三体云-连麦直播API

API功能

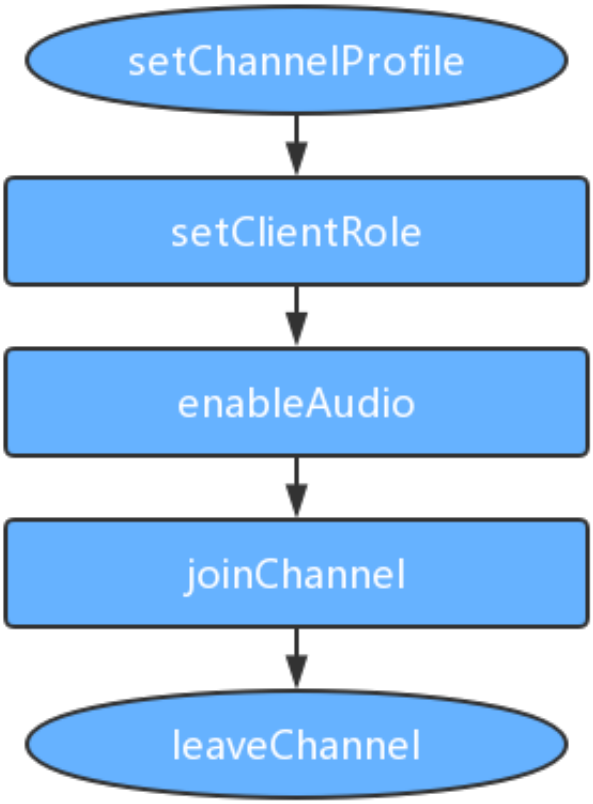
1. 接口功能

1. 通用功能
2. 房间操作
3. 语音设置
4. 伴奏设置
5. 视频设置
6. 聊天信息
7. 跨房间连麦
8. cdn推流设置

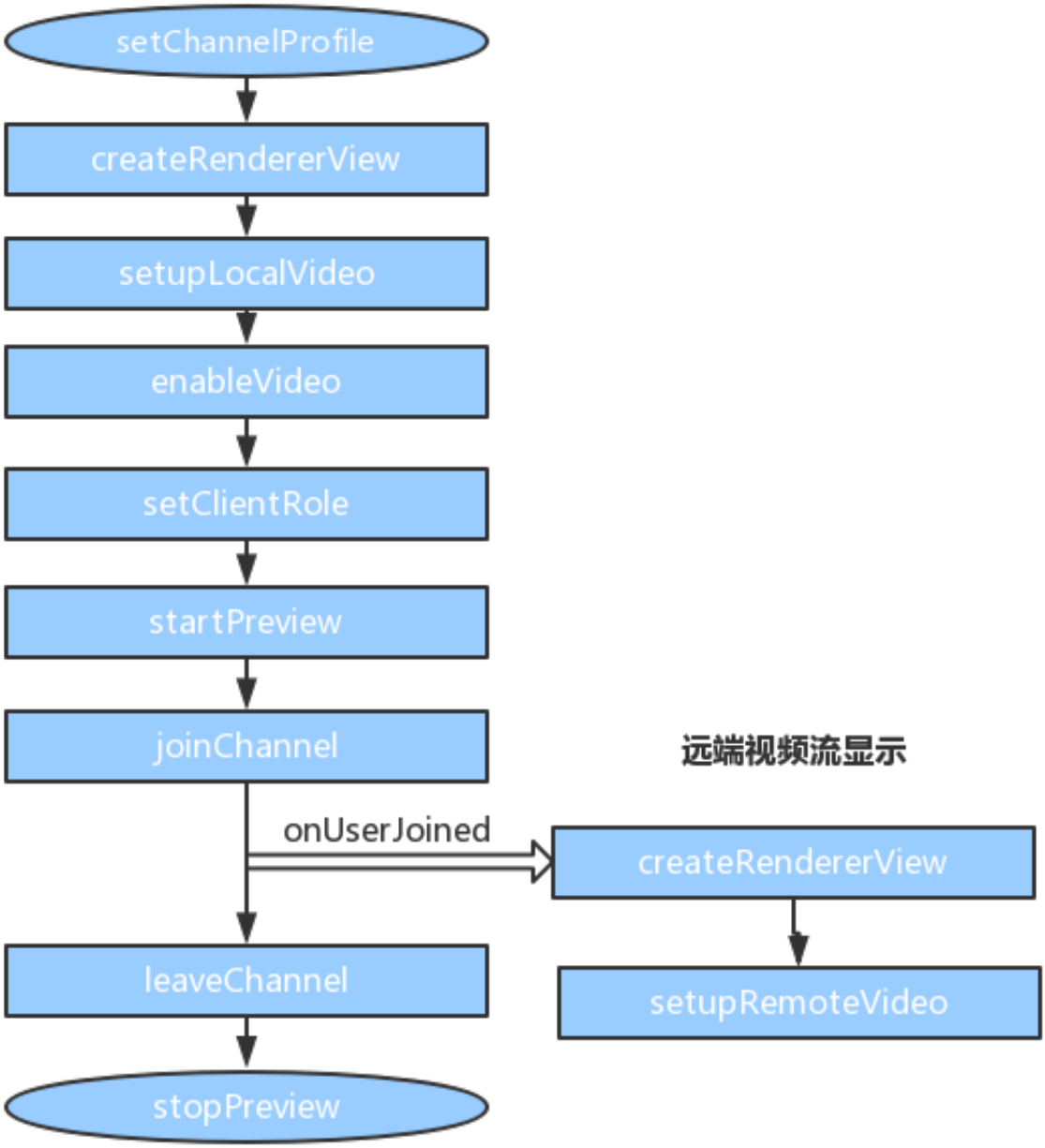
2. 回调功能

1. 基本回调
2. 音频回调
3. 视频回调
4. 聊天回调
5. 跨房间连麦回调
6. 其他回调

3. 语音直播流程


setChannelProfile
setClientRole
enableAudio
joinChannel
leaveChannel

4. 视频直播流程



5. SDK接入工程

SDK包含**TTTRtcEngineKit.framework**和**TTTPlayerKit.framework**，两个framework只支持真机，不支持模拟器

1. 把SDK导入工程
2. 设置Bitcode为NO
3. 设置后台音频模式
4. 导入系统库
    - libxml2.tbd
    - libstdc++.tbd

- libz.tbd
- ReplayKit.framework
- CoreTelephony.framework
- SystemConfiguration.framework

一. 接口功能

1. 通用功能

初始化SDK

```
1. + (instancetype)sharedEngineWithAppId:(NSString *)appId delegate:(id<TTTRtcEngi
neDelegate>)delegate

2. + (instancetype)sharedEngineWithAppId:(NSString *)appId
                                delegate:(id<TTTRtcEngineDelegate>)delegate
                              enableChat:(BOOL)enableChat
                            enableSignal:(BOOL)enableSignal
```

初始化 TTTRtcEngineKit 为一个单例。使用 TTTRtcEngineKit，必须先调用该接口进行初始化。 通过指定的 delegate 通知应用程序引擎运行时的事件。delegate 中定义的所有方法都是可选实现的。

参数	描述
appId	连麦平台分配，用于区分不同的客户和应用，在同一个连麦平台内保证唯一
TTTRtcEngineDelegate	SDK回调代理
enableChat	打开发送聊天功能
enableSignal	打开发送信令功能

销毁引擎实例

```
+ (void)destroy;
```

Note: 该版本，调用该方法并不会销毁TTTRtcEngineKit对象，会释放一些SDK内部的资源

获取SDK版本号

```
+ (NSString *)getSdkVersion;
```

设置服务器地址

```
- (void)setServerIp:(NSString*)ip port:(int)port;
```

参数	描述
ip	ip地址或域名
port	端口

Note:如不需要指定特定的服务器，请不要调用该接口

设置频道模式

```
- (int)setChannelProfile:(TTTRtcChannelProfile)profile;
```

profile当前有三种模式:。

模式	Enum参数	描述
通信	TTTRtc*ChannelProfile*Communication	用于一对一或群聊，用户可以自由发言，互看视频
直播	TTTRtc*ChannelProfile*LiveBroadcasting	用于直播
游戏	TTTRtc*ChannelProfile*Game_FreeMode	用于游戏中语音，视频

设置用户角色

```
- (int)setClientRole:(TTTRtcClientRole)role withKey:(NSString *)permissionKey;
```

role说明

角色	Enum参数	描述
主播	TTTRtc*ClientRole*Anchor	仅可用于直播模式下
副播	TTTRtc*ClientRole*Broadcaster	直播模式下，副播可以上传音视频，主播控制副播位置
观众	TTTRtc*ClientRole*Audience	直播模式下，观众只有观看音视频的权限，不能上传音视频

直播模式下，默认角色是观众。 **主播**角色拥有最高权限，可以对其他用户进行禁言，踢出等操作

参数	描述
role	角色
permissionKey	音视频通话，直接传nil

设置日志文件路径

```
- (int)setLogFile:(NSString*)filePath;
```

参数	描述
filePath	日志文件的完整路径。该日志文件为UTF-8编码。

设置日志文件过滤器

```
- (int)setLogFilter:(TTTRtcLogFilter)filter;
```

filter说明

Enum参数	描述
TTTRtc*LogFilter*Off	不打印日志
TTTRtc*LogFilter*Debug	打印Debug日志
TTTRtc*LogFilter*Info	打印Info日志
TTTRtc*LogFilter*Warning	打印Warning日志
TTTRtc*LogFilter*Error	打印Error日志
TTTRtc*LogFilter*Critical	打印Critical日志

追加自定义日志

```
- (int)appendLogContent:(NSString *)content;
```

参数	描述
content	自定义日志内容

Note:进入房间后才可使用

设置信令超时时间

```
- (int)setSignalTimeout:(NSUInteger)seconds;
```

参数	描述
seconds	超时时间

踢出房间

```
- (int)kickChannelUser:(int64_t)uid;
```

角色为**TTTRtc*ClientRole*Anchor**有权限做该操作，被踢着会收到**didKickedOutOfUid**回调

参数	描述
uid	被踢者userID

2. 房间操作

加入通话频道

```
- (int)joinChannelByKey:(NSString *)channelKey
            channelName:(NSString *)channelName
                    uid:(int64_t)uid
            joinSuccess:(void(^)(NSString *channel, int64_t uid, NSInteger elapsed
))joinSuccessBlock;
```

该方法让用户加入通话频道，在同一个频道内的用户可以互相通话，多个用户加入同一个频道，可以群聊。使用不同 App ID 的应用程序是不能互通的。如果已在通话中，用户必须调用 leaveChannel 退出当前通话，才能进入下一个频道。 SDK 在音频通话中使用 iOS 系统的 AVAudioSession 共享对象进行录音和播放，用户对该对象的操作可能会影响 SDK 的音频相关功能。

参数	描述
channelKey	此为程序生成的Channel Key,直接传nil
channelName	标识通话的频道名称，长度在64字节以内的字符串
uid	用户ID
joinSuccessBlock	用户加入成功回调

Note:同一个频道里不能出现两个相同的 uid,如果使用 joinSuccessBlock，就不会响应加入房间的代理回调

离开频道

```
- (int)leaveChannel:(void(^)(TTTRtcStats *stat))leaveChannelBlock;
```

当调用 joinChannelByKey API 方法后，必须调用 leaveChannel 结束通话，否则无法开始下一次通话。 调用 leaveChannel，没有副作用。该方法会把会话相关的所有资源释放掉。该方法是异步操作，调用返回时并没有真正退出频道。在真正退出频道后，SDK 会触发 didLeaveChannelWithStats 回调。

参数	描述
leaveChannelBlock	成功离开频道的回调

Note: 使用leaveChannelBlock，就不会响应退出房间的代理回调

3. 语音设置

静音/取消静音

```
- (int)muteLocalAudioStream:(BOOL)mute;
```

参数	描述
mute	YES: 麦克风静音, NO: 取消静音

静音所有远端音频/对所有远端音频取消静音

```
- (int)muteAllRemoteAudioStreams:(BOOL)mute;
```

该方法用于允许/禁止播放远端用户的音频流，即对所有远端用户进行静音与否

参数	描述
mute	YES: 停止播放所接收的音频流, NO: 恢复播放所接收的音频流

静音指定远端用户/对指定远端用户取消静音

```
- (int)muteRemoteAudioStream:(int64_t)uid mute:(BOOL)mute;
```

参数	描述
uid	远端用户的ID
mute	YES: 停止播放指定用户的音频流, NO: 恢复播放指定用户的音频流

禁言指定远端用户/对指定远端用户取消禁言

```
- (int)muteRemoteSpeaking:(int64_t)uid mute:(BOOL)mute;
```

直播模式下，主播可以禁言用户，被禁言的用户不会上传音频流，其他端会收到该用户被禁言的回调

参数	描述
uid	远端用户的ID
mute	YES: 禁止发言, NO: 允许发言

切换音频输出方式：扬声器或听筒

```
- (int)setEnableSpeakerphone:(BOOL)enableSpeaker;
```

参数	描述
enableSpeaker	YES: 音频输出至扬声器，NO: 语音会根据默认路由出声

Note:在插入耳机的状态下，应禁止调用该方法，拔下耳机的时候声音默认从扬声器出来

是否是扬声器状态

```
- (BOOL)isSpeakerphoneEnabled;
```

设置默认的语音路由

```
- (int)setDefaultAudioRouteToSpeakerphone:(BOOL)defaultToSpeaker;
```

参数	描述
defaultToSpeaker	YES: 从扬声器出声，NO: 语音聊天：从听筒出声；视频聊天：从扬声器出声

启用说话者音量提示

```
- (int)enableAudioVolumeIndication:(NSInteger)interval smooth:(NSInteger)smooth;
```

参数	描述
interval	指定音量提示的时间间隔（<=0: 禁用音量提示功能；>0: 提示间隔，单位为毫秒。建议设置到大于200毫秒。）
smooth	Y平滑系数。默认可以设置为3

设置音频高音质选项

```
- (int)setHighQualityAudioParametersWithFullband:(BOOL)fullband stereo:(BOOL)stere
o fullBitrate:(BOOL)fullBitrate;
```

参数	描述
fullband	全频带编解码器（48kHz采样率）
stereo	立体声编解码器
fullBitrate	高码率模式，建议仅在纯音频模式下使用

Note:请在加入频道前开启高音质，需要全部的参数设置为YES。高音质会占用较大的带宽

打开/关闭耳返功能

```
- (int)enableAudioEarBack:(BOOL)enable;
```

参数	描述
enable	YES: 打开耳返功能，NO: 关闭耳返功能

Note:该方法只有在插入耳机的状态下有效果，如果中间拔掉耳机，耳返会自动停止

4. 伴奏设置

开始客户端本地混音

```
- (int)startAudioMixing:(NSString *)filePath loopback:(BOOL)loopback replace:(BOOL
)replace cycle:(NSInteger)cycle;
```

参数	描述
filePath	指定需要混音的本地音频文件名和文件路径
loopback	True: 只有本地可以听到混音或替换后的音频流，False: 本地和对方都可以听到混音或替换后的音频流
replace	True: 音频文件内容将会替换本地录音的音频流，False: 音频文件内容将会和麦克风采集的音频流进行混音
cycle	指定音频文件循环播放的次数

停止客户端本地混音

```
- (int)stopAudioMixing;
```

暂停播放伴奏

```
- (int)pauseAudioMixing;
```

恢复播放伴奏

```
- (int)resumeAudioMixing;
```

调节伴奏音量

```
- (int)adjustAudioMixingVolume:(NSInteger)volume;
```

参数	描述
volume	伴奏音量范围为0~100。默认100为原始文件音量

获取伴奏时长

```
- (int)getAudioMixingDuration;
```

获取伴奏播放进度

```
- (int)getAudioMixingCurrentPosition;
```

拖动语音进度条

```
- (int)setAudioMixingPosition:(NSInteger)pos;
```

参数	描述
pos	进度条位置，单位为毫秒

5. 视频设置

开启视频模式

```
- (int)enableVideo;
```

Note:可以在加入频道前或者通话中调用，在加入频道前调用，则自动开启视频模式，在通话中调用则由音频模式切换为视频模式。

关闭视频，开启纯音频模式

```
- (int)disableVideo;
```

Note:可以在加入频道前或者通话中调用，在加入频道前调用，则自动开启纯音频模式，在通话中调用则由视频模式切换为纯音频频模式

设置视频编码属性

```
- (int)setVideoProfile:(TTTRtcVideoProfile)profile swapWidthAndHeight:(BOOL)swapWi
dthAndHeight;
```

profile说明

Enum参数	分辨率	帧率	带宽kbps
TTTRtc*VideoProfile*120P	160x120	15	65
TTTRtc*VideoProfile*180P	320x180	15	140
TTTRtc*VideoProfile*240P	320x240	15	200
TTTRtc*VideoProfile*360P	640x360	15	400
TTTRtc*VideoProfile*480P	640x480	15	500
TTTRtc*VideoProfile*720P	1280x720	15	1130
TTTRtc*VideoProfile*1080P	1920x1080	15	2080
TTTRtc*VideoProfile*Default	640x360	15	400

该方法设置视频编码属性（Profile）。每个属性对应一套视频参数，如分辨率、帧率、码率等。 当设备的摄像头不支持指定的分辨率时，SDK 会自动选择一个合适的摄像头分辨率，但是编码分辨率仍然用 setVideoProfile() 指定的。 该方法仅设置编码器编出的码流属性，可能跟最终显示的属性不一致，例如编码码流分辨率为 640x480，码流的旋转属性为 90 度，则显示出来的分辨率为竖屏模式。

参数	描述
profile	Profile对应一套视频参数，如分辨率、帧率、码率
swapWidthAndHeight	是否交换宽和高，在竖屏采集时需交换宽高

自定义视频编码属性

```
- (int)setVideoProfile:(CGSize)videoSize frameRate:(NSUInteger)frameRate bitRate:(
NSUInteger)bitRate;
```

我们支持用户自定义视频的宽高，帧率和码率

参数	描述
videoSize	视频分辨率
frameRate	视频帧率
bitRate	视频码率kbps

禁用/启用本地视频功能

```
- (int)enableLocalVideo:(BOOL)enabled;
```

禁用本地视频视频流不会上传服务器，其它端不能观看本端视频

参数	描述
enabled	YES: 启用本地视频（默认），NO: 禁用本地视频

启动本地视频预览

```
- (int)startPreview;
```

Note:需要上传本地视频建议开启本地预览。该方法可重复调用，需要对应调用stopPreview，否则无法关闭预览

停止本地视频预览

```
- (int)stopPreview;
```

设置本地视频显示属性

```
- (int)setupLocalVideo:(TTTRtcVideoCanvas*)local;
```

该方法设置本地视频显示信息。应用程序通过调用此接口绑定本地视频流的显示视窗(view)，并设置视频显示模式。 在应用程序开发中，通常在初始化后调用该方法进行本地视频设置，然后再加入频道。如果需要解除绑定，把参数设置为nil

local说明：

```
@interface TTTRtcVideoCanvas : NSObject
// 视频显示窗口。SDK不维护view的生命周期，应用程序应保证view在通话中是有效的。
@property (strong, nonatomic) TImageView *view;
@property (assign, nonatomic) TTTRtcRenderMode renderMode; // 视频显示模式
@property (assign, nonatomic) int64_t uid;                 // 用户ID

@end
```

设置远端视频显示属性

```
- (int)setupRemoteVideo:(TTTRtcVideoCanvas*)remote;
```

切换摄像头

```
- (int)switchCamera;
```

Note:第一次默认开启前摄像头，中间切换摄像头操作之后，SDK内部会保留当前设置的值，不会在下次加入房间重置为前摄像头

暂停所有远端视频流

```
- (int)muteAllRemoteVideoStreams:(BOOL)mute;
```

参数	描述
mute	YES: 停止播放接收到的所有视频流，NO: 允许播放接收到的所有视频流

允许/禁止播放指定的远端视频流

```
- (int)muteRemoteVideoStream:(int64_t)uid mute:(BOOL)mute;
```

参数	描述
uid	用户ID
mute	YES: 停止播放接收到的视频流，NO: 允许播放接收到的视频流

设置视频优化选项

```
- (int)setVideoQualityParameters:(BOOL)preferFrameRateOverImageQuality;
```

参数	描述
preferFrameRateOverImageQuality	True: 画质和流畅度里，优先保证流畅度，False: 画质和流畅度里，优先保证画质(默认)。

配置外部视频源

```
- (int)setExternalVideoSource:(BOOL)enable useTexture:(BOOL)useTexture;
```

参数	描述
enable	是否使用外部视频源
useTexture	是否使用 Texture 作为输入

Note:如果使用外部视频源，在“enableVideo/startPreview”之前调用本API。

推送外部视频帧

```
- (BOOL)pushExternalVideoFrame:(TTTRtcVideoFrame *)videoFrame;
```

** videoFrame**说明：

```
@interface TTTRtcVideoFrame : NSObject

@property (assign, nonatomic) TTTRtcVideoFrameFormat format;  // 视频帧的格式
@property (assign, nonatomic) CMTime time;                    // 视频帧的时间戳，以毫秒为单位。不正确的时间戳会导致丢帧或者音视频不同步
@property (assign, nonatomic) CVPixelBufferRef textureBuffer;
@property (strong, nonatomic) NSData *dataBuffer;             // raw data buffer. in case of ios texture, it is not used
@property (assign, nonatomic) int strideInPixels;             // 视频帧的行间距，单位为像素而不是字节。
@property (assign, nonatomic) int height;                     // how many rows of pixels, in case of ios texture, it is not used
@property (assign, nonatomic) int cropLeft;                   // how many pixels to crop on the left boundary
@property (assign, nonatomic) int cropTop;                    // how many pixels to crop on the top boundary
@property (assign, nonatomic) int cropRight;                  // how many pixels to crop on the right boundary
@property (assign, nonatomic) int cropBottom;                 // how many pixels to crop on the bottom boundary
@property (assign, nonatomic) int rotation;                   // 0, 90, 180, 270.

@end
```

设置本地视频帧采集格式

```
- (int)setLocalVideoFrameCaptureFormat:(TTTRtcVideoFrameFormat)format;
```

如果不设置，回调rtcEngine:localVideoFrameCaptured:videoFrame:默认格式为“BGRA”

format说明：

```
typedef NS_ENUM(NSUInteger, TTTRtcVideoFrameFormat) {
    TTTRtc_VideoFrameFormat_Texture = 0,
    TTTRtc_VideoFrameFormat_I420    = 1,
    TTTRtc_VideoFrameFormat_NV12    = 2,
    TTTRtc_VideoFrameFormat_NV21    = 3,
    TTTRtc_VideoFrameFormat_RGBA    = 4,
    TTTRtc_VideoFrameFormat_BGRA    = 5,
    TTTRtc_VideoFrameFormat_ARGB    = 6,
};
```

设置远端视频帧输出格式

```
- (int)setRemoteVideoFrameOutputFormat:(TTTRtcVideoFrameFormat)format;
```

设置是否启用视频双流模式

```
- (int)enableDualStreamMode:(BOOL)enabled;
```

参数	描述
enable	是否启用视频双流模式

设置视频大小流

```
- (int)setRemoteVideoStream:(int64_t)uid type:(TTTRtcVideoStreamType)streamType;
```

streamType说明：

```
typedef NS_ENUM(NSUInteger, TTTRtcVideoStreamType) {
    TTTRtc_VideoStream_High = 0, // 视频大流
    TTTRtc_VideoStream_Low  = 1, // 视频小流
};
```

如果远端用户选择发送双流(视频大流和小流), 该方法指定接收远端用户的视频流大小。

参数	描述
uid	用户ID
streamType	视频流类型

设是否正在屏幕录制

```
- (BOOL)isScreenRecording;
```

开始录制屏幕用作视频源

```
- (int)startRecordScreen;
```

Note:调用该方法，录屏的视频流取代摄像头采集视频流作为本端的输出，仅支持ios11.0之后的系统

开始录制屏幕并保存

```
- (int)startRecordScreenAndSave:(NSString *)path width:(NSInteger)width height:(NS
Integer)height;
```

参数	描述
path	保存路径，如果值为nil，保存到系统相册
width	视频宽度
height	视频高度

停止录制屏幕

```
- (int)stopRecordScreen;
```

6. 聊天信息

发送聊天消息

```
- (int)sendChatMessageWithUserID:(int64_t)userID chatType:(TTTRtcChatType)chatType
 seqID:(NSString *)seqID data:(NSString *)data;
```

chatType说明

Enum参数	描述
TTTRtc*ChatType*Text	文字消息
TTTRtc*ChatType*Picture	图片
TTTRtc*ChatType*Audio	短语音
TTTRtc*ChatType*Custom	自定义消息

参数	描述
userID	用户ID，0会发送给所有用户
chatType	聊天消息类型
seqID	唯一标识
data	消息内容

发送信令

```
- (int)sendSignalWithUserID:(int64_t)userID seqID:(NSString *)seqID data:(NSString
 *)data;
```

参数	描述
userID	用户ID，0会发送给所有用户
seqID	唯一标识
data	消息内容

开始采集语音消息

```
- (int)startRecordChatAudio;
```

停止采集并开始发送消息

```
- (int)stopRecordAndSendChatAudioWithUserID:(int64_t)userID seqID:(NSString *)seqI
D;
```

参数	描述
userID	用户ID，0会发送给所有用户
seqID	唯一标识

Note:调用方法之后，SDK内部会上传短语音消息到服务器，上传成功会收到发送消息成功的回调

取消语音消息录制

```
- (int)cancelRecordChatAudio;
```

Note:取消录制，并删除录制的内容

开始播放语音消息

```
- (int)startPlayChatAudioFileName:(NSString *)fileName;
```

参数	描述
fileName	具有完整路径的段语音(xxx/.../xxx.wav)

Note:如果当前正在播放短语音，必须停止播放，才能播放下一条语音

停止播放语音消息

```
- (int)stopPlayChatAudio;
```

是否正在播放语音消息

```
- (BOOL)isChatAudioPlaying;
```

7. 跨房间连麦

跨房间连麦

```
- (int)linkOtherAnchor:(int64_t)sessionID userID:(int64_t)userID;
```

参数	描述
sessionID	对方主播的会话ID
userID	对方主播的用户ID

结束与其他主播连麦

```
- (int)unlinkOtherAnchor:(int64_t)sessionID userID:(int64_t)userID;
```

参数	描述
sessionID	对方主播的会话ID
userID	对方主播的用户ID

8. cdn推流设置

配置旁路直播推流

```
- (int)configPublisher:(TTTPublisherConfiguration *)config;
```

该方法用于在加入频道前为引擎创建一份推流设置。在你调用本 API 之前: 请确保已联系 3TLive 开通旁路直播推流功能。

请确保用户已经调用 setClientRole() 且已将用户角色设为主播，主播必须在加入频道前调用本 API

config说明 ```objc @interface TTTPublisherConfiguration : NSObject

@property (assign, nonatomic) NSInteger bitrate; // 旁路直播输出码流的码率。默认设置为 500 Kbps @property (copy, nonatomic) NSString *publishUrl; // 合图推流地址 @property (assign, nonatomic) BOOL isPureAudio; // 推送纯音频流

@end ```

设置画中画布局(cdn布局)

```
- (int)setVideoCompositingLayout:(TTTRtcVideoCompositingLayout*)layout;
```

1. 定义一个画布(canvas): 画布的宽和高(即视频的分辨率), 背景颜色，和您想在屏幕上显示的视频总数。
2. 在画布上定义每个视频的位置和尺寸(无论画布定义的宽和高有多大，每个视频用 0 到 1 的相对位置和尺寸进行定义)，图片所在的图层，图片的透明度，视频是经过裁减的还是缩放到合适大小等等。
3. 服务端推流程序在生成 H.264 视频流并通过 RTMP 协议推送给 CDN 服务商时，会将自定义布局的详细信息格式化为 JSON 字符串，打包到在每个关键帧的 SEI 信息中，SEI 的类型为 100 。 **layout**说明：

```
@interface TTTRtcVideoCompositingLayout : NSObject

@property (assign, nonatomic) NSInteger canvasWidth;   // 整个屏幕(画布)的宽度
@property (assign, nonatomic) NSInteger canvasHeight;  // 整个屏幕(画布)的高度
@property (copy, nonatomic) NSString* backgroundColor; // 屏幕(画布)的背景颜色，可根据 RGB 填写所需颜色对应的6位符号。e.g. "#c0c0c0"
@property (strong, nonatomic) NSMutableArray *regions; // 视频合成区域列表
@property (copy, nonatomic) NSString* appData;         // 应用程序自定义的数据

@end
```

Note:请确保在频道内调用该方法。如果同一频道内有多人调用该方法，其他调用了该方法的用户需调用 clearVideoCompositingLayout 取消已设置的画中画布局，仅留一人保留布局设置

取消画中画布局设置

```
- (int)clearVideoCompositingLayout;
```

设置SEI

```
- (int)setSEI:(NSString *)SEI;
```

设置画中画布局即SEI内容

参数	描述
SEI	JSON格式的SEI

设置CDN推流视频参数，在加入房间之前调用

```
- (void)setVideoMixerParams:(CGSize)videoSize videoFrameRate:(NSUInteger)videoFram
eRate videoBitRate:(NSUInteger)videoBitRate;
```

参数	描述
videoSize	视频分辨率
videoFrameRate	视频帧率
videoBitRate	视频码率kbps

设置CDN推流音频参数

```
- (void)setAudioMixerParams:(NSUInteger)samplerate channels:(NSUInteger)channels;
```

参数	描述
samplerate	采样率
channels	1单声道，2双声道

设置混屏背景图片

```
- (void)setVideoMixerBackgroundImgUrl:(NSString*)url;
```

当前版本仅支持进房间前进行设置，退出房间清空，再次进房间需要重新设置

参数	描述
url	背景图片所在的地址

向h.264码流中插入sei内容

```
- (int)insertH264SeiContent:(NSString*)content;
```

参数	描述
content	字符串内容

开始RTMP推流

```
- (int)startRtmpPublish:(NSString *)rtmpUrl;
```

停止RTMP推流

```
- (int)stopRtmpPublish;
```

暂停RTMP推流

```
- (int)pauseRtmpPublish;
```

恢复RTMP推流

```
- (int)resumeRtmpPublish;
```

二. 回调功能

1. 基本回调

发生错误回调

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine didOccurError:(TTTRtcErrorCode)errorCo
de;
```

errorCode说明：

Enum参数	描述
TTTRtc*Error*NoError	没有错误
TTTRtc*Error*InvalidChannelName	无效的房间名称
TTTRtc*Error*Enter_TimeOut	超时,10秒未收到服务器返回结果
TTTRtc*Error*Enter_Failed	无法连接服务器
TTTRtc*Error*Enter_VerifyFailed	验证码错误
TTTRtc*Error*Enter_BadVersion	版本错误
TTTRtc*Error*Enter_Unknown	未知错误
TTTRtc*Error*Unknown	未知错误

该回调有joinChannel不成功触发

参数	描述
engine	TTTRtcEngineKit对象
errorCode	错误原因

加入频道成功回调

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine didJoinChannel:(NSString*)channel with Uid:(int64_t)uid elapsed:(NSInteger) elapsed;
```

该回调有joinChannel成功触发

参数	描述
engine	TTTRtcEngineKit对象
channel	频道名
uid	用户ID
elapsed	从joinChannel开始到该事件产生的延迟（毫秒）

网络连接丢失回调

```
- (void)rtcEngineConnectionDidLost:(TTTRtcEngineKit *)engine;
```

Note:在房间内发生断网，SDK会自动重连，如果重连不成功触发此回调

网络异常断开后，超时连接成功

```
- (void)rtcEngineReconnectServerTimeout:(TTTRtcEngineKit *)engine;
```

Note:当网络异常断开后，将尝试重连，若在服务器容忍的超时范围外才重连上服务器，服务器将会拒绝，其房间状态将不可用。此时触发该回调，上层应该在收到此回调后退出房间。

成功离开频道

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine didLeaveChannelWithStats:(TTTRtcStats*
)stats;
```

stats说明

```
@interface TTTRtcStats : NSObject
@property (assign, nonatomic) NSUInteger duration;        // 通话时长，累计值
@property (assign, nonatomic) NSUInteger txBytes;         // 发送字节数，累计值
@property (assign, nonatomic) NSUInteger rxBytes;         // 接收字节数，累计值
@property (assign, nonatomic) NSUInteger txAudioKBitrate;
@property (assign, nonatomic) NSUInteger rxAudioKBitrate;
@property (assign, nonatomic) NSUInteger users;
@end
```

用户加入回调

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine didJoinedOfUid:(int64_t)uid clientRole
:(TTTRtcClientRole)clientRole
    isVideoEnabled:(BOOL)isVideoEnabled elapsed:(NSInteger)elapsed;
```

参数	描述
engine	TTTRtcEngineKit对象
uid	用户ID
clientRole	用户角色
isVideoEnabled	用户是否启用本地视频
elapsed	加入频道开始到该回调触发的延迟（毫秒)

用户离线回调

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine didOfflineOfUid:(int64_t)uid reason:(TTTRtcUserOfflineReason)reason;
```

reason说明：

Enum参数	描述
TTTRtc*UserOffline*Quit	用户主动离开
TTTRtc*UserOffline*Dropped	因过长时间收不到对方数据包，超时掉线
TTTRtc*UserOffline*BecomeAudience	当用户身份从主播切换为观众时触发

用户被踢出回调

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine didKickedOutOfUid:(int64_t)uid reason:(TTTRtcKickedOutReason)reason;
```

参数	描述
engine	TTTRtcEngineKit对象
uid	用户ID
reason	用户被踢出的原因

2. 音频回调

用户音频静音

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine didAudioMuted:(BOOL)muted byUid:(int64_t)uid;
```

参数	描述
engine	TTTRtcEngineKit对象
muted	YES: 静音，NO: 取消静音
uid	用户ID

禁止/允许用户发言回调

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine didSpeakingMuted:(BOOL)muted ofUid:(int64_t)uid;
```

主播对用户做禁言，可以发言操作，房间内其他用户也会收到

参数	描述
engine	TTTRtcEngineKit对象
muted	YES: 禁止发言，NO: 允许发言
uid	用户ID

音频输出路由发生变化

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine didAudioRouteChanged:(TTTRtcAudioOutputRouting)routing;
```

routing说明：

Enum参数	描述
TTTRtc*AudioOutput*Headset	耳机或蓝牙
TTTRtc*AudioOutput*Speaker	扬声器
TTTRtc*AudioOutput*Headphone	手机听筒

本地音频统计

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine localAudioStats:(TTTRtcLocalAudioStats *)stats;
```

stats说明：

```
@interface TTTRtcLocalAudioStats : NSObject

@property (assign, nonatomic) NSUInteger encodedBitrate;  // 编码的码率(kbps)
@property (assign, nonatomic) NSUInteger sentBitrate;     // 发送的码率(kbps)
@property (assign, nonatomic) NSUInteger receivedBitrate; // 接收的码率(kbps)

@end
```

远端音频统计

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine remoteAudioStats:(TTTRtcRemoteAudioSta
ts*)stats;
```

stats说明：

```
@interface TTTRtcRemoteAudioStats : NSObject

@property (assign, nonatomic) int64_t uid;
@property (assign, nonatomic) NSUInteger receivedBitrate;

@end
```

远端用户音量

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine reportAudioLevel:(int64_t)userID
      audioLevel:(NSUInteger)audioLevel audioLevelFullRange:(NSUInteger)audioLeve
lFullRange;
```

提示谁在说话及其音量，默认禁用。可通过enableAudioVolumeIndication方法设置。

参数	描述
engine	TTTRtcEngineKit对象
userID	用户ID
audioLevel	非线性区间[0,9]
audioLevelFullRange	线性区间[0,32768]

远端首次音频数据接收

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine firstRemoteAudioDataIncommingOfUid:(in
t64_t)uid;
```

参数	描述
engine	TTTRtcEngineKit对象
uid	用户ID

3. 视频回调

启用/关闭视频

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine didVideoEnabled:(BOOL)enabled byUid:(int64_t)uid;
```

参数	描述
engine	TTTRtcEngineKit对象
enabled	YES: 该用户已启用了视频功能，NO: 该用户已关闭了视频功能
uid	用户ID

本地视频统计

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine localVideoStats:(TTTRtcLocalVideoStats *)stats;
```

stats说明：

```
@interface TTTRtcLocalVideoStats : NSObject

@property (assign, nonatomic) NSUInteger encodedBitrate;  // 编码的码率(kbps)
@property (assign, nonatomic) NSUInteger sentBitrate;     // 发送的码率(kbps)
@property (assign, nonatomic) NSUInteger sentFrameRate;   // 发送的帧率(fps)
@property (assign, nonatomic) NSUInteger receivedBitrate; // 接收的码率(kbps)
@property (assign, nonatomic) float sentLossRate;         // 发送的丢包率
@property (assign, nonatomic) int bufferDuration;         // 视频缓冲区大小

@end
```

远端视频统计

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine remoteVideoStats:(TTTRtcRemoteVideoStats*)stats;
```

stats说明：

```
@interface TTTRtcRemoteVideoStats : NSObject

@property (assign, nonatomic) int64_t uid;
@property (assign, nonatomic) NSUInteger delay;
@property (assign, nonatomic) NSUInteger width;
@property (assign, nonatomic) NSUInteger height;
@property (assign, nonatomic) NSUInteger receivedBitrate;
@property (assign, nonatomic) NSUInteger receivedFrameRate;
@property (assign, nonatomic) NSUInteger receivedFrames;    // 接收的帧数
@property (assign, nonatomic) NSUInteger lostFrames;        // 丢掉的帧数

@end
```

本地首帧视频显示

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine firstLocalVideoFrameWithSize:(CGSize)s
ize elapsed:(NSInteger)elapsed;
```

参数	描述
engine	TTTRtcEngineKit对象
size	视频流尺寸（宽度和高度）
elapsed	加入频道开始到该回调触发的延迟（毫秒)

远端首帧视频接收解码

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine firstRemoteVideoFrameDecodedOfUid:(int
64_t)uid size:(CGSize)size elapsed:(NSInteger)elapsed;
```

参数	描述
engine	TTTRtcEngineKit对象
uid	用户ID
size	视频流尺寸（宽度和高度）
elapsed	加入频道开始到该回调触发的延迟（毫秒)

摄像头启用

```
- (void)rtcEngineCameraDidReady:(TTTRtcEngineKit *)engine;
```

视频功能停止

```
- (void)rtcEngineVideoDidStop:(TTTRtcEngineKit *)engine;
```

本地视频采集

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine localVideoFrameCaptured:(TTTRtcVideoFr
ame *)videoFrame;
```

参数	描述
engine	TTTRtcEngineKit对象
videoFrame	视频帧

远端视频接收解码

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine remoteVideoFrameDecodedOfUid:(int64_t)
uid videoFrame:(TTTRtcVideoFrame *)videoFrame;
```

参数	描述
engine	TTTRtcEngineKit对象
uid	用户ID
videoFrame	视频帧

远端用户是否启用双流的

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine dualStreamModeEnabled:(BOOL)enabled us
erID:(int64_t)userID;
```

参数	描述
engine	TTTRtcEngineKit对象
enabled	是否启用了双流
userID	用户ID

请求改变视频编码参数

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine changeVideoEncodeParams:(NSUInteger)fr
ameRate bitRate:(NSUInteger)bitRate;
```

当网络发生拥塞时，会请求下调码率，反之会请求上调码率

参数	描述
engine	TTTRtcEngineKit对象
frameRate	帧率
bitRate	码率

跟视频媒体服务器连接断开

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine videoConnectFailed:(NSString *)mediaID;
```

会尝试重连

参数	描述
engine	TTTRtcEngineKit对象
mediaID	视频设备ID

4. 聊天回调

发送聊天消息成功

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine chatMessageSentOfChatInfo:(TTTRtcChatI
nfo *)chatInfo errorCode:(TTTRtcErrorCode)errorCode;
```

本端发送的聊天消息成功触发该回调

chatInfo说明：

```
@interface TTTRtcChatInfo : NSObject

@property (assign, nonatomic) TTTRtcChatType chatType;  // 聊天类型
@property (copy, nonatomic) NSString *seqID;            // 唯一标识
@property (copy, nonatomic) NSString *chatData;         // 聊天内容,语音消息是其路径
@property (assign, nonatomic) NSUInteger audioDuration; // 音频时长（单位“秒”，chatTy
pe为“Audio”）

@end
```

参数	描述
engine	TTTRtcEngineKit对象
chatInfo	聊天信息
errorCode	错误代码

收到聊天消息

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine chatMessageReceivedOfUserID:(int64_t)u
serID chatInfo:(TTTRtcChatInfo *)chatInfo;
```

参数	描述
engine	TTTRtcEngineKit对象
userID	用户ID
chatInfo	聊天信息

发送信令成功

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine signalSentOfSeqID:(NSString *)seqID da
ta:(NSString *)data errorCode:(TTTRtcErrorCode)errorCode;
```

参数	描述
engine	TTTRtcEngineKit对象
seqID	唯一标识
data	信令内容
errorCode	错误代码

收到信令

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine signalReceivedOfUserID:(int64_t)userID
 seqID:(NSString *)seqID data:(NSString *)data;
```

参数	描述
engine	TTTRtcEngineKit对象
userID	用户ID
seqID	唯一标识
data	信令内容

语音消息播放完成

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine chatAudioDidFinishPlaying:(NSString *)
fileName;
```

参数	描述
engine	TTTRtcEngineKit对象
fileName	语音消息文件名

5. 跨房间连麦回调

跨房间连麦，与其他主播连麦成功或失败的回调

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine onAnchorEnter:(int64_t)sessionID userI
D:(int64_t)userID error:(TTTRtcErrorCode)error;
```

参数	描述
engine	TTTRtcEngineKit对象
sessionID	对方主播的会话ID
userID	对方主播的用户ID
error	错误码

跨房间连麦，结束与其他主播连麦的回调

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine onAnchorExit:(int64_t)sessionID userID
:(int64_t)userID;
```

参数	描述
engine	TTTRtcEngineKit对象
sessionID	对方主播的会话ID
userID	对方主播的用户ID

跨房间连麦，其他主播向“我”发起连麦的回调

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine onAnchorLinkResponse:(int64_t)sessionI
D userID:(int64_t)userID;
```

参数	描述
engine	TTTRtcEngineKit对象
sessionID	对方主播的会话ID
userID	对方主播的用户ID

跨房间连麦，其他主播结束与“我”连麦的回调

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine onAnchorUnlinkResponse:(int64_t)sessio
nID userID:(int64_t)userID reason:(TTTRtcAnchorExitReason)reason;
```

reason说明：

```
typedef NS_ENUM(NSUInteger, TTTRtcAnchorExitReason) {
    TTTRtc_AnchorExit_Unlink    = 0, // 其他主播结束与“我”连麦，正常离开
    TTTRtc_AnchorExit_Timeout   = 1, // 心跳超时断开
    TTTRtc_AnchorExit_LinkClose = 2, // 网络异常断开
};
```

参数	描述
engine	TTTRtcEngineKit对象
sessionID	对方主播的会话ID
userID	对方主播的用户ID
reason	连麦主播离开原因

6. 其他回调

设置SEI的回调

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine onSetSEI:(NSString *)SEI;
```

参数	描述
engine	TTTRtcEngineKit对象
SEI	JSON格式的SEI

RTMP推流回调

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine onStatusOfRtmpPublish:(TTTRtcRtmpPublishStatus)status;
```

status说明：

```
typedef NS_ENUM(NSUInteger, TTTRtcRtmpPublishStatus) {
    TTTRtc_RtmpPublishStatus_InitError     = 0, // 初始化RTMP发送器失败
    TTTRtc_RtmpPublishStatus_OpenError     = 1, // 打开RTMP链接失败
    TTTRtc_RtmpPublishStatus_AudioNoBuf    = 2, // 音频数据缓冲区空间不足
    TTTRtc_RtmpPublishStatus_VideoNoBuf    = 3, // 视频数据缓冲区空间不足
    TTTRtc_RtmpPublishStatus_LinkFailed    = 4, // 发送视频数据失败
    TTTRtc_RtmpPublishStatus_LinkSuccessed = 5, // 推流成功
};
```

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine reportH264SeiContent:(NSString*)content uid:(int64_t)uid;
```

```
- (void)rtcEngineOnMediaSending:(TTTRtcEngineKit *)engine;
```