

# iOS SDK 1.2.1 集成步骤

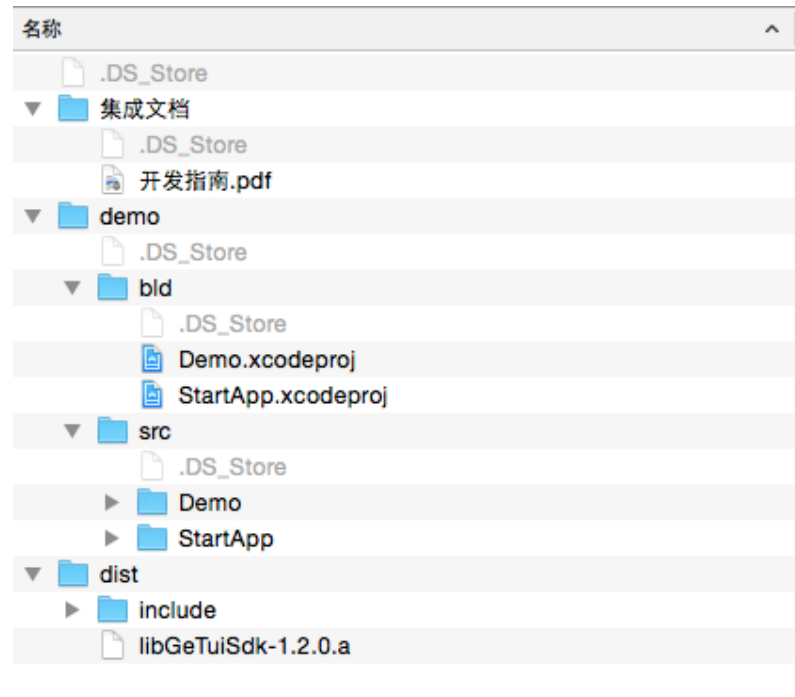
## 目录

iOS SDK 1.2.1 集成步骤.....	1
概述:.....	2
项目设置:.....	2
1. 将 dist 目录拷贝到项目工程目录下 , 导入 dist/include/所有的头文件、libGeTuiSdk-{version}.a 文件和几个系统库到 XCode 项目中。 2	
2. 添加头文件搜索目录.....	2
3. 添加依赖库 : ( 必须 ) .....	3
4. SDK 后台运行权限设置 .....	3
5. SDK 地理围栏功能 : ( 可选 ) .....	4
接入流程:.....	6
1. AppDelegate 中启动个推 SDK.....	6
2. 启动 SDK , 并设置后台开关和电子围栏开关.....	7
3. 向服务器注册 DeviceToken .....	7
4. Background Fetch 接口回调.....	8
5. 个推 SDK 支持用户设置标签 , 标示一组标签用户 , 可以针对标签用户进行推送.....	9
6.个推 SDK 支持绑定别名功能 , 对用户设置别名 , 可以针对具体别名进行推送.....	9
7. 设置 SDK Delegate 回调.....	9
iOS 应用&Server&getui SDK&getui Server 和 Apple Push Notification Server 的交互过程 .....	12

## 概述:

个推推送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。

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：



其中 dist 目录包含集成 SDK 所需的静态库和头文件。

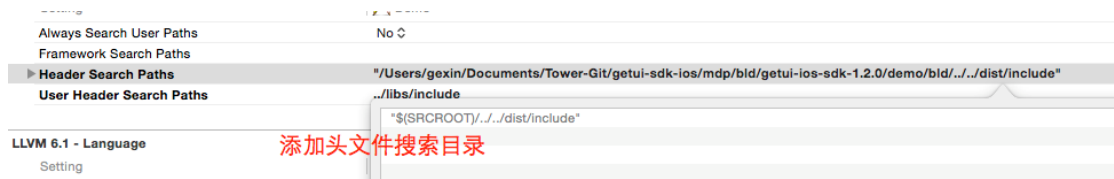
注意：libGeTuiSdk-{version}.a （version 为具体的 sdk 版本号）使用 libo 工具将支持 i386、arm64、armv7 和 armv7s 的代码打包到了一起，所以这个库将同时支持 simulator 和 device。

## 项目设置:














### 1. 个推 SDK 头文件和.a 库设置

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

### 2. 添加头文件搜索目录



### 3. 添加依赖库 : ( 必须 )

Name	Status
 libGeTuiSdk-1.2.0.a	Required ⇅
 SystemConfiguration.framework	Required ⇅
 CoreTelephony.framework	Required ⇅
 CoreLocation.framework	Required ⇅
 MapKit.framework	Required ⇅
 AVFoundation.framework	Required ⇅
 libz.dylib	Required ⇅
 libsqlite3.0.dylib	Required ⇅
 Security.framework	Required ⇅
 CFNetwork.framework	Required ⇅
 UIKit.framework	Required ⇅
 Foundation.framework	Required ⇅
 CoreGraphics.framework	Required ⇅
+ -	

系统库支持 :

ibz.dylib

libsqlite3.dylib

Security.framework

SystemConfiguration.framework

CFNetwork.framework

CoreTelephony.framework

CoreLocation.framework

AVFoundation.framework

### 4. SDK 后台运行权限设置

#### 4.1 Background Fetch 权限 : ( 必选 )

为了更好支持 SDK 推送 , APP 定期抓取离线数据 , 添加 Background Fetch 功能。

添加 background Fetch 权限 :

- Modes: ☐ Audio and AirPlay  
☐ 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

---

#### 4.2 Audio and AirPlay 权限：（可选）

Audio and AirPlay 权限开启，可以保证 SDK 后台长时间运行，保证消息实时到达。开发者后台发送透传消息将直接通过 GeTuiSdkDidReceivePayload 接口送达。（注：该功能视 APP 应用具体功能而定，如果 App 不支持后台多媒体播放，请不要勾选，勾选将有概率通不过 AppStore 审核！）

- Modes: ☒ Audio and AirPlay  
☐ 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

---

#### 5. SDK 地理围栏功能：（可选）

描述：本功能为使用个推 2.0 智能标签和个推 3.0 应景推送的必选功能，建议勾选此功能。

##### 5.1 GPS 定位权限设置（可选）

为了适配 iOS8 及以上系统：Info.plist 中需要添加 NSLocationWhenInUseUsageDescription 或者 NSLocationAlwaysUsageDescription key。

NSLocationWhenInUseUsageDescription 允许 App 前台获取 GPS 信息

NSLocationAlwaysUsageDescription 允许 App 前/后台获取 GPS 信息

Key	Type	Value
▼ Information Property List	Dictionary (18 items)	
Localization native development region	String	en
Bundle display name	String	\$(PRODUCT_NAME)
Executable file	String	\$(EXECUTABLE_NAME)
▶ Icon files	Array	(0 items)
Bundle identifier	String	com.igexin.getui.demo
InfoDictionary version	String	6.0
Bundle name	String	\$(PRODUCT_NAME)
Bundle OS Type code	String	APPL
Bundle versions string, short	String	1.0
Bundle creator OS Type code	String	????
Bundle version	String	1.0
Application requires iPhone environment	Boolean	YES
NSLocationAlwaysUsageDescription	String	请求定位Always
NSLocationWhenInUseUsageDescription	String	请求定位InUse
▶ Required background modes	Array	(1 item)
▶ Required device capabilities	Array	(1 item)
▶ Supported interface orientations	Array	(2 items)
aps-environment	String	development

其中 NSLocationAlwaysUsageDescription 或者 NSLocationWhenInUseUsageDescription key 对应的描述将会出现在请求窗口中，如果不需要描述可以设置 value 为 空。



接入流程:

## 1. AppDelegate 中启动个推 SDK

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 {  
    // [1]:使用 APPID/APPKEY/APPSECRET 创建个推实例  
    [self startSdkWith:kAppId appKey:kAppKey appSecret:kAppSecret];  
  
    // [2]:注册 APNS  
    [self registerRemoteNotification];  
  
    // [2-EXT]: 获取启动时收到的 APN 数据  
    NSDictionary*message=[launchOptionsobjectForKey:UIApplicationLaunchOptionsRemoteNotificationKey];  
  
    if (message) {  
        NSString *payloadMsg = [message objectForKey:@"payload"];  
        NSString *record = [NSString stringWithFormat:@"[APN]%@", %@, [NSDate date], payloadMsg];  
        [_viewController logMsg:record];  
    }  
    return YES;  
}
```

## 2. 启动 SDK ，并设置后台开关和地理围栏开关

```
- (void)startSdkWith:(NSString *)appId appKey:(NSString *)appKey appSecret:(NSString *)appSecret
{
    NSError *err = nil;

    //[1-1]:通过 AppId、 appKey 、 appSecret 启动 SDK
    [GeTuiSdk startSdkWithAppId:appId appKey:appKey appSecret:appSecret delegate:self
    error:&err];

    //[1-2]:设置是否后台运行开关
    [GeTuiSdk runBackgroundEnable:YES];

    //[1-3]:设置地理围栏功能，开启 LBS 定位服务和是否允许 SDK 弹出用户定位请求，请求
    NSLocationAlwaysUsageDescription 权限,如果 UserVerify 设置为 NO,需第三方负责提示用户定位授权。
    [GeTuiSdk lbsLocationEnable:YES andUserVerify:YES];

    if (err) {
        [_viewController logMsg:[NSString stringWithFormat:@"%@", [err localizedDescription]]];
    }
}
```

## 3. 当应用进入后台时通知个推 SDK 进入后台

```
- (void)applicationDidEnterBackground:(UIApplication *)application
{
    // [EXT] APP 进入后台时，通知个推 SDK 进入后台
    [GeTuiSdk enterBackground];
}
```

注：原先 stopSDK 接口。

## 4. 向服务器注册 DeviceToken

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

推送 APN 消息。

```
-(void)application:(UIApplication*)application
didRegisterForRemoteNotificationsWithDeviceToken:(NSData*)deviceToken
{
    NSString *token = [[deviceToken description]
stringByTrimmingCharactersInSet:[NSCharacterSet
characterSetWithCharactersInString:@"<>"]];
    [_deviceToken release];
    _deviceToken = [[token stringByReplacingOccurrencesOfString:@" " withString:@""]
retain];
    NSLog(@"deviceToken:%@", _deviceToken);
    // [3]:向个推服务器注册 deviceToken
    [GeTuiSdk registerDeviceToken:_deviceToken];
}
```

如果获取 DeviceToken 获取失败，也需要通知个推服务器。

```
-(void)application:(UIApplication*)application
didFailToRegisterForRemoteNotificationsWithError:(NSError*)error
{
    // [3-EXT]:如果 APNS 注册失败，通知个推服务器
    [GeTuiSdk registerDeviceToken:@""];
}
```

## 5. Background Fetch 接口回调

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 performFetchWithCompletionHandler:(void
(^)(UIBackgroundFetchResult))completionHandler {
    [GeTuiSdk resume]; // 恢复个推 SDK 运行
    completionHandler(UIBackgroundFetchResultNewData);
}
```



6. 个推 SDK 支持用户设置标签，标示一组标签用户，可以针对标签用户进行推送

```
NSArray *tagNames = [tagName componentsSeparatedByString:@"tag1,tag2"];  
[GeTuiSdk setTags: tagNames];
```

接口： + (BOOL)setTags:(NSArray \*)tags;

7.个推 SDK 支持绑定别名功能，对用户设置别名，可以针对具体别名进行推送

接口： + (void)bindAlias:(NSString \*)alias; //绑定别名

接口： + (void)unbindAlias:(NSString \*)alias; //解绑别名

```
NSString* aAlias = @"张三";  
[GeTuiSdk bindAlias:aAlias];
```

8. 设置 SDK Delegate 回调，实现 GeTuiSdkDelegate 各个接口方法

8.1 SDK 启动成功返回 cid

接口： - (void) GeTuiSdkDidRegisterClient:(NSString \*)clientId;

```
- (void)GeTuiSdkDidRegisterClient:(NSString *)clientId // SDK 返回 clientId  
{  
    // [4-EXT-1]: 个推 SDK 已注册，返回 clientId  
    [_clientId release];  
    _clientId = [clientId retain];  
}
```

8.2 SDK 收到透传消息回调

接口： - (void)GeTuiSdkDidReceivePayload:(NSString\*)payloadId andTaskId:(NSString\*) taskId  
andMessageId:(NSString\*)aMsgId fromApplication:(NSString \*)appId;

```

-(void)GeTuiSdkDidReceivePayload:(NSString*)payloadIdandTaskId:(NSString*)taskId
andMessageId:(NSString *)aMsgId fromApplication:(NSString *)appId
{
    // [4]: 收到个推消息
    [_payloadId release];
    _payloadId = [payloadId retain];
    NSData* payload = [GeTuiSdk retrivePayloadById:payloadId]; //根据 payloadId 取回 Payload
    NSString *payloadMsg = nil;
    if (payload) {
        payloadMsg = [[NSString alloc] initWithBytes:payload.bytes
                                                    length:payload.length
                                                    encoding:NSUTF8StringEncoding];
    }
    NSString *record = [NSString stringWithFormat:@"%d, %@, %@", ++_lastPaylodIndex, [self
formateTime:[NSDate date]], payloadMsg];
    NSLog(@"task id : %@, messageId:%@", taskId, aMsgId);
    [payloadMsg release];
}

```

### 8.3 SDK 收到 sendMessage 消息回调

**接口：** - (void) GeTuiSdkDidSendMessage:(NSString \*)messageId result:(int)result;

```

- (void)GeTuiSdkDidSendMessage:(NSString *)messageId result:(int)result {
    // [4-EXT]:发送上行消息结果反馈
    NSString *record = [NSString stringWithFormat:@"Received sendmessage:%@ result:%d",
messageId, result];
    [_viewController logMsg:record];
}

```

### 8.4 SDK 遇到错误回调

**接口：** - (void) GeTuiSdkDidOccurError:(NSError \*)error;

```

- (void)GeTuiSdkDidOccurError:(NSError *)error
{
    // [EXT]: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。

    [_viewController logMsg:[NSString stringWithFormat:@">>>[GexinSdk error]:%@", [error
localizedDescription]]];
}

```

## 8.5. SDK 运行状态通知

状态类型：

1. SdkStatusStarting // 正在启动
1. SdkStatusStarted // 启动
2. SdkStatusStoped // 停止

接口：- (void) GeTuiSdkDidNotifySdkState:(SdkStatus)aStatus;

```

- (void)GeTuiSdkDidNotifySdkState:(SdkStatus)aStatus {
    // [EXT]:通知 SDK 运行状态

    _sdkStatus = aStatus;

    [_viewController updateStatusView:self];
}

```

## iOS 应用&Server&getui SDK&getui Server 和 Apple Push Notification Server 的交互

### 过程

