、libGeTuiSdk-{version}.a文件和几个系统库到XCode项目中。

添加头文件搜索目录

注：头文件搜索目录根据不同项目或目录结构不同会有不一样，请开发者根据自己项目需求自行修改。












Library Search Paths		/Users/zhaowei/Desktop/GetuiSdk iOS/GtSdkDemo-objc/ ../GtSdkLib
Rez Search Paths		
Sub-Directories to Exclude in Recursive Searches	\$(inherited)	non-recursive
Sub-Directories to Include in Recursive Searches	\$(SRCROOT)/ ../GtSdkLib	non-recursive

## 2.3、添加依赖库（必须，如下图）

添加系统库支持：

- libz.tbd

- libsqlite3.tbd
- Security.framework
- MobileCoreServices.framework
- SystemConfiguration.framework
- CoreTelephony.framework
- AVFoundation.framework
- JavaScriptCore.framework
- CoreLocation.framework
- CoreBluetooth.framework

Name	Status
 libsqlite3.tbd	Required ⌵
 libz.tbd	Required ⌵
 libGeTuiSdk-1.4.1.a	Required ⌵
 Security.framework	Required ⌵
 JavaScriptCore.framework	Required ⌵
 MobileCoreServices.framework	Required ⌵
 SystemConfiguration.framework	Required ⌵
 CoreTelephony.framework	Required ⌵
 AVFoundation.framework	Required ⌵
 CoreLocation.framework	Required ⌵
 CoreBluetooth.framework	Required ⌵

## 2.4、SDK后台运行权限设置

为了更好支持SDK 推送，APP定期抓取离线数据，需要配置后台运行权限： Background fetch： 后台获取  
Remote notifications： 推送唤醒（静默推送， Silent Remote Notifications）

General
Capabilities
Resource Tags
Info
Build Settings
Build Phases
Build Rules

Background Modes
ON

Modes:
☐ Audio, AirPlay and Picture in Picture
☐ Location updates
☐ Voice over IP
☐ Newsstand downloads
☐ External accessory communication
☐ Uses Bluetooth LE accessories
☐ Acts as a Bluetooth LE accessory
☒ Background fetch
☒ Remote notifications

Steps: ✓ Add the "Required Background Modes" key to your info plist file

## 3.基本集成

### 3.1、AppDelegate 中注册 GeTuiSdkDelegate

```
#import <UIKit/UIKit.h>
#import "GeTuiSdk.h"      // GetuiSdk头文件，需要使用的地方需要添加此代码

/// 个推开发者网站中申请App时，注册的AppId、AppKey、AppSecret
#define kGtAppId           @"iMahVVxurw6BNr7XSn9EF2"
#define kGtAppKey          @"yIPfqwq6OMAPp6dkqgLpG5"
#define kGtAppSecret       @"G0aBqAD6t79JfzTB6Z5lo5"

/// 需要使用个推回调时，需要添加"GeTuiSdkDelegate"
@interface AppDelegate : UIResponder <UIApplicationDelegate, GeTuiSdkDelegate>
```

### 3.2、App运行时启动个推SDK并注册APNS

在AppDelegate didFinishLaunchingWithOptions 方法中:通过平台分配的AppId、AppKey、AppSecret启动个推SDK，并完成注册APNS通知和处理启动时拿到的APNS透传数据。

```
- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(
NSDictionary *)launchOptions {
    // 通过个推平台分配的appId、 appKey 、 appSecret 启动SDK，注：该方法需要在主线程中调用
    [GeTuiSdk startSdkWithAppId:kGtAppId appKey:kGtAppKey appSecret:kGtAppSecret
    delegate:self];
    // 注册APNS
    [self registerUserNotification];
    return YES;
}
```

注：注册APNs获取DeviceToken不同项目或版本会有所不同，可以参考如下方式注册APNs。

```

/** 注册APNS */
- (void)registerRemoteNotification {
#ifdef __IPHONE_8_0
    if ([[UIDevice currentDevice] systemVersion] floatValue] >= 8.0) {
        NSMutableUserNotificationCategory *actionCategory;
        actionCategory = [[NSMutableUserNotificationCategory alloc] init];
        [actionCategory setIdentifier:NotificationCategoryId];
        [actionCategory setActions:nil
            forContext:UIUserNotificationActionContextDefault];

        NSMutableSet *categories = [NSMutableSet setWithObject:actionCategory];
        UIUserNotificationType types = (UIUserNotificationTypeAlert |
            UIUserNotificationTypeSound |
            UIUserNotificationTypeBadge);

        UIUserNotificationSettings *settings;
        settings = [UIUserNotificationSettings settingsForTypes:types categories:categories];
        [[UIApplication sharedApplication] registerForRemoteNotifications];
        [[UIApplication sharedApplication] registerUserNotificationSettings:settings];

    } else {
        UIRemoteNotificationType apn_type = (UIRemoteNotificationType)(UIRemoteNotificationTypeAlert |
            UIRemoteNotificationTypeSound |
            UIRemoteNotificationTypeBadge);
        [[UIApplication sharedApplication] registerForRemoteNotificationTypes:apn_type];
    }
#else
    UIRemoteNotificationType apn_type = (UIRemoteNotificationType)(UIRemoteNotificationTypeAlert |
        UIRemoteNotificationTypeSound |
        UIRemoteNotificationTypeBadge);
    [[UIApplication sharedApplication] registerForRemoteNotificationTypes:apn_type];
#endif
}

```

### 3.3、向个推服务器注册DeviceToken

免除开发者管理 DeviceToken 的麻烦，需要向 GeTui Server 上报DeviceToken。并可通过个推开发者平台推

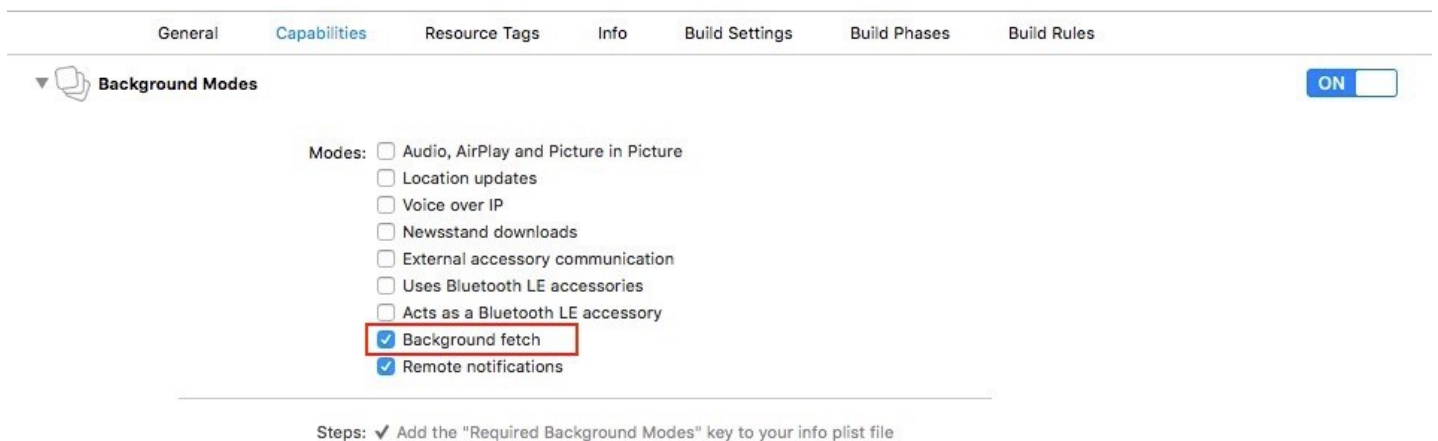
送APN消息。

```
/** 远程通知注册成功委托 */
- (void)application:(UIApplication *)application didRegisterForRemoteNotifications
WithDeviceToken:(NSData *)deviceToken {
    NSString *token = [[deviceToken description] stringByTrimmingCharactersInSet:[
NSCharacterSet characterSetWithCharactersInString:@"<>"]];
    token = [token stringByReplacingOccurrencesOfString:@" " withString:@""];
    NSLog(@"\n>>>[DeviceToken Success]:%@", token);

    //向个推服务器注册deviceToken
    [GeTuiSdk registerDeviceToken:token];
}
```

### 3.4、Background Fetch 接口回调

注: iOS7.0 以后支持APP后台刷新数据, 会回调 `performFetchWithCompletionHandler` 接口, 此处为保证个推数据刷新需调用`[GeTuiSdk resume]` 接口恢复个推SDK 运行刷新数据。



```
- (void)application:(UIApplication *)application performFetchWithCompletionHandle
r:(void (^)(UIBackgroundFetchResult))completionHandler {
    /// Background Fetch 恢复SDK 运行
    [GeTuiSdk resume];
    completionHandler(UIBackgroundFetchResultNewData);
}
```

### 3.5、GeTuiSdk注册回调, 获取CID信息

在不确定是否启动个推SDK成功, 可以通过回调查看注册结果

```

/** SDK启动成功返回cid */
- (void)GeTuiSdkDidRegisterClient:(NSString *)clientId {
    //个推SDK已注册，返回clientId
    NSLog(@"\n>>>[GeTuiSdk RegisterClient]:%@\\n\\n", clientId);
}

/** SDK遇到错误回调 */
- (void)GeTuiSdkDidOccurError:(NSError *)error {
    //个推错误报告，集成步骤发生的任何错误都在这里通知，如果集成后，无法正常收到消息，查看这里的
    通知。
    NSLog(@"\n>>>[GexinSdk error]:%@\\n\\n", [error localizedDescription]);
}

```

## 4.高级功能

### 4.1、使用个推SDK透传消息, 由个推通道下发 (非APNS)

SDK 在线状态时（App在前台运行），个推服务器会直接给您的App发送透传消息，不发送苹果APNS消息，可以更快的把消息发送到手机端；SDK离线状态时（停止SDK 或 App后台运行 或 App停止），个推服务器会给App发送苹果APNS消息，同时保存个推的离线消息，当SDK在线后，SDK会获取所有的个推透传消息，offLine字 段就是表明该条消息是否为离线消息。

```

/** SDK收到透传消息回调 */
- (void)GeTuiSdkDidReceivePayloadData:(NSData *)payloadData andTaskId:(NSString *)
taskId andMsgId:(NSString *)msgId andOffLine:(BOOL)offLine fromGtAppId:(NSString *)
)appId {
    //收到个推消息
    NSString *payloadMsg = nil;
    if (payloadData) {
        payloadMsg = [[NSString alloc] initWithBytes:payloadData.bytes
                                                    length:payloadData.length
                                                    encoding:NSUTF8StringEncoding];
    }

    NSString *msg = [NSString stringWithFormat:@"taskId=%@,messageId=%@,payloadMs
g:%@%@",taskId,msgId, payloadMsg,offLine ? @"<离线消息>" : @""];
    NSLog(@"\n>>>[GexinSdk ReceivePayload]:%@\\n\\n", msg);

    /**
    *汇报个推自定义事件
    *actionId: 用户自定义的actionid, int类型, 取值90001-90999。
    *taskId: 下发任务的任务ID。
    *msgId: 下发任务的消息ID。
    *返回值: BOOL, YES表示该命令已经提交, NO表示该命令未提交成功。注: 该结果不代表服务器收到该条
命令
    */
    /**
    [GeTuiSdk sendFeedbackMessage:90001 taskId:taskId msgId:msgId];
    }

```

注：个推透传获取的消息内容为下图中“消息内容”

The screenshot shows the Gexin SDK web interface for creating a透传消息 (透传消息). The interface includes a sidebar with navigation options: 创建推送, 推送通知, 透传消息, 分组对比测试, and 数据统计. The main content area is titled '透传消息' and contains two input fields: '描述' (Description) and '消息内容' (Message Content). The '消息内容' field is highlighted with a red box and contains the placeholder text '请输入透传消息的命令代码'. A '参数生成工具' (Parameter Generation Tool) link is visible at the bottom right of the '消息内容' field.

## 4.2、苹果官方静默推送

如果需要使用推送唤醒/Apns透传/静默推送 (Remote Notifications) “content-available:1”,需要配置



- Modes:
- ☐ Audio, AirPlay and Picture in Picture
  - ☐ Location updates
  - ☐ Voice over IP
  - ☐ Newsstand downloads
  - ☐ External accessory communication
  - ☐ Uses Bluetooth LE accessories
  - ☐ Acts as a Bluetooth LE accessory
  - ☒ Background fetch
  - ☒ Remote notifications

Steps: ✓ Add the "Required Background Modes" key to your info plist file

```
/** APP已经接收到“远程”通知(推送) - 透传推送消息 */
- (void)application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary *)userInfo fetchCompletionHandler:(void (^)(UIBackgroundFetchResult result))completionHandler {
    // 处理APNs代码, 通过userInfo可以取到推送的信息(包括内容, 角标, 自定义参数等)。如果需要弹窗等其他操作, 则需要自行编码。
    NSLog(@"\n>>>[Receive RemoteNotification - Background Fetch]:%@\\n\\n",userInfo);
    completionHandler(UIBackgroundFetchResultNewData);
}
```

### 4.3、指定标签推送

用户设置标签, 标示一组标签用户, 可以针对该标签用户进行推送

```
NSString *tagName = @"个推,推送,iOS";
NSArray *tagNames = [tagName componentsSeparatedByString:@","];
if (![GeTuiSdk setTags:tagNames]) {
    UIAlertView *alertView = [[UIAlertView alloc] initWithTitle:@"Failed" message:@"设置失败" delegate:nil cancelButtonTitle:@"OK" otherButtonTitles:nil];
    [alertView show];
}
```

### 4.4、设置别名, 别名推送

对用户设置别名, 可以针对具体别名进行推送

```
// 绑定别名
[GeTuiSdk bindAlias:@"个推研发"];
// 取消绑定别名
[GeTuiSdk unbindAlias:@"个推研发"];
```

### 4.5、设置角标

badge是iOS用来标记应用程序状态的一个数字，出现在程序图标右上角。sdk封装badge功能，允许应用上传badge值至个推服务器，由个推后台帮助开发者管理每个用户所对应的推送badge值，简化了设置推送badge的操作。实际应用中，开发者只需将变化后的Badge值通过setBadge接口同步个推服务器，无需自己维护用户与badge值之间的对应关系，方便运营维护。

支持版本: v1.4.1及后续版本

```
[GeTuiSdk setBadge:badge]; //同步本地角标值到服务器
[[UIApplication sharedApplication] setApplicationIconBadgeNumber:badge]; //APP 显示角标需开发者调用系统方法进行设置
```

## 4.6、重置角标

重置角标, 重置服务器角标计数，计数变更为0。

支持版本: v1.4.1及后续版本

```
[GeTuiSdk resetBadge]; //重置角标计数
[[UIApplication sharedApplication] setApplicationIconBadgeNumber:0]; // APP 清空角标
```

## 5.使用CocoaPods集成

### 5.1、安装Cocopods

安装方式异常简单，Mac 下都自带 ruby，使用 ruby 的 gem 命令即可下载安装：

```
$ sudo gem install cocoapods
$ pod setup
```

### 5.2、配置Cocopods Podfile文件，导入GTSDK

使用时需要新建一个名为 Podfile 的文件，以如下格式，将依赖的库名字依次列在文件中即可：

```
platform :ios
pod 'GTSDK'
```

### 5.3、执行 Install，完成GTSDK 导入

然后将编辑好的 Podfile 文件放到你的项目根目录中，执行如下命令即可：

```
$ cd "your project home"
$ pod install
```

注：CocoaPods详细使用参考[“CocoaPods安装和使用”](#)

5.4、SDK 代码接入

注：请查看第二点 基本使用 和 第三点 高级使用 完成SDK接入。

6.iOS SDK 推送流程

iOS应用、Server、getui SDK、getui Server、Apple Push Notification Server的交互过程，如下图

