GeTuiSDK1.3.1 集成指南

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：

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 的代码打包到了一起，所以这个库将同时支持 simulator 和 device。

1、项目设置:

注意事项

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。

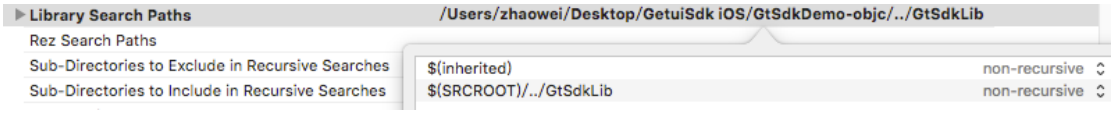
```
<key>NSAppTransportSecurity</key>
<dict>
    <key>NSAllowsArbitraryLoads</key>
    <true/>
</dict>
```

1.1 个推 SDK 头文件和.a 库设置

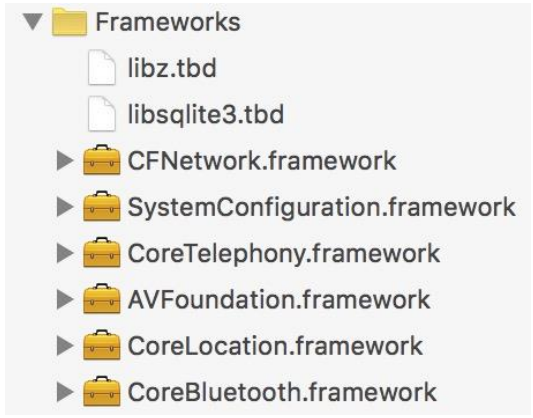
将 GtSdkLib 目录拷贝到项目工程目录下，导入 GtSdkLib 目录所有的头文件、libGeTuiSdk-{version}.a 文件和几个系统库到 XCode 项目中。

1.2 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。



1.3 添加依赖库 ：（必须）

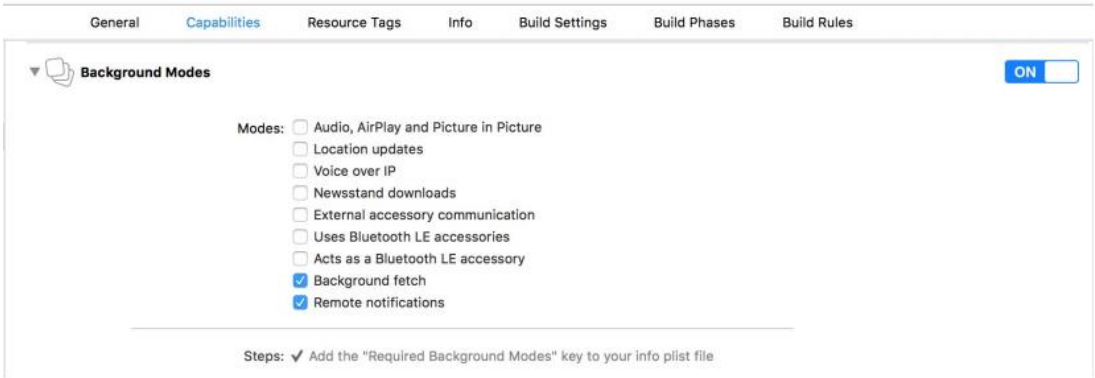


系统库支持：

libz.tbd
libsqlite3.tbd
CFNetwork.framework
SystemConfiguration.framework
CoreTelephony.framework
AVFoundation.framework
CoreLocation.framework
CoreBluetooth.framework

1.4 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：



2、基本集成:

2.1 在您的 AppDelegate.h 文件中添加以下代码

```
#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>
```

2.2 在你的 AppDelegate.m 文件中启动个推 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 {
    // Override point for customization after application launch.

    // 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];

    // 处理远程通知启动 APP
    [self receiveNotificationByLaunchingOptions:launchOptions];

    return YES;
}
```

2.3 在您的 AppDelegate.m 文件中向个推服务器注册 DeviceToken（必须）

注：注册 APNs 获取 DeviceToken 不同项目或版本会有所不同，可以参考个推“集成 Demo”中方式注册 APNs。

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。

```
/** 远程通知注册成功委托 */
```

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

如果获取 DeviceToken 获取失败，也需要通知个推服务器

```
/** 远程通知注册失败委托 */
- (void)application:(UIApplication *)application
didFailToRegisterForRemoteNotificationsWithError:(NSError *)error {
    [GeTuiSdk registerDeviceToken:@""];     /// 如果 APNS 注册失败，通
知个推服务器
    NSLog(@"\n>>>[DeviceToken Error]:%@\n\n",error.description);
}
```

2.4 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 {
    /// Background Fetch 恢复 SDK 运行
    [GeTuiSdk resume];
    completionHandler(UIBackgroundFetchResultNewData);
}
```

2.5 GeTuiSdk 注册回调，可以验证注册结果（可选）

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 {
    // [4-EXT-1]: 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 {
```

```
    // [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。
    NSLog(@"\n>>>[GexinSdk error]:%@\n\n", [error localizedDescription]);
}
```

3、更多集成方案：个推 SDK 透传消息(非 APNS)（可选）

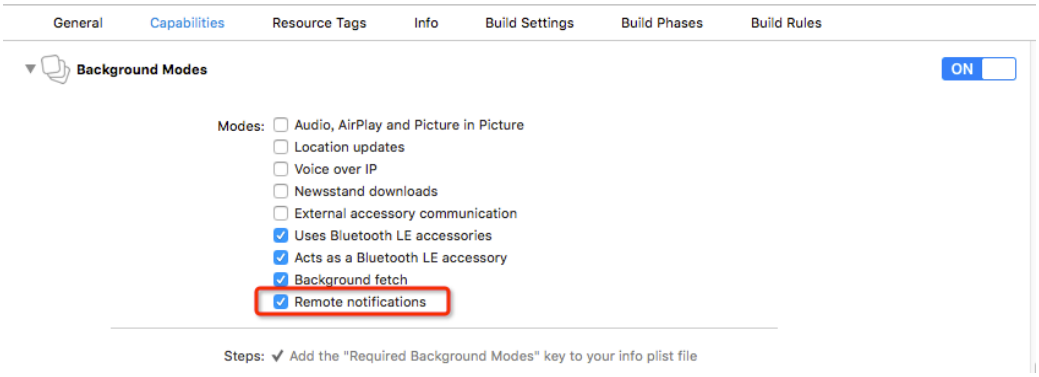
SDK 在线状态时（App 在前台运行），个推服务器会直接给您的 App 发送透传消息，不发送苹果 APNS 消息，可以更快的把消息发送到手机端；SDK 离线状态时（停止 SDK 或 App 后台运行 或 App 停止），个推服务器会给 App 发送苹果 APNS 消息，同时保存个推的离线消息（苹果 APNS 可能会推送到手机 APP 上或者用户不点击或者触发该条消息），在 SDK 在线后，SDK 会获取所有的个推透传消息，offLine 字段就是表明该条消息是否是离线消息。

```
/** SDK 收到透传消息回调 */
- (void)GeTuiSdkDidReceivePayload:(NSString *)payloadId andTaskId:(NSString *)taskId andMessageId:(NSString *)aMsgId andOffLine:(BOOL)offLine fromApplication:(NSString *)appId {

    // [4]: 收到个推消息
    NSData *payload = [GeTuiSdk retrivePayloadById:payloadId];
    NSString *payloadMsg = nil;
    if (payload) {
        payloadMsg = [[NSString alloc] initWithBytes:payload.bytes length:payload.length encoding:NSUTF8StringEncoding];
    }

    NSString *msg = [NSString stringWithFormat:@"payloadId=%@,taskId=%@,messageId:%@,payloadMsg:%@%@",payloadId,taskId,aMsgId,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:aMsgId];
}
```

注：个推透传获取的消息内容为下图中”消息内容“

4、更多集成方案：苹果 APNS 透传（可选）

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（可选）




```
/** 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);
}
```

5、更多集成方案：用户设置标签（可选）

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

```
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];
}
```

6、更多集成方案：用户设置别名（可选）

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"];
```

iOS SDK 推送流程

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