SDK 开发指南

—IOS

日期	操作人	对应 sdk 版本	备注
2014/07/14	nick	V1.0.0	初稿
2014/07/28	nick	V1.1.0	优化流程和 API 接口、支持群组聊天
2014/08/09	nick	V1.2.0	增加联系人管理功能 优化 API 和内部流程，修复部分 BUG。 为提高兼容性，发布包从原来的 static framework 结构改变为 static library 结构 。

1. 前言.....5
2. 准备工作.....5
2.1. 申请应用的appID和appKey.....5
2.2. 导入SDK包.....5
3. 代码使用说明.....7
3.1. 简单样例.....7
3.1.1. 引入恩布API头文件.....7
3.1.2. 环境初始化.....7
3.1.3. 用户登录.....8
3.1.4. 呼叫对方用户.....8
3.1.5. 发送信息.....8
3.1.6. 登出系统.....9
3.1.7. 重载实现回调代理方法.....9
4. API说明.....11
4.1. 初始化应用环境.....11
4.1.1. 设置服务器访问地址.....11
4.1.2. 获取SDK入口全局实例.....11
4.1.3. 初始化应用参数.....11
4.2. 登录授权.....12
4.2.1. 普通用户登录.....12
4.2.2. 用户登出.....12
4.2.3. 获取当前登录用户信息.....12

4.2.4. 获取当前登录用户组织架构信息.....12
4.3. 会话操作.....13
4.3.1. 发起会话呼叫邀请(一对一会话).....13
4.3.2. 发起会话呼叫邀请(群组会话).....13
4.3.3. 应答被呼叫邀请.....13
4.3.4. 挂断已有会话.....14
4.3.5. 发送富文本消息.....14
4.3.6. 发送文件.....15
4.3.7. 取消发送文件.....15
4.3.8. 接收富文本信息.....16
4.3.9. 接收文件.....16
4.3.10. 获取会话信息.....16
4.4. 回调代理事件.....17
4.4.1. 接收到会话邀请.....17
4.4.2. 主动发起会话邀请中.....17
4.4.3. 会话已经接通.....17
4.4.4. 对方拒绝会话邀请.....17
4.4.5. 对方忙，未响应会话邀请.....18
4.4.6. 会话断开.....18
4.4.7. 接收到聊天消息.....18
4.4.8. 即将开始接收文件.....18
4.5. 联系人管理.....19

4.5.1. 增加或编辑联系人.....19

4.5.2. 删除联系人.....19

4.5.3. 获取联系人信息.....19

1. 前言

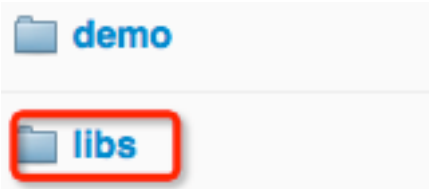
说明文档适用于所有合作伙伴的苹果 IOS 版客户端应用，在接入之前，请确认您已在恩布开放平台（http://www.entboost.com/）注册，并已获得 App ID, App Key。如还未获取，请参考业务文档。

2. 准备工作

2.1. 申请应用的appID和appKey

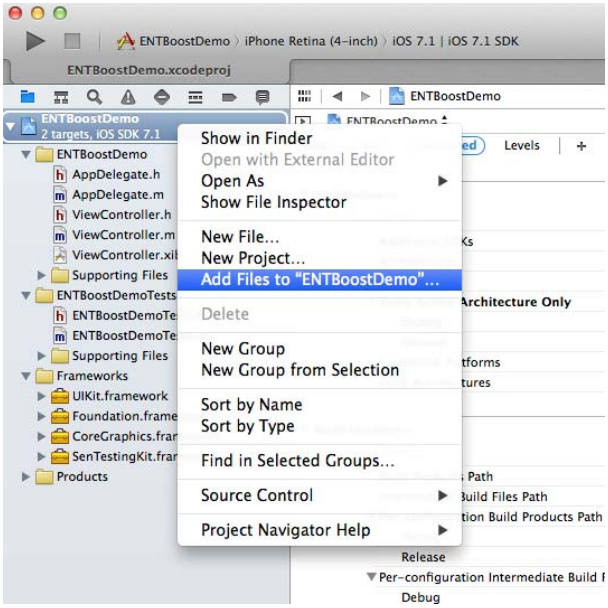
见恩布官方网站 http://www.entboost.com

2.2. 导入SDK包

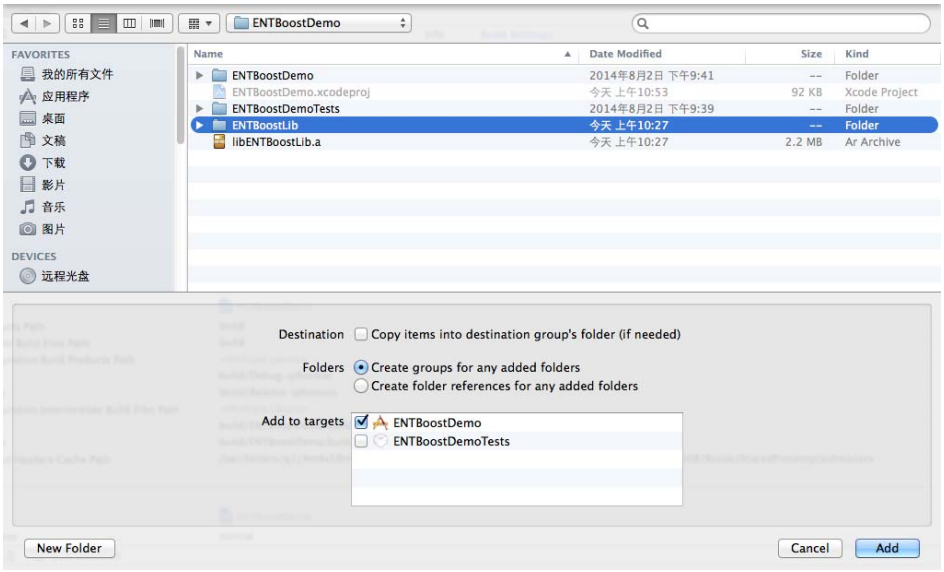


(获取地址：http://git.oschina.net/akee/entboost/tree/master/src/ios)

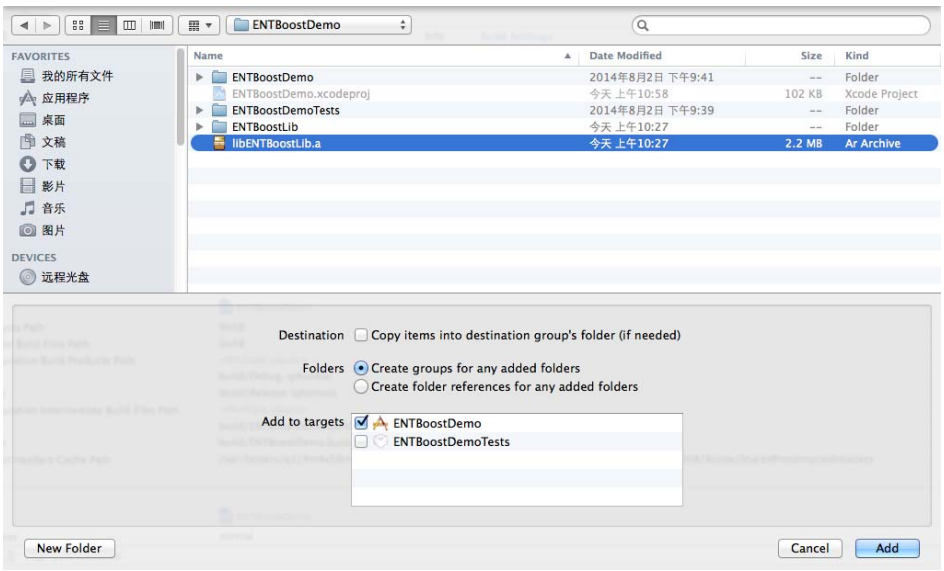
. 将获取的工具包解压，提取出 libs 文件夹下的 IOS static library 包(ENTBoostLib)
. 将获取到的 ENTBoostLib 包解压缩后复制到您的工程目录下并导入。

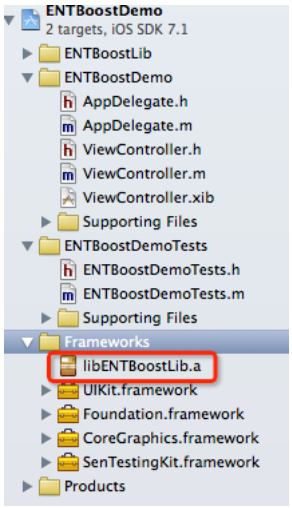


第一步：先导入头文件

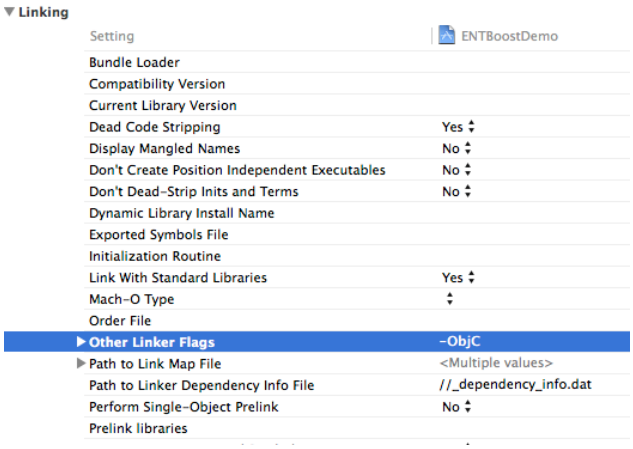


第二步：导入静态库文件





第三步：在 Build Settings -> Linking -> Other Linker Flags 中增加-ObjC 值



3. 代码使用说明

3.1. 简单样例

3.1.1. 引入恩布API头文件

```
#import "ENTBoost.h"
```

3.1.2. 环境初始化

(由于初始化过程需要一些时间，因此尽量不要把这个工作放主线程)

```
//设置服务器访问地址
[ENTBoostKit setServerAddress:@"entboost.entboost.com:18012"];
```

```
//获取恩布 API 全局单实例
ENTBoostKit* _ebkit = [ENTBoostKit sharedToolKit];

//初始化应用环境
[_ebkit initWithAppId:appId appKey:appKey andDelegate:self onCompletion:^{
        NSLog(@"init app success");
    } onFailure:^(NSError *error) {
    }];
```

3.1.3. 用户登录

```
[_ebkit logonSyncWithAccount:account password:password onCompletion:^(EBAccountInfo *accountInfo) {
        NSLog(@"logon success, username = %@", accountInfo.userName);
    } onFailure:^(NSError *error) {
    }];
```

3.1.4. 呼叫对方用户

执行呼叫后可能将有几种事件返回: onCallConnected: onCallReject: onCallBusy: onCallAlerting:

```
//呼叫单个用户
[_ebkit callUserWithAccount: @"auto-reply-test@entboost.com"];

//呼叫群组
[_ebkit callGroupWithDepCode:99089];
```

3.1.5. 发送信息

```
//检查会话是否已经存在
//一对一会话，depCode=0
EBCallInfo* callInfo = [_ebkit callInfoWithAccount:toAccount depCode:0];
if(!callInfo) { //还没有与对方建立会话
    [_ebkit callUserWithAccount:toAccount];
} else { //已经接通
    EBChatText* chat = [[EBChatText alloc] initWithText:text];
    EBMessage* message = [[EBMessage alloc] init];
    [message addChatDot:chat];
    [_ebkit sendMessage:message forCallId:callInfo.callId onCompletion:^{
        NSLog(@"发送文本完毕");
    } onFailure:^(NSError *error) {
```

```
        }];
}
```

3.1.6. 登出系统

```
[_ebkit logoff];
```

3.1.7. 重载实现回调代理方法

```
//接收到被邀请对话的事件
- (void)onCallIncoming:(const EBCallInfo *)callInfo fromUid:(uint64_t)fromUid fromAccount:(NSString *)fromAccount
vCard:(EBVCard *)vCard clientAddress:(NSString *)clientAddress
{
    //应答是否接受邀请
    [_ebkit ackTheCall:callInfo.callId accept:TRUE onCompletion:^{
        NSLog(@"onCallIncoming ack the call success");
    } failure:^(NSError *error) {
    }];
}

//会话已连通事件
- (void)onCallConnected:(const EBCallInfo *)callInfo
{
    NSLog(@"onCallConnected event callId = %llu", callInfo.callId);
}
//会话已挂断事件
- (void)onCallHangup:(const EBCallInfo *)callInfo
{
    NSLog(@"onCallHangup event callId = %llu", callInfo.callId);
}

//接收到信息
- (void)onRecevieMessage:(EBMessage *)message
{
    NSArray* chats = message.chats;
    [chats enumerateObjectsUsingBlock:^(id obj, NSUInteger idx, BOOL *stop) {
        EBChat* chatDot = obj;
        switch (chatDot.type) {
            case EB_CHAT_ENTITY_TEXT:
            {
                NSString* txt = ((EBChatText*)chatDot).text;
                NSLog(@"[%i]文本段:%@", idx, txt);
```

```
                });
            }
                break;
            case EB_CHAT_ENTITY_RESOURCE:
            {
                EBChatResource* resDot = (EBChatResource*)chatDot;
                EBEmotion* expression = resDot.expression;
                NSLog(@"[%i]资源(表情)段:resId = %llu,  文件路径  = %@", idx, expression.resId,
expression.filepath);
            }
                break;
            case EB_CHAT_ENTITY_IMAGE:
            {
                EBChatImage* imageDot = (EBChatImage*)chatDot;
                //NSData* imageData    = imageDot.data;
                UIImage* image = imageDot.image;
                CGSize size = image.size;
                NSLog(@"[%i]图片段:大小%fX%f", idx, size.width, size.height);
            }
                break;
            case EB_CHAT_ENTITY_AUDIO:
            {
                EBChatAudio* audioDot = (EBChatAudio*)chatDot;
                NSLog(@"[%i]语音段:长度(字节)%i", idx, audioDot.data.length);
                dispatch_async(dispatch_get_main_queue(), ^{
            }
                break;
        }
    }];
}

//即将接收文件，询问是否接收并要求传入要保存的绝对路径
- (void)onWillRecevieFileForCall:(uint64_t)callId msgId:(uint64_t)msgId msgTime:(NSDate*)msgTime
fromUid:(uint64_t)fromUid fileName:(NSString *)fileName fileSize:(uint64_t)fileSize accept:(BOOL *)accept
savedFilePath:(NSString**)savedFilePath
{
    *accept = TRUE;
    *savedFilePath = [EBFileUtility receivedFilePathWithFileName:fileName]; //可以是其它路径，由上层应用决定
}
```

4. API说明

4.1. 初始化应用环境

4.1.1. 设置服务器访问地址

```
/**设置服务器访问地址,在首次获取 ENTBoostKit实例前设置才有效，不设置默认访问官方测试服务
 * @param serverAddress 服务器访问地址，格式如：192.168.0.1:18012
 */
+ (void)setServerAddress:(NSString*)serverAddress;
```

4.1.2. 获取SDK入口全局实例

```
//获取全局单例
+ (ENTBoostKit*)sharedToolKit;
```

用法：

```
ENTBoostKit * _ebKit = [ENTBoostKit sharedClient];
```

本实例将是所有 `API` 的调用入口。

4.1.3. 初始化应用参数

```
/**初始化应用参数
 * @param appId 应用ID
 * @param appKey 应用key
 * @param delegate SDK事件回调代理
 * @param completionBlock 成功后的回调函数
 * @param failureBlock 失败后的回调函数
 */
- (void)initWithAppId:(NSString*)appId appKey:(NSString*)appKey andDelegate:(id)delegate
onCompletion:(void(^)(void))completionBlock onFailure:(void(^)(NSError *error))failureBlock;
```

4.2. 登录授权

4.2.1. 普通用户登录

```
/**用户登录(同步调用方式)
 * @param account 用户账号
 * @param password 用户密码
 * @param completionBlock 成功后的回调函数
 * @param failureBlock 失败后的回调函数
 */
- (void)logonSyncWithAccount:(NSString*)account password:(NSString*)password
onCompletion:(void(^)(EBAccountInfo *accountInfo))completionBlock onFailure:(void(^)(NSError *error))failureBlock;
```

成功回调参数：

accountInfo 当前登录用户信息

失败回调参数:

error 失败原因

注意：本 API 是同步模式，登录过程视网络情况而持续几秒时间

4.2.2. 用户登出

```
///用户登出
- (void)logoff;
```

4.2.3. 获取当前登录用户信息

```
///当前登录用户信息
-  (EBAccountInfo*)accountInfo;
```

4.2.4. 获取当前登录用户组织架构信息

```
///企业信息
- (EBEnterpriseInfo*)enterpriseInfo;

///企业部门信息字典, {key = depCode的NSNumber对象, obj = EBGroupInfo对象}
- (NSDictionary*)entGroupInfos;

///个人群组(非企业部门)信息字典, {key = depCode的NSNumber对象, obj = EBGroupInfo对象}
- (NSDictionary*)personalGroupInfos;
```

```
///企业部门或个人群组中成员信息字典, {key = empCode的NSNumber对象, obj = EBMemberInfo对象}
- (NSDictionary*)memberInfos;
```

4.3. 会话操作

4.3.1. 发起会话呼叫邀请(一对一会话)

```
/**邀请对方用户进行一对一会话
 * 邀请过程中，如果持续时间较长，将执行回调事件onCallAlerting:
 * 如果邀请成功后将执行回调事件onCallConnected:
 * 如果失败后将执行回调事件onCallReject:或者onCallBusy:
 * @param account 用户账号
 * @param failureBlock 失败后的回调函数
 */
- (void)callUserWithAccount:(NSString*)account onFailure:(void(^)(NSError *error))failureBlock;
```

相关事件有:

```
- (void)onCallAlerting
- (void)onCallConnected;
- (void)onCallReject;
- (void)onCallBusy;
```

详细说明请见回调代理事件部分

4.3.2. 发起会话呼叫邀请(群组会话)

```
/**邀请对方用户进行一对一会话
 * 如果邀请成功后将执行回调事件onCallConnected:
 * @param depCode 企业部门或群组编号
 * @param failureBlock 失败后的回调函数
 */
- (void)callGroupWithDepCode:(uint64_t)depCode onFailure:(void(^)(NSError *error))failureBlock;
```

相关事件有:

```
- (void)onCallConnected;
```

详细说明请见回调代理事件部分

4.3.3. 应答被呼叫邀请

```
/**回复会话邀请，用于收到onCallIncoming:事件时回复对方
```

```
 * @param callId  会话编号
 * @param accept  是否接受通话, TRUE=接受, FALSE=拒绝
 * @param completionBlock  成功后的回调函数
 * @param failureBlock  失败后的回调函数
 */
- (void)ackTheCall:(uint64_t)callId accept:(BOOL)accept onCompletion:(void(^)(void))completionBlock
failure:(void(^)(NSError* error))failureBlock;
```

失败回调参数:

error	失败原因

用途：当收到呼叫请求时，用于回应

4.3.4. 挂断已有会话

```
/**挂断会话
 * @param callId  会话编号
 * @param completionBlock  成功后的回调函数
 * @param failureBlock  失败后的回调函数
 */
- (void)closeTheCall:(uint64_t)callId onCompletion:(void(^)(void))completionBlock failure:(void(^)(NSError*
error))failureBlock;
```

失败回调参数:

error	失败原因

注意：只有一对一会话才可以挂断，如不想继续和对方通话可使用本 API

4.3.5. 发送富文本消息

```
/**发送富文本消息
 * @param message  富文本消息
 * @param callId  会话编号
 * @param completionBlock  发送成功后调用的block
 * @param failureBlock      发送失败后调用的block
 */
- (void)sendMessage:(EBMessage*)message forCallId:(uint64_t)callId onCompletion:(void(^)(void))completionBlock
onFailure:(void(^)(NSError *error))failureBlock;
```

失败回调参数:

error	失败原因

4.3.6. 发送文件

```
/**发送文件
 * @param path 文件路径
 * @param callId 会话编号
 * @param offFile 是否发送离线文件
 * @param beginBlock 开始发送文件时调用的block
 * @param processingBlock 发送过程中不定期调用的block，用于提醒发送进度
 * @param completionBlock 发送成功后调用的block
 * @param cancelBlock 发送被拒绝或取消后调用的block
 * @param failureBlock 发送失败后调用的block
 */
- (void)sendFileAtPath:(NSString*)path forCallId:(uint64_t)callId offFile:(BOOL)offFile onBegin:(void(^)(uint64_t msgId))beginBlock onProcessing:(void(^)(double_t percent, uint64_t callId, uint64_t msgId))processingBlock onCompletion:(void(^)(void))completionBlock onCancel:(void(^)(BOOL initiative))cancelBlock onFailure:(void(^)(NSError *error))failureBlock;
```

[beginBlock]开始发送文件回调参数:

msgId 信息编号

[processingBlock]发送过程回调参数:

percent 完成百分比，完成时为100
callId 会话编号
msgId 信息编号

[cancelBlock]取消发送回调参数:

initative 是否发送方主动取消, TRUE=发送方取消,FALSE=接收方取消

[failureBlock]失败回调参数:

error 失败原因

4.3.7. 取消发送文件

```
/**取消发送文件
 * @param msgId 消息编号
 * @param callId 会话编号
 * @param completionBlock 发送成功后调用的block
 * @param failureBlock 发送失败后调用的block
 */
- (void)cancelSendingFileWithMsgId:(uint64_t)msgId forCallId:(uint64_t)callId onCompletion:(void(^)(void))completionBlock onFailure:(void(^)(NSError *error))failureBlock;
```

[failureBlock]失败回调参数:

error 失败原因

4.3.8. 接收富文本信息

当收到如下事件，表示接收到富文本信息

- (void)onRecevieMessage:(EBMessage*)message;详细说明请见回调代理事件部分

4.3.9. 接收文件

收到如下事件，表示将要接收文件；应对*accept 和*savedFilepath 进行赋值以告知 SDK 是否接收文件及保存位置

```
/**即将开始接收文件
 * @param callId 会话编号
 * @param msgId 消息编号
 * @param msgTime 消息发送时间
 * @param fromUid 文件发送方的用户编号
 * @param fileName 文件名
 * @param fileSize 文件大小(字节s)
 * @param accept (输出参数)是否接收文件,TRUE=接收，FALSE=拒绝
 * @param savedFilePath (输出参数)如确认接收文件，应返回将要保存文件的全路径
 */
- (void)onWillRecevieFileForCall:(uint64_t)callId msgId:(uint64_t)msgId msgTime:(NSDate*)msgTime
fromUid:(uint64_t)fromUid fileName:(NSString *)fileName fileSize:(uint64_t)fileSize accept:(BOOL *)accept
savedFilePath:(NSString**)savedFilePath;
```

4.3.10. 获取会话信息

```
/**获取会话信息
 * @param callId 会话编号
 * @return 会话信息
 */
- (EBCallInfo*)callInfoWithCallId:(uint64_t)callId;

/**获取会话信息
 * @param account 用户账号，寻找一对一会话时不可以为nil
 * @param depCode 群组编号，等于0表示寻找一对一会话，大于0表示寻找群组会话
 * @return 会话信息
 */
- (EBCallInfo*)callInfoWithAccount:(NSString*)account depCode:(uint64_t)depCode;
```

4.4. 回调代理事件

4.4.1. 接收到会话邀请

```
/**接收到会话邀请
 * @param callInfo 会话信息
 * @param fromUid 发起邀请的用户编号
 * @param fromAccount 发起邀请的用户账号
 * @param vCard 电子名片
 * @param clientAddress 发起方用户客户端登录地址信息
 */
- (void)onCallIncoming:(const EBCallInfo*)callInfo fromUid:(uint64_t)fromUid fromAccount:(NSString*)fromAccount
vCard:(EBVCard*)vCard clientAddress:(NSString*)clientAddress;
```

当接收到本事件时，应使用以下API进行回应

```
- (void)ackTheCall:(uint64_t)callId accept:(BOOL)accept onCompletion:(void(^)(void))completionBlock
failure:(void(^)(NSError* error))failureBlock;
```

4.4.2. 主动发起会话邀请中

```
/**正在进行会话邀请
 * @param callInfo 会话信息
 * @param toUid 被邀请方
 */
- (void)onCallAlerting:(const EBCallInfo*)callInfo toUid:(uint64_t)toUid;
```

当我方呼叫他方用户(非同企业、非同群组内的用户)时，触发本事件；可用于弹出等待窗口等场景。

4.4.3. 会话已经接通

```
/**会话已经接通
 * @param callInfo 会话信息
 */
- (void)onCallConnected:(const EBCallInfo*)callInfo;
```

收到本事件，表示可以与对方进行信息交流(互发)

4.4.4. 对方拒绝会话邀请

```
/**被邀请方拒绝通话
 * @param callInfo 会话信息
 */
```

```
- (void)onCallReject:(const EBCallInfo*)callInfo;
```

4.4.5. 对方忙，未响应会话邀请

```
/**被邀请方忙，未应答
 * @param callInfo 会话信息
 */
- (void)onCallBusy:(const EBCallInfo*)callInfo;
```

4.4.6. 会话断开

```
/**断开会话
 * @param callInfo 会话信息
 */
- (void)onCallHangup:(const EBCallInfo*)callInfo;
```

无论会话是主动断开还是被动断开，如果需要再次通话都需要重新发起会话邀请 CALL

4.4.7. 接收到聊天消息

```
/**接收到聊天消息
 * @param message 消息
 */
- (void)onRecevieMessage:(EBMessage*)message;
```

4.4.8. 即将开始接收文件

```
/**即将开始接收文件
 * @param callId 会话编号
 * @param msgId 消息编号
 * @param msgTime 消息发送时间
 * @param fromUid 文件发送方的用户编号
 * @param fileName 文件名
 * @param fileSize 文件大小(字节s)
 * @param accept 是否接收文件(输出参数),TRUE=接收，FALSE=拒绝
 * @param savedFilePath 如确认接收文件，应返回将要保存文件的全路径
 */
- (void)onWillRecevieFileForCall:(uint64_t)callId msgId:(uint64_t)msgId msgTime:(NSDate*)msgTime
fromUid:(uint64_t)fromUid fileName:(NSString *)fileName fileSize:(uint64_t)fileSize accept:(BOOL *)accept
savedFilePath:(NSString**)savedFilePath;
```

4.5. 联系人管理

4.5.1. 增加或编辑联系人

```
/**新增或编辑联系人
    如contactInfo.contact已经存在，则是编辑操作；
    如contactInfo.contact不存在，则是新增操作
 * @param contactInfo  联系人信息
 * @param completionBlock  发送成功后调用的block
 * @param failureBlock      发送失败后调用的block
 */
- (void)saveOrUpdateContactInfo:(EBContactInfo*)contactInfo completion:(void (^)(void))completionBlock failure:(void (^)(NSError *error))failureBlock;
```

[failureBlock]失败回调参数:

error 失败原因

4.5.2. 删除联系人

```
/**新增或编辑联系人
 * @param contact  联系人账号
 * @param completionBlock  发送成功后调用的block
 * @param failureBlock      发送失败后调用的block
 */
- (void)delContact:(NSString*)contact completion:(void (^)(void))completionBlock failure:(void (^)(NSError *error))failureBlock;
```

[failureBlock]失败回调参数:

error 失败原因

4.5.3. 获取联系人信息

```
//联系人信息, {key = contact, obj = EBContactInfo对象}
- (NSDictionary*)contactInfos;
```