

iOS SDK 集成指南

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。

1. 解压缩后的文件目录结构

- 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：

1、包含集成SDK所需的静态库和头文件。

2、使用情况：

(1)、在 App 内投放广告，获取 IDFA 可通过苹果审核。

(2)、App 内无广告，但由于先前投放的特定广告，可参考如下勾选，通过苹果审核。

勾选如图：

广告标识符

此 App 是否使用广告标识符 (IDFA)?

☒ 是 ☐ 否

广告标识符 (IDFA) 是每台 iOS 设备的唯一 ID，是投放定向广告的唯一方法。用户可以选择在其 iOS 设备上限制广告定位。

如果您的 App 使用广告标识符，请在提交您的代码（包括任何第三方代码）之前进行检查，以确保您的 App 仅出于下面列出的目的使用广告标识符，并尊重“限制广告跟踪”设置。如果您在 App 中加入了第三方代码，则您将对这类代码的行为负责。因此，请务必与您的第三方提供商核实，确认此类代码是否遵循广告标识符和“限制广告跟踪”设置的使用限制。

此 App 使用广告标识符来实现以下目的（选择所有适用项）：

- ☐ 在 App 内投放广告
- ☒ 标明此 App 安装来自先前投放的特定广告
- ☒ 标明此 App 中发生的操作来自先前投放的广告

如果您认为自己还有其他可以接受的广告标识符使用方式，请[联系我们](#)。

iOS 中的“限制广告跟踪”设置

- ☒ 本人，在此确认，此 App（以及与此 App 交互的任何第三方）使用广告标识符检查功能并尊重用户在 iOS 中的“限制广告跟踪”设置。当用户启用广告标识符后，此 App 不会用于 [iOS 开发人员计划许可协议](#) 中规定的“有限广告目的”之外的任何目的，以任何方式使用广告标识符，以及通过使用广告标识符获取的任何信息。

对于广告标识符 (IDFA) 的使用，请务必选择正确的答案。如果您的 App 包含 IDFA 而您选择了“否”，此二进制文件将永久被拒绝，您必须提交另一个二进制文件。

注意: 获取 IDFA 需添加 AdSupport.framework 库支持

- GtSdkLib-noidfa:

1、包含集成SDK所需的静态库和头文件。

2、使用情况:

由于 IDFA(identifier for advertising) 能够较精准的识别用户, 尤其对于广告主追踪广告转化率提供了很大帮助, 在 App 内无广告情况下还是建议开发者使用获取 IDFA 版本, 并参考(2)中所说的方式提交 AppStore 审核, 当然开发者不想使用 IDFA 或者担忧采集 IDFA 而未集成任何广告服务遭到 Apple 拒绝, 我们也准备了该无 IDFA 版本供开发者集成。

注意: 不获取 IDFA 需删除 AdSupport.framework 库支持

2. 资源内容图示



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 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 ↕

2.2 添加依赖库（必须，如下图）

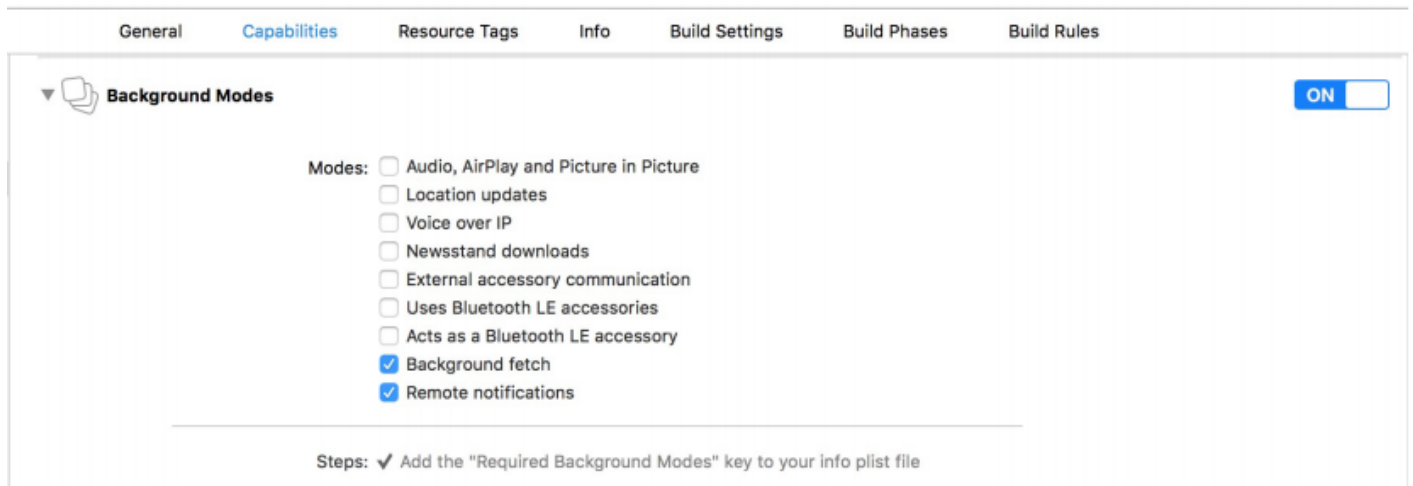
添加系统库支持：

- libc++.tbd
- libz.tbd
- libsqlite3.tbd
- Security.framework
- MobileCoreServices.framework
- SystemConfiguration.framework
- CoreTelephony.framework
- AVFoundation.framework
- JavaScriptCore.framework
- CoreLocation.framework
- CoreBluetooth.framework
- AdSupport.framework（使用无 IDFA 版本接入需删除该 AdSupport 库）

 libc++.tbd	Required ↕
 libz.tbd	Required ↕
 libsqlite3.tbd	Required ↕
 Security.framework	Required ↕
 MobileCoreServices.framework	Required ↕
 SystemConfiguration.framework	Required ↕
 CoreTelephony.framework	Required ↕
 AVFoundation.framework	Required ↕
 JavaScriptCore.framework	Required ↕
 CoreLocation.framework	Required ↕
 CoreBluetooth.framework	Required ↕
 AdSupport.framework	Required ↕
+ —	

2.3 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）



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 @"yIPfqwg6OMAPp6dkqgLpG5"
#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 registerRemoteNotification];
    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) {

        UIUserNotificationType types = (UIUserNotificationTypeAlert |
                                         UIUserNotificationTypeSound |
                                         UIUserNotificationTypeBadge);

        UIUserNotificationSettings *settings;
        settings = [UIUserNotificationSettings settingsForTypes:types categories:n
il];

        [[UIApplication sharedApplication] registerForRemoteNotifications];
        [[UIApplication sharedApplication] registerUserNotificationSettings:settin
gs];

    } else {
        UIRemoteNotificationType apn_type = (UIRemoteNotificationType)(UIRemoteNot
ificationTypeAlert |
                                                                           UIRemoteNot
ificationTypeSound |
                                                                           UIRemoteNot
ificationTypeBadge);
        [[UIApplication sharedApplication] registerForRemoteNotificationTypes:apn_
type];
    }
#else
    UIRemoteNotificationType apn_type = (UIRemoteNotificationType)(UIRemoteNotific
ationTypeAlert |
                                                                           UIRemoteNotific
ationTypeSound |
                                                                           UIRemoteNotific
ationTypeBadge);
    [[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。

```

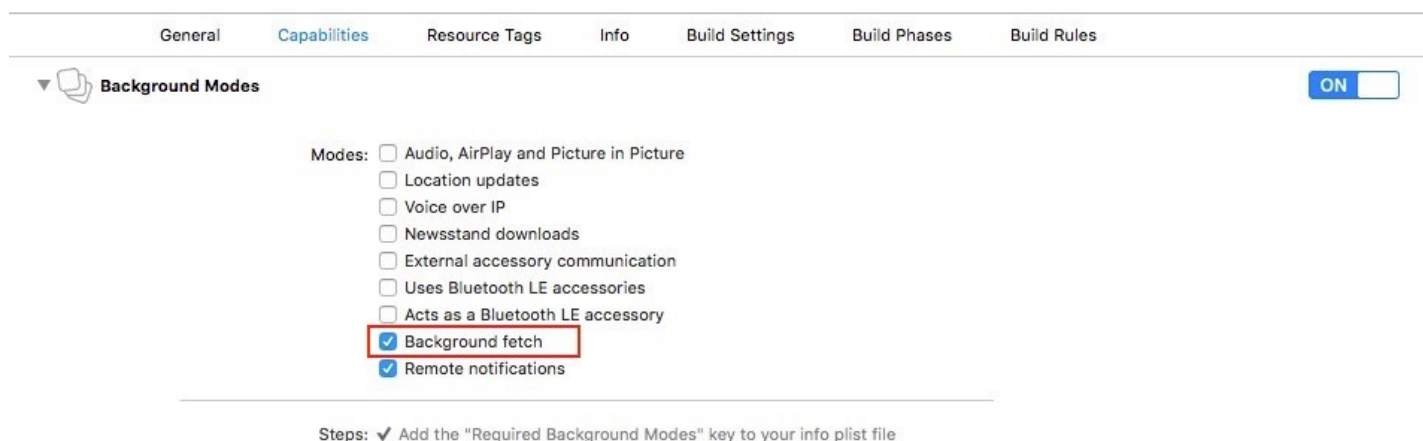
/** 远程通知注册成功委托 */
- (void)application:(UIApplication *)application didRegisterForRemoteNotifications
WithDeviceToken:(NSData *)deviceToken {
    NSString *token = [[deviceToken description] stringByTrimmingCharactersInSet:[
    NSCharacterSet characterSetWithCharactersInString:@"<>"]];
    token = [token 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 统计远程推送消息

处理APNs展示点击, 统计有效用户点击数。

```

- (void)application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary *)userInfo fetchCompletionHandler:(void (^)(UIBackgroundFetchResult))completionHandler {
    // 将收到的APNs信息传给个推统计
    [GeTuiSdk handleRemoteNotification:userInfo];
    completionHandler(UIBackgroundFetchResultNewData);
}

```

3.6 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 andTaskId:taskId andMsgId: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 configuring transparent messages. The sidebar on the left includes a dropdown menu with 'com.igexin....' and several buttons: '创建推送' (Create Push), '推送通知' (Push Notification), '透传消息' (Transparent Message), '分组对比测试' (Group Comparison Test), and '数据统计' (Data Statistics). The main content area is titled '透传消息 ?' (Transparent Message ?). It contains a form with two main fields: '描述' (Description) and '消息内容' (Message Content). The '描述' field has a placeholder '只用于后续查找识别,类似于备注' (Only for subsequent search and identification, similar to a note) and a character count of '0/30'. The '消息内容' field has a placeholder '请输入透传消息的命令代码' (Please enter the command code for the transparent message) and a character count of '0 / 800'. A red box highlights the '消息内容' field. At the bottom right of the form, there is a link labeled '参数生成工具' (Parameter Generation Tool).

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]:%@",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" andSequenceNum:@"seq-1"];
// 取消绑定别名
[GeTuiSdk unbindAlias:@"个推" andSequenceNum:@"seq-2"];
```

处理 绑定/解绑 返回:

```

- (void)GeTuiSdkDidAliasAction:(NSString *)action result:(BOOL)isSuccess sequenceNum:(NSString *)aSn error:(NSError *)aError {
    if ([kGtResponseBindType isEqualToString:action]) {
        NSLog(@"绑定结果 : %@ !, sn : %@", isSuccess ? @"成功" : @"失败", aSn);
        if (!isSuccess) {
            NSLog(@"失败原因: %@", aError);
        }
    } else if ([kGtResponseUnBindType isEqualToString:action]) {
        NSLog(@"绑定结果 : %@ !, sn : %@", isSuccess ? @"成功" : @"失败", aSn);
        if (!isSuccess) {
            NSLog(@"失败原因: %@", aError);
        }
    }
}
}

```

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

```

4.7 设置渠道

设置渠道信息，方便服务器根据渠道统计信息。

支持版本: v1.5.0及后续版本

```

[GeTuiSdk setChannelId:@"GT-Channel"];

```

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'
```

如果需要使用 不获取 IDFA 版本的库，请如下配置：

```
platform :ios
pod 'GTSDK', '1.5.0-noidfa'
```

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. 推送流程

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

