互动直播

MAC 端集成

产品文档

腾讯云

文档目录

MAC 端集成
下载代码

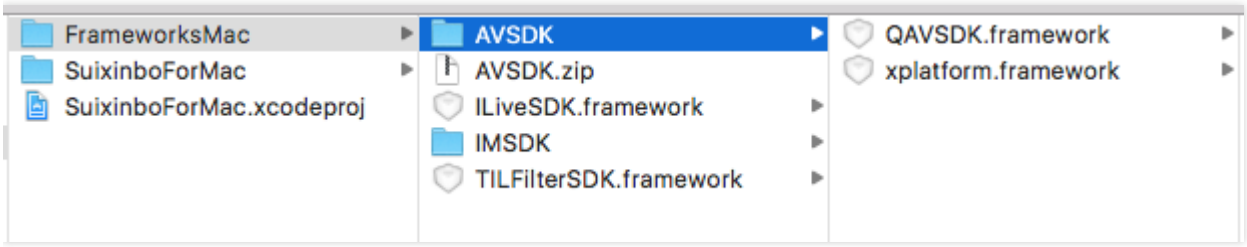
最近更新时间：2018-06-15 18:33:48

下载 Demo

单击下载 Mac Demo 的代码。演示了包含界面和后台交互的完整直播流程。

解压 SDK 压缩包

由于 GitHub 上有 100MB 文件上传限制，所以随心播工程中将 QAVSDK 压缩后再上传，要使随心播正常运行，请开发人员解压 iLiveSDK_Mac_Suixinbo/SuixinboForMac/FrameworksMac/AVSDK.zip 到当前目录，解压后目录如下图所示。

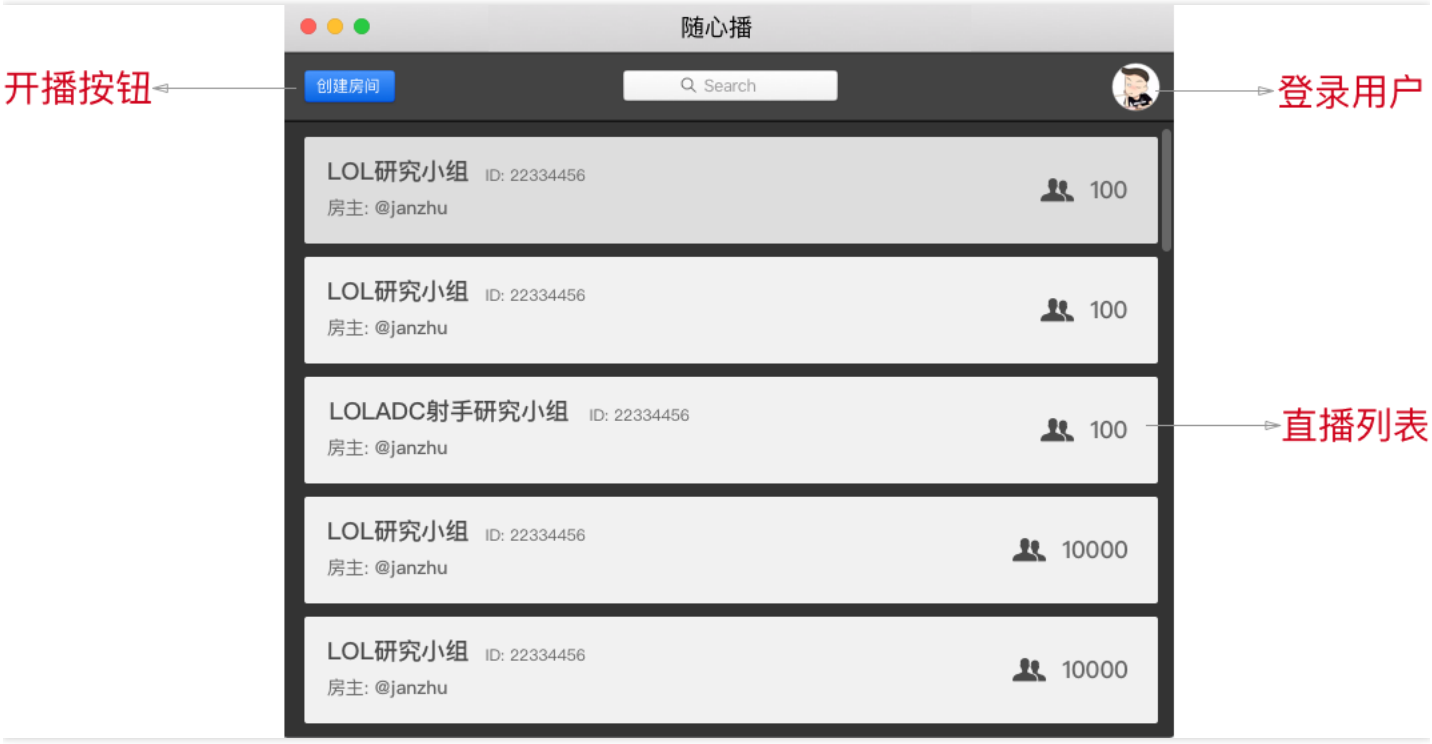


运行

直接运行安装包（下载的 Demo 工程中，SuiXinBoForMac.dmg 可以直接运行）。

编译源码运行工程（最低支持 MacOS 10.7）。

效果图如下。



集成到开发者自己的代码工程里

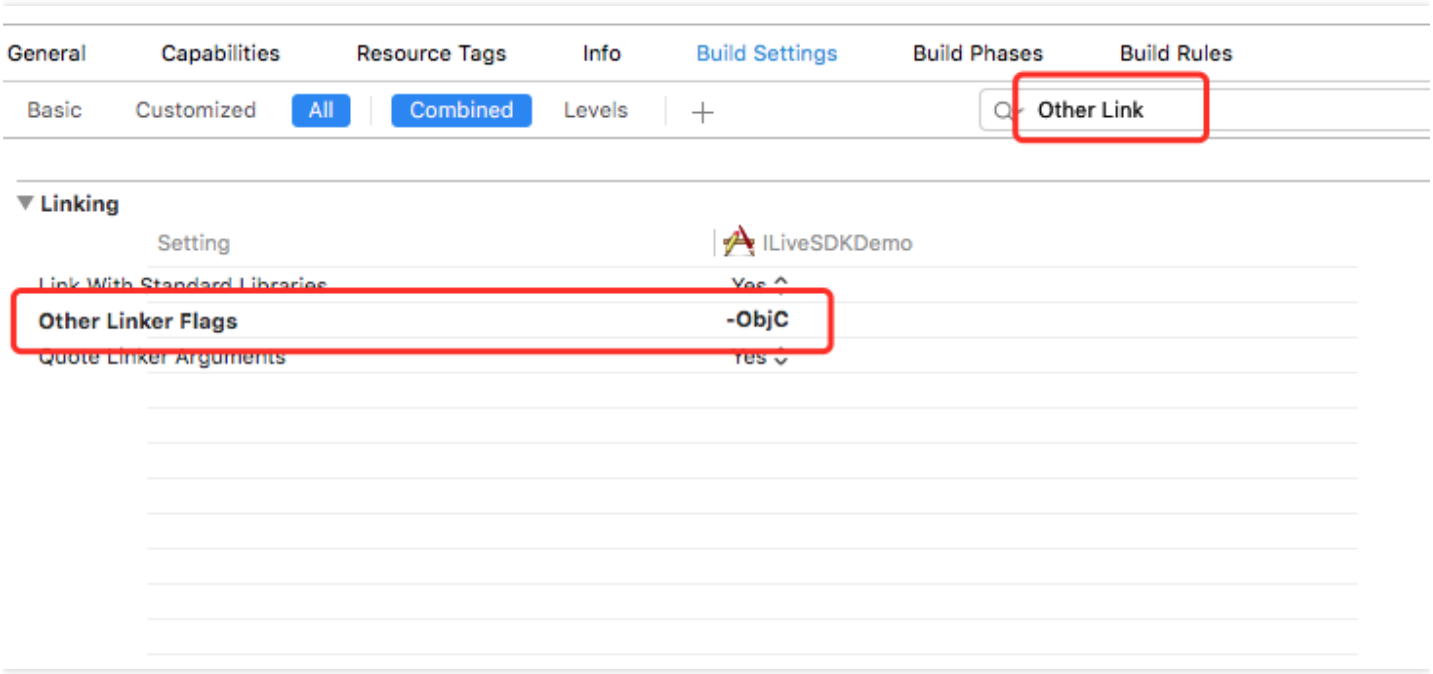
引入 SDK 并导入项目

参照以上步骤，并将 FrameworksMac 中的所有 SDK 添加到自己的工程中。

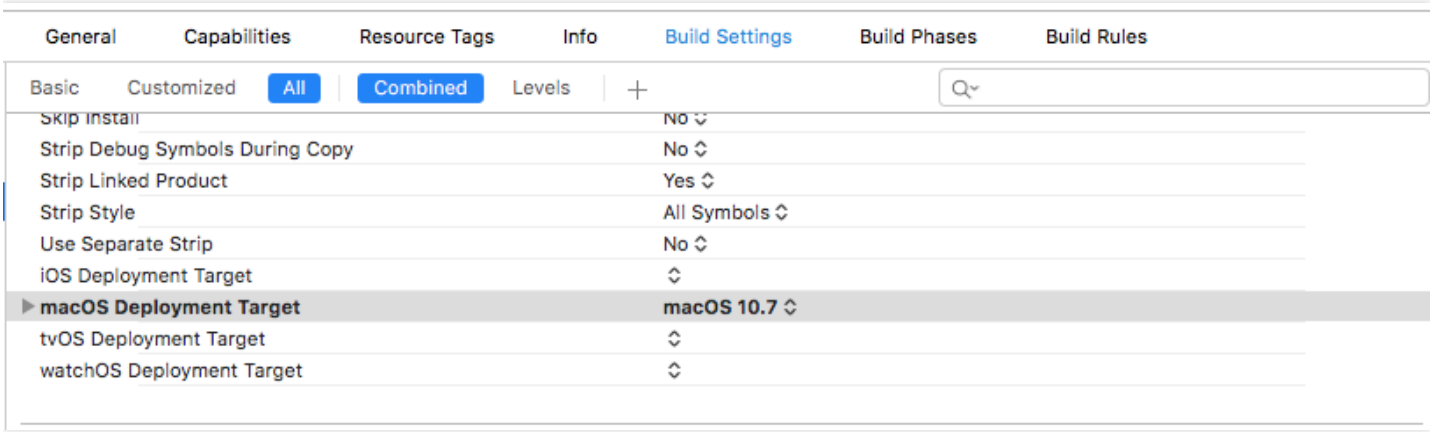
修改工程配置

将下载好的 SDK 复制到工程目录下，工程目录右键，Add Files to " you projectname"，在 Demo 中如下图所示。

Build Settings/Linking/Other Linker Flags，增加 -ObjC 配置，如下图所示。



设置最低版本大于或等 10.7，【Build Settings】>【macOS Deployment Target】>【macOS 10.7】，如下图所示。



若上述步骤均无误，则工程编译可以通过了。

添加系统库

添加以下系统库比较方便的方法是直接从随心播工程中，将 SystemLibrarys 组拖到自己的工程目录下。

需要增加的系统库
QuartzCore.framework
CoreTelephony.framework
CoreWLAN.framework
Foundation.framework
SystemConfiguration.framework
libc++.tbd
libiconv.tbd
libresolv.9.tbd
libsqlite3.tbd
libstdc++.6.tbd
libz.tbd

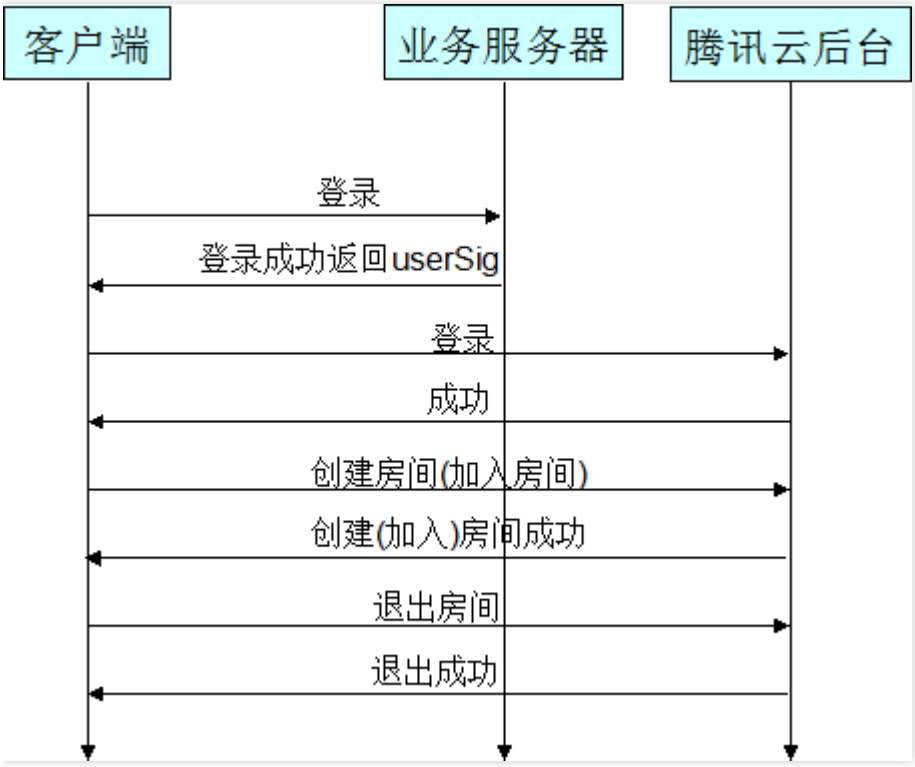
添加腾讯相关依赖库

名称	所在文件夹	说明	是否必须
QAVSDK.framework	FrameworksMac/AVSDK/	音视频 SDK	必须
xplatform.framework	FrameworksMac/AVSDK/	音视频跨平台支持库	必须
IMCore.framework	FrameworksMac/IMSDK/	即时通讯核心库	必须
ImSDK.framework	FrameworksMac/IMSDK/	即时通讯 Mac 平台封装库	必须
QALSDK.framework	FrameworksMac/IMSDK/	即时通讯网络模块 SDK	必须
TLSSDK.framework	FrameworksMac/IMSDK/	即时通讯登录服务 SDK	必须
ILiveSDK.framework	FrameworksMac/	互动直播基础功能 SDK	必须
TILFilterSDK.framework	FrameworksMac/	互动直播美颜依赖库	非必须

直播接口

最近更新时间：2018-09-14 16:11:52

ILiveSDK 直播流程图



初始化 ILiveSDK

在应用启动时初始化 ILiveSDK。

接口名	接口描述
initSdk: accountType:	ILiveSDK 内部类初始化，告知 AppId。内部包含了 IMSDK 的初始化

参数类型	参数名	说明
int	appId	传入业务方 appid
int	accountType	传入业务方 accountType

示例：

```
[[ILiveSDK getInstance] initSdk:SuixinboSdkAppId accountType:SuixinboAccountType];
```

帐号登录

托管模式

托管模式：用户帐号系统托管到腾讯云。详情请参阅 托管模式。

接口名	接口描述
tlsLogin: pwd: succ: failed:	托管模式登录到腾讯云后台

参数类型	参数名	说明
NSString	uid	用户在托管模式下注册的帐号
NSString	pwd	用户在托管模式下注册帐号的密码
TCIVoidBlock	succ	登录成功回调
TCIErrorBlock	failed	登录失败回调

示例：

```
[[ILiveLoginManager getInstance] tlsLogin:@"这里是帐号 id" pwd:@"这里是登录密码" succ:^{
NSLog(@"登录成功");
} failed:^(NSString *moudle, int errId, NSString *errMsg) {
NSLog(@"登录失败");
}];
```

独立模式

独立模式：用户帐号系统由用户自己的服务器维护。独立模式需要业务后台生成 Sig，客户端拿到这个 Sig 再登录腾讯云后台。详情请参阅 独立模式。

接口名	接口描述
iLiveLogin: sig: succ: failed:	独立模式登录到腾讯云后台

参数类型	参数名	说明
NSString	uid	用户在独立模式下注册的帐号
NSString	sig	用户在业务方后台获取到的签名
TCIVoidBlock	succ	登录成功回调
TCIErrorBlock	failed	登录失败回调

示例：

```
[[ILiveLoginManager getInstance] iLiveLogin:@"这里是帐号 id" sig:@"这里是签名字符串" succ:^{
NSLog(@"登录成功");
} failed:^(NSString *moudle, int errId, NSString *errMsg) {
NSLog(@"登录失败");
}];
```

创建房间（进入房间）

主播创建房间

接口名	接口描述
createRoom: option: succ: failed:	主播创建直播间，自动渲染本地画面，开始直播

参数类型	参数名	说明
int	roomId	房间号。业务方后台生成的房间号，需保证唯一性
ILiveRoomOption	option	主播创建房间时的配置项，使用 defaultHostLiveOption 接口获取主播默认配置即可
TCIVoidBlock	succ	创建房间成功回调
TCIErrorBlock	failed	创建房间失败回调

示例：

```
//如果使用美颜 SDK，需要设置本地画面代理，详情参考 LiveWindowController+Beauty.m 中的实现
//[[ILiveRoomManager getInstance] setLocalVideoDelegate:self];
```

```
//开发者可以详细了解下 option 的各个参数配置，这很重要
ILiveRoomOption *option = [ILiveRoomOption defaultHostLiveOption];
option.controlRole = @"user"; //这里填写开发者自己的账号系统下的角色名
option.roomDisconnectListener = self; //房间失去连接的回调通知
option.memberStatusListener = self; //房间内用户的事件回调
__weak typeof(self) ws = self;
[[ILiveRoomManager getInstance] createRoom:(int)_item.info.roomnum option:option succ:^{
NSLog(@"创建房间成功");
//如果要设置麦克风音量，则需要设置下面两个代理，详细实现参照 Demo 的 LiveWindowController+Audio.m 文件
[[[ILiveSDK getInstance] getAVContext].audioCtrl registerAudioDataCallback:QAVAudioDataSource_VoiceDispose];
[[[ILiveSDK getInstance] getAVContext].audioCtrl registerAudioDataCallback:QAVAudioDataSource_NetStream];
} failed:^(NSString *module, int errId, NSString *errMsg) {
NSLog(@"创建房间失败,module=%@,code=%d,msg=%@",module,errId,errMsg);
}];
```

观众进入房间

接口名	接口描述
joinRoom: option: succ: failed:	观众进入直播间，自动拉去远程画面并渲染，开始观看直播

参数类型	参数名	说明
int	roomId	房间号。业务方后台生成的房间号，需保证唯一性
ILiveRoomOption	option	观众进入房间时的配置项，使用 defaultGuestLiveOption 接口获取观众默认配置即可
TCIVoidBlock	succ	进入房间成功回调
TCIErrorBlock	failed	进入房间失败回调

特别注意: 普通观众加入房间时，应该配置authBits无上行权限，仅观看权限。否则会和主播一样走核心机房DC，产生高额费用。

示例：

```
//参数意义见创建房间
//用户成功加入房间后，如果打开摄像头，则会收到 onEndpointsUpdateInfo 回调，在 onEndpointsUpdateInfo 回调中，添加上渲染视图即可
ILiveRoomOption *option = [ILiveRoomOption defaultGuestLiveOption];
option.controlRole = _item.info.roleName;
option.memberStatusListener = self;
[[ILiveRoomManager getInstance] joinRoom:(int)_item.info.roomnum option:option succ:^{
NSLog(@"加入房间成功");
} failed:^(NSString *module, int errId, NSString *errMsg) {
NSLog(@"加入房间失败.M=%@,errId=%d,errMsg=%@",module,errId,errMsg);
}];
```

若以上步骤均无误，则主播开始直播，观众观看直播的整个流程就结束了。

互动消息和上麦接口

最近更新时间：2018-06-15 18:33:58

互动消息

iLiveSDK(Windows)提供了消息通讯的功能。基于消息通讯可以实现房间内成员的群消息，两个用户之间的 C2C 消息（不用加好友）。当前 SDK 只支持发文本消息和自定义消息。

发送 C2C 消息

C2C 消息指的是两个用户之间的点对点聊天消息。邀请房间成员上麦，点赞等消息可以在业务层通过 C2C 自定义消息实现。具体的实现可以参考 Demo。

接口名	接口描述
sendC2CMessage	发送 C2C 消息

参数类型	参数名	说明
const char *	dstUser	接收方 ID
const Message &	message	IM 消息内容
iLiveSuccCallback	suc	发送消息成功回调
iLiveErrCallback	err	发送消息失败回调
void*	data	用户自定义数据的指针，在成功和失败的回调函数中原封不动地返回

示例：

```
void OnSendC2CMsgSuc( void* data )
{
//发送 C2C 消息成功
}
void OnSendC2CMsgErr( int code, const char *desc, void* data )
{
//发送 C2C 消息失败
}
Message message;
MessageTextElem *elem = new MessageTextElem("hello");
```

```
message.elems.push_back(elem);
GetILive()->sendC2CMessage( message, OnSendC2CMsgSuc, OnSendC2CMsgErr, NULL );
```

发送群消息

当前仅支持给当前所在房间发群消息，房间内其他成员都会收到该消息。

接口名	接口描述
sendGroupMessage	发送群消息

参数类型	参数名	说明
const Message &	message	消息内容
iLiveSuccCallback	suc	发送消息成功回调
iLiveErrCallback	err	发送消息失败回调
void*	data	用户自定义数据的指针，在成功和失败的回调函数中原封不动地返回

示例：

```
void OnSendGroupMsgSuc( void* data )
{
//发送群消息成功
}
void OnSendGroupMsgErr( int code, const char *desc, void* data )
{
//发送群消息失败
}
Message message;
MessageTextElem *elem = new MessageTextElem("hello");
message.elems.push_back(elem);
GetILive()->sendGroupMessage( message, OnSendGroupMsgSuc, OnSendGroupMsgErr, NULL );
```

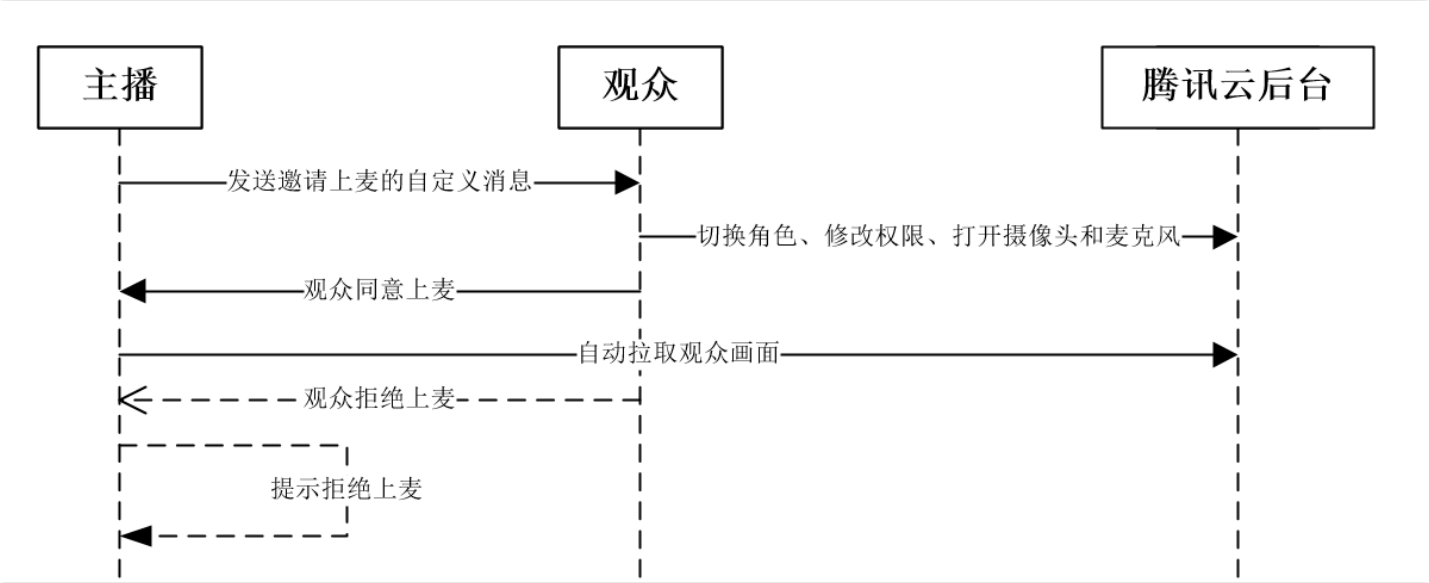
接收消息

设置消息监听回调后，SDK 每次收到消息都会通知上层。

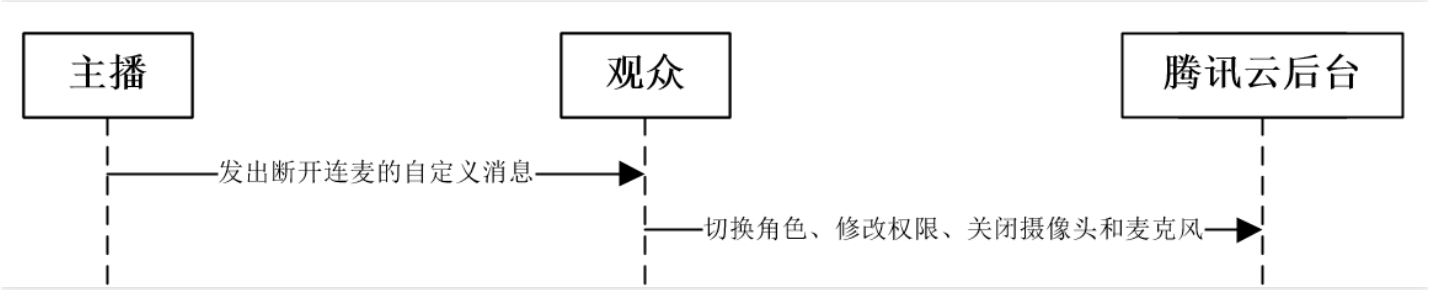
```
void OnMessage(const Message& message)
{
std::string szSender = message.sender.c_str();
for (size_t i = 0; i < message.elems.size(); ++i)
{
MessageElem *pElem = message.elems[i];
switch( pElem->type )
{
case TEXT:
{
MessageTextElem *elem = static_cast<MessageTextElem*>(pElem);
std::cout << elem->content.c_str() << std::endl;
break;
}
case CUSTOM:
{
MessageCustomElem *elem = static_cast<MessageCustomElem*>(pElem);
std::string szDate = elem->data.c_str();
break;
}
default:
break;
}
}
}
GetILive()->setMessageCallBack(OnMessage, NULL);
```

连麦互动

主播邀请上麦流程



主播断开连麦观众流程



接口

iLiveSDK 未对上/下麦进行封装，用户可以参考随心播的 sendC2CCustomCmd() 和 sendGroupCustomCmd() 函数，发送自定义消息作为邀请上麦和接受邀请的信令；观众上麦和下麦，需要切换用户角色和修改用户权限。

切换角色：

接口名	接口描述
changeRole	更改角色

参数类型	参数名	说明
const char *	szControlRole	角色字符串(由用户 App 的控制台配置)
iLiveSuccCallback	suc	成功的回调函数

参数类型	参数名	说明
iLiveErrCallback	err	失败的回调函数
void*	data	用户自定义数据的指针，在成功和失败的回调函数中原封不动地返回

示例：

```
void OnChangeRoleSuc( void* data )
{
//切换角色成功
}
void OnChangeRoleErr( int code, const char *desc, void* data )
{
//切换角色失败
}
GetILive()->changeRole(Role, OnChangeRoleSuc, OnChangeRoleErr, NULL);
```