三体云-音频通话API

API功能

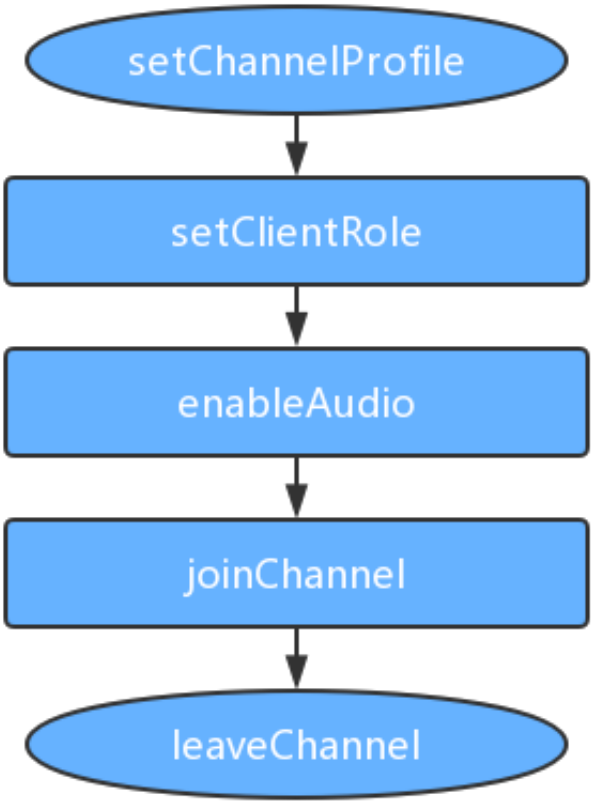
1. 接口功能

1. 通用功能
2. 房间操作
3. 语音设置
4. 伴奏设置
5. 聊天信息

2. 回调功能

1. 基本回调
2. 音频回调
3. 聊天回调
4. 其他回调

3. 语音通话流程



4.SDK接入工程

SDK包含**TTTRtcEngineVoiceKit.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
    - libc++.tbd
    - libz.tbd
    - AudioToolbox.framework
    - AVFoundation.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: 现阶段，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，音频通话请选择**通信模式**。

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
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	通信模式下，默认是该角色
观众	TTTRtc*ClientRole*Audience	在通信模式下，观众不能发言

音频通话在设置为通信模式下，默认角色是副播，如果不需要观众角色，可忽略此方法

参数	描述
role	角色
permissionKey	连麦鉴权密钥，语音通话，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

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

踢出房间

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

角色为“TTTRtc*ClientRole*Anchor”调用有效

配置旁路直播推流

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

有**TTTPublisherConfigurationBuilder**生成**TTTPublisherConfiguration** 必须调用-(TTTPublisherConfigurationBuilder *)setPublishPureAudio:(BOOL)isPureAudio;设置为纯音频

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

启用/关闭本地音频和远端音频数据回调

```
- (int)enableAudioDataReport:(BOOL)enableLocal remote:(BOOL)enableRemote;
```

参数	描述
enableLocal	YES: 获取本地音频数据，NO: 关闭获取本地音频数据
enableRemote	YES: 获取远端音频数据，NO: 关闭获取远端音频数据

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:开启高音质会占用较大的带宽

开启网络质量检测

```
- (int)enableLastmileTest;
```

关闭网络质量检测

```
- (int)disableLastmileTest;
```

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. 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;
```

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*NoAudioData	长时间没有上行音频数据
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; // 音频发送码率 (kbps)，瞬
时值
@property (assign, nonatomic) NSUInteger rxAudioKBitrate; // 音频接收码率 (kbps)，瞬
时值
@property (assign, nonatomic) NSUInteger users;           // 房间内的瞬时人数
@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

音频连麦主播拥有踢出用户的权限

Enum参数	描述
TTTRtc*KickedOut*KickedByHost	被主播踢出
TTTRtc*KickedOut*PushRtmpFailed	rtmp推流失败
TTTRtc*KickedOut*ServerOverload	服务器过载
TTTRtc*KickedOut*MasterExit	主播已退出
TTTRtc*KickedOut*ReLogin	重复登录
TTTRtc*KickedOut*NoAudioData	长时间没有上行音频数据
TTTRtc*KickedOut*NoVideoData	长时间没有上行视频数据
TTTRtc*KickedOut*NewChairEnter	其他人以主播身份进入
TTTRtc*KickedOut*ChannelKeyExpired	Channel Key失效

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

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)
@property (assign, nonatomic) NSUInteger captureDataSize; // push数据量

@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;
@property (assign, nonatomic) NSUInteger loseRate;          //丢包率
@property (assign, nonatomic) NSUInteger bufferDuration;    //缓存时常
@property (nonatomic, assign) NSUInteger delay;

@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]

本端音频采集数据回调

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine localAudioData:(char *)data dataSize:(
NSUInteger)size sampleRate:(NSUInteger)sampleRate channels:(NSUInteger)channels;
```

参数	描述
engine	TTTRtcEngineKit对象
data	PCM数据
size	PCM数据长度
sampleRate	采样率
channels	声道数

远端音频数据回调

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine remoteAudioData:(char *)data dataSize:
(NSUInteger)size sampleRate:(NSUInteger)sampleRate channels:(NSUInteger)channels;
```

参数	描述
engine	TTTRtcEngineKit对象
data	音频数据
size	数据长度
sampleRate	采样率
channels	声道数

3. 聊天回调

发送聊天消息成功

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine chatMessageSentOfChatInfo:(TTTRtcChatInfo *)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”，chatType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)userID 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

4. 其他回调

网络质量检测回调

```
- (void)rtcEngine:(TTTRtcEngineKit *)engine lastmileQuality:(TTTNetworkQuality)qua
lity;
```

quality说明：

Enum参数	描述
TTTNetworkQualityExcellent	延迟<=50ms
TTTNetworkQualityGood	50ms>延迟<=100ms
TTTNetworkQualityCommon	100ms>延迟<=200ms
TTTNetworkQualityPoor	200ms>延迟<=500ms
TTTNetworkQualityBad	延迟>500ms
TTTNetworkQualityDown	没有网络