

# 互动直播

## iOS 端集成

### 产品文档



腾讯云

---

**【版权声明】**

©2013-2019 腾讯云版权所有

本文档著作权归腾讯云单独所有，未经腾讯云事先书面许可，任何主体不得以任何形式复制、修改、抄袭、传播全部或部分本文档内容。

**【商标声明】**

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。

**【服务声明】**

本文档意在向客户介绍腾讯云全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或模式的承诺或保证。

## 文档目录

iOS 端集成

下载代码

直播接口

互动消息和上麦接口

# iOS 端集成

## 下载代码

最近更新时间：2019-05-16 11:01:28

## 下载 Demo

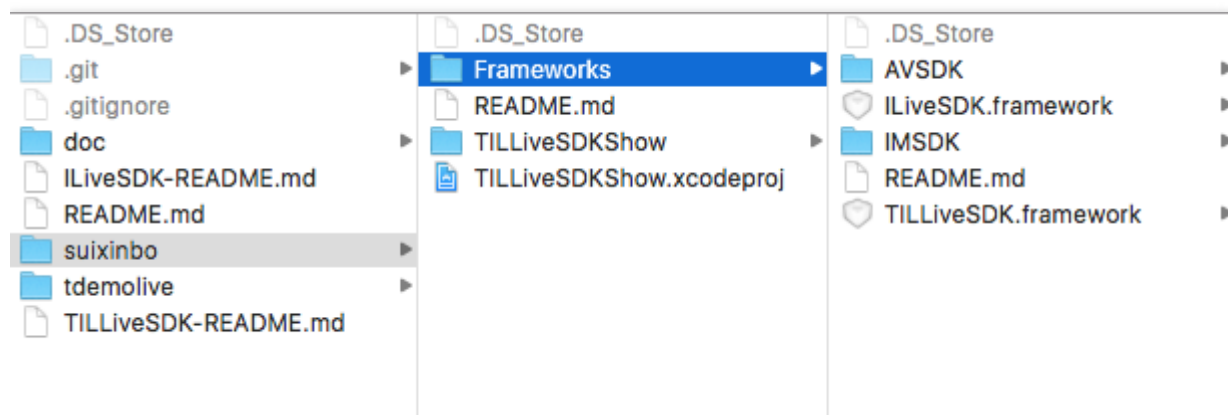
单击下载 [iOS Demo](#) 的代码。代码里包含两个示例。

- tdemolive 目录下是一个最简单的互动直播示例，演示了最关键的几个接口的调用。使用方法可以参考 GitHub 上的说明。
- suixinbo 目录下是新版随心播代码，演示了包括界面和后台交互的完整的直播流程。

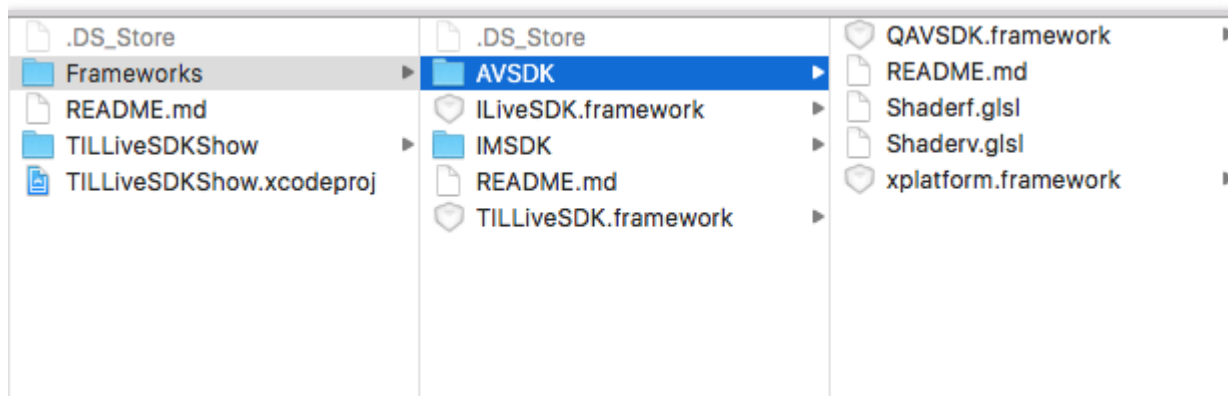
## 下载并导入 Frameworks

下载 [ILiveSDK](#) , [TILLiveSDK](#) , [AVSDK](#) , [IMSDK](#) , 并解压到工程目录 suixinbo/Frameworks 下，工程最后的目录如下图：

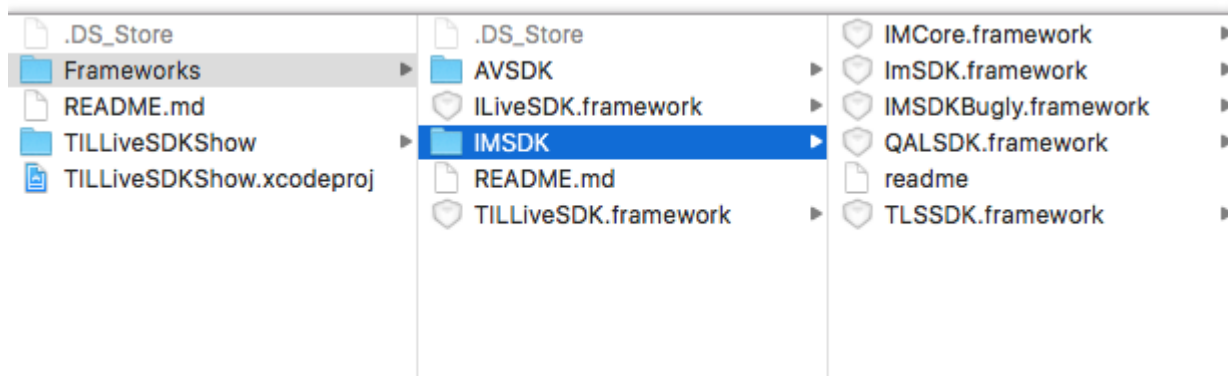
Frameworks 目录



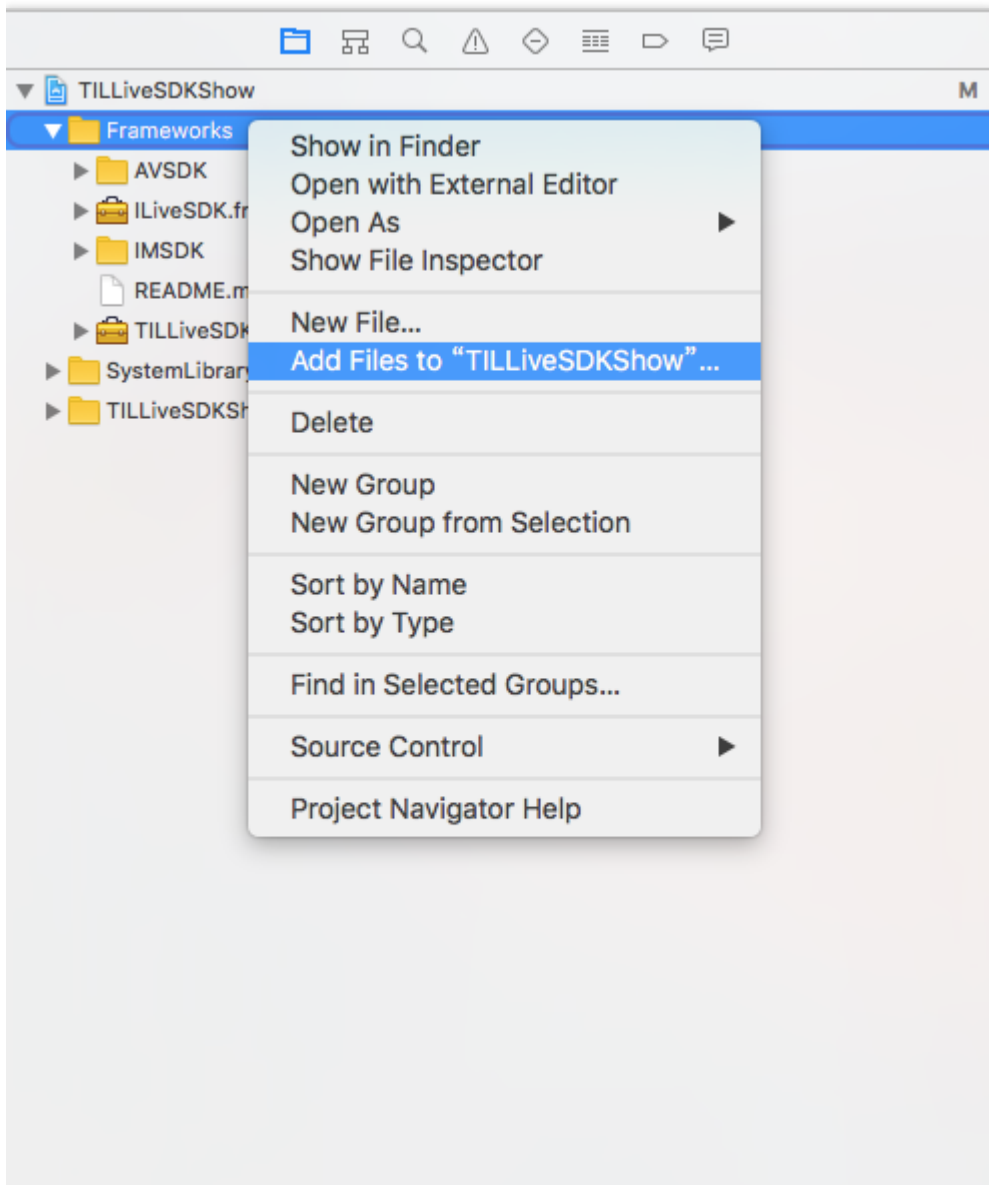
AVSDK 目录



## IMSDK 目录



导入 Frameworks，将下载好的 SDK 复制到工程目录下，单击右键，Add Files to "your projectname"，在 Demo 中如下图所示：



## 运行

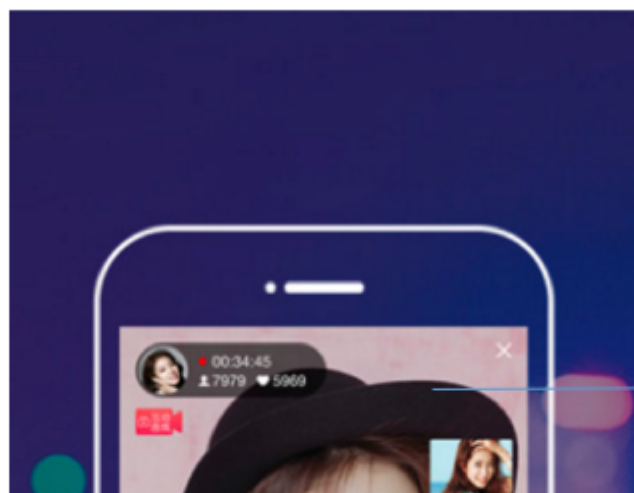
编译运行工程。（如果 xcode8 编译不过，修改 Bundle Identifier，随心播工程上的 Bundle Identifier 在用户真机上可能无法运行，用户重新修改下 Bundle Identifier 即可，比如在原有 ID 后面加 1）。





正在直播的列表

开直播按钮



主播



## 集成到开发者自己的代码工程里

### 引入 SDK 并导入项目

参照 [下载并导入 Frameworks](#)。

### 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 (如果有 -all\_load 配置，需要去掉)，如下图所示。

General Capabilities Resource Tags Info Build Settings Build Phases Build Rules

Basic Customized All Combined Levels + Other Link

## ▼ Linking

Setting

ILiveSDKDemo

Link With Standard Libraries

Yes ^

Other Linker Flags

-ObjC

Quote Linker Arguments

yes v

Build Settings/Linking/Bitcode , 增加 Bitcode 配置 , 设置为 NO , 如下图所示。

General Capabilities Resource Tags Info Build Settings Build Phases Build Rules

Basic Customized All Combined Levels + Bitcode

## ▼ Build Options

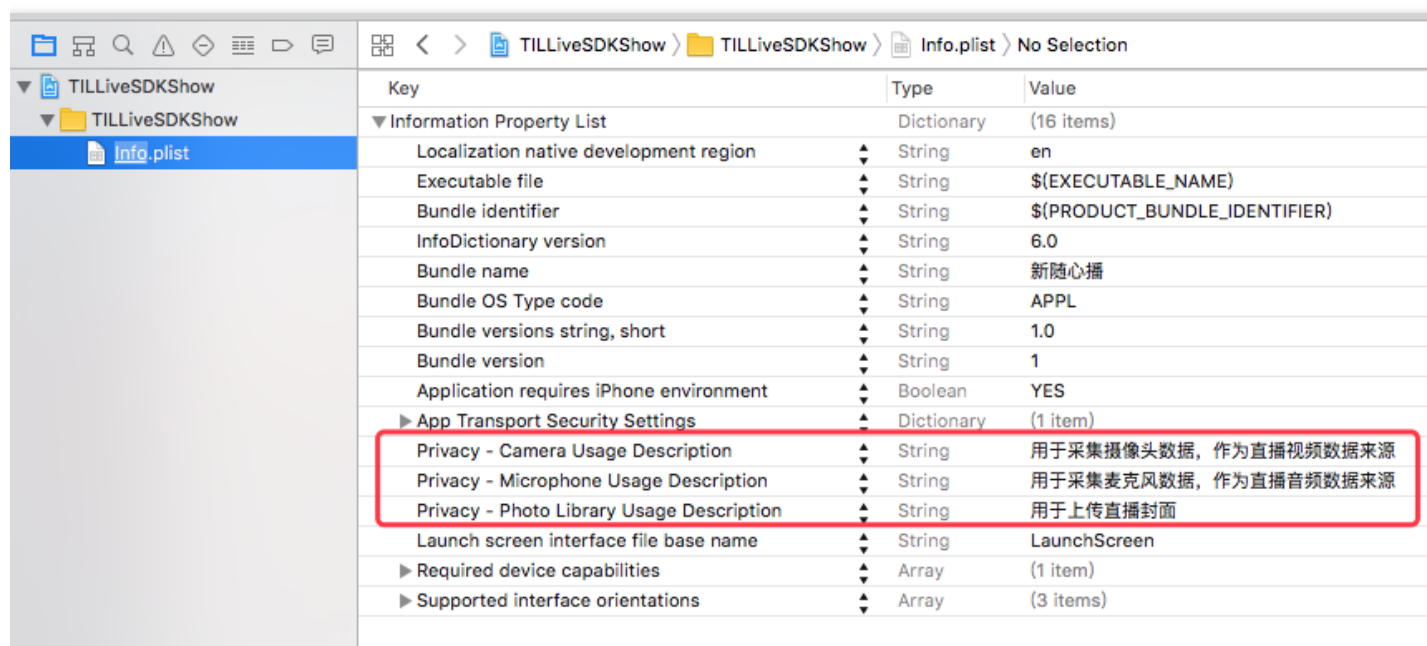
Setting

TILiveSDKShow

Enable Bitcode

No ^

iOS10 及以上系统，需在 Info.plist 中增加设备访问权限配置。



| Key                                                  | Type    | Value                         |
|------------------------------------------------------|---------|-------------------------------|
| Information Property List (Dictionary, 16 items)     |         |                               |
| Localization native development region               | String  | en                            |
| Executable file                                      | String  | \$(EXECUTABLE_NAME)           |
| Bundle identifier                                    | String  | \$(PRODUCT_BUNDLE_IDENTIFIER) |
| InfoDictionary version                               | String  | 6.0                           |
| Bundle name                                          | String  | 新随心播                          |
| Bundle OS Type code                                  | String  | APPL                          |
| Bundle versions string, short                        | String  | 1.0                           |
| Bundle version                                       | String  | 1                             |
| Application requires iPhone environment              | Boolean | YES                           |
| App Transport Security Settings (Dictionary, 1 item) |         |                               |
| Privacy - Camera Usage Description                   | String  | 用于采集摄像头数据，作为直播视频数据来源          |
| Privacy - Microphone Usage Description               | String  | 用于采集麦克风数据，作为直播音频数据来源          |
| Privacy - Photo Library Usage Description            | String  | 用于上传直播封面                      |
| Launch screen interface file base name               | String  | LaunchScreen                  |
| Required device capabilities (Array, 1 item)         |         |                               |
| Supported interface orientations (Array, 3 items)    |         |                               |

若上述步骤均无误，则工程编译可以通过了。

## 修改后台地址

目前随心播后台主要用来维护直播房间列表。如果复用随心播客户端代码，需要修改随心播后台地址为业务方自己部署的服务器地址。

| 请求类                      | 说明         | 修改文件                       | 修改方法              |
|--------------------------|------------|----------------------------|-------------------|
| LiveAVRoomIDRequest      | 获取自己分配的房间号 | LiveAVRoomIDRequest.m      | - (NSString *)url |
| LiveStartRequest         | 创建新房间      | LiveStartRequest.m         | - (NSString *)url |
| LiveEndRequest           | 退出房间       | LiveEndRequest.m           | - (NSString *)url |
| LiveListRequest          | 获取房间列表     | LiveListRequest.m          | - (NSString *)url |
| LiveHostHeartBeatRequest | 房间心跳       | LiveHostHeartBeatRequest.m | - (NSString *)url |
| LiveImageSignRequest     | 图片上传相关     | LiveImageSignRequest.m     | - (NSString *)url |

## 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                                                             |
|----------------------------------------------------------------------|
| libc++.tbd                                                           |
| libstdc++.tbd                                                        |
| libstdc++.6.tbd                                                      |
| libz.tbd                                                             |
| libbz2.tbd                                                           |
| libiconv.tbd                                                         |
| libresolv.tbd                                                        |
| libsqlite3.tbd                                                       |
| UIKit.framework                                                      |
| CoreVideo.framework                                                  |
| CoreMedia.framework                                                  |
| Accelerate.framework                                                 |
| Foundation.framework                                                 |
| AVFoundation.framework                                               |
| VideoToolbox.framework                                               |
| CoreGraphics.framework                                               |
| CoreTelephony.framework                                              |
| SystemConfiguration.framework                                        |
| AssetsLibrary.framework ( 如果需要使用 QVEffect ( AVSDK 内置美颜 ) , 则需要添加此库 ) |
| OpenAL.framework ( AVSDK 1.9.5 版本新增的依赖库 , 添加时注意确认 )                  |

## 库类介绍

| Frameworks 文件夹 | 说明                      |
|----------------|-------------------------|
| AVSDK          | 包括音视频相关的所有 SDK          |
| IMSDK          | 包括即时通讯相关的所有 SDK         |
| ILiveSDK       | 包括 ILiveSDK 和 TILiveSDK |

## 库类清单

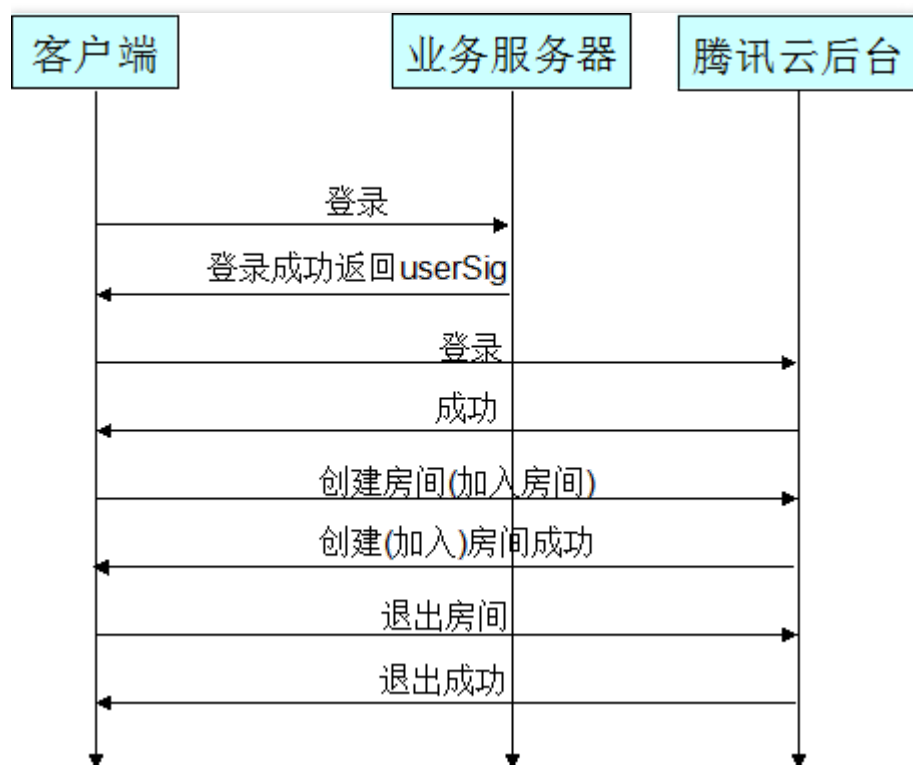
| 名称                   | 所在文件夹                | 说明                              |
|----------------------|----------------------|---------------------------------|
| QAVSDK.framework     | Frameworks/AVSDK/    | 音视频 SDK                         |
| IMCore.framework     | Frameworks/IMSDK/    | 即时通讯核心库                         |
| ImSDK.framework      | Frameworks/IMSDK/    | 即时通讯 SDK                        |
| IMSDKBugly.framework | Frameworks/IMSDK/    | 上报 SDK                          |
| QALSDK.framework     | Frameworks/IMSDK/    | 即时通讯网络模块 SDK                    |
| TLSSDK.framework     | Frameworks/IMSDK/    | 登录服务 SDK                        |
| ILiveSDK.framework   | Frameworks/ILiveSDK/ | 互动直播基础功能 SDK                    |
| TILiveSDK.framework  | Frameworks/ILiveSDK/ | 直播 SDK(针对直播场景封装的 SDK，包括互动连麦等功能) |

**AVSDK 1.9.5 版本 xplatform.framework 已私有化打包到 AVSDK.framework，升级新版本的用户注意去掉工程中的 xplatform.framework。**

# 直播接口

最近更新时间：2019-05-16 10:55:00

## 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 sdkAppId。内部包含了 IMSDK 的初始化 |

| 参数类型 | 参数名         | 说明                |
|------|-------------|-------------------|
| int  | sdkAppId    | 传入业务方 sdkAppId    |
| int  | accountType | 传入业务方 accountType |

示例：

```
[[ILiveSDK getInstance] initWithSdkId:1400027849 accountId:11656];//需替换为用户自己的sdkappid
```

## 帐号登录

| 接口名                            | 接口描述     |
|--------------------------------|----------|
| iLiveLogin: sig: succ: failed: | 登录到腾讯云后台 |

| 参数类型          | 参数名    | 说明                           |
|---------------|--------|------------------------------|
| NSString      | uid    | 登录帐号                         |
| NSString      | sig    | 登录鉴权签名, <a href="#">生成规则</a> |
| 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 |
|------|-----|----|
|------|-----|----|

| 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：

```
ILiveRoomOption *option = [ILiveRoomOption defaultHostLiveOption]; //默认主播配置
TILLiveManager *manager = [TILLiveManager getInstance];
[manager setAVRootView:self.view]; //设置渲染承载的视图
//添加渲染视图，userid：画面所属者id type:相机/屏幕共享
[manager addAVRenderView:[UIScreen mainScreen].bounds forIdentifier:userId srcType:type];
[manager createRoom:self.roomId option:option succ:^(
    NSLog(@"创建房间成功");
) failed:^(NSString *moudle, int errId, NSString *errMsg) {
    NSLog(@"创建房间失败");
}];
```

#### 观众进入房间

| 接口名                             | 接口描述                       |
|---------------------------------|----------------------------|
| joinRoom: option: succ: failed: | 观众进入直播间，自动拉取远程画面并渲染，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：**

```
ILiveRoomOption *option = [ILiveRoomOption defaultGuestLiveOption]; //默认观众配置
TILLiveManager *manager = [TILLiveManager getInstance];
[manager setAVRootView:self.view]; //设置渲染承载的视图
//添加渲染视图，userid：画面所属者id type:相机/屏幕共享
[manager addAVRenderView:[UIScreen mainScreen].bounds forIdentifier:userId srcType:type];
[manager joinRoom:self.roomId option:option succ:^(
    NSLog(@"进入房间成功");
} failed:^(NSString *moudle, int errId, NSString *errMsg) {
    NSLog(@"进入房间失败");
}];
```

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 18:33:10

## 准备

无论是发送消息还是上麦，都需要在创建或进入房间前设置事件和消息监听。**示例：**

```
TILLiveManager *manager = [TILLiveManager getInstance];
[manager setAVListener:self]; //设置音视频事件监听
[manager setIMListener:self]; //设置消息监听
```

## 发送文本消息

| 接口名                        | 接口描述   |
|----------------------------|--------|
| sendMessage: succ: failed: | 发送文本消息 |

| 参数类型               | 参数名    | 说明       |
|--------------------|--------|----------|
| ILVLiveTextMessage | msg    | 文本消息类型   |
| TCIVoidBlock       | succ   | 发送消息成功回调 |
| TCIErrorBlock      | failed | 发送消息失败回调 |

**示例：**

```
// 1. 发送文本消息
TILLiveManager *manager = [TILLiveManager getInstance];
ILVLiveTextMessage *msg = [[ILVLiveTextMessage alloc] init];

msg.type = ILVLIVE_IMTYPE_GROUP; //群消息 (也可发 C2C 消息)
msg.text = @"这里是消息的内容"; //消息内容
msg.sendId = @"这里是消息的发送方 id";
msg.recvId = @"这里是消息的接收方 id";

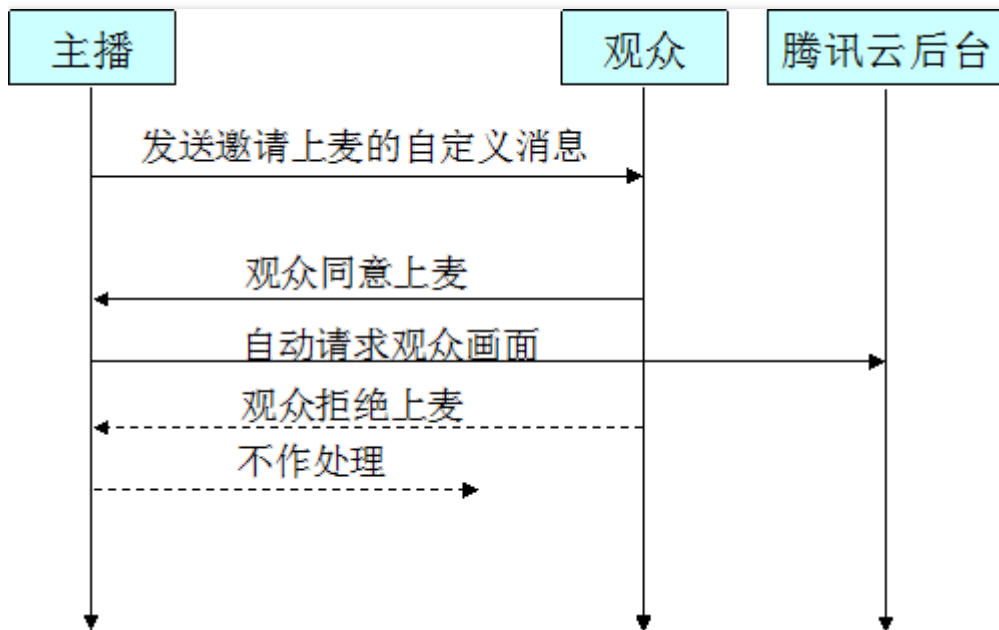
[manager sendMessage:msg succ:^(
    NSLog(@"发送成功");
    )];
```

```
} failed:^(NSString *moudle, int errId, NSString *errMsg) {  
    NSLog(@"发送失败");  
};
```

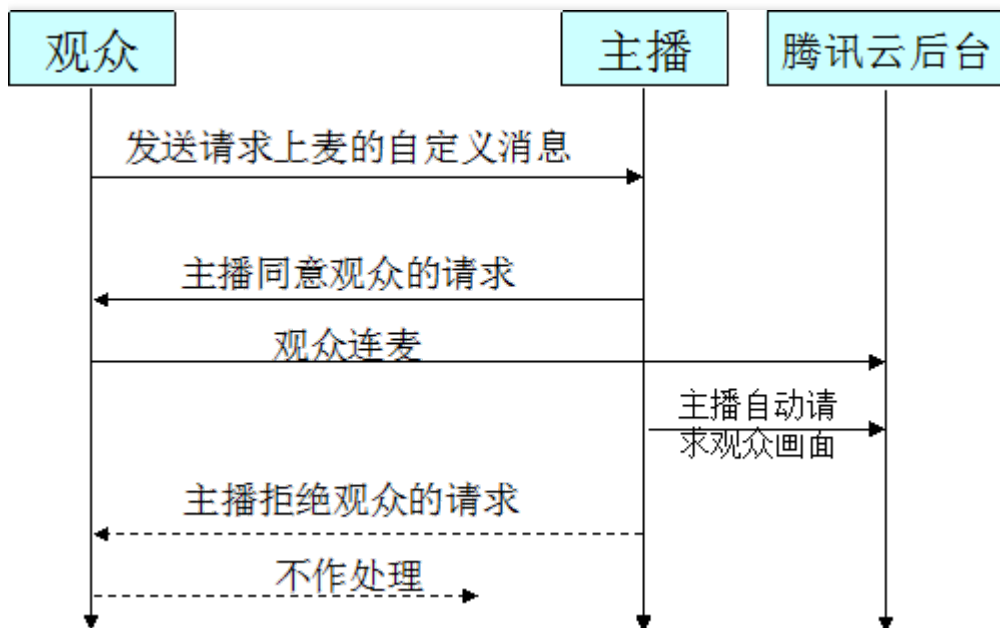
```
// 2. 文本消息接收 ( 在文本消息回调中接受文本消息 )  
- (void)onTextMessage:(ILVLiveTextMessage *)msg  
{  
    NSLog(@"收到消息 : %@", msg.text);  
}
```

## 上麦

### 主播邀请上麦流程图



### 观众请求上麦流程图



## 接口

### 发送自定义消息接口

| 接口名                              | 接口描述    |
|----------------------------------|---------|
| sendCustomMessage: succ: failed: | 发送自定义消息 |

| 参数类型                 | 参数名    | 说明          |
|----------------------|--------|-------------|
| ILVLiveCustomMessage | msg    | 自定义消息体      |
| TCIVoidBlock         | succ   | 发送自定义消息成功回调 |
| TCIErrorBlock        | failed | 发送自定义消息失败回调 |

### 示例：

```

TILLiveManager *manager = [TILLiveManager getInstance];
ILVLiveCustomMessage *msg = [[ILVLiveCustomMessage alloc] init];
msg.cmd = ILVLIVE_IMCMD_INVITE; //邀请信令
msg.type = ILVLIVE_IMTYPE_C2C; //C2C 消息类型
msg.recvid = @"这里是消息的接收方 id";

[manager sendCustomMessage:msg succ:^(

```

```
NSLog(@"邀请成功");
} failed:^(NSString *moudle, int errId, NSString *errMsg) {
NSLog(@"邀请失败");
};
```

## 上麦接口

| 接口名                                  | 接口描述                          |
|--------------------------------------|-------------------------------|
| upToVideoMember: role: succ: failed: | 观众上麦，包括打开摄像头、麦克风，切换角色，切换权限等操作 |

| 参数类型          | 参数名    | 说明                                   |
|---------------|--------|--------------------------------------|
| uint64_t      | auth   | 通话能力权限位,在 QAVCommon.h 文件中可以查看所有权限位定义 |
| NSString      | role   | 角色字符串（由业务方 App 的控制台生成）               |
| TCIVoidBlock  | succ   | 上麦成功回调                               |
| TCIErrorBlock | failed | 上麦失败回调                               |

## 示例：

```
//观众在消息回调中可以收到主播发送自定义消息
- (void)onCustomMessage:(ILVLiveCustomMessage *)msg
{
    TILLiveManager *manager = [TILLiveManager getInstance];
    switch (msg.cmd)
    {
        case ILVLIVE_IMCMD_INVITE:
        {
            //收到邀请调用上麦接口
            [manager upToVideoMember:ILVLIVEAUTH_INTERACT role:@"腾讯云后台配置的角色" succ:^(
                NSLog(@"上麦成功");
            ) failed:^(NSString *moudle, int errId, NSString *errMsg) {
                NSLog(@"上麦失败");
            }];
        }
        break;
    }
}
```

## 设置上麦着渲染画面的区域接口

| 接口名                         | 接口描述                                        |
|-----------------------------|---------------------------------------------|
| addAVRenderView:<br>forKey: | 添加渲染界面的区域，以及设置渲染视图对应的 key(一般使用渲染视图对应用户的 ID) |

| 参数类型     | 参数名   | 说明                              |
|----------|-------|---------------------------------|
| CGRect   | frame | 视图渲染区域                          |
| NSString | key   | 视图对应的 key，用作业务逻辑，一般使用画面对应的用户 ID |

示例：

```

- (void)onUserUpdateInfo:(ILVLiveAVEvent)event users:(NSArray *)users
{
    TILLiveManager *manager = [TILLiveManager getInstance];
    switch (event)
    {
        case ILVLIVE_AVEVENT_CAMERA_ON:
        {
            for (NSString *user in users)
            {
                //因为主播的渲染位置创建或进入房间的时候已经指定，这里不需要再指定。
                //当然也可根据自己的逻辑再此处指定主播的渲染位置。
                if (![user isEqualToString:self.host])
                {
                    [manager addAVRenderView:CGRectMakeMake(20, 20, 120, 160) forKey:user];
                }
            }
            break;
        }
    }
}

```

若上述步骤均无误，则可以实现主播邀请观众连麦，观众成功上麦，直播间中出现多路画面(主播和互动观众的)。