iOS百度慧推SDK集成文档

1. 背景

iOS百度慧推SDK用于协助业务方iOS APP更方便的使用Apple。
主要功能： APNs消息： 提供APNs推送服务。
其它功能： 1. 自定义消息： 提供自定义透传消息服务。
2. 多媒体消息： 提供多媒体消息推送服务。

> 注 : 本文档代码支持最低Xcode8进行编译集成，更低版本Xcode请调整适配。

本文档对应推送SDK Version：v1.0.8，发布日期：2021.2.3。

2. 项目工程集成SDK
2.1 资料文件结构
```shell
SPSDK_iOS_v1.0.8
|
 | - Document
 |     | - iOS推送SDK集成文档.pdf (推送SDK集成时参考)
 |     |
 | - SPSDKLib
 |     | - SPSDKLib.h (推送SDK头文件，通过Framework直接集成)
 |     | - SPSDK.framework (推送SDK静态库，通过Xcode直接集成)
|   |- SPSDKMessage.h (推送SDK自定义消息头文件，通过Framework直接集成)
|   |- SPSDKMessage.framework (推送SDK自定义消息静态库，通过Xcode直接集成)
|   |- SPSDKExtension.h (推送SDK多媒体扩展消息头文件，通过Framework直接集成)
|   |- SPSDKExtension.framework (推送SDK多媒体扩展消息静态库，通过Xcode直接集成))
|   |- SPLocationCenter.h(推送SDK地理围栏头文件，通过Framework直接集成)
|   |- SPSDKGeo.framework(推送SDK地理围栏静态库，通过Xcode直接集成)
|- Dependency
| |- ProtocolBuffer.framework (推送SDK序列化依赖库，通过Framework直接集成)
  |- GCD socket Source （推送SDK socket依赖库，通过Framework直接集成）
  ```

2.2 SDK集成
2.2.1 创建应用
请到“百度智能云“注册App，同时申请AppKey和SecretKey，具体请参考：
```
https://cloud.baidu.com/doc/SPP/GettingStarted.html#.E5.88.9B.E5.BB.BA.E5.BA.94.E7.94.A8
```

2.2.2 Xcode集成
推送SDK以静态库的形式分发，解压后按照以下步骤通过Xcode集成到项目工程中:

1. 将libSPSDKLib.a、SPSDKLib.h添加进工程
2. 设置【Other Linker Flags】选项为-ObjC标识
3. 在【Build Phases】->【Link Binary With Libraries】中添加系统库：
  - libz.tbd
  - UserNotifications.framework （Xcode8）
  - libsqlite3.tbd
4. 请打开【 Targets】->【Capabilities】->【Push Notifications】 处的开关设置为`ON`状态（Xcode 8）

5. 选择集成：
  - 自定义消息：将SPSDKLib_Message.a、SPSDKMessage.h 添加进工程
  - 多媒体消息：将SPSDKLib_Extension.a、SPServiceExtension.h 添加进工程

注意：

> 如需支持HTTP资源或接口进行测试，请在工程的info.plist和Extension的info.plist中增加配置：
> 1. info.plist 中添加Key：App Transport Security Settings ，类型为Dictionary
> 2. 在Dictionary中添加Key：Allow Arbitrary Loads，类型为Boolean，并设置Value为YES

3. SDK使用流程

业务方使用Xcode把推送APNs SDK集成好后，建议按照如下流程进行代码集成调用。涉及到的推送APNs SDK相关接口，在后续**"4.SDK接口介绍"**章节中详细说明。

3.1 启动推送SDK并注册APNs

```objc
#import "SPSDKLib.h" // 引入头文件

- (BOOL)application:(UIApplication *)application didFinishLaunchingWithOptions:(NSDictionary *)launchOptions {
    [SPSDKLib startSDKWithAppkey:@"appkey" secretKey:@"secretKey" CUID:@"cuid"]; // 启动SDK

    [self registerAPNs]; // 注册APNs
}

// 注册APNs，针对不同iOS版本采取不同注册方式，以下为举例代码
- (void)registerAPNs {
  if ([[UIDevice currentDevice].systemVersion floatValue] >= 10.0) {
    UNUserNotificationCenter *center = [UNUserNotificationCenter currentNotificationCenter];
    center.delegate = self;
```

```
        [center requestAuthorizationWithOptions:(UNAuthorizationOptionBadge |
UNAuthorizationOptionSound | UNAuthorizationOptionAlert | UNAuthorizationOptionCarPlay)
completionHandler:^(BOOL granted, NSError *_Nullable error) {
            // 请根据error自行处理
        }];
        [[UIApplication sharedApplication] registerForRemoteNotifications];
    } else if ([[[UIDevice currentDevice] systemVersion] floatValue] >= 8.0) {
        UIUserNotificationType types = (UIUserNotificationTypeAlert |
UIUserNotificationTypeSound | UIUserNotificationTypeBadge);
        UIUserNotificationSettings *settings = [UIUserNotificationSettings settingsForTypes:types
categories:nil];
        [[UIApplication sharedApplication] registerUserNotificationSettings:settings];
        [[UIApplication sharedApplication] registerForRemoteNotifications];
    } else {
        UIRemoteNotificationType types = (UIRemoteNotificationType)
(UIRemoteNotificationTypeAlert | UIRemoteNotificationTypeSound |
UIRemoteNotificationTypeBadge);
         [[UIApplication sharedApplication] registerForRemoteNotificationTypes:types];
    }
}
```

3.2 向推送服务器注册DeviceToken

```objc
- (void)application:(UIApplication *)application
didRegisterForRemoteNotificationsWithDeviceToken:(NSData *)deviceToken {
    //Xcode11打的包，iOS13获取Token有变化
    if ([[[UIDevice currentDevice] systemVersion] floatValue] >= 13) {
        if (![deviceToken isKindOfClass:[NSData class]]) {
            //记录获取token失败的描述
            return;
        }
        const unsigned *tokenBytes = (const unsigned *)[deviceToken bytes];
        NSString *strToken = [NSString
stringWithFormat:@"%08x%08x%08x%08x%08x%08x%08x%08x",
                              ntohl(tokenBytes[0]), ntohl(tokenBytes[1]), ntohl(tokenBytes[2]),
                              ntohl(tokenBytes[3]), ntohl(tokenBytes[4]), ntohl(tokenBytes[5]),
                              ntohl(tokenBytes[6]), ntohl(tokenBytes[7])];
        NSLog(@"deviceToken1:%@", strToken);
        [SPSDKLib registerDeviceToken:strToken];
    } else {
        NSString *token = [NSString
                           stringWithFormat:@"%@",deviceToken];
        token = [token stringByReplacingOccurrencesOfString:@"<" withString:@""];
        token = [token stringByReplacingOccurrencesOfString:@">" withString:@""];
        token = [token stringByReplacingOccurrencesOfString:@" " withString:@""];
        [SPSDKLib registerDeviceToken:token];
    }
}
```

swift

```
static func registerToken(deviceToken: Data) {
        var deviceTokenString = String()
        if #available(iOS 13.0, *) {
            let bytes = [UInt8](deviceToken)
            for item in bytes {
                deviceTokenString += String(format: "%02x", item&0x000000FF)
            }
        } else {
            let dtDataStr = NSData.init(data: deviceToken)
            deviceTokenString = dtDataStr.description.replacingOccurrences(of: "<", with:
"").replacingOccurrences(of: ">", with: "").replacingOccurrences(of: " ", with: "")
        }
        SPSDKLib.registerDeviceToken(deviceTokenString) { (error) in
            if error != nil {
                DuLogd("\(String(describing: error))")
            } else {
                DuLogd("注册设备ID 成功")
            }
        }
    }
```

注意⚠️ ： 如果Xcode版本在11以上，iOS13系统获取Token有变化需要添加如上判断。
```

### 3.3 上报收到APNs推送事件

需要在AppDelegate.m中接收到远程推送消息的代理方法中执行，跟据iOS版本有所区分。

iOS10以前处理方式：

```objc
- (void)application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary
*)userInfo fetchCompletionHandler:(void (^)(UIBackgroundFetchResult))completionHandler {

    [SPSDKLib didRevRemoteNotification:userInfo
applicationState:application.applicationState];
}
```

 iOS10及以后处理方式：

```objc
// App前台运行时触发
- (void)userNotificationCenter:(UNUserNotificationCenter *)center willPresentNotification:
(UNNotification *)notification withCompletionHandler:(void (^)
(UNNotificationPresentationOptions))completionHandler {
    NSDictionary * userInfo = notification.request.content.userInfo;
    [SPSDKLib didRevRemoteNotification:userInfo applicationState:[[UIApplication
sharedApplication] applicationState]];
}
```
```

```
// 点击通知消息，App进入前台处理时触发
- (void)userNotificationCenter:(UNUserNotificationCenter *)center
didReceiveNotificationResponse:(UNNotificationResponse *)response withCompletionHandler:
(void (^)(void))completionHandler {
    NSDictionary * userInfo = response.notification.request.content.userInfo;
        [SPSDKLib didRevRemoteNotification:userInfo applicationState:[[UIApplication
sharedApplication] applicationState]];
}
```

3.4 自定义消息接入流程

业务方需导入protocolbuffer 3.3.0的framework或者pod 'Protobuf', '~> 3.3.0'.
业务方在完整接入APNs 推送SDK、并且按照如上步骤配置完成的基础上。如需接入自定义消息，请按照以下步骤：
具体接入请参照4.11的接口介绍。

```
#define SPSDK_MESSAGE @"spsdk_message"
#define SPSDK_MESSAGEDESC @"spsdk_messageDesc"

[[SPSDKMessage sharedInstance] receiveSocketMessage:^(NSDictionary * _Nullable message,
NSError * _Nullable error) {
    if (!error) {
        NSLog(@"收到的消息:%@", message[SPSDK_MESSAGE]);
        NSLog(@"收到的描述:%@", message[SPSDK_MESSAGEDESC]);
    } else {
        NSLog(@"错误消息:%@", error);
    }
}];
```

3.5 多媒体消息接入流程

业务方在完整接入APNs 推送SDK、并且按照APNs SDK相关步骤配置完成的基础上。如需接入多媒体消息，请借鉴以下示例:
注意⚠️：这里只给出相关示例，具体接入请参照4.12的接口介绍。

```
- (void)didReceiveNotificationRequest:(UNNotificationRequest *)request withContentHandler:
(void (^)(UNNotificationContent * _Nonnull))contentHandler {
    self.contentHandler = contentHandler;
    // copy发来的通知，开始做一些处理
    self.bestAttemptContent = [request.content mutableCopy];

    // Modify the notification content here...
    self.bestAttemptContent.title = [NSString stringWithFormat:@"%@ [modified]",
self.bestAttemptContent.title];
```

```
    // 重写一些东西
    self.bestAttemptContent.title = @"我是标题";
    self.bestAttemptContent.subtitle = @"我是子标题";
    self.bestAttemptContent.body = @"我们不一样;

    // 附件
    NSDictionary *dict =  self.bestAttemptContent.userInfo;
    NSDictionary *notiDict = dict[@"aps"];
    NSString *imgUrl = [NSString stringWithFormat:@"%@",notiDict[@"media_url"]];

    // 这里添加一些点击事件，可以在收到通知的时候，添加，也可以在拦截通知的这个扩展中添
加

     self.bestAttemptContent.categoryIdentifier = @"category1";
    if (!imgUrl.length) {
        [self apnsDeliverWith:request imgUrl:NO];
        self.contentHandler(self.bestAttemptContent);
    }
    [self loadAttachmentForUrlString:imgUrl withType:@"image"
completionHandle:^(UNNotificationAttachment *attach) {
        if (attach) {
            self.bestAttemptContent.attachments = [NSArray arrayWithObject:attach];
            [self  apnsDeliverWith:request imgUrl:YES];
            NSLog(@"收到");
        } else {
            [self  apnsDeliverWith:request imgUrl:NO];
        }
    }];
}

- (void)apnsDeliverWith:(UNNotificationRequest *)request imgUrl:(BOOL)url {
    //配置
    [[SPServiceExtension sharedInstance] initSPServiceExtensionWithAppkey:@"appkey"
secretKey:@"appSecrect" cuid:nil];
    //上报
    [[SPServiceExtension sharedInstance]spserviceExtensionRecieveRequest:request
attachment:url completion:^(BOOL success) {
        if (success) {
            NSLog(@"receice extension message successful");
        }
        self.contentHandler(self.bestAttemptContent);
    }];

}

- (void)serviceExtensionTimeWillExpire {
    // Called just before the extension will be terminated by the system.
    // Use this as an opportunity to deliver your "best attempt" at modified content, otherwise
the original push payload will be used.
    self.contentHandler(self.bestAttemptContent);
```

```
}


- (void)loadAttachmentForUrlString:(NSString *)urlStr
                          withType:(NSString *)type
                 completionHandle:(void(^)(UNNotificationAttachment *attach))completionHandler{

    __block UNNotificationAttachment *attachment = nil;
    NSURL *attachmentURL = [NSURL URLWithString:urlStr];
    NSString *fileExt = [self fileExtensionForMediaType:type];

    NSURLSession *session = [NSURLSession sessionWithConfiguration:
[NSURLSessionConfiguration defaultSessionConfiguration]];
    [[session downloadTaskWithURL:attachmentURL
             completionHandler:^(NSURL *temporaryFileLocation, NSURLResponse *response,
NSError *error) {
                 if (error != nil) {
                     NSLog(@"%@", error.localizedDescription);
                 } else {
                     NSFileManager *fileManager = [NSFileManager defaultManager];
                     NSURL *localURL = [NSURL fileURLWithPath:[temporaryFileLocation.path
stringByAppendingString:fileExt]];
                      NSLog(@"sss:%@", attachmentURL);
                     [fileManager moveItemAtURL:temporaryFileLocation toURL:localURL
error:&error];

                     NSError *attachmentError = nil;
                     attachment = [UNNotificationAttachment attachmentWithIdentifier:@""
URL:localURL options:nil error:&attachmentError];
                     if (attachmentError) {
                         NSLog(@"%@", attachmentError.localizedDescription);
                     }
                 }

                  completionHandler(attachment);


             }] resume];
}

- (NSString *)fileExtensionForMediaType:(NSString *)type {
    NSString *ext = type;


     if ([type isEqualToString:@"image"]) {
        ext = @"jpg";
    }

    if ([type isEqualToString:@"video"]) {
        ext = @"mp4";
    }

    if ([type isEqualToString:@"audio"]) {
        ext = @"mp3";
```

```
    }

    return [@"." stringByAppendingString:ext];
}
```

4. SDK接口介绍及示例

APNs SDK使用的唯一入口类为`SPSDKLib`，本地通知配置内容请使用`SPNotificationConfig`对象，下面详细介绍。

自定义消息SDK使用的唯一入口类为`SPSDKMessage`。

多媒体消息SDK使用的唯一入口类为`SPServiceExtension`。

4.1 启动SDK接口

接口描述
```objc
+ (void)startSDKWithAppkey:(NSString *)appkey secretKey:(NSString *)secretKey CUID:(NSString *)cuid;
```
用于启动SDK，调用一次即可，建议`- application:didFinishLaunchingWithOptions:`方法中调用。

AppKey和SecretKey生成，请联系@陈明东[chenmingdong@baidu.com]。

参数说明
参数名	参数类型	示例	说明
appkey	NSString*	@"1000123"	注册App时生成
secretKey	NSString*	@"d392x1b0a822963506d2dbcxdd5b0336"	注册App时生成
cuid	NSString*	@"cud-xxx"	业务线唯一用户标识

返回值说明
无

4.2 注册DeviceToken接口

接口描述
```objc
+ (void)registerDeviceToken:(NSString *)deviceToken;
```
App向Apple APNs申请Device Token成功后，需要调用此接口把Device Token上传到推送服务上，用于后续推送。

由于推送服务需借助APNs实现推送消息功能，**务必调用此接口**，将客户端收到的Device Token上传到推送服务。

参数说明
参数名	参数类型	示例	说明
deviceToken	NSString*	@"9b6c8c18a0ecfabcdf68fecf61497ec3f527930f701fd1720ae60a09df7c051a"	新申请或已使用的都可以

返回值说明
无

4.3 接收远程推送消息回执接口

接口描述
```objc
+ (void)didRevRemoteNotification:(NSDictionary *)userInfo applicationState:(UIApplicationState)applicationState;
```
当客户端App接收到远程通知时，通过此接口进行统计上报。

userInfo 的获取分为两种：

- iOS10及以上举例：

  ```objc
  - (void)userNotificationCenter:(UNUserNotificationCenter *)center didReceiveNotificationResponse:(UNNotificationResponse *)response withCompletionHandler:(void (^)(void))completionHandler {
        NSDictionary * userInfo = response.notification.request.content.userInfo; // 获取userInfo
         [SPSDKLib didRevRemoteNotification:userInfo applicationState:[[UIApplication sharedApplication] applicationState]];
  }
  ```

- iOS10以下举例：

  ```objc
  - (void)application:(UIApplication *)application didReceiveRemoteNotification:(NSDictionary *)userInfo fetchCompletionHandler:(void (^)(UIBackgroundFetchResult))completionHandler {
      [SPSDKLib didRevRemoteNotification:notification.userInfo applicationState:[[UIApplication sharedApplication] applicationState]]; // 直接使用userInfo

  }
```

```

#### 参数说明
| 参数名 | 参数类型 | 示例 | 说明 |
| --- | --- |:---:|:---:|
|userInfo|  NSDictionary*  | aps消息体|接收到的远程推送消息的userInfo对象 |
|applicationState|  NS_ENUM |    UIApplicationStateActive | App处理推送消息所处的状态 |


#### 返回值说明
无

### 4.4 添加本地消息接口

#### 接口描述
```objc
+ (void)addLocalNotification:(SPNotificationConfig *)notificationConfig;
```
需要立即或定时触发本地消息时，调用此接口，notificationConfig为本地消息配置。

参数说明
参数名	参数类型	示例	说明
notificationConfig	SPNotificationConfig*	如下	包含所有消息内容和属性

SPNotificationConfig 使用举例

SPNotificationConfig作为配置本地消息使用，包含本地消息的内容，以及触发的时间配置。

* iOS10及以上举例：

```objc
// 举例1: 每周一上午8点30分提醒
SPNotificationConfig *config = [[SPNotificationConfig alloc] init];
config.repeat = YES; // 设置是否重复

NSDateComponents *components = [[NSDateComponents alloc] init];
components.weekday = 2;
components.hour = 8;
components.minute = 30;
config.dateComponents = components; // 设置触发时间

SPNotificationContent *content = [[SPNotificationContent alloc] init];
content.title = @"local notification";
```
```

```
content.body = @"xxxxxxxxxxxxxxxx";
config.content = content; // 设置消息内容

[SPSDKLib addLocalNotification:config]; // 添加本地消息
```

```objc
// 举例2: 每小时提醒一次
SPNotificationConfig *config = [[SPNotificationConfig alloc] init];
config.repeat = YES; // 设置是否重复
config.timeInterval = 60 * 60; // 设置重复时间间隔为1个小时

SPNotificationContent *content = [[SPNotificationContent alloc] init];
content.title = @"local notification";
content.body = @"xxxxxxxxxxxxxxxx";
config.content = content; // 设置消息内容

[SPSDKLib addLocalNotification:config]; // 添加本地消息
```

注意：

> dateComponents和timeInterval同时设置时，会按照dateComponents的时间设置本地推送消息

* iOS10以下举例：

```objc
// 举例3: 30秒后触发本地通知
SPNotificationConfig *config = [[SPNotificationConfig alloc] init];

NSDate *now = [NSDate date];
config.fireDate = [NSDate dateWithTimeInterval:30 sinceDate:now]; // 30秒后触发

SPNotificationContent *content = [[SPNotificationContent alloc] init];
content.title = @"local notification";
content.body = @"xxxxxxxxxxxxxxxx";
config.content = content; // 设置消息内容

[SPSDKLib addLocalNotification:config]; // 添加本地消息
```

注意：

> 针对iOS10以下， repeat 无效（config.repeat ），需App从SDK外多次创建本地消息

返回值说明

无

4.5 删除本地消息接口

接口描述
```objc
+ (void)cancelLocalNotificationWithIdentifier:(NSArray *)notificationIdentifiers; // 删除指定未出发本地消息

+ (void)clearAllLocalNotifications; // 删除全部未触发本地消息
```
两个接口均用于删除已添加未触发的本地消息。

- `+ (void)cancelLocalNotificationWithIdentifier:`方法按照指定notificationIdentifier进行删除

- `+ (void)clearAllLocalNotifications` 清空所有本地未触发消息

SPNotificationConfig对象创建时，会生成notificationIdentifier，App需要记下notificationIdentifier，用于后续的指定删除动作。

举例：

```objc
SPNotificationConfig *config = [[SPNotificationConfig alloc] init];
... // 通知配置
config.notificationIdentifier; // config创建成功，会自动生成notificationIdentifier
[SPSDKLib cancelLocalNotificationWithIdentifier:@[config.notificationIdentifier]]; // 删除
```

参数说明
参数名	参数类型	示例	说明
notificationIdentifiers	NSArray*	@[@"identifier1",@"identifier2"]	注册App时生成

返回值说明
无

4.6 badge同步接口
接口描述
```objc
+ (void)setBadge:(NSInteger)badge withCompletion:
(SPAsyncOperationCompletion)completion;
```

推送平台具备向客户端推送角标（badge）的能力，如果希望推送平台进行个性化推送，可以在客户端调用此接口同步客户端当前角标数量到推送平台。

`SPAsyncOperationCompletion`为block，用于异步回调使用，回调方法参数为`BOOL success`，表示接口是否执行完成。

举例：

```objc
[SPSDKLib setBadge:9 withCompletion:^(BOOL success) {
    // success 同步是否成功，继续执行自己的逻辑
}];
```

参数说明
参数名	参数类型	示例	说明
badge	NSInteger	9	注册App时生成

回调说明
参数名	参数类型	示例	说明
SPAsyncOperationCompletion	NSArray*	success	向推送平台同步后进行回调

4.7 批量增加Tag接口
接口描述
```objc
+ (NSString *)addTags:(NSSet<NSString *> *)tags withCompletion:
(SPAsyncOperationCompletion)completion;
```
批量设置Tag，用于根据Tag进行分类推送。

举例：

```objc
NSSet *tags = [NSSet setWithObjects:@"110", @"120", nil];
[SPSDKLib addTags:tags withCompletion:^(BOOL success) {
  // success 同步是否成功，继续执行自己的逻辑
}];
```

参数说明

参数名	参数类型	示例	说明
tags	NSSet		需要增加的tag集合

回调说明
参数名	参数类型	示例	说明
SPAsyncOperationCompletion	BOOL	success	向推送平台同步后进行回调

4.8 其他Tag操作接口
接口描述
```objc
// 删除指定Tag集合
+ (NSString *)deleteTags:(NSSet<NSString *> *)tags withCompletion:
(SPAsyncOperationCompletion)completion;

// 批量更新Tag，删除旧Tag
+ (NSString *)setTags:(NSSet<NSString *> *)tags withComletion:
(SPAsyncOperationCompletion)completion;

// 清除所有Tags
+ (NSString *)clearTagsWithComletion:(SPAsyncOperationCompletion)completion;

// 查询现有Tags，结果在completion中返回
+ (NSString *)getTagsWithCompletion:(void(^)(BOOL, NSSet<NSString *> *))completionBlock;
```
上述四个接口为对tag的修改和删除操作，传入参数和block使用方法同上4.7.

参数说明

参数名	参数类型	示例	说明
tags	NSSet		需要增加的tag集合

回调说明
参数名	参数类型	示例	说明
SPAsyncOperationCompletion	BOOL	success	向推送平台同步后进行回调

4.9 获取PushId接口
接口描述
```objc
+ (NSString *)getPushId;
```
参数说明
无

返回说明
| 返回值 | 类型 | 示例 | 说明 |

| --- | --- |:---:|:---:|
|PushId| NSString | CEA0CCA6F89D33BFE35F22592B6237401CA662CC | |

4.10 别名Alias接口操作
接口描述

```objc
// 设置Alias
+ (void)setAlias:(NSString *)alias
completion:(SPAsyncOperationCompletion)completion;

//  删除alias
+ (void)deleteAlias:(SPAsyncOperationCompletion)completion;

//  查询当前alias
+ (void)getAlias:(void(^)(BOOL success, NSString *aliasName))completion;
```
上述三个接口为对alias的设置、删除、查询操作。

参数说明

参数名	参数类型	示例	说明
alias	NSString		需要设置的alias别名

回调说明
参数名	参数类型	示例	说明
SPAsyncOperationCompletion| BOOL	success	向推送平台同步后进行回调	NSString
aliasName | 返回绑定后的别名 |

4.11 自定义消息接口
接口描述

```
//单例方法
+ (SPSDKMessage *)sharedInstance;

/**
接收透传消息:
@param completion 回调函数
*/
- (void)receiveSocketMessage:(SPAsyncMessageCompletion)completion;
```
上述接口为接收透传消息内容与描述。

回调说明
参数名	参数类型	示例	说明
SPAsyncMessageCompletion| 异步消息回调函数 | message | 透传消息 | error | 错误代码 |

| 错误代码描述:
-1001 连接认证失败
-1002 绑定服务器失败
-1003 读取消息失败
-1004 心跳消息失败
-1005 重启服务限流
-1006 服务器配置失败
-1007 ecc认证加密错误
-1008 服务端空消息错误
|

4.12 多媒体消息接口
接口描述

```
//单例方法
+ (instancetype)sharedInstance;

/**
 @param appkey  渠道Appkey，需要提前申请
 @param secretKey 配合渠道Appkey, 需要提前申请
 @param cuid 用于统计设备的cuid
 */
- (void)initSPServiceExtensionWithAppkey:(NSString *_Nonnull)appkey secretKey:(NSString *_Nonnull)secretkey cuid:(NSString *_Nullable)cuid;

/**
@param request 多媒体消息请求参数
@param status 是否可以接收到多媒体资源
 1：通过链接下载可以收到多媒体资源
 0：通过链接下载无法收到多媒体资源
@param completion 回执是否成功
*/
- (void)spserviceExtensionRecieveRequest:(UNNotificationRequest *_Nonnull)request attachment:(BOOL)status completion:(completionBlock)completion;
```

上述接口为启动extension注册与回执结果接口。

参数说明

参数名	参数类型	示例	说明
appkey	NSString	渠道Appkey
secretKey	NSString	配合渠道Appkey的绑定码
cuid	NSString	用于统计设备的cuid
request	UNNotificationRequest	多媒体消息请求参数
status	BOOL	是否可以接收到多媒体资源
completionBlock	Block	回执是否成功

回调说明
参数名	参数类型	示例	说明
completionBlock | 上报回执回调函数 | BOOL | success | 回执成功 |

5 地理围栏
接口描述
```
+ (void)geoStart
```

6 业务支持

集成和使用过程中，又任何问题，请及时联系：

- 申请AppKey，SecretKey，请联系@惠宇[huiyu@baidu.com]
- 客户端集成问题，请联系@龚佳亮[gongjialiang@baidu.com]、@臧剑超[zangjianchao@baidu.com]
- 推送服务集成问题，请联系@梁骏宇[liangjunyu@baidu.com]