目录

目录 1
使用Token鉴权 6
一、鉴权流程概览 6
实现鉴权详情 6
获取App ID 6
获取App证书 6
部署Token服务器 7
概览 7
快速开始 9
一、准备事项 9
注意 9
二、集成SDK 9
1. 复制ksy_rtcsdk.aar包到项目中 9
2. gradle中依赖sdk_debug.aar 9
3. 添加权限 9
三、初始化KRTC SDK 9
1. 申请动态权限 9
2. 初始化SDK 10
3. 创建KRtcEngine 11
四、加入RTC房间 11
1. 采集&编码相关配置 11
2. 本地音视频相关设置 11
音频设置 12
视频设置 12
开始音频、视频捕获及开启预览 12
3. 加入房间 12
五、离开RTC房间 12
1. 离开RTC房间需先断开房间的推流，然后离开房间，都需要传入房间ID 12
六、实现音视频通话 12
1. 配置远端用户的视频 12
2. 远端用户加入房间 12
3. 远端用户离开房间 13
七、关闭KRTCEngine 13
八、混淆 13
KRtcEngine 13
音乐文件播放及混音-方法 13
1 adjustAudioMixingPublishVolume 13
2 startAudioMixing 13
3 stopAudioMixing 13
4 pauseAudioMixing 13
5 resumeAudioMixing 13
6 getAudioMixingPlayoutVolume 13
7 getAudioMixingPublishVolume 13
8 getAudioMixingDuration 13
9 getAudioMixingCurrentPosition 13
10 setAudioMixingPosition 14
11 setAudioMixingPitch 14
自定义视频模块 14
1 setLocalVideoRenderer 14
2 setRemoteVideoRenderer 14

原始音频数据-方法 14
1 registerAudioFrameObserver 14
2 setRecordingAudioFrameParameters 14
3 setPlaybackAudioFrameParameters 14
CDN推流 14
1 setLiveTranscoding 14
2 addPublishStreamUrl 14
3 removePublishStreamUrl 15
音频管理 15
1 setAudioProfile 15
2 adjustRecordingSignalVolume 15
3 adjustAudioMixingPlayoutVolume 15
4 adjustPlaybackSignalVolume 15
5 adjustUserPlaybackSignalVolume 15
6 enableLocalAudio 15
7 muteLocalAudioStream 15
8 muteRemoteAudioStream 15
9 muteAllRemoteAudioStreams 15
10 enableAudio 16
视频管理 16
1 setupLocalVideo 16
2 setupRemoteVideo 16
3setVideoEncoderConfiguration 16
4 enableLocalVideo 16
5 startPreview 16
6 stopPreview 16
7 muteLocalVideoStream 16
8 muteRemoteVideoStream 16
9 muteAllRemoteVideoStreams 16
10 setLocalRenderMode 17
11 setRemoteRenderMode 17
12 setLocalRenderMode 17
13 setRemoteRenderMode 17
14 enableVideo 17
直播水印 17
1 addVideoWatermark 17
2 clearVideoWatermarks 17
摄像头控制 17
1 switchCamera 17
2 isCameraZoomSupported 17
3 isCameraTorchSupported 18
4 isCameraFocusSupported 18
5 isCameraExposurePositionSupported 18
6 isCameraAutoFocusFaceModeSupported 18
7 setCameraZoomFactor 18
8 getCameraMaxZoomFactor 18
9 setCameraFocusPositionInPreview 18
10 setCameraExposurePosition 18
11 setCameraTorchOn 18
12 setCameraAutoFocusFaceModeEnabled 18
其他SDK通用 18
1 getSdkVersion 18

日志文件 19
1 setLogFilter 19
核心方法 19
1 joinChannel 19
2 leaveChannel 19
3 getConnectionState 19
4 renewToken 19
5 destroy 19
6 switchChannel 19
7 setClientRole 19
8 createKRtcEngine 19
IKRtcEngineEventHandler 20
本地媒体-事件 20
1 onAudioPublishStateChanged 20
2 onVideoPublishStateChanged 20
3 onLocalAudioStateChanged 20
远端媒体-事件 20
1 onRemoteVideoStateChanged 20
2 onRemoteAudioStateChanged 20
3 onAudioSubscribeStateChanged 20
4 onVideoSubscribeStateChanged 20
数据统计-事件 21
1 onRtcStats 21
2 onRemoteAudioStats 21
3 onRemoteVideoStats 21
4 onLocalAudioStats 21
5 onLocalVideoStats 21
CDN推流 21
1 onRtmpStreamingStateChanged 21
2 onTranscodingUpdated 21
音频管理 21
1、onAudioVolumeIndication 21
2、onActiveSpeaker 21
其他SDK通用 21
1 onWarning 21
概览 21
快速开始 28
一、 前提条件 28
二、创建 iOS 项目 28
三、获取 SDK 31
四、运行项目 31
五、基本流程 38
KRtcEngine 38
音乐文件播放及混音-方法 38
1 adjustAudioMixingPublishVolume 38
2 startAudioMixing 38
3 stopAudioMixing 39
4 pauseAudioMixing 39
5 resumeAudioMixing 39
6 getAudioMixingPlayoutVolume 39
7 getAudioMixingPublishVolume 39
8 getAudioMixingDuration 39

9 getAudioMixingCurrentPosition 39
10 setAudioMixingPosition 39
11 setAudioMixingPitch 39
自定义视频模块 39
1 setLocalVideoRenderer 39
2 setRemoteVideoRenderer 39
原始音频数据-方法 39
1 registerAudioFrameObserver 39
2 setRecordingAudioFrameParameters 39
3 setPlaybackAudioFrameParameters 40
CDN推流 40
1 setLiveTranscoding 40
2 addPublishStreamUrl 40
3 removePublishStreamUrl 40
音频管理 40
1 setAudioProfile 40
2 adjustRecordingSignalVolume 40
3 adjustAudioMixingPlayoutVolume 40
4 adjustPlaybackSignalVolume 40
5 adjustUserPlaybackSignalVolume 40
6 enableLocalAudio 41
7 muteLocalAudioStream 41
8 muteRemoteAudioStream 41
9 muteAllRemoteAudioStreams 41
10 enableAudio 41
视频管理 41
1 setupLocalVideo 41
2 setupRemoteVideo 41
3setVideoEncoderConfiguration 41
4 enableLocalVideo 41
5 startPreview 42
6 stopPreview 42
7 muteLocalVideoStream 42
8 muteRemoteVideoStream 42
9 muteAllRemoteVideoStreams 42
10 setLocalRenderMode 42
11 setRemoteRenderMode 42
12 setLocalRenderMode 42
13 setRemoteRenderMode 42
14 enableVideo 43
直播水印 43
1 addVideoWatermark 43
2 clearVideoWatermarks 43
摄像头控制 43
1 switchCamera 43
2 isCameraZoomSupported 43
3 isCameraTorchSupported 43
4 isCameraFocusSupported 43
5 isCameraExposurePositionSupported 43
6 isCameraAutoFocusFaceModeSupported 43
7 setCameraZoomFactor 43
8 getCameraMaxZoomFactor 43

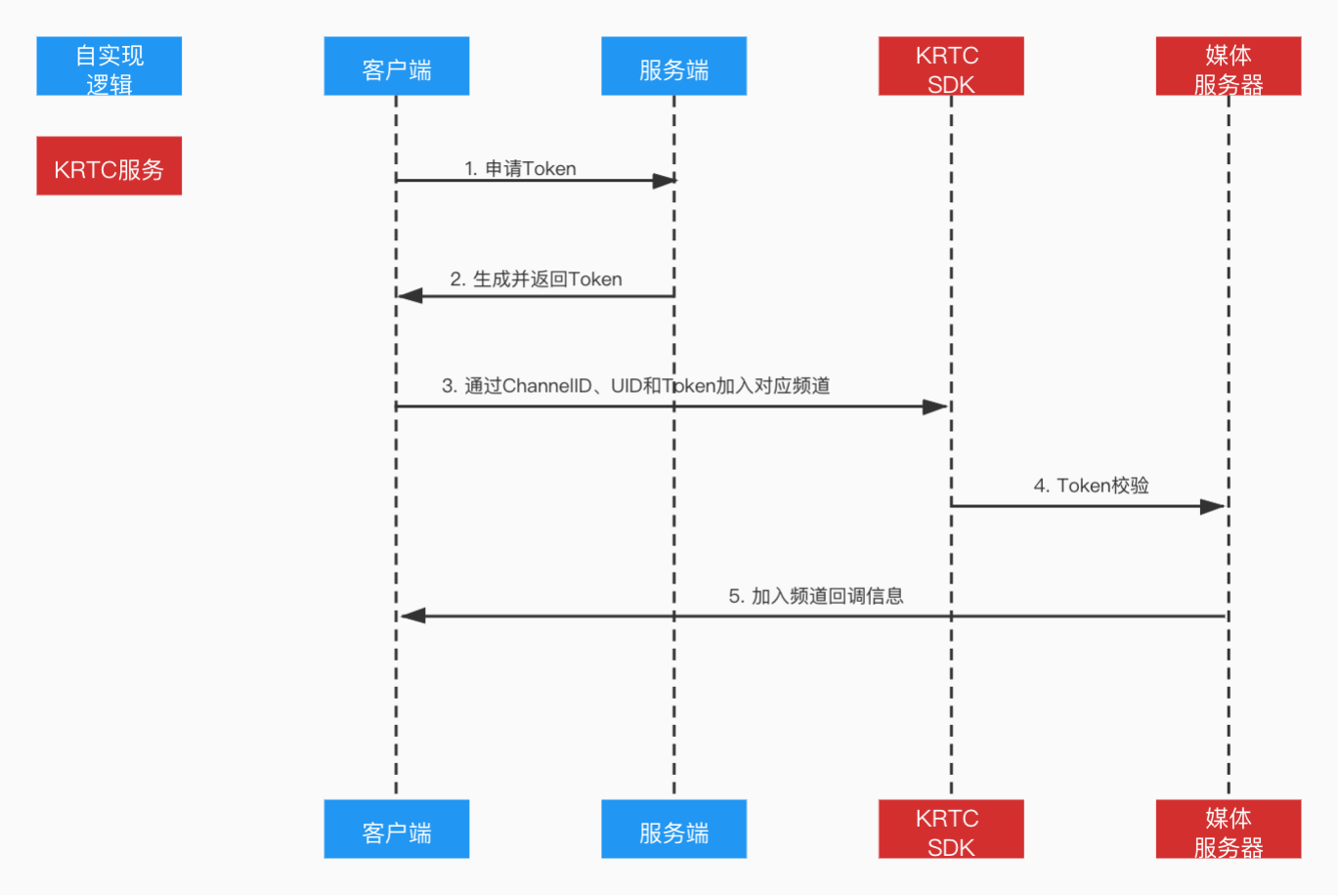
9 setCameraFocusPositionInPreview 43
10 setCameraExposurePosition 44
11 setCameraTorchOn 44
12 setCameraAutoFocusFaceModeEnabled 44
其他SDK通用 44
1 getSdkVersion 44
日志文件 44
1 setLogFilter 44
核心方法 44
1 joinChannel 44
2 leaveChannel 44
3 getConnectionState 45
4 renewToken 45
5 destroy 45
6 switchChannel 45
7 setClientRole 45
8 createKRtcEngine 45
IKRtcEngineEventHandler 45
本地媒体-事件 45
1 onAudioPublishStateChanged 45
2 onVideoPublishStateChanged 45
3 onLocalAudioStateChanged 45
远端媒体-事件 46
1 onRemoteVideoStateChanged 46
2 onRemoteAudioStateChanged 46
3 onAudioSubscribeStateChanged 46
4 onVideoSubscribeStateChanged 46
数据统计-事件 46
1 onRtcStats 46
2 onRemoteAudioStats 46
3 onRemoteVideoStats 46
4 onLocalAudioStats 46
5 onLocalVideoStats 46
CDN推流 46
1 onRtmpStreamingStateChanged 46
2 onTranscodingUpdated 46
音频管理 47
1、onAudioVolumeIndication 47
2、onActiveSpeaker 47
其他SDK通用 47
1 onWarning 47

使用Token鉴权

当您的用户使用KRTC服务，例如加入频道进行音视频通话，需要使用Token进行鉴权。

本文档将会讲述如何在您的服务端部署Token生成器、同时如何搭建使用Token鉴权的客户端。

一、鉴权流程概览



Token 需要在您的app 服务器上生成，有效期为 24 小时。当用户 app 的客户端与KRTC channel建联时，KRTC 平台会对该 Token进行鉴权。Token 包含以下信息：

- 你在 KRTC 控制台创建项目时生成的 App ID
- 你的项目的 App 证书
- 频道名
- 用户 ID
- 用户权限，如是否能发流或收流
- Token 过期的 Unix 时间戳

实现鉴权详情

获取App ID

在KRTC控制台，找到新建的项目，复制App ID。

获取App证书

点击“编辑”，进入项目编辑页面。点击眼镜图标即可复制App证书。

部署Token服务器

Token 需要在你的服务端部署生成。当客户端发送请求时，服务端会生成相应的 Token，并返回给客户端。

1.解压文件 2.进入工程文件，运行时sh release/make.sh 3.使用rpm工具安装生的的rpm 包 rpm -Uhv xxx.rpm —force 4.在启动token服务 /usr/local/krtc_token_demo/krtc_center start

概览

KRTC通过全球部署的虚拟网络，提供可以灵活搭配的 API 组合，为移动端到移动端以及移动端到 Web 端提供质量可靠的实时音视频通信。

核心方法	序号	方法	描述
1	joinChannel	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，可以群聊核心	
2	leaveChannel	离开频道核心	
3	getConnectionState	获取当前频道连接状态核心	
4	renewToken	更新 Token核心	
5	destroy	销毁 KRtcEngine对象核心	
6	switchChannel	快速切换直播频道核心	
7	setClientRole	设置用户角色核心	
8	createKRtcEngine	创建 KRtcEngine 对象（单例）并返回实例	

核心事件	序号	方法	描述
1	onJoinChannelSuccess	表示该客户端成功加入了指定的频道核心	
2	nRejoinChannelSuccess	重新加入频道回调核心有时候由于网络原因，客户端可能会和服务器失去连接，SDK 会进行自动重连，自动重连成功后触发此回调方法核心	
3	onConnectionLost	网络连接丢失回调	
4	onLeaveChannel	离开频道回调	
5	onUserOffline	远端用户（通信场景）/主播（直播场景）离开当前频道回调核心提示有远端用户/主播离开了频道（或掉线）核心	
6	onConnectionStateChanged	网络连接状态已改变回调核心该回调在网络连接状态发生改变的时候触发，并告知用户当前的网络连接状态和引起网络状态改变的原因核心	
7	onUserJoined	远端用户（通信场景）/主播（直播场景）加入当前频道回调核心	
8	onTokenPrivilegeWillExpire	Token 服务即将过期回调核心	
9	onRequestToken	Token 已过期回调核心	
10	onClientRoleChanged	直播场景下用户角色已切换回调核心	
11	onNetworkTypeChanged	本地网络类型发生改变回调核心	

音频管理	序号	事件	描述
1	setAudioProfile	根据场景需求配置适合的音质属性	
2	adjustRecordingSignalVolume	【设置采集音量】调节录音信号音量，需要对应方式获取目前音量便于管理，下同	
3	adjustAudioMixingPlayoutVolume	调节音乐文件在本地的播放音量	
4	adjustPlaybackSignalVolume	【设置播放音量】调节所有远端用户的音量	
5	adjustUserPlaybackSignalVolume	【设置播放音量】调节指定远端用户的音量	
6	enableLocalAudio	开关本地音频采集核心	
7	muteLocalAudioStream	开关本地音频发送核心	
8	muteRemoteAudioStream	接收 / 停止接收指定音频流核心	

9	muteAllRemoteAudioStreams	接收 / 停止接收所有音频流核心
10	setDefaultMuteAllRemoteAudioStreams	设置是否默认接收所有音频流核心
13	enableAudio	启用音频模块

序号	事件	描述
1	onAudioVolumeIndication	【获取用户音量（回调）】回调报告频道内瞬时音量最高的几个用户（即说话者）的用户 ID 及他们的音量
2	onActiveSpeaker	回调报告特定时间段内累积音量最高的用户 ID

视频管理	序号	事件	描述
1	setupLocalVideo		初始化本地视图：设置本地视频镜像，使用前置摄像头，SDK 默认启用镜像模式；如果你使用后置摄像头，SDK 默认关闭镜像模式
2	setupRemoteVideo		初始化远端用户视图核心
3	setVideoEncoderConfiguration		设置视频编码属性核心每个属性对应一套视频参数，如分辨率、帧率、码率、视频方向等核心
4	enableLocalVideo		开关本地视频采集核心
5	startPreview		开启视频预览核心
6	stopPreview		停止视频预览核心
7	muteLocalVideoStream		开关本地视频发送核心
8	muteRemoteVideoStream		接收 / 停止接收指定远端用户的视频流核心
9	muteAllRemoteVideoStreams		接收 / 停止接收所有远端视频流核心
10	setLocalVideoMirrorMode		设置本地视频镜像(2020.3.1，xuxiaoguang注,声网已废弃该接口，调用setupLocalVideo(开始时设置) 或 setLocalRenderMode(过程中设置))
11	setLocalRenderMode		更新本地视图显示模式核心
12	setRemoteRenderMode		更新远端视图显示模式核心
13	setDefaultMuteAllRemoteVideoStreams		设置是否默认停止接收视频流核心
14	setLocalRenderMode		更新本地视图显示模式核心
15	setRemoteRenderMode		更新远端视图显示模式核心
16	setDefaultMuteAllRemoteVideoStreams		设置是否默认停止接收视频流核心
17	enableVideo		开关本地音频采集

本地媒体	序号	事件	描述
1	onAudioPublishStateChanged	音频发布状态改变回调核心	
2	onVideoPublishStateChanged	视频发布状态改变回调核心	
3	onLocalAudioStateChanged	报告当前本地音频状态核心当本地音频出现故障时，你可以通过错误码排查问题核心	
4	onLocalVideoStateChanged	报告当前本地视频状态核心当本地视频出现故障时，你可以通过错误码排查问题核心	

远端媒体	序号	事件	描述
1	onRemoteVideoStateChanged	报告远端视频流当前状态和状态改变的原因核心	
2	onRemoteAudioStateChanged	远端用户音频流状态已变化回调核心	
3	onAudioSubscribeStateChanged	音频订阅状态发生改变回调核心	
4	onVideoSubscribeStateChanged	视频订阅状态发生改变回调核心	

数据统计	序号	事件	描述
1	onRtcStats	报告本地通话统计信息核心你可以了解到通话时长、当前通话频道中的人数、当前系统的 CPU 使用率、当前 App 的 CPU 使用率和以下重要参数核心	
2	onRemoteAudioStats	报告当前通话中每个远端用户/主播音频流的统计信息核心你可以了解到每个远端用户/主播发送的音频流质量、声道数（单声道或双声道）和其他重要参数信息核心	
3	onRemoteVideoStats	通话中远端视频流统计信息回调	
4	onLocalAudioStats	通话中本地音频流统计信息回调	
5	onLocalVideoStats	报告本地设备发送视频流的统计信息核心你可以了解视频编码宽/高和以下重要参数信息核心	

音乐文件播放及混音	序号	方法	描述
1	adjustAudioMixingPublishVolume	调节音乐文件在远端的播放音量	
2	startAudioMixing	开始播放音乐文件及混音，同时让频道内的其他人听到此音乐	
3	stopAudioMixing	停止播放音乐文件及混音核心	
4	pauseAudioMixing	暂停播放音乐文件	
5	resumeAudioMixing	恢复播放音乐文件	
6	getAudioMixingPlayoutVolume	获取音乐文件的本地播放音量	
7	getAudioMixingPublishVolume	获取音乐文件的远端播放音量	
8	getAudioMixingDuration	获取音乐文件播放时长	
9	getAudioMixingCurrentPosition	获取音乐文件的播放进度	
10	setAudioMixingPosition	设置音乐文件的播放位置	
11	setAudioMixingPitch	调整本地播放的音乐文件的音调	

CDN推流	序号	方法	描述
1	setLiveTranscoding	设置音视频流的直播参数，如分辨率、码率、帧率、水印和背景色位置核心如果是多主播转码合图，需设置每个主播的参数，再次调用可以更新参数	
2	addPublishStreamUrl	主播设置CDN推流地址	
3	removePublishStreamUrl	删除推流地址	

序号	方法	描述
1	onRtmpStreamingStateChanged	回调当前推流状态
2	onTranscodingUpdated	旁路推流设置被更新回调

语音播放路由	序号	方法	描述
1	setDefaultAudioRoutetoSpeakerphone	设置默认的音频播放路由	
2	setEnableSpeakerphone	启用/关闭扬声器播放	
3	isSpeakerphoneEnabled	查询扬声器启用状态	

序号	事件	描述
1	onAudioRouteChanged	语音路由已改变回调
2	onAudioRouteChanged	语音路由已改变回调

通话前网络测试	序号	事件	描述
1	onLastmileQuality	通话前网络上下行 last mile 质量报告回调，该回调描述本地用户在加入频道前的 last mile 网络探测的结果	
2	onLastmileProbeResult	通话前网络质量探测报告回调，通话前网络上下行 Last mile 质量探测报告回调核心	

自定义视频模块	序号	方法	描述
1	setLocalVideoRenderer	自定义本地视频渲染器	
2	setRemoteVideoRenderer	自定义远端视频渲染器	

视频自采集（仅适用于 Push 模式）	序号	方法	描述
1	pushExternalVideoFrame	完成视频数据处理后，再通过 pushExternalVideoFrame 发送给 SDK 进行后续操作	

音频自采集（仅适用于 Push 模式）	序号	方法	描述
1	setExternalAudioSource	自定义音频采集，指定外部音频采集设备	
2	pushExternalAudioFrame	音频数据处理后发送给SDK	

原始音频数据	序号	方法	描述
1	registerAudioFrameObserver	注册语音观测器对象	
2	setRecordingAudioFrameParameters	设置采集的音频格式核心	
3	setPlaybackAudioFrameParameters	设置播放的声音格式核心	

原始视频数据	序号	方法	描述
1	registerVideoFrameObserver	加入频道前调用 registerVideoFrameObserver 方法注册视频观测器，并在该方法中实现一个 IVideoFrameObserver 类核心	

序号	事件	描述

1	onCaptureVideoFrame	获取本地采集的视频数据
2	onPreEncodeVideoFrame	获取本地编码前的视频数据
3	onRenderVideoFrame	获取接收到的远端视频数据

媒体附属信息	序号	方法	描述
1	registerMediaMetadataObserver	直播的视频帧中同步添加 Metadata，注册媒体 Metadata 观测器	

直播水印	序号	方法	描述
1	addVideoWatermark	添加本地视频水印	
2	clearVideoWatermarks	删除已添加的视频水印	

摄像头控制	序号	方法	描述
1	switchCamera	切换前置/后置摄像头	
2	isCameraZoomSupported	检测设备是否支持摄像头缩放功能	
3	isCameraTorchSupported	检测设备是否支持闪光灯常开	
4	isCameraFocusSupported	检测设备是否支持手动对焦功能	
5	isCameraExposurePositionSupported	检测设备是否支持手动曝光功能	
6	isCameraAutoFocusFaceModeSupported	检测设备是否支持人脸对焦功能	
7	setCameraZoomFactor	设置摄像头缩放比例	
8	getCameraMaxZoomFactor	获取摄像头支持最大缩放比例	
9	setCameraFocusPositionInPreview	设置手动对焦位置，并触发对焦	
10	setCameraExposurePosition	设置手动曝光位置	
11	setCameraTorchOn	设置是否打开闪光灯	
12	setCameraAutoFocusFaceModeEnabled	设置是否开启人脸对焦功能	

其他SDK通用	序号	方法	描述
1	getSdkVersion	查询 SDK 版本号	
2	getErrorDescription	获取 SDK 警告或错误描述	

序号	事件	描述
-	-	-
1	onWarning	SDK 发生警告回调

日志文件	序号	方法	描述
1	setLogFile	设置日志文件	
2	setLogFilter	设置日志输出等级	
3	setLogFileSize	设置日志文件大小	

快速开始

一、准备事项

1. 创建自己将要开发的工程。
2. 获取ksy_rtcsdk.aar包。

注意

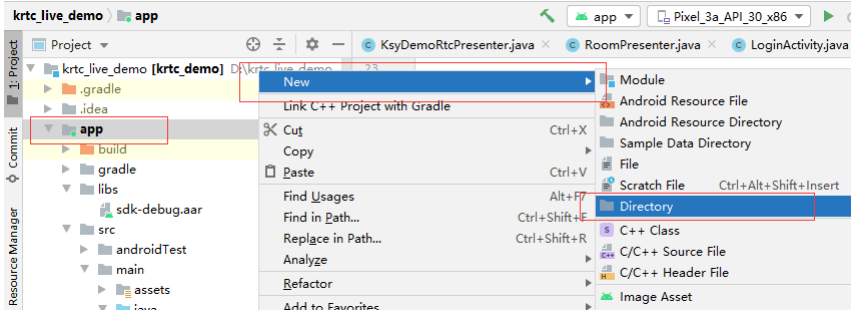
ksy_rtcsdk支持最小支持Android的sdk版本19：minSdkVersion 19

二、集成SDK

以AndroidStudio为例

1. 复制ksy_rtcsdk.aar包到项目中

复制sdk_debug.aar到需要集成的项目中Moudle的libs文件夹中，libs文件夹需要与src文件夹同层级。如果不存在libs文件夹，右键当前的Moulde创建libs文件夹（Moudle-New-Directory），如下图所示：



2. gradle中依赖sdk_debug.aar

在所导入的Moudle的build.gradle中加入

```
repositories {
    flatDir{
        dirs 'libs'
    }
}
```

在dependencies的节点下加入：implementation(name:'ksy_rtcsdk',ext:'aar')

3. 添加权限

在导入moudle的AndroidMainfest.xml文件中添加权限

```
<uses-permission android:name="android.permission.RECORD_AUDIO" />
<uses-permission android:name="android.permission.CAMERA" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.MODIFY_AUDIO_SETTINGS" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

三、初始化KRTC SDK

1. 申请动态权限

创建直播前需要获取以下检测摄像头权限、音频权限、读写权限

```
int cameraPerm = ActivityCompat.checkSelfPermission(mContext, Manifest.permission.CAM
int audioPerm = ActivityCompat.checkSelfPermission(mContext, Manifest.permission.RECO
if (cameraPerm != PackageManager.PERMISSION_GRANTED ||
        audioPerm != PackageManager.PERMISSION_GRANTED) {
    if (Build.VERSION.SDK_INT < Build.VERSION_CODES.M) {
        Log.e(TAG, "No CAMERA or AudioRecord permission, please check");
        Toast.makeText(mContext.getApplicationContext(), "No CAMERA or AudioRecord pe
                Toast.LENGTH_LONG).show();
    } else {
        String[] permissions = {Manifest.permission.CAMERA,
                Manifest.permission.RECORD_AUDIO,
                Manifest.permission.READ_EXTERNAL_STORAGE,
                Manifest.permission.WRITE_EXTERNAL_STORAGE};
        ActivityCompat.requestPermissions((Activity) mContext, permissions,
                RoomActivity.PERMISSION_REQUEST_CAMERA_AUDIOREC);
    }
} else {
    //有相应权限进行业务操作
}
```

2. 初始化SDK

实现rtc返回的相关回调IKRtcEngineEventHandler，在createKRTcEngine时注册。

```
private IKRtcEngineEventHandler ikRtcEngineEventHandler = new IKRtcEngineEventHandle
    @Override
    public void onSDKInitFinish(boolean result) {
        super.onSDKInitFinish(result);
    }

    @Override
    public void onConnectionStateChanged(int state, int reason) {
        super.onConnectionStateChanged(state, reason);
    }

    @Override
    public void onJoinChannelSuccess(String channel, String uid, int elapsed) {
        super.onJoinChannelSuccess(channel, uid, elapsed);
    }

    @Override
    public void onLeaveChannel(RtcStats stats) {
        super.onLeaveChannel(stats);
    }

    @Override
    public void onUserJoined(String uid, int elapsed) {
        super.onUserJoined(uid, elapsed);
    }

    @Override
    public void onUserOffline(String uid, int reason) {
        super.onUserOffline(uid, reason);
    }
```

3. 创建KRtcEngine

设置日志文件大小，可决定是否开启，并对文件进行大小设置。

KRtcEngine.setLogFileSize(5* 1024);

创建KRtcEngine,传入自己的App ID，传入回调接口：

this.mRtcEngine = KRtcEngine.createKRtcEngine(context, appId, ikRtcEngineEventHandler);

四、加入RTC房间

当SDK初始化完成之后，在onSDKInitFinish回调方法中进行以下操作：

1. 采集&编码相关配置

在sdk初始化完成之后的回调方法中调用。 设置摄像头，默认为前置

```
mRtcEngine.setCameraCaptureConfiguration(new KCameraCaptureConfiguration(KCameraCaptu

mRtcEngine.setDefaultAudioRouteToSpeakerphone(true);

mRtcEngine.setAudioProfile(AudioEncoderConfiguration.AudioProfile.getValue(AudioEncoc
AudioEncoderConfiguration.AudioScenario.getValue(AudioEncoderConfiguration.AudioScena

mRtcEngine.setVideoEncoderConfiguration(new VideoEncoderConfiguration(
VideoEncoderConfiguration.VD_320x240,
VideoEncoderConfiguration.FRAME_RATE.FRAME_RATE_FPS_15,
VideoEncoderConfiguration.STANDARD_BITRATE, 400));
```

2. 本地音视频相关设置

音视频相关配置及增加本地的展示view传给sdk。

音频设置

```
mRtcEngine.setupLocalAudio();
```

视频设置

创建KSurfaceView对象

```
mLocalView = KRtcEngine.createRendererView(getBaseContext());
```

将当前surfaceView放在最上层

```
mLocalView.setZOrderMediaOverlay(true);
```

设置surfaceView的缩放类型

```
mLocalView.setScalingType(RendererCommon.ScalingType.SCALE_ASPECT_FILL);
```

本地帧布局中放入KSurfaceView对象mLocalView

```
mLocalContainer.addView(mLocalView);
```

将KSurfaceView传给KVdieoCanvas,并设置本地视频

```
mRtcEngine.setupLocalVideo(new KVideoCanvas(mLocalView, ""));
```

```
mRtcEngine.setClientRole(Role.CLIENT_ROLE_BROADCASTER);
```

开始音频、视频捕获及开启预览

```
mRtcEngine.enableLocalAudio(true);
mRtcEngine.enableLocalVideo(true);
mRtcEngine.startPreview();
```

3. 加入房间

在SDK初始化成功的回调中及设置好相关配置后，传入Token、Channel ID、用户名、用户ID

```
mRtcEngine.joinChannel("token", "channelID", "userName", "userId");
```

加入房间成功后会收到onJoinChannelSuccess回调，如果重新加入房间会收到onRejoinChannelSuccess回调，在这个两个方法中设置音频、视频推流：

```
mRtcEngine.muteLocalAudioStream(false);
mRtcEngine.muteLocalVideoStream(false);
mRtcEngine.muteAllRemoteAudioStreams(false);
mRtcEngine.muteAllRemoteVideoStreams(false);
```

五、离开RTC房间

1. 离开RTC房间需先断开房间的推流，然后离开房间，都需要传入房间ID

```
mRtcEngine.leaveChannelStream(channelId);
```

```
mRtcEngine.leaveChannel(channelId);
```

离开房间成功会收到onLeaveChannel()回调，界面需要将KSurfaceView对象删除

```
mLocalContainer.removeAllViews();
```

六、实现音视频通话

1. 配置远端用户的视频

创建远端用户的KSurfaceView,界面显示的位置可自行决定。

```
mRemoteView = KRtcEngine.createRendererView(getBaseContext());
```

```
mRemoteView.setScalingType(RendererCommon.ScalingType.SCALE_ASPECT_FILL);
mRemoteContainer.addView(mRemoteView);
```

将远端用户的KSurfaceView对象设置到setupRemoteVideo中

```
mRtcEngine.setupRemoteVideo(new KVideoCanvas(mRemoteView, uid));
```

2. 远端用户加入房间

有远端用户加入房间回调onUserJoined，在回调中需设置：

```
new Thread(() -> {

mRtcEngine.muteRemoteAudioStream("uid",false);
mRtcEngine.muteRemoteVideoStream("uid",false);

}).start();
```

3. 远端用户离开房间

有远端用户离开房间会回调onUserOffline，然后界面需要释放远端的KSurfaceView对象。

```
mRemoteContainer.removeAllViews();
```

七、关闭KRTCEngine

当退出rtc视频通话后需要关闭销毁KRTCEngine

```
mRtcEngine.destroy();
```

八、混淆

```
-keep class com.ksyun.krtc.**{ *;}

-dontwarn com.ksyun.krtc.**

-keep class org.krtc.**{ *;}

-dontwarn org.krtc.**
```

KRtcEngine

音乐文件播放及混音-方法

1 adjustAudioMixingPublishVolume

调节音乐文件在远端的播放音量。 public int adjustAudioMixingPublishVolume(int volume)

参数	参数	说明
volume	范围为[0,400]，默认值为 100，表示原始音量。最大支持4倍音量。不等于0设为0，大于400则设为400	content3

返回值 0表示成功，不等于0表示失败。

2 startAudioMixing

开始播放音乐文件及混音。 public int startAudioMixing(String filePath, boolean loopback, boolean replace, int cycle)

参数	参数	说明
filePath	指定需要混音的本地或在线音频文件的绝对路径（包含文件名后缀），如 /sdcard/emulated/0/audio.mp4。支持的音频格式包括：mp3、mp4、m4a、aac、3gp、mkv 及 wav。如果用户提供的目录以 /assets/ 开头，则去 assets 里面查找该文件。如果用户提供的目录不是以 /assets/ 开头，一律认为是在绝对路径里查找该文件。	
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：只推动设置的本地音频文件或者线上音频文件，不传输麦克风收录的音频;false：音频文件内容将会和麦克风采集的音频流进行混音	
cycle	音频文件循环播放的次数：正整数：循环的次数。-1：无限循环。	

返回值 0表示成功，不等于0表示失败。

3 stopAudioMixing

停止播放音乐文件及混音。 public int stopAudioMixing()

返回值 0表示成功，不等于0表示失败。

4 pauseAudioMixing

暂停播放音乐文件及混音。该方法暂停播放伴奏。请在频道内调用该方法。 public int pauseAudioMixing() 返回值。0表示成功，不等于0表示失败。

5 resumeAudioMixing

恢复播放音乐文件及混音。该方法恢复混音，继续播放伴奏。请在频道内调用该方法。 public int resumeAudioMixing()

返回值 0表示成功，不等于0表示失败。

6 getAudioMixingPlayoutVolume

获取音乐文件的本地播放音量。 public int getAudioMixingPlayoutVolume()

返回值 方法调用成功则返回音量值，范围为 [0, 400]。默认值为 100，表示原始音量。 不等于0：方法调用失败。

7 getAudioMixingPublishVolume

获取音乐文件的远端播放音量。 public int getAudioMixingPublishVolume()

返回值 方法调用成功则返回音量值，范围为 [0, 400]。默认值为 100，表示原始音量。 不等于0：方法调用失败。

8 getAudioMixingDuration

获取音乐文件的时长。 该方法获取音乐文件时长，单位为毫秒。请在频道内调用该方法。 public int getAudioMixingDuration()

返回值 方法调用成功，并返回伴奏时长。不等于0：方法调用失败。

9 getAudioMixingCurrentPosition

获取音乐文件的播放进度。该方法获取当前伴奏播放进度，单位为毫秒。请在频道内调用该方法。 public int getAudioMixingCurrentPosition()

返回值 方法调用成功并返回伴奏播放进度。不等于0：方法调用失败。

10 setAudioMixingPosition

设置音乐文件的播放位置。该方法可以设置音频文件的播放位置，这样你可以根据实际情况播放文件，而不是非得从头到尾播放一个文件。 public int setAudioMixingPosition(int pos)

参数	参数	说明
pos	进度条位置，单位为毫秒。	

返回值 0表示成功，不等于0表示失败。

11 setAudioMixingPitch

调整本地播放的音乐文件的音调。该方法需在 startAudioMixing 后调用。 public int setAudioMixingPitch(int pitch)

参数	参数	说明
pitch	按半音音阶调整本地播放的音乐文件的音调，默认值为 1.0，即不调整音调。取值范围为 [-12,12]，每相邻两个值的音高距离相差半音。取值的绝对值越大，音调升高或降低得越多。	

返回值 0表示成功，不等于0表示失败。

自定义视频模块

1 setLocalVideoRenderer

推送外部视频帧。该方法同setupLocaleVideo()方法，只能同时使用一个，且使用时机相同。 public int setLocalVideoRenderer(IKVideoSink render)

参数	参数	说明
render	基于模板方法的自定义渲染器	

返回值 0表示成功，不等于0表示失败。

2 setRemoteVideoRenderer

推送外部视频帧。该方法同setupRemoteVideo()方法，只能同时使用一个，且使用时机相同。 public int setRemoteVideoRenderer(String uid, IKVideoSink render)

参数	参数	说明
uid	用户ID	
render	基于模板方法的自定义渲染器	

返回值 0表示成功，不等于0表示失败。

原始音频数据-方法

1 registerAudioFrameObserver

注册音频数据观察者. public int registerAudioFrameObserver(IKRtcAudioFrameObserver observer)

参数	参数	说明
observer	音频数据观察者	

返回值 0表示成功，不等于0表示失败。

2 setRecordingAudioFrameParameters

设置采集的音频格式。该方法会关联 onRecordFrame 回调数据的格式。 需在加入频道前调用。 public int setRecordingAudioFrameParameters(int sampleRate, int channel, int mode, int samplesPerCall)

参数	参数	说明
sampleRate	onRecordFrame	中返回数据的采样率，可设置为 8000，16000，32000，44100 或 48000。
channel	onRecordFrame	中返回数据的通道数，可设置为：1：单声道。 2：双声道。
mode	onRecordFrame	的使用模式，0：只读模式，1：只写模式，2：读写模式
samplesPerCall	onRecordFrame	中返回数据的采样点数，如 RTMP/RTMPS 推流应用中通常为 1024。

返回值 0表示成功，不等于0表示失败。

3 setPlaybackAudioFrameParameters

设置播放的声音格式。 该方法指定 onPlaybackFrame 回调数据的格式。 需在加入频道前调用。 public int setPlaybackAudioFrameParameters(int sampleRate, int channel, int mode, int samplesPerCall)

参数	参数	说明
sampleRate	onRecordFrame	中返回数据的采样率，可设置为 8000，16000，32000，44100 或 48000。
channel	onRecordFrame	中返回数据的通道数，可设置为：1：单声道。 2：双声道。
mode	onRecordFrame	的使用模式，0：只读模式，1：只写模式，2：读写模式
samplesPerCall	onRecordFrame	中返回数据的采样点数，如 RTMP/RTMPS 推流应用中通常为 1024。

返回值 0表示成功，不等于0表示失败。

CDN推流

1 setLiveTranscoding

增加旁路推流地址。该方法用于旁路推流的视图布局及音频设置等。调用该方法更新转码参数 LiveTranscoding 时，SDK 会触发 onTranscodingUpdated 回调。 首次调用该方法设置转码参数时，不会触发 onTranscodingUpdated 回调。 public int setLiveTranscoding(LiveTranscoding transcoding)

参数	参数	说明
transcoding	旁路推流布局相关设置。	

返回值 0表示成功，不等于0表示失败。

2 addPublishStreamUrl

增加旁路推流地址。调用该方法后，你可以向 CDN 推送 RTMP 或 RTMPS 协议的媒体流。SDK 会在本地触发 onRtmpStreamingStateChanged 回调，报告增加旁路推流地址的状态 public int addPublishStreamUrl(String url, boolean transcodingEnabled)

参数	参数	说明
url	CDN 推流地址，格式为 RTMP。该字符长度不能超过 1024 字节。url 不支持中文等特殊字符	
transcodingEnabled	是否转码。如果设为 true，则需要在该方法前先调用setLiveTranscoding 方法。true：转码。转码是指在旁路推流时对音视频流进行转码处理后，再推送到其他 CDN 服务器。多适用于频道内有多个主播，需要进行混流、合图的场景。false：不转码	

返回值 0表示成功，不等于0表示失败。

3 removePublishStreamUrl

移除旁路推流地址。 public int removePublishStreamUrl(String url)

参数	参数	说明
url	CDN 推流地址，格式为 RTMP。该字符长度不能超过 1024 字节。url 不支持中文等特殊字符	

返回值 0表示成功，不等于0表示失败。

音频管理

1 setAudioProfile

设置音频编解码属性

public int setAudioProfile(int profile, int scenario)

参数	参数	说明
profile	设置采样率，码率，编码模式和声道数	

返回值 0表示成功，不等于0表示失败。需要在setupLocalAudio()方法调用前使用，才能生效。

2 adjustRecordingSignalVolume

【设置采集音量】调节录音信号音量，需要对应方式获取目前音量便于管理，下同

public int adjustRecordingSignalVolume(int volume)

参数	参数	说明
volume	范围为[0,400]，默认值为 100，表示原始音量。最大支持4倍音量。不等于0设为0，大于400则设为400	

返回值 0表示成功，不等于0表示失败

3 adjustAudioMixingPlayoutVolume

调节音乐文件在本地的播放音量

public int adjustAudioMixingPlayoutVolume(int volume)

参数	参数	说明
volume	范围为[0,400]，默认值为 100，表示原始音量。最大支持4倍音量。不等于0设为0，大于400则设为400	

返回值 0表示成功，不等于0表示失败

4 adjustPlaybackSignalVolume

【设置播放音量】调节所有远端用户的音量

public int adjustPlaybackSignalVolume(int volume)

参数	参数	说明
volume	范围为[0,400]，默认值为 100，表示原始音量。最大支持4倍音量。不等于0设为0，大于400则设为400	

返回值 0表示成功，不等于0表示失败

5 adjustUserPlaybackSignalVolume

【设置播放音量】调节指定远端用户的音量

public int adjustUserPlaybackSignalVolume(String uid, int volume)

参数	参数	说明
uid	用户ID	
volume	范围为[0,400]，默认值为 100，表示原始音量。最大支持4倍音量。不等于0设为0，大于400则设为400	

返回值 0表示成功，不等于0表示失败

6 enableLocalAudio

开关本地音频采集

public int enableLocalAudio(boolean enabled)

参数	参数	说明
enabled	true开启音频采集，false关闭音频采集	

返回值 0表示成功，不等于0表示失败

7 muteLocalAudioStream

开关本地音频发送核心

public int muteLocalAudioStream(boolean muted)

参数	参数	说明
muted	true关闭音频发送，false开启音频发送	

返回值 0表示接口调用成功，不等于0表示接口调用失败。（此返回值仅代表接口调用是否成功，实际音频发送是否成功，仍需要通过IKRtcEngineEventHandler的onAudioPublishStateChanged()回调来进行判断。）

8 muteRemoteAudioStream

接收/停止接收指定音频流核心

public int muteRemoteAudioStream(String uid, boolean muted)

参数	参数	说明
uid	用户ID	
muted	true关闭接收，false开启接收	

返回值 0表示成功，不等于0表示失败。（此返回值仅代表接口调用是否成功，实际音频接收/停止接收是否成功，仍需要通过IKRtcEngineEventHandler的onAudioSubscribeStateChanged()回调来进行判断。）

9 muteAllRemoteAudioStreams

接收 / 停止接收所有音频流核心

注意:同muteRemoteAudioStream接口，操作需通过回调接口确认

public int muteAllRemoteAudioStreams(boolean muted)

参数	参数	说明
muted	true关闭音频发送，false开启音频发送	

返回值 表示成功，不等于0表示失败。

10 enableAudio

启用音频模块

注意：默认为开启状态

public int enableVideo()

返回值：0表示成功，不等于0表示失败

视频管理

1 setupLocalVideo

初始化本地渲染视图

public int setupLocalVideo(KVideoCanvas kVideoCanvas)

参数	参数	说明
kVideoCanvas	本地渲染视图。	

返回值 0表示成功，不等于0表示失败。

2 setupRemoteVideo

初始化远端用户渲染视图

public int setupRemoteVideo(KVideoCanvas remote)

参数	参数	说明
remote	远端视频渲染视图。	

返回值 0表示成功，不等于0表示失败。

3setVideoEncoderConfiguration

设置视频编解码配置

public int setVideoEncoderConfiguration(VideoEncoderConfiguration config)

参数	参数	说明
config	视频编解码配置。	

返回值 0表示成功，不等于0表示失败。需要在setupLocalVideo()方法调用前使用，才能生效。

4 enableLocalVideo

开关本地视频采集。 public int enableLocalVideo(boolean enabled)

参数	参数	说明
enabled	true开启采集，false关闭采集	

返回值 0表示成功，不等于0表示失败

5 startPreview

开启视频预览

public int startPreview()

返回值：0表示成功，不等于0表示失败

6 stopPreview

停止视频预览

public int stopPreview()

返回值：0表示成功，不等于0表示失败

7 muteLocalVideoStream

开关本地视频发送

public int muteLocalVideoStream(boolean muted)

参数	参数	说明
muted	true关闭发送，false开启发送	

返回值 0表示成功，不等于0表示失败。注意，此返回值仅代表接口调用是否成功，实际视频发送是否成功，仍需要通过IKRtcEngineEventHandler的onVideoPublishStateChanged()回调来进行判断。

8 muteRemoteVideoStream

接收 / 停止接收指定远端用户的视频流

public int muteRemoteVideoStream(String uid, boolean muted)

参数	参数	说明
uid	用户ID	
muted	true关闭接收，false开启接收	

返回值 0表示成功，不等于0表示失败。注意，此返回值仅代表接口调用是否成功，实际视频接收/停止接收是否成功，仍需要通过IKRtcEngineEventHandler的onVideoSubscribeStateChanged()回调来进行判断。

9 muteAllRemoteVideoStreams

接收 / 停止接收所有远端视频流

public int muteAllRemoteVideoStreams(boolean muted)

参数	参数	说明
muted	true关闭发送，false开启发送	

返回值 0表示成功，不等于0表示失败。同muteRemoteVideoStream接口，操作需通过回调接口确认

10 setLocalRenderMode

更新本地视图显示模式

public int setLocalRenderMode(int renderMode, int mirrorMode)

参数	参数	说明
enderMode	本地渲染模式。1，比例和填充之间的妥协。2，比例优先，不保证视图完整性。3，填充优先，不保证原视频比例。-1，忽略，不设置渲染模式	
mirrorMode	镜像模式。0：关闭，1开启。-1忽略，不设置镜像模式	

返回值 0表示成功，不等于0表示失败

11 setRemoteRenderMode

public int setRemoteRenderMode(String uid, int renderMode, int mirrorMode)

参数	参数	说明
uid	用户ID	
renderMode	本地渲染模式。1，比例和填充之间的妥协。2，比例优先，不保证视图完整性。3，填充优先，不保证原视频比例。-1，忽略，不设置渲染模式	
mirrorMode	镜像模式。0：关闭，1开启	

返回值 0表示成功，不等于0表示失败。

12 setLocalRenderMode

更新本地视图显示模式

public int setLocalRenderMode(int renderMode, int mirrorMode)

参数	参数	说明
renderMode	本地渲染模式。1，比例和填充之间的妥协。2，比例优先，不保证视图完整性。3，填充优先，不保证原视频比例。-1，忽略，不设置渲染模式	
mirrorMode	镜像模式。0：关闭，1开启。-1忽略，不设置镜像模式	
renderMode	本地渲染模式。1，比例和填充之间的妥协。2，比例优先，不保证视图完整性。3，填充优先，不保证原视频比例。-1，忽略，不设置渲染模式	
mirrorMode	镜像模式。0：关闭，1开启	

返回值：0表示成功，不等于0表示失败

13 setRemoteRenderMode

更新远端视图显示模式

public int setRemoteRenderMode(String uid, int renderMode, int mirrorMode)

参数	参数	说明
renderMode	本地渲染模式。1，比例和填充之间的妥协。2，比例优先，不保证视图完整性。3，填充优先，不保证原视频比例。-1，忽略，不设置渲染模式	
mirrorMode	镜像模式。0：关闭，1开启	

返回值 0表示成功，不等于0表示失败

14 enableVideo

启用视频模块

public int enableVideo()

返回值 0表示成功，不等于0表示失败。

直播水印

1 addVideoWatermark

添加本地水印

public int addVideoWatermark(String watermarkUrl, WatermarkOptions options)

参数	参数	说明
watermarkUrl	待添加的水印图片的本地路径	
options	待添加的水印图片的设置选项	

返回值 0表示成功，不等于0表示失败

2 clearVideoWatermarks

清除水印

public int clearVideoWatermarks()

返回值：0表示成功，不等于0表示失败

摄像头控制

1 switchCamera

切换前置/后置摄像头

public int switchCamera()

返回值：0表示成功，不等于0表示失败

2 isCameraZoomSupported

检测设备是否支持摄像头缩放功能

public boolean isCameraZoomSupported()

返回值 true：设备支持相机缩放功能 false：设备不支持相机缩放功能

3 isCameraTorchSupported

检测设备是否支持闪光灯常开

public boolean isCameraTorchSupported()

返回值 true：设备支持闪光灯常开；false：设备不支持闪光灯常开

4 isCameraFocusSupported

检测设备是否支持手动对焦功能

public boolean isCameraFocusSupported()

返回值：true：设备支持手动对焦功能 false：设备不支持手动对焦功能。

5 isCameraExposurePositionSupported

检测设备是否支持手动曝光功能。

public boolean isCameraExposurePositionSupported()

返回值、true：设置支持手动曝光功能 false：设备不支持手动曝光功能

6 isCameraAutoFocusFaceModeSupported

检测设备是否支持人脸对焦功能

public boolean isCameraAutoFocusFaceModeSupported()

返回值 true：设备支持人脸对焦功能 false：设备不支持人脸对焦功能

7 setCameraZoomFactor

设置摄像头缩放比例

注意：相机缩放比例，有效范围从 1.0 到最大缩放

public int setCameraZoomFactor(float camZoomf) 参数	参数	说明
camZoomf		缩放比例

返回值 方法调用成功：返回设置的 Factor 值 方法调用失败：返回值 < 0

8 getCameraMaxZoomFactor

获取摄像头支持最大缩放比例

public float getCameraMaxZoomFactor()

返回值：该相机支持的最大缩放比例

9 setCameraFocusPositionInPreview

置手动对焦位置，并触发对焦

public int setCameraFocusPositionInPreview(float positionX, float positionY, boolean var3)

参数	参数	说明
positionX	触摸点相对于视图的横坐标	
positionY	触摸点相对于视图的纵坐标	
var3	是否开启此功能	

返回值 0：方法调用成功 < 0：方法调用失败

10 setCameraExposurePosition

设置手动曝光位置

public int setCameraExposurePosition(float positionXinView, float positionYinView)

参数	参数	说明
positionXinView	触摸点相对于视图的横坐标	
positionYinView	触摸点相对于视图的纵坐标	

返回值 0：方法调用成功 < 0：方法调用失败

11 setCameraTorchOn

设置是否打开闪光灯。

public int setCameraTorchOn(boolean isOn)

参数	参数	说明
isOn	true：打开 false：关闭	

返回值 0：方法调用成功 < 0：方法调用失败。

12 setCameraAutoFocusFaceModeEnabled

设置是否开启人脸对焦功能

public int setCameraAutoFocusFaceModeEnabled(boolean enabled)

参数	参数	说明
enabled	true：打开 false：关闭	

返回值 false：（默认）关闭人脸对焦功能

其他SDK通用

1 getSdkVersion

获取SDK版本号。createKRtcEngine（）之后，无需等待结果即可调用

public String getSdkVersion()

返回值：SDK版本号

日志文件

1 setLogFilter

设置日志输出等级

注意：此接口必须在createKRtcEngine()方法之前调用

public static int setLogFilter(int filter)

参数	参数	说明
filter	日志输出等级。最小可关闭日志。	

返回值 0表示成功，不等于0表示失败。

2 setLogFileSize

设置 SDK 输出的日志文件大小

注意：此接口必须在createKRtcEngine()方法之前调用

public static int setLogFileSize(int fileSizeInKBytes)

参数	参数	说明
fileSizeInKBytes	单个日志文件的大小，单位为 KB。默认值为 1024 KB。如果你将 fileSizeInKByte 设为 1024 KB，SDK 会最多输出 5 MB 的日志文件。	

返回值 0表示成功，不等于0表示失败。

核心方法

1 joinChannel

加入频道

public int joinChannel(String token, String channelID, String userName, String userId)

参数	参数	说明
token	客户签名	
channelID	频道ID。当前频道ID，格式校验规则为：英文、数字及下划线。且开头不为下划线，长度不超过64个字符	
userName	用户名。不为空	
userId	用户ID。不为空	

返回值 0表示方法调用成功，不等于0表示失败。注意，此返回值仅代表接口调用是否成功，实际加入频道是否成功，仍需要通过IKRtcEngineEventHandler的onJoinChannelSuccess()回调来进行判断。

2 leaveChannel

离开频道

public int leaveChannel(String channelId)

参数	参数	说明
channelId	频道ID。当前频道ID，格式校验规则为：英文、数字及下划线。且开头不为下划线，长度不超过64个字符	

返回值 0表示成功，不等于0表示失败。注意，此返回值仅代表接口调用是否成功，实际离开频道是否成功，仍需要通过IKRtcEngineEventHandler的onLeaveChannel()回调来进行判断。

3 getConnectionState

获取当前频道连接状态

public int getConnectionState()

返回值 频道连接状态。1 - 网络连接断开。网络连接断开。2 - 建立网络连接中。3 - 网络已连接。4 - 重新建立网络连接中。5 - 网络连接失败

4 renewToken

更新 Token核心

public int renewToken(String token)

5 destroy

销毁SDK实例。

注意：此方法为同步方法，调用需一定耗时，建议在非UI线程调用。

public static void destroy()

6 switchChannel

快速切换直播频道。

public int switchChannel(String token, String channelId)

参数	参数	说明
token	客户签名	
channelId	频道号	

返回值 0表示成功，不等于0表示失败。

7 setClientRole

设置用户角色。

注意：SDK加入频道前调用，则为设置默认角色。SDK加入频道后调用，则为切换角色。默认为观众，即只拉流不能推流。

public int setClientRole(int role)

参数	参数	说明
role	角色	

返回值 0表示成功，不等于0表示失败。

8 createKRtcEngine

创建SDK单例并初始化。初始化完毕之后，会调用IKRtcEngineEventHandler的onSDKInitFinish(bool result)方法，返回SDK初始化结果。

public static KRtcEngine createKRtcEngine(Context context, String appId, IKRtcEngineEventHandler handler)

参数	参数	说明
context	上下文环境	
appId	用户ID	
handler	事件回调handler	

返回值 KRtcEngine单例

IKRtcEngineEventHandler

本地媒体-事件

1 onAudioPublishStateChanged

音频发布状态改变回调。

public void onAudioPublishStateChanged(String channel, int oldState, int newState)

参数	参数	说明
channel	频道ID	
oldState	旧状态	
newState	新状态	

2 onVideoPublishStateChanged

视频发布状态改变回调。

public void onVideoPublishStateChanged(String channel, int oldState, int newState)

参数	参数	说明
channel	频道ID	
oldState	旧状态	
newState	新状态	

3 onLocalAudioStateChanged

本地音频状态发生改变回调。

public void onLocalAudioStateChanged(int state, int error)

参数	参数	说明
state	当前的本地音频状态	
error	本地音频出错原因	

4 onLocalVideoStateChanged 本地视频状态发生改变回调。

public void onLocalVideoStateChanged(int localVideoState, int error)

参数	参数	说明
localVideoState	当前的本地视频状态	
error	本地视频出错原因	

远端媒体-事件

1 onRemoteVideoStateChanged

远端用户视频流状态已变化回调。

public void onRemoteVideoStateChanged(String uid, int state, int reason)

参数	参数	说明
uid	发生视频状态改变的远端用户 ID	
state	远端视频流状态	
reason	远端视频流状态改变的具体原因	

2 onRemoteAudioStateChanged

远端用户音频流状态已变化回调。

public void onRemoteAudioStateChanged(String uid, int state, int reason)

参数	参数	说明
uid	发生视频状态改变的远端用户 ID	
state	远端视频流状态	
reason	远端视频流状态改变的具体原因	

3 onAudioSubscribeStateChanged

音频订阅状态发生改变回调。

public void onAudioSubscribeStateChanged(String channel, String uid, int oldState, int newState)

参数	参数	说明
uid	用户ID	
oldState	旧状态	
newState	新状态	

4 onVideoSubscribeStateChanged

视频订阅状态发生改变回调。

public void onVideoSubscribeStateChanged(String channel, String uid, int oldState, int newState)

参数	参数	说明
uid	用户ID	
oldState	旧状态	
newState	新状态	

数据统计-事件

1 onRtcStats

当前通话统计回调。 该回调在通话中每两秒触发一次。 public void onRtcStats(IKRtcEngineEventHandler.RtcStats stats)

参数	参数	说明
stats	统计信息	

2 onRemoteAudioStats

通话中远端音频流的统计信息回调。该回调描述远端用户在通话中端到端的音频流统计信息，针对每个远端用户/主播每 2 秒触发一次。如果远端同时存在多个用户/主播，该回调每 2 秒会被触发多次。 public void onRemoteAudioStats(IKRtcEngineEventHandler.RemoteAudioStats stats)

参数	参数	说明
stats	统计信息	

3 onRemoteVideoStats

通话中远端视频流的统计信息回调。该回调描述远端用户在通话中端到端的音频流统计信息，针对每个远端用户/主播每 2 秒触发一次。如果远端同时存在多个用户/主播，该回调每 2 秒会被触发多次。 public void onRemoteVideoStats(IKRtcEngineEventHandler.RemoteVideoStats stats)

参数	参数	说明
stats	统计信息	

4 onLocalAudioStats

通话中本地音频流的统计信息回调。该回调描述本地设备发送音频流的统计信息，每 2 秒触发一次。 public void onLocalAudioStats(IKRtcEngineEventHandler.LocalAudioStats stats)

参数	参数	说明
stats	统计信息	

5 onLocalVideoStats

通话中本地视频流的统计信息回调。该回调描述本地设备发送视频流的统计信息，每 2 秒触发一次。 public void onLocalVideoStats(IKRtcEngineEventHandler.LocalVideoStats stats)

参数	参数	说明
stats	统计信息	

CDN推流

1 onRtmpStreamingStateChanged

回调当前推流状态

public void onRtmpStreamingStateChanged(String url, int state, int errCode)

参数	参数	说明
url	推流状态发生改变的 URL 地址	
state	详细的推流错误信息	

2 onTranscodingUpdated

旁路推流设置被更新回调。调用setLiveTranscoding方法更新转码参数 LiveTranscoding 时，SDK 会触发 onTranscodingUpdated 回调。 首次调用该方法设置转码参数时，不会触发 onTranscodingUpdated 回调。

public void onTranscodingUpdated()

音频管理

1、onAudioVolumeIndication

【获取用户音量（回调）】回调报告频道内瞬时音量最高的几个用户（即说话者）的用户 ID 及他们的音量

注意：*该回调默认禁用。可以通过 enableAudioVolumeIndication 方法开启。开启后，只要频道内有发流用户，SDK 会在加入频道后按 enableAudioVolumeIndication 中设置的时间间隔触发 onAudioVolumeIndication 回调。* 每次会触发两个 onAudioVolumeIndication 回调，一个报告本地发流用户的音量相关信息，另一个报告瞬时音量最高的远端用户（最多 3 位）的音量相关信息。 *本地用户静音后，SDK 立即停止报告本地用户的音量提示回调。* 瞬时音量最高的远端用户静音后 20 秒，远端的音量提示回调中将不再包含该用户；如果远端所有用户都将自己静音，20 秒后 SDK 停止报告远端用户的音量提示回调。

public void onAudioVolumeIndication(IKRtcEngineEventHandler.AudioVolumeInfo[] speakers, int totalVolume)

参数		参数	说明
speakers	用户音量信息，详见 AudioVolumeInfo 数组。如果 speakers 为空，则表示远端用户不发流或没有远端用户。		
totalVolume	混音后的总音量，取值范围为 [0,255]		

2、onActiveSpeaker

回调报告特定时间段内累积音量最高的用户 ID

public void onActiveSpeaker(String uid)

参数	参数	说明
uid	远端最活跃用户的ID	

其他SDK通用

1 onWarning

SDK 发生警告回调

public void onWarning(int warn)

参数	参数	说明
warn	警告码	

概览

KRTC通过全球部署的虚拟网络，提供可以灵活搭配的 API 组合，为移动端到移动端以及移动端到 Web 端提供质量可靠的实时音视频通信。

核心方法

序号	方法	描述
1	joinChannel	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，可以群聊核心
2	leaveChannel	离开频道核心
3	getConnectionState	获取当前频道连接状态核心
4	renewToken	更新 Token 核心
5	destroy	销毁 KRtcEngine 对象核心
6	switchChannel	快速切换直播频道核心
7	setClientRole	设置用户角色核心
10	createKRtcEngine	创建 KRtcEngine 对象（单例）并返回实例

核心事件

序号	事件	描述
1	onJoinChannelSuccess	表示该客户端成功加入了指定的频道核心
2	onRejoinChannelSuccess	重新加入频道回调核心有时候由于网络原因，客户端可能会和服务器失去连接，SDK 会进行自动重连，自动重连成功后触发此回调方法核心
3	onConnectionLost	网络连接丢失回调
4	onLeaveChannel	离开频道回调
5	onUserOffline	远端用户（通信场景）/主播（直播场景）离开当前频道回调核心提示有远端用户/主播离开了频道（或掉线）核心
6	onConnectionStateChanged	网络连接状态已改变回调核心该回调在网络连接状态发生改变的时候触发，并告知用户当前的网络连接状态和引起网络状态改变的原因核心
7	onUserJoined	远端用户（通信场景）/主播（直播场景）加入当前频道回调核心
8	onTokenPrivilegeWillExpire	Token 服务即将过期回调核心
9	onRequestToken	Token 已过期回调核心
10	onClientRoleChanged	直播场景下用户角色已切换回调核心
11	onNetworkTypeChanged	本地网络类型发生改变回调核心

音频管理

序号	**方法**	**描述**
1	setAudioProfile	根据场景需求配置适合的音质属性，profile 重点是 48kHZ 的采样率，最好支持双声道，参考声网的 AUDIO_SCENARIO_GAME_STREAMING，这是娱乐使用重点
2	adjustRecordingSignalVolume	【设置采集音量】调节录音信号音量，需要对应方式获取目前音量便于管理，下同
3	adjustAudioMixingPlayoutVolume	调节音乐文件在本地的播放音量
4	adjustPlaybackSignalVolume	【设置播放音量】调节所有远端用户的音量
5	adjustUserPlaybackSignalVolume	【设置播放音量】调节指定远端用户的音量
6	enableLocalAudio	开关本地音频采集核心
7	muteLocalAudioStream	开关本地音频发送核心
8	muteRemoteAudioStream	接收 / 停止接收指定音频流核心
9	muteAllRemoteAudioStreams	接收 / 停止接收所有音频流核心
10	setDefaultMuteAllRemoteAudioStreams	设置是否默认接收所有音频流核心
13	enableAudio	启用音频模块
序号	**事件**	**描述**
1	onAudioVolumeIndication	【获取用户音量（回调）】回调报告频道内瞬时音量最高的几个用户（即说话者）的用户 ID 及他们的音量
2	onActiveSpeaker	回调报告特定时间段内累积音量最高的用户 ID

视频管理

序号	方法	描述
1	setupLocalVideo	初始化本地视图：设置本地视频镜像，使用前置摄像头，SDK 默认启用镜像模式；如果你使用后置摄像头，SDK 默认关闭镜像模式
2	setupRemoteVideo	初始化远端用户视图核心
3	setVideoEncoderConfiguration	设置视频编码属性核心每个属性对应一套视频参数，如分辨率、帧率、码率、视频方向等核心
4	enableLocalVideo	开关本地视频采集核心
5	startPreview	开启视频预览核心
6	stopPreview	停止视频预览核心
7	muteLocalVideoStream	开关本地视频发送核心
8	muteRemoteVideoStream	接收 / 停止接收指定远端用户的视频流核心
9	muteAllRemoteVideoStreams	接收 / 停止接收所有远端视频流核心
10	setLocalVideoMirrorMode	设置本地视频镜像(2020.3.1，xuxiaoguang注,声网已废弃该接口，调用setupLocalVideo(开始时设置) 或setLocalRenderMode(过程中设置))
11	setLocalRenderMode	更新本地视图显示模式核心
12	setRemoteRenderMode	更新远端视图显示模式核心
13	setDefaultMuteAllRemoteVideoStreams	设置是否默认停止接收视频流核心
14	setLocalRenderMode	更新本地视图显示模式核心
15	setRemoteRenderMode	更新远端视图显示模式核心
16	setDefaultMuteAllRemoteVideoStreams	设置是否默认停止接收视频流核心
17	enableVideo	开关本地音频采集

本地媒体

序号	事件	描述
1	onAudioPublishStateChanged	音频发布状态改变回调核心
2	onVideoPublishStateChanged	视频发布状态改变回调核心
3	onLocalAudioStateChanged	报告当前本地音频状态核心当本地音频出现故障时，你可以通过错误码排查问题核心
4	onLocalVideoStateChanged	报告当前本地视频状态核心当本地视频出现故障时，你可以通过错误码排查问题核心

远端媒体

序号	事件	描述
1	onRemoteVideoStateChanged	报告远端视频流当前状态和状态改变的原因核心
2	onRemoteAudioStateChanged	远端用户音频流状态已变化回调核心
3	onAudioSubscribeStateChanged	音频订阅状态发生改变回调核心
4	onVideoSubscribeStateChanged	视频订阅状态发生改变回调核心

数据统计

序号	事件	描述
1	onRtcStats	报告本地通话统计信息核心你可以了解到通话时长、当前通话频道中的人数、当前系统的 CPU 使用率、当前 App 的 CPU 使用率和以下重要参数核心
2	onRemoteAudioStats	报告当前通话中每个远端用户/主播音频流的统计信息核心你可以了解到每个远端用户/主播发送的音频流质量、声道数（单声道或双声道）和其他重要参数信息核心
3	onRemoteVideoStats	通话中远端视频流统计信息回调
4	onLocalAudioStats	通话中本地音频流统计信息回调
5	onLocalVideoStats	报告本地设备发送视频流的统计信息核心你可以了解视频编码宽/高和以下重要参数信息核心

音乐文件播放及混音

序号	方法	描述
1	adjustAudioMixingPublishVolume	调节音乐文件在远端的播放音量
2	startAudioMixing	开始播放音乐文件及混音，同时让频道内的其他人听到此音乐
3	stopAudioMixing	停止播放音乐文件及混音核心
4	pauseAudioMixing	暂停播放音乐文件
5	resumeAudioMixing	恢复播放音乐文件
6	getAudioMixingPlayoutVolume	获取音乐文件的本地播放音量
7	getAudioMixingPublishVolume	获取音乐文件的远端播放音量
8	getAudioMixingDuration	获取音乐文件播放时长
9	getAudioMixingCurrentPosition	获取音乐文件的播放进度
10	setAudioMixingPosition	设置音乐文件的播放位置
11	setAudioMixingPitch	调整本地播放的音乐文件的音调

CDN 推流

序号	方法	描述
1	setLiveTranscoding	设置音视频流的直播参数，如分辨率、码率、帧率、水印和背景色位置核心如果是多主播转码合图，需设置每个主播的参数，再次调用可以更新参数
2	addPublishStreamUrl	主播设置 CDN 推流地址
3	removePublishStreamUrl	删除推流地址
序号	事件	描述
1	onRtmpStreamingStateChanged	回调当前推流状态
2	onTranscodingUpdated	旁路推流设置被更新回调

语音播放路由

序号	方法	描述
1	setDefaultAudioRoutetoSpeakerphone	设置默认的音频播放路由
2	setEnableSpeakerphone	启用/关闭扬声器播放
3	isSpeakerphoneEnabled	查询扬声器启用状态
序号	事件	描述
1	onAudioRouteChanged	语音路由已改变回调
2	onAudioRouteChanged	语音路由已改变回调

通话前网络测试

序号	事件	描述
1	onLastmileQuality	通话前网络上下行 last mile 质量报告回调，该回调描述本地用户在加入频道前的 last mile 网络探测的结果
2	onLastmileProbeResult	通话前网络质量探测报告回调，通话前网络上下行 Last mile 质量探测报告回调核心

自定义视频模块

序号	方法	描述
1	setLocalVideoRenderer	自定义本地视频渲染器
2	setRemoteVideoRenderer	自定义远端视频渲染器

视频自采集（仅适用于 Push 模式）

序号	方法	描述
1	pushExternalVideoFrame	完成视频数据处理后，再通过 pushExternalVideoFrame 发送给 SDK 进行后续操作

音频自采集（仅适用于 Push 模式）

序号	方法	描述
1	setExternalAudioSource	自定义音频采集，指定外部音频采集设备
2	pushExternalAudioFrame	音频数据处理后发送给 SDK

原始音频数据

序号	方法	描述
1	registerAudioFrameObserver	注册语音观测器对象
2	setRecordingAudioFrameParameters	设置采集的音频格式核心
3	setPlaybackAudioFrameParameters	设置播放的声音格式核心

原始视频数据

序号	方法	描述
1	registerVideoFrameObserver	加入频道前调 用 registerVideoFrameObserver 方法注册视频观测器，并在该方法中实现一 个 IVideoFrameObserver 类核心
序号	事件	描述
1	onCaptureVideoFrame	获取本地采集的视频数据
2	onPreEncodeVideoFrame	获取本地编码前的视频数据
3	onRenderVideoFrame	获取接收到的远端视频数据

媒体附属信息

序号	方法	描述
1	registerMediaMetadataObserver	直播的视频帧中同步添加 Metadata，注册媒体 Metadata 观测器

直播水印

序号	方法	描述
1	addVideoWatermark	添加本地视频水印
2	clearVideoWatermarks	删除已添加的视频水印

摄像头控制

序号	方法	描述
1	switchCamera	切换前置/后置摄像头
2	isCameraZoomSupported	检测设备是否支持摄像头缩放功能
3	isCameraTorchSupported	检测设备是否支持闪光灯常开
4	isCameraFocusSupported	检测设备是否支持手动对焦功能
5	isCameraExposurePositionSupported	检测设备是否支持手动曝光功能
6	isCameraAutoFocusFaceModeSupported	检测设备是否支持人脸对焦功能
7	setCameraZoomFactor	设置摄像头缩放比例
8	getCameraMaxZoomFactor	获取摄像头支持最大缩放比例
9	setCameraFocusPositionInPreview	设置手动对焦位置，并触发对焦
10	setCameraExposurePosition	设置手动曝光位置
11	setCameraTorchOn	设置是否打开闪光灯
12	setCameraAutoFocusFaceModeEnabled	设置是否开启人脸对焦功能

其他 SDK 通用

序号	方法	描述
1	getSdkVersion	查询 SDK 版本号
2	getErrorDescription	获取 SDK 警告或错误描述
序号	事件	描述
1	onWarning	SDK 发生警告回调

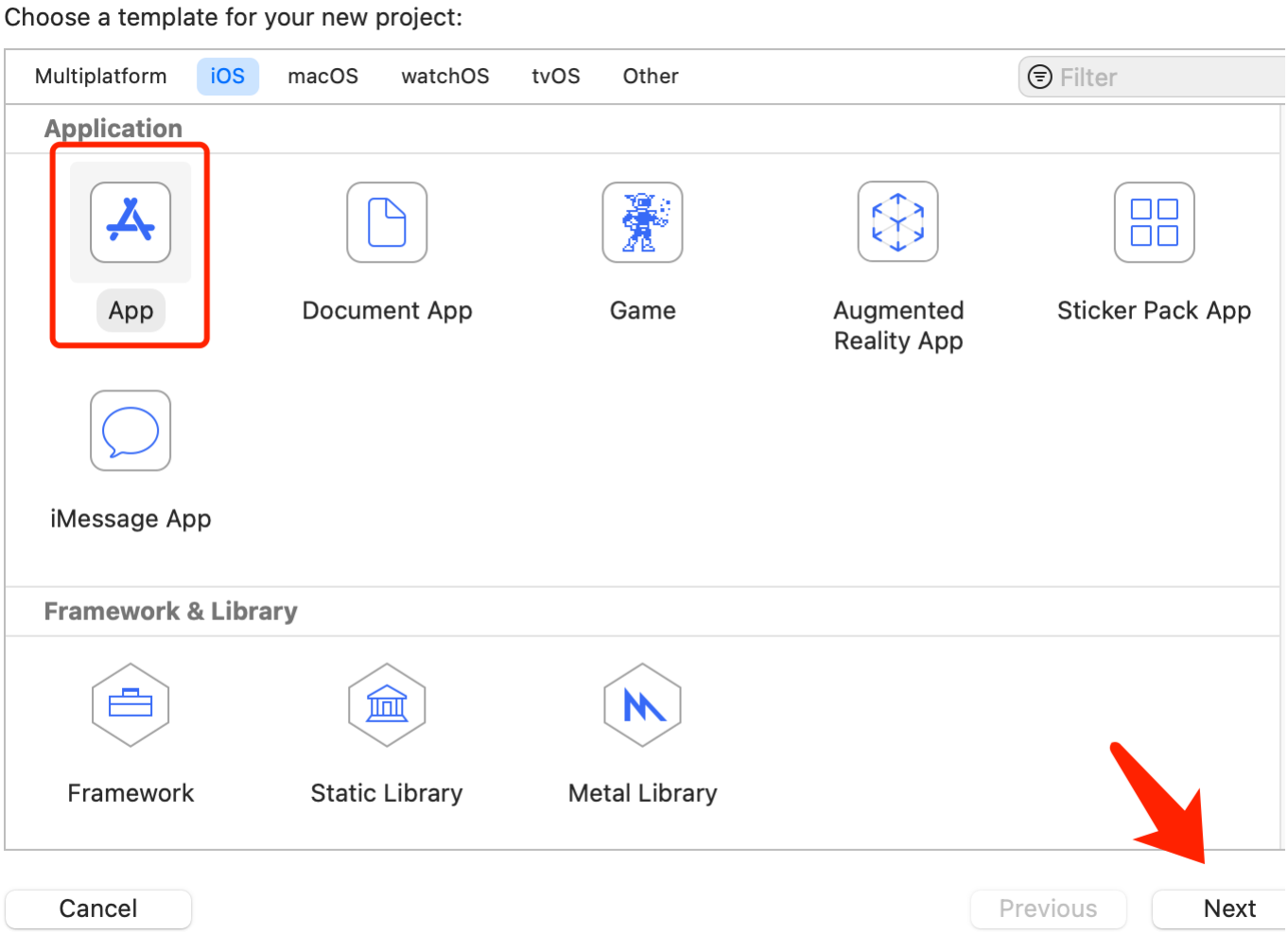
快速开始

一、 前提条件

1. Xcode 10.0或以上版本
2. iOS 9.0或以上版本的设备

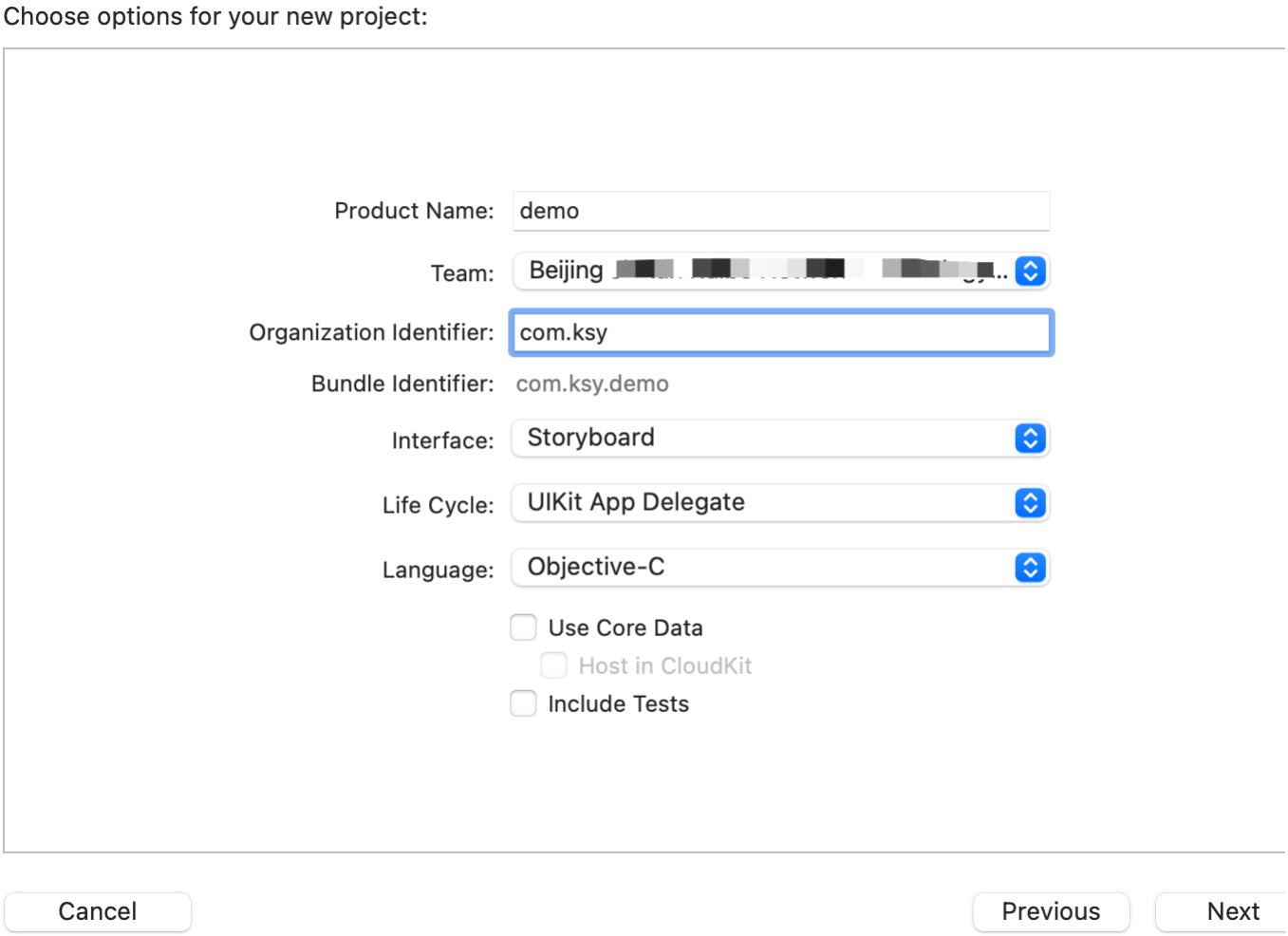
二、创建 iOS 项目

3. 打开 Xcode 并点击 Create a new Xcode project。
4. 选择平台类型为 iOS、项目类型为 Single View App，并点击 Next。

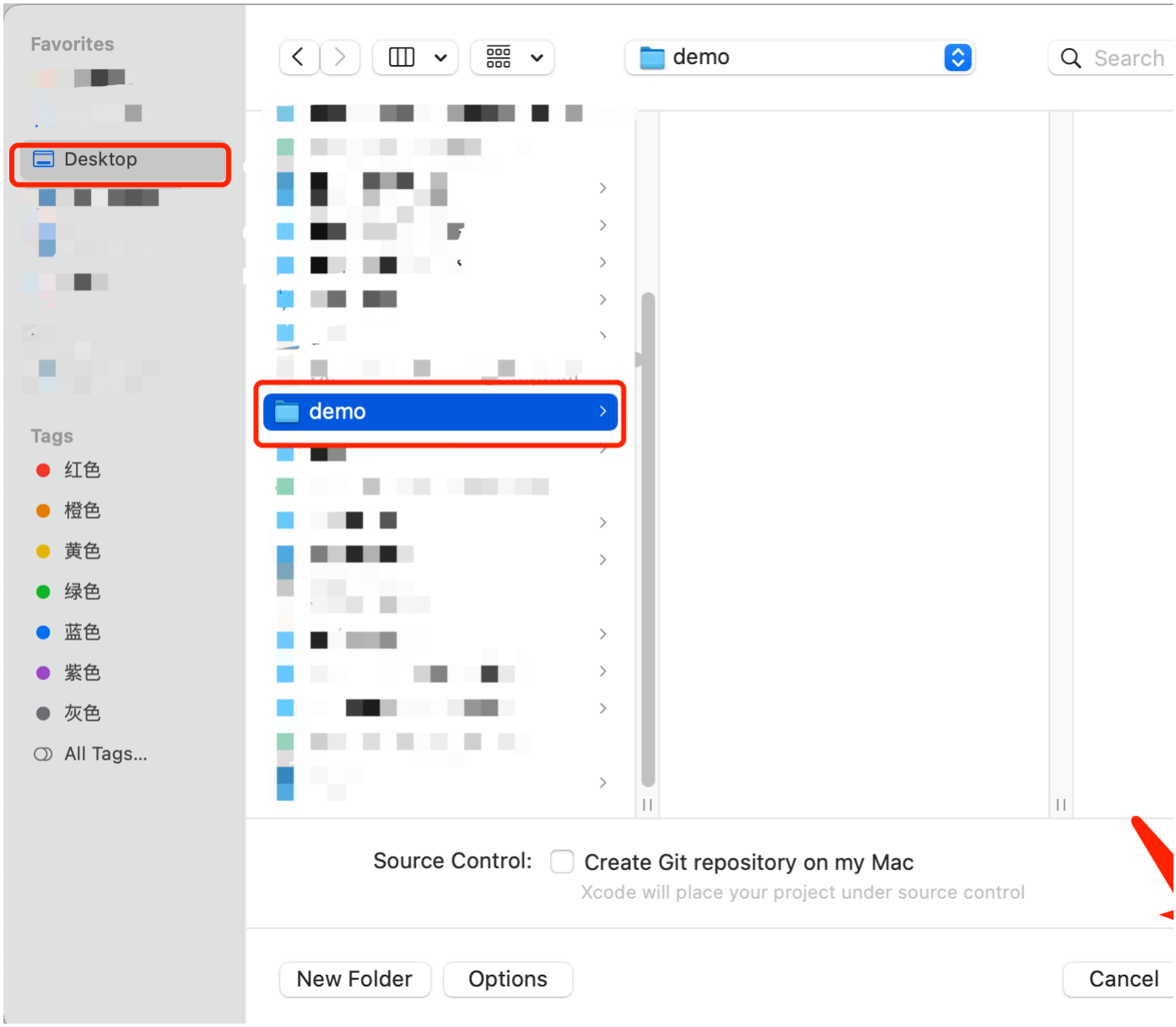


5. 输入项目名称（Product Name）、开发团队信息（Team）、组织名称（Organization Name）和语言（Language）等项目信息，并点击 Next。（注：如果你没有添加过开发团队信息，会看到 Add

account… 按钮。点击该按钮并按照屏幕提示登入 Apple ID，完成后即可选择你的 Apple 账户作为开发团队。）



6. 选择项目存储路径，并点击 Create。



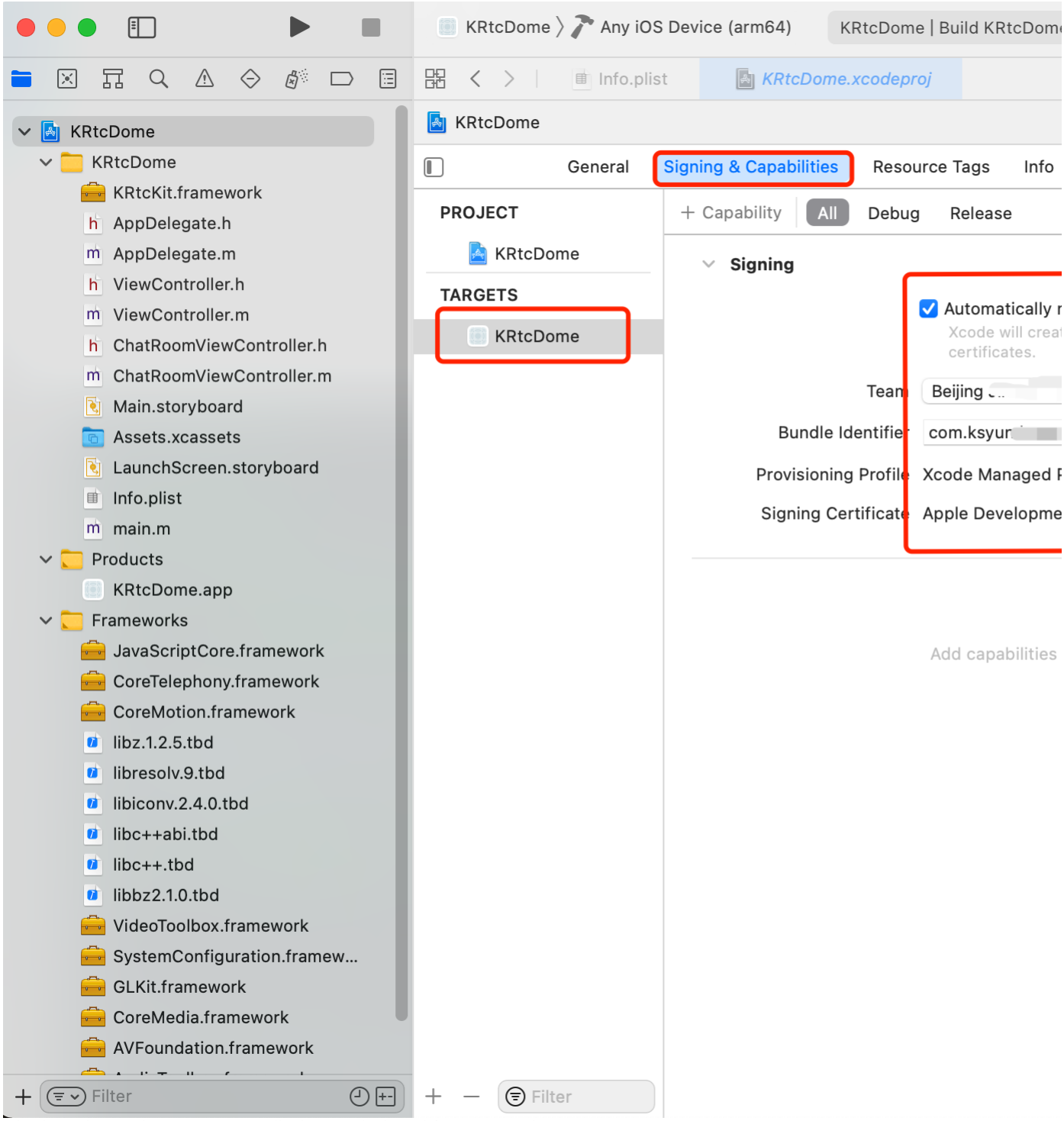
三、获取 SDK

1. 从官网获取 SDK

四、运行项目

在运行项目前，你需要设置签名和开发团队，并添加设备权限。

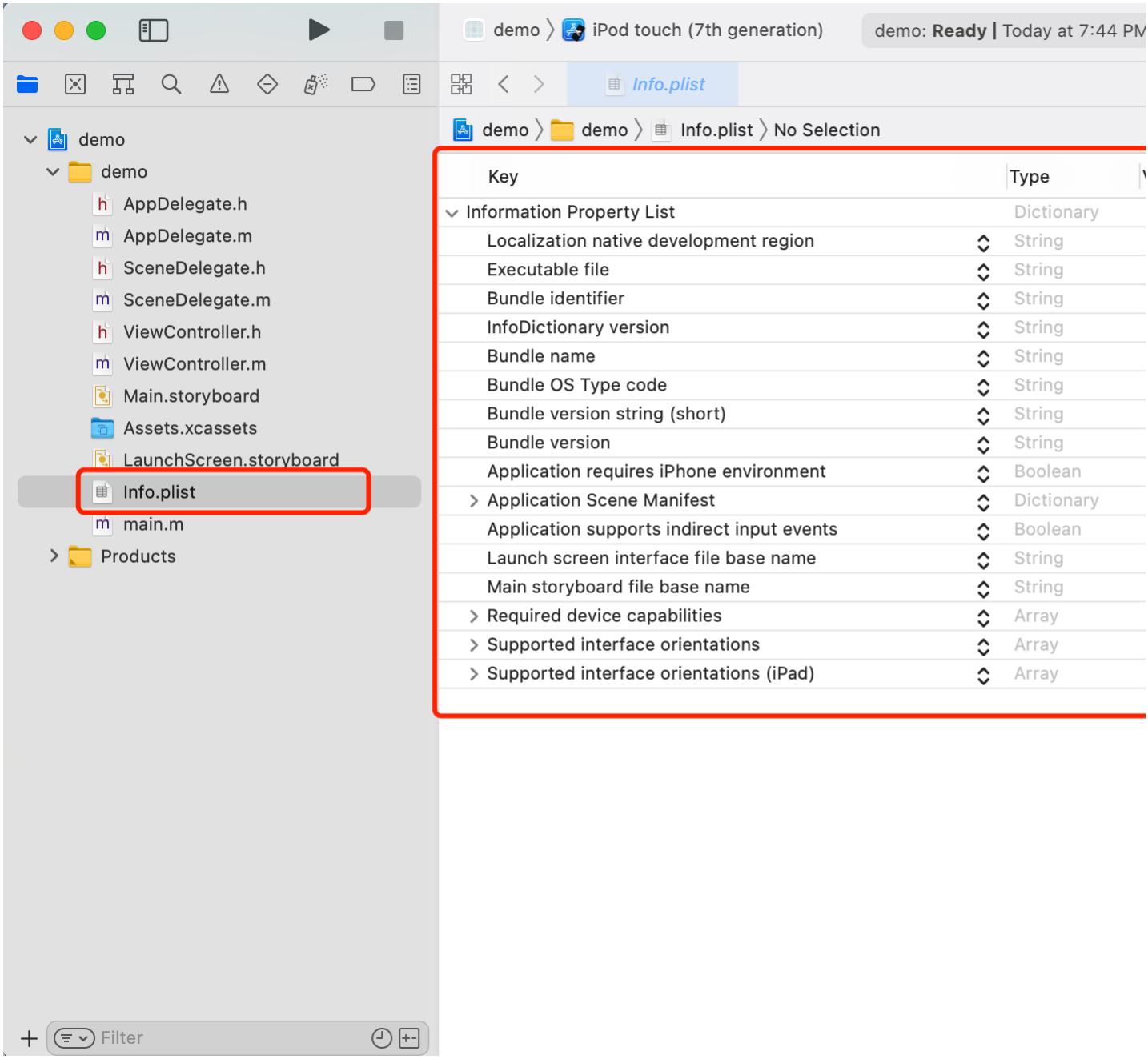
2. 设置签名和开发团队 （1）在 Xcode 中，进入 TARGETS > Project Name > Signing & Capabilities > Signing 菜单，勾选 Automatically manage signing。 （2）仔细阅读弹窗提示，并点击 Enable Automatic。 （3）成功设置签名后，在 Team 处选择你的开发团队。



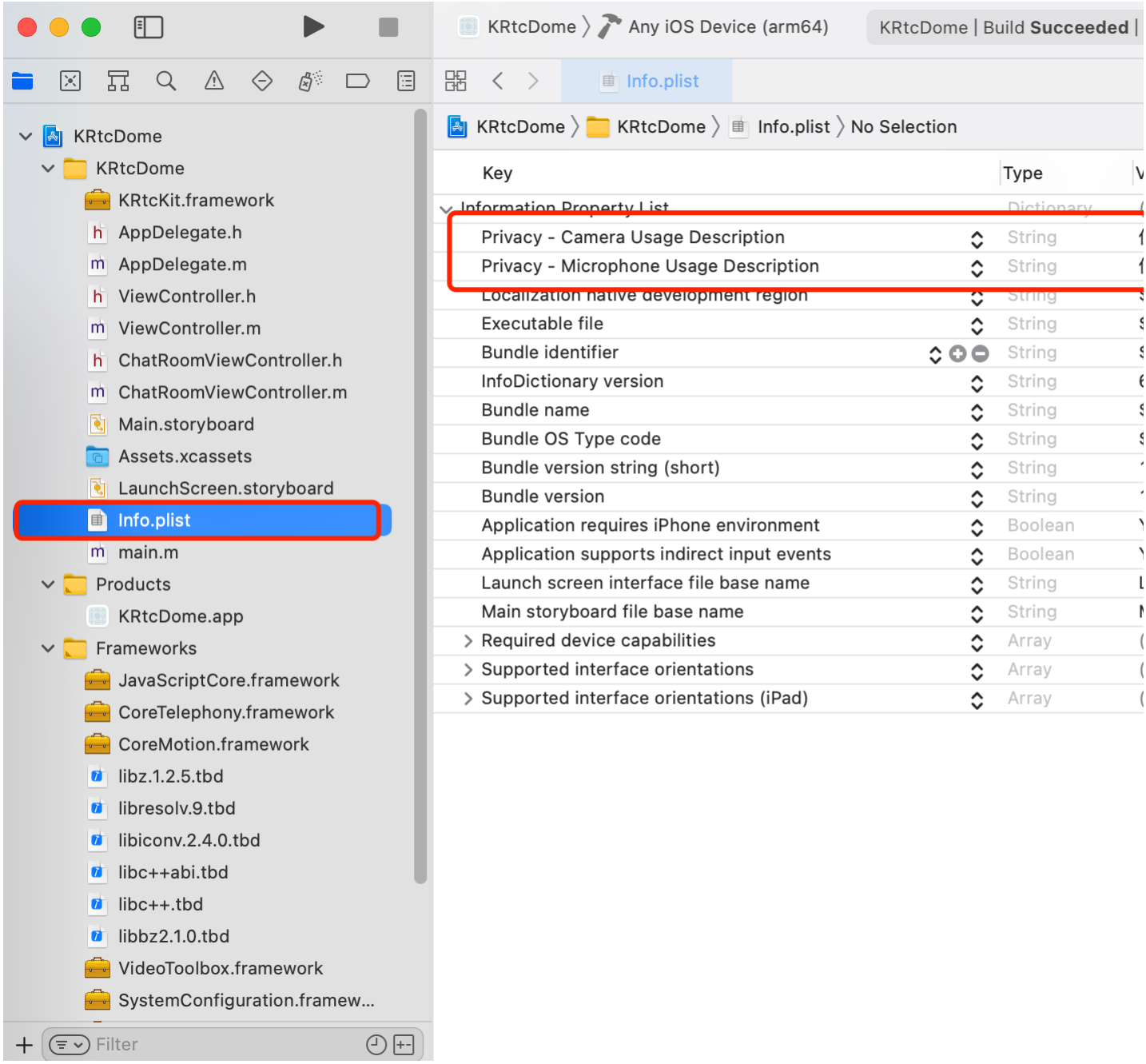
3. 添加设备权限 在 Xcode 中，打开 info.plist 文件。在右侧列表中添加如下内容，获取相应的设备权限：

Key	Type	Value
Privacy - Microphone Usage Description	String	"使用麦克风的目的"
Privacy - Camera Usage Description	String	"使用摄像头的目的"

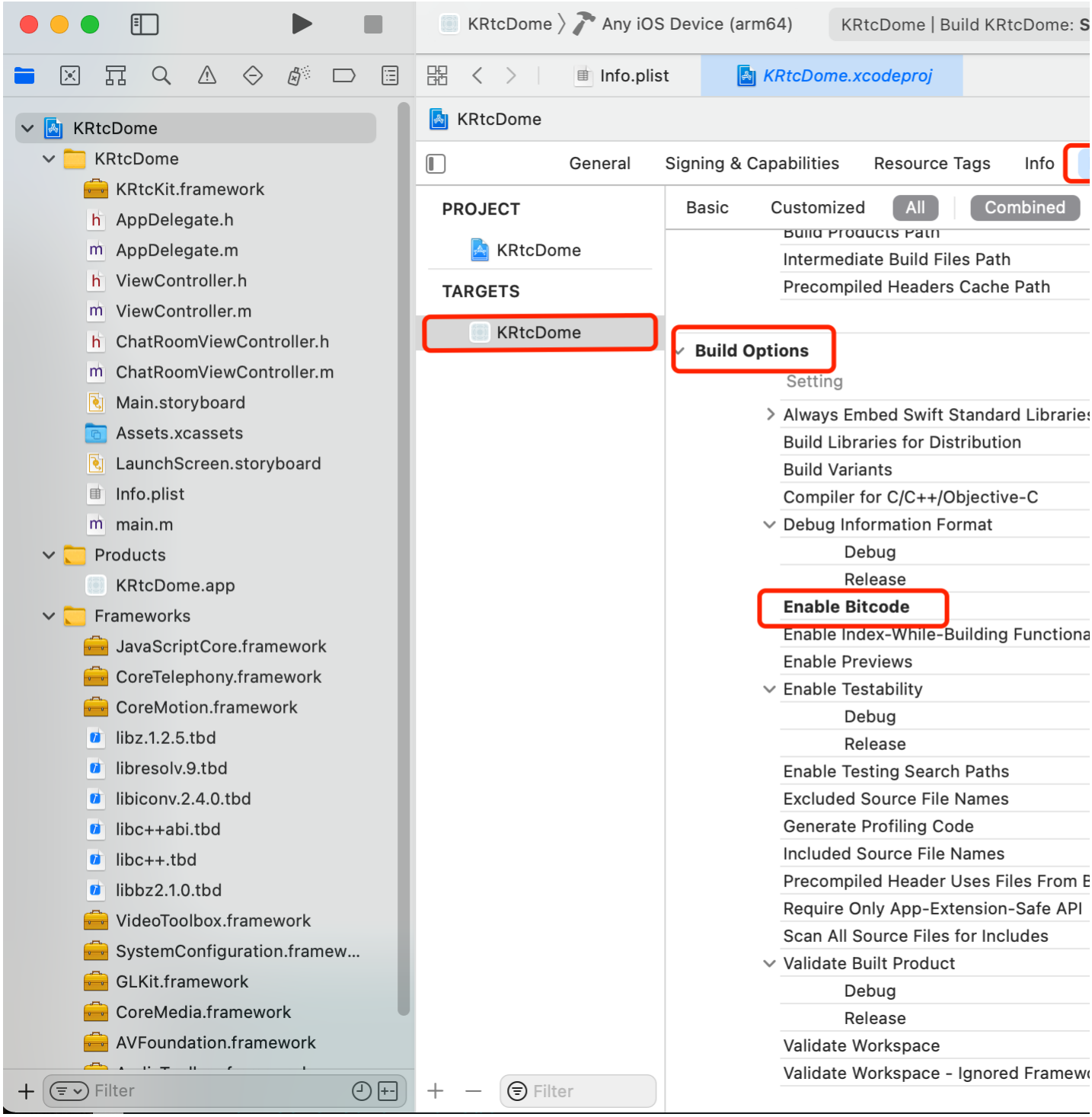
添加前


demo 〉 iPod touch (7th generation)
demo: Ready | Today at 7:44 PM
Info.plist
demo
demo
AppDelegate.h
AppDelegate.m
SceneDelegate.h
SceneDelegate.m
ViewController.h
ViewController.m
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
main.m
Products
Filter
demo 〉 demo 〉 Info.plist 〉 No Selection
Key
Type
Information Property List
Dictionary
Localization native development region
String
Executable file
String
Bundle identifier
String
InfoDictionary version
String
Bundle name
String
Bundle OS Type code
String
Bundle version string (short)
String
Bundle version
String
Application requires iPhone environment
Boolean
Application Scene Manifest
Dictionary
Application supports indirect input events
Boolean
Launch screen interface file base name
String
Main storyboard file base name
String
Required device capabilities
Array
Supported interface orientations
Array
Supported interface orientations (iPad)
Array

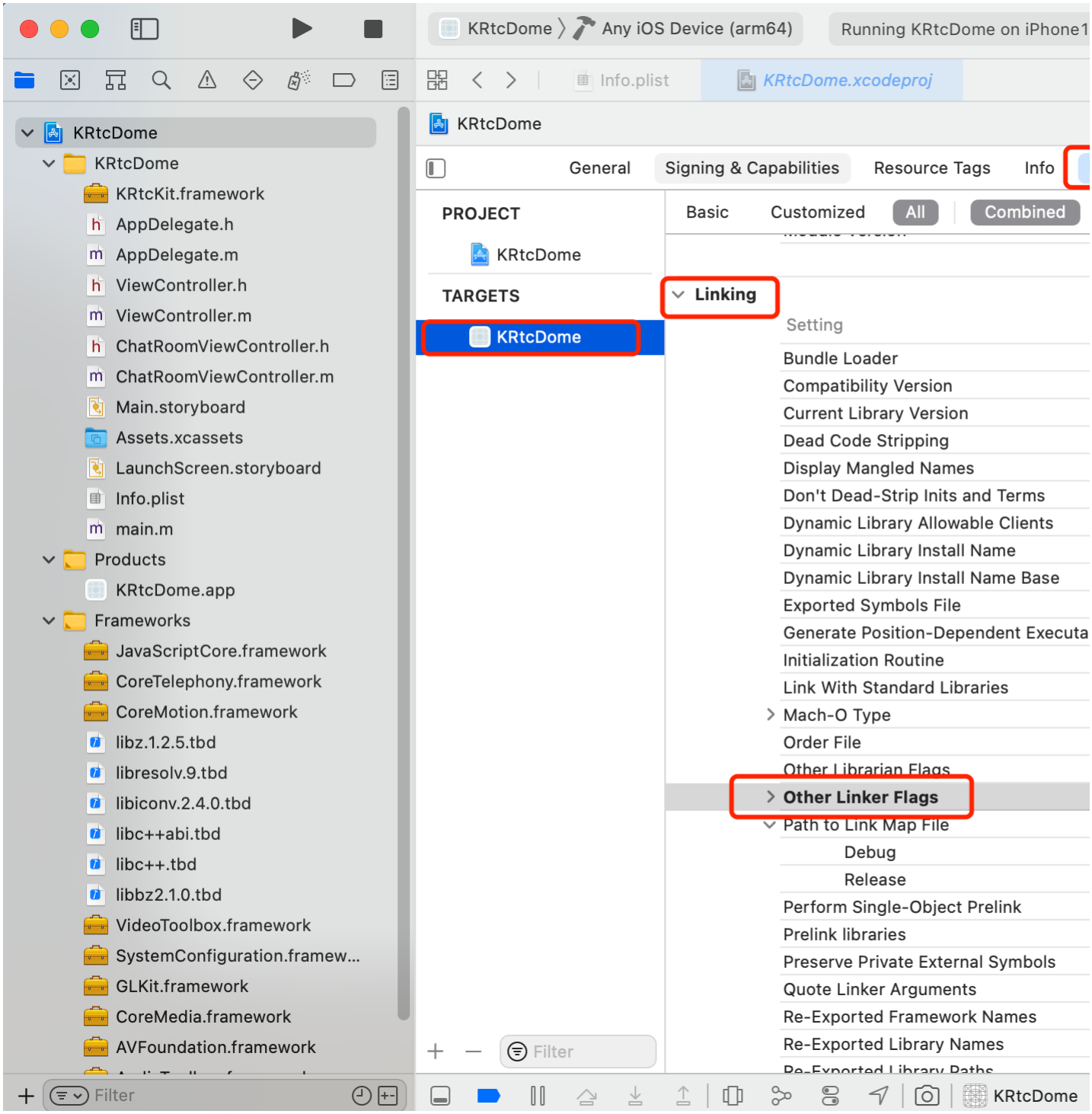
添加后



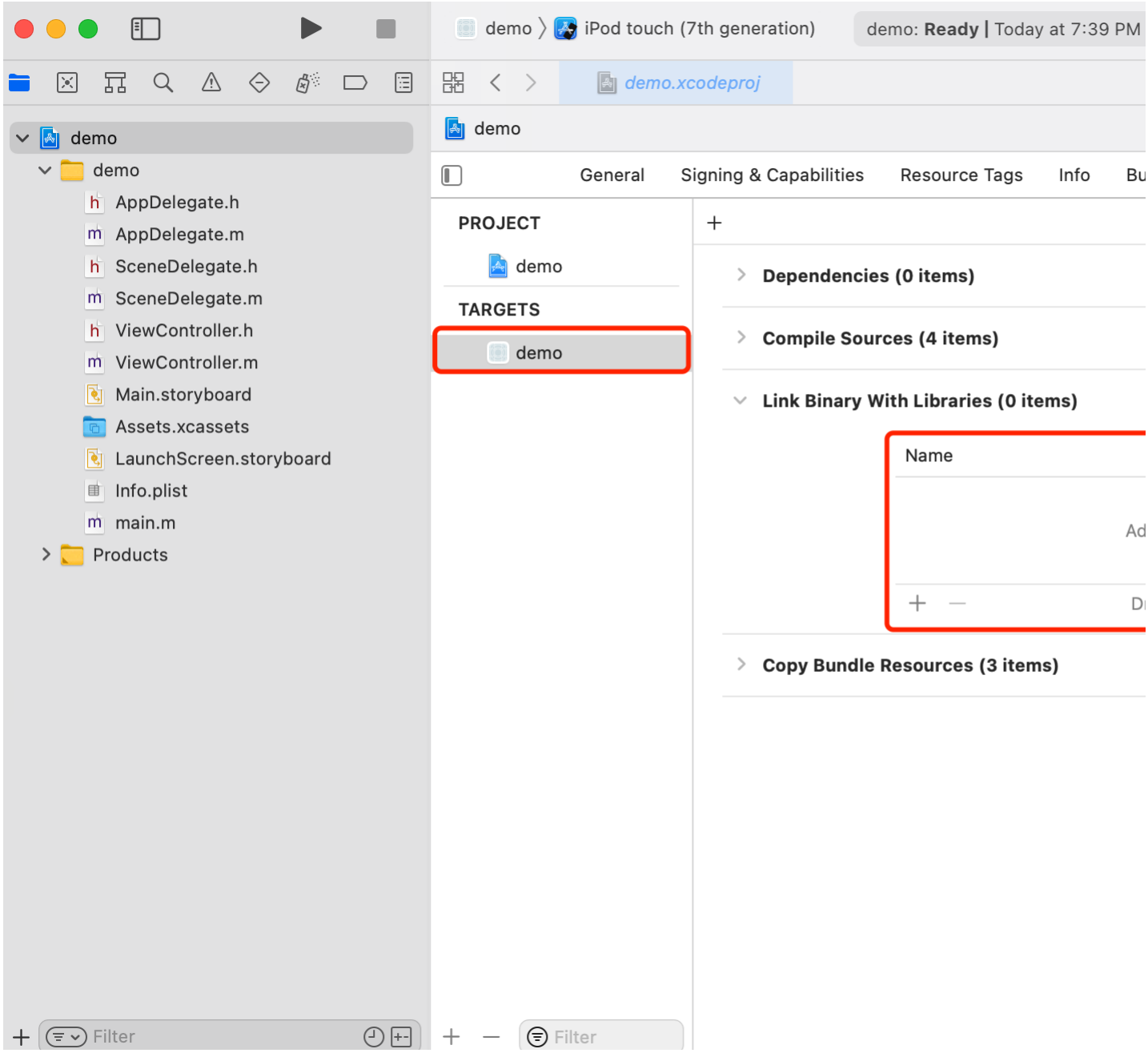
3.工程设置 （1）打开Xcode，进入TAGETS > Project Name > Build Settings > Build Options > Enable Bitcode > NO



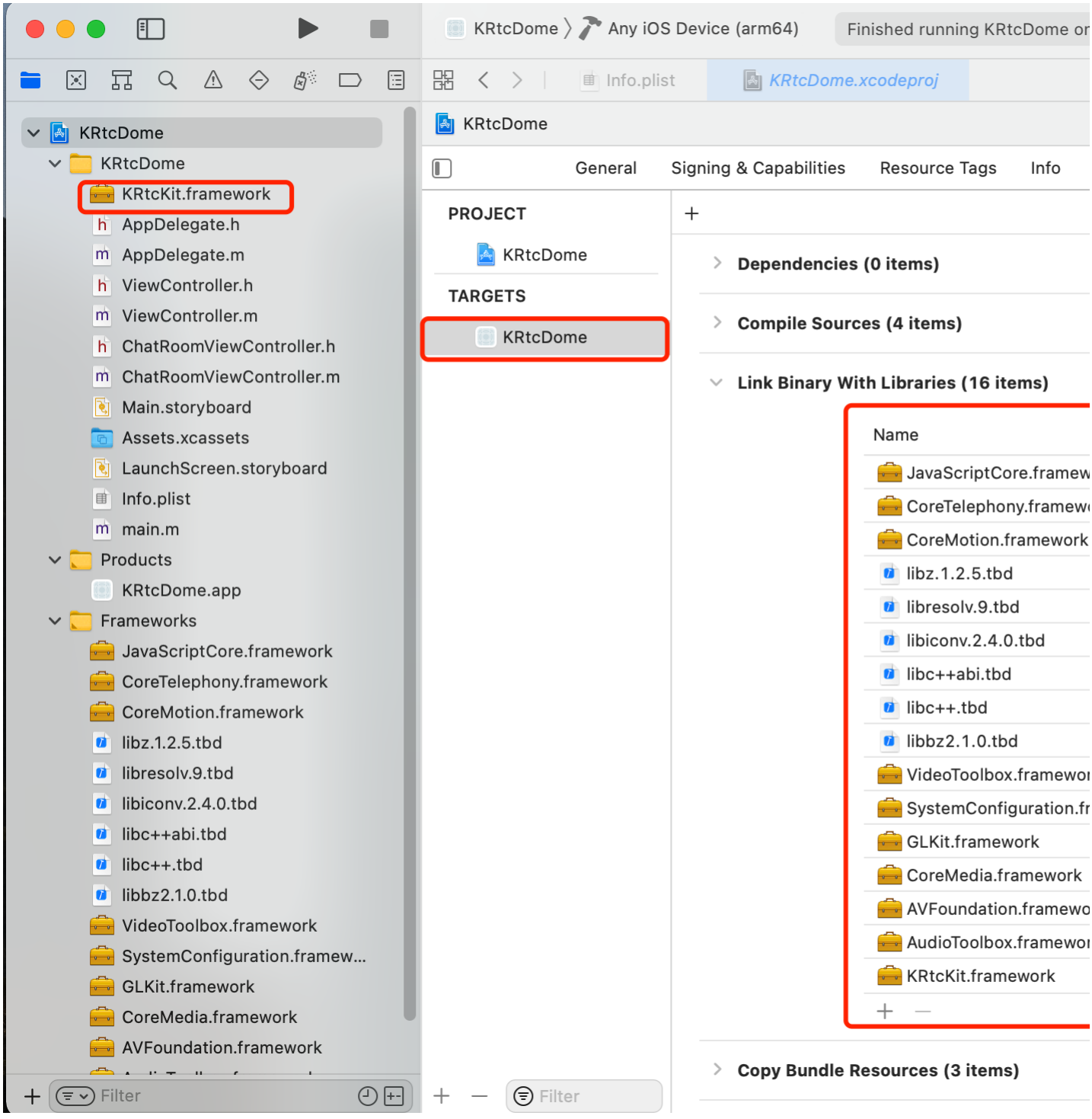
(2) 打开Xcode，进入TAGETS > Project Name > Build Settings > Linking > Other Linker Flags > -ObjC



4.添加库 （1）将 KRtcKit.framework 静态库复制到 ./project_name 文件夹下（./project_name 为你的项目名称）。 （2）打开 Xcode，进入 TARGETS > Project Name > Build Phases > Link Binary with Libraries 菜单，点击 + 添加如下库。在添加 KRtcKit.framework 时，还需在点击 + 后点击 Add Other… > Add Files，找到本地文件并打开。 （3）系统库 a. libz.1.2.5.tbd b. libresolv.9.tbd c. libiconv.2.4.0.tbd d. libc++abi.tbd e. libc++.tbd f. libbz2.1.0.tbd g. VideoToolbox.framework h. SystemConfiguration.framework i. GLKit.framework j. CoreMedia.framework k. CoreMotion.framework l. CoreTelephony.framework m. AVFonundation.framework n. AudioToolbox.framework o. JavaScriptCore.framework 添加前


demo 〉 iPod touch (7th generation)
demo: Ready | Today at 7:39 PM
demo.xcodeproj
demo
demo
AppDelegate.h
AppDelegate.m
SceneDelegate.h
SceneDelegate.m
ViewController.h
ViewController.m
Main.storyboard
Assets.xcassets
LaunchScreen.storyboard
Info.plist
main.m
Products
Filter
General
Signing & Capabilities
Resource Tags
Info
PROJECT
demo
TARGETS
demo
Dependencies (0 items)
Compile Sources (4 items)
Link Binary With Libraries (0 items)
Name
Copy Bundle Resources (3 items)
Filter

添加后



五、基本流程

1. 导入 #import <KRtcKit/KRtcKit.h>
2. 添加代理方法
3. 初始化SDK sharedEngineWithAppId
4. 设置相关配置后加入频道 joinChannelByToken
5. 离开频道leaveChannel
6. 销毁SDK[KRtcEngineKit destroy]; 详见示例工程。

KRtcEngine

音乐文件播放及混音-方法

1 adjustAudioMixingPublishVolume

调节音乐文件在远端的播放音量。 public int adjustAudioMixingPublishVolume(int volume)

参数	参数	说明
volume	范围为[0,400]，默认值为 100，表示原始音量。最大支持4倍音量。不等于0设为0，大于400则设为400	content3

返回值 0表示成功，不等于0表示失败。

2 startAudioMixing

开始播放音乐文件及混音。 public int startAudioMixing(String filePath, boolean loopback, boolean replace, int cycle)

参数	参数	说明
filePath	指定需要混音的本地或在线音频文件的绝对路径（包含文件名后缀），如 /sdcard/emulated/0/audio.mp4。支持的音频格式包括：mp3、mp4、m4a、aac、3gp、mkv 及 wav。如果用户提供的目录以 /assets/ 开头，则去 assets 里面查找该文件。如果用户提供的目录不是以 /assets/ 开头，一律认为是在绝对路径里查找该文件。	
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：只推动设置的本地音频文件或者线上音频文件，不传输麦克风收录的音频;false：音频文件内容将会和麦克风采集的音频流进行混音	
cycle	音频文件循环播放的次数：正整数：循环的次数。-1：无限循环。	

返回值 0表示成功，不等于0表示失败。

3 stopAudioMixing

停止播放音乐文件及混音。 public int stopAudioMixing()

返回值 0表示成功，不等于0表示失败。

4 pauseAudioMixing

暂停播放音乐文件及混音。该方法暂停播放伴奏。请在频道内调用该方法。 public int pauseAudioMixing() 返回值。0表示成功，不等于0表示失败。

5 resumeAudioMixing

恢复播放音乐文件及混音。该方法恢复混音，继续播放伴奏。请在频道内调用该方法。 public int resumeAudioMixing()

返回值 0表示成功，不等于0表示失败。

6 getAudioMixingPlayoutVolume

获取音乐文件的本地播放音量。 public int getAudioMixingPlayoutVolume()

返回值 方法调用成功则返回音量值，范围为 [0, 400]。默认值为 100，表示原始音量。 不等于0：方法调用失败。

7 getAudioMixingPublishVolume

获取音乐文件的远端播放音量。 public int getAudioMixingPublishVolume()

返回值 方法调用成功则返回音量值，范围为 [0, 400]。默认值为 100，表示原始音量。 不等于0：方法调用失败。

8 getAudioMixingDuration

获取音乐文件的时长。 该方法获取音乐文件时长，单位为毫秒。请在频道内调用该方法。 public int getAudioMixingDuration()

返回值 方法调用成功，并返回伴奏时长。不等于0：方法调用失败。

9 getAudioMixingCurrentPosition

获取音乐文件的播放进度。该方法获取当前伴奏播放进度，单位为毫秒。请在频道内调用该方法。 public int getAudioMixingCurrentPosition()

返回值 方法调用成功并返回伴奏播放进度。不等于0：方法调用失败。

10 setAudioMixingPosition

设置音乐文件的播放位置。该方法可以设置音频文件的播放位置，这样你可以根据实际情况播放文件，而不是非得从头到尾播放一个文件。 public int setAudioMixingPosition(int pos)

参数	参数	说明
pos	进度条位置，单位为毫秒。	

返回值 0表示成功，不等于0表示失败。

11 setAudioMixingPitch

调整本地播放的音乐文件的音调。该方法需在 startAudioMixing 后调用。 public int setAudioMixingPitch(int pitch)

参数	参数	说明
pitch	按半音音阶调整本地播放的音乐文件的音调，默认值为 1.0，即不调整音调。取值范围为 [-12,12]，每相邻两个值的音高距离相差半音。取值的绝对值越大，音调升高或降低得越多。	

返回值 0表示成功，不等于0表示失败。

自定义视频模块

1 setLocalVideoRenderer

推送外部视频帧。该方法同setupLocaleVideo()方法，只能同时使用一个，且使用时机相同。 public int setLocalVideoRenderer(IKVideoSink render)

参数	参数	说明
render	基于模板方法的自定义渲染器	

返回值 0表示成功，不等于0表示失败。

2 setRemoteVideoRenderer

推送外部视频帧。该方法同setupRemoteVideo()方法，只能同时使用一个，且使用时机相同。 public int setRemoteVideoRenderer(String uid, IKVideoSink render)

参数	参数	说明
uid	用户ID	
render	基于模板方法的自定义渲染器	

返回值 0表示成功，不等于0表示失败。

原始音频数据-方法

1 registerAudioFrameObserver

注册音频数据观察者. public int registerAudioFrameObserver(IKRtcAudioFrameObserver observer)

参数	参数	说明
observer	音频数据观察者	

返回值 0表示成功，不等于0表示失败。

2 setRecordingAudioFrameParameters

设置采集的音频格式。该方法会关联 onRecordFrame 回调数据的格式。 需在加入频道前调用。 public int setRecordingAudioFrameParameters(int sampleRate, int channel, int mode, int samplesPerCall)

参数	参数	说明
sampleRate	onRecordFrame	中返回数据的采样率，可设置为 8000，16000，32000，44100 或 48000。
channel	onRecordFrame	中返回数据的通道数，可设置为：1：单声道。 2：双声道。
mode	onRecordFrame	的使用模式，0：只读模式，1：只写模式，2：读写模式
samplesPerCall	onRecordFrame	中返回数据的采样点数，如 RTMP/RTMPS 推流应用中通常为 1024。

返回值 0表示成功，不等于0表示失败。

3 setPlaybackAudioFrameParameters

设置播放的声音格式。 该方法指定 onPlaybackFrame 回调数据的格式。 需在加入频道前调用。 public int setPlaybackAudioFrameParameters(int sampleRate, int channel, int mode, int samplesPerCall)

参数	参数	说明
sampleRate	onRecordFrame	中返回数据的采样率，可设置为 8000，16000，32000，44100 或 48000。
channel	onRecordFrame	中返回数据的通道数，可设置为：1：单声道。 2：双声道。
mode	onRecordFrame	的使用模式，0：只读模式，1：只写模式，2：读写模式
samplesPerCall	onRecordFrame	中返回数据的采样点数，如 RTMP/RTMPS 推流应用中通常为 1024。

返回值 0表示成功，不等于0表示失败。

CDN推流

1 setLiveTranscoding

增加旁路推流地址。该方法用于旁路推流的视图布局及音频设置等。调用该方法更新转码参数 LiveTranscoding 时，SDK 会触发 onTranscodingUpdated 回调。 首次调用该方法设置转码参数时，不会触发 onTranscodingUpdated 回调。 public int setLiveTranscoding(LiveTranscoding transcoding)

参数	参数	说明
transcoding		旁路推流布局相关设置。

返回值 0表示成功，不等于0表示失败。

2 addPublishStreamUrl

增加旁路推流地址。调用该方法后，你可以向 CDN 推送 RTMP 或 RTMPS 协议的媒体流。SDK 会在本地触发 onRtmpStreamingStateChanged 回调，报告增加旁路推流地址的状态 public int addPublishStreamUrl(String url, boolean transcodingEnabled)

参数	参数	说明
url		CDN 推流地址，格式为 RTMP。该字符长度不能超过 1024 字节。url 不支持中文等特殊字符
transcodingEnabled		是否转码。如果设为 true，则需要在该方法前先调用setLiveTranscoding 方法。true：转码。转码是指在旁路推流时对音视频流进行转码处理后，再推送到其他 CDN 服务器。多适用于频道内有多个主播，需要进行混流、合图的场景。false：不转码

返回值 0表示成功，不等于0表示失败。

3 removePublishStreamUrl

移除旁路推流地址。 public int removePublishStreamUrl(String url)

参数	参数	说明
url		CDN 推流地址，格式为 RTMP。该字符长度不能超过 1024 字节。url 不支持中文等特殊字符

返回值 0表示成功，不等于0表示失败。

音频管理

1 setAudioProfile

设置音频编解码属性

public int setAudioProfile(int profile, int scenario)

参数	参数	说明
profile		设置采样率，码率，编码模式和声道数

返回值 0表示成功，不等于0表示失败。需要在setupLocalAudio()方法调用前使用，才能生效。

2 adjustRecordingSignalVolume

【设置采集音量】调节录音信号音量，需要对应方式获取目前音量便于管理，下同

public int adjustRecordingSignalVolume(int volume)

参数	参数	说明
volume		范围为[0,400]，默认值为 100，表示原始音量。最大支持4倍音量。不等于0设为0，大于400则设为400

返回值 0表示成功，不等于0表示失败

3 adjustAudioMixingPlayoutVolume

调节音乐文件在本地的播放音量

public int adjustAudioMixingPlayoutVolume(int volume)

参数	参数	说明
volume		范围为[0,400]，默认值为 100，表示原始音量。最大支持4倍音量。不等于0设为0，大于400则设为400

返回值 0表示成功，不等于0表示失败

4 adjustPlaybackSignalVolume

【设置播放音量】调节所有远端用户的音量

public int adjustPlaybackSignalVolume(int volume)

参数	参数	说明
volume		范围为[0,400]，默认值为 100，表示原始音量。最大支持4倍音量。不等于0设为0，大于400则设为400

返回值 0表示成功，不等于0表示失败

5 adjustUserPlaybackSignalVolume

【设置播放音量】调节指定远端用户的音量

public int adjustUserPlaybackSignalVolume(String uid, int volume)

参数	参数	说明
uid	用户ID	
volume	范围为[0,400]，默认值为 100，表示原始音量。最大支持4倍音量。不等于0设为0，大于400则设为400	

返回值 0表示成功，不等于0表示失败

6 enableLocalAudio

开关本地音频采集

public int enableLocalAudio(boolean enabled)

参数	参数	说明
enabled	true开启音频采集，false关闭音频采集	

返回值 0表示成功，不等于0表示失败

7 muteLocalAudioStream

开关本地音频发送核心

public int muteLocalAudioStream(boolean muted)

参数	参数	说明
muted	true关闭音频发送，false开启音频发送	

返回值 0表示接口调用成功，不等于0表示接口调用失败。（此返回值仅代表接口调用是否成功，实际音频发送是否成功，仍需要通过IKRtcEngineEventHandler的onAudioPublishStateChanged()回调来进行判断。）

8 muteRemoteAudioStream

接收/停止接收指定音频流核心

public int muteRemoteAudioStream(String uid, boolean muted)

参数	参数	说明
uid	用户ID	
muted	true关闭接收，false开启接收	

返回值 0表示成功，不等于0表示失败。（此返回值仅代表接口调用是否成功，实际音频接收/停止接收是否成功，仍需要通过IKRtcEngineEventHandler的onAudioSubscribeStateChanged()回调来进行判断。）

9 muteAllRemoteAudioStreams

接收/停止接收所有音频流核心

注意:同muteRemoteAudioStream接口，操作需通过回调接口确认

public int muteAllRemoteAudioStreams(boolean muted)

参数	参数	说明
muted	true关闭音频发送，false开启音频发送	

返回值 表示成功，不等于0表示失败。

10 enableAudio

启用音频模块

注意：默认为开启状态

public int enableVideo()

返回值：0表示成功，不等于0表示失败

视频管理

1 setupLocalVideo

初始化本地渲染视图

public int setupLocalVideo(KVideoCanvas kVideoCanvas)

参数	参数	说明
kVideoCanvas	本地渲染视图。	

返回值 0表示成功，不等于0表示失败。

2 setupRemoteVideo

初始化远端用户渲染视图

public int setupRemoteVideo(KVideoCanvas remote)

参数	参数	说明
remote	远端视频渲染视图。	

返回值 0表示成功，不等于0表示失败。

3setVideoEncoderConfiguration

设置视频编解码配置

public int setVideoEncoderConfiguration(VideoEncoderConfiguration config)

参数	参数	说明
config	视频编解码配置。	

返回值 0表示成功，不等于0表示失败。需要在setupLocalVideo()方法调用前使用，才能生效。

4 enableLocalVideo

开关本地视频采集。 public int enableLocalVideo(boolean enabled)

参数	参数	说明
enabled	true开启采集，false关闭采集	

返回值 0表示成功，不等于0表示失败

5 startPreview

开启视频预览

public int startPreview()

返回值：0表示成功，不等于0表示失败

6 stopPreview

停止视频预览

public int stopPreview()

返回值：0表示成功，不等于0表示失败

7 muteLocalVideoStream

开关本地视频发送

public int muteLocalVideoStream(boolean muted)

参数	参数	说明
muted	true关闭发送，false开启发送	

返回值 0表示成功，不等于0表示失败。注意，此返回值仅代表接口调用是否成功，实际视频发送是否成功，仍需要通过IKRtcEngineEventHandler的onVideoPublishStateChanged()回调来进行判断。

8 muteRemoteVideoStream

接收 / 停止接收指定远端用户的视频流

public int muteRemoteVideoStream(String uid, boolean muted)

参数	参数	说明
uid	用户ID	
muted	true关闭接收，false开启接收	

返回值 0表示成功，不等于0表示失败。注意，此返回值仅代表接口调用是否成功，实际视频接收/停止接收是否成功，仍需要通过IKRtcEngineEventHandler的onVideoSubscribeStateChanged()回调来进行判断。

9 muteAllRemoteVideoStreams

接收 / 停止接收所有远端视频流

public int muteAllRemoteVideoStreams(boolean muted)

参数	参数	说明
muted	true关闭发送，false开启发送	

返回值 0表示成功，不等于0表示失败。同muteRemoteVideoStream接口，操作需通过回调接口确认

10 setLocalRenderMode

更新本地视图显示模式

public int setLocalRenderMode(int renderMode, int mirrorMode)

参数	参数	说明
enderMode	本地渲染模式。1，比例和填充之间的妥协。2，比例优先，不保证视图完整性。3，填充优先，不保证原视频比例。-1，忽略，不设置渲染模式	
mirrorMode	镜像模式。0：关闭，1开启。-1忽略，不设置镜像模式	

返回值 0表示成功，不等于0表示失败

11 setRemoteRenderMode

public int setRemoteRenderMode(String uid, int renderMode, int mirrorMode)

参数	参数	说明
uid	用户ID	
renderMode	本地渲染模式。1，比例和填充之间的妥协。2，比例优先，不保证视图完整性。3，填充优先，不保证原视频比例。-1，忽略，不设置渲染模式	
mirrorMode	镜像模式。0：关闭，1开启	

返回值 0表示成功，不等于0表示失败。

12 setLocalRenderMode

更新本地视图显示模式

public int setLocalRenderMode(int renderMode, int mirrorMode)

参数	参数	说明
renderMode 丨本地渲染模式。1，比例和填充之间的妥协。2，比例优先，不保证视图完整性。3，填充优先，不保证原视频比例。-1，忽略，不设置渲染模式		
mirrorMode 丨镜像模式。0：关闭，1开启。-1忽略，不设置镜像模式		
renderMode	本地渲染模式。1，比例和填充之间的妥协。2，比例优先，不保证视图完整性。3，填充优先，不保证原视频比例。-1，忽略，不设置渲染模式	
mirrorMode	镜像模式。0：关闭，1开启	

返回值：0表示成功，不等于0表示失败

13 setRemoteRenderMode

更新远端视图显示模式

public int setRemoteRenderMode(String uid, int renderMode, int mirrorMode)

参数	参数	说明

renderMode 本地渲染模式。1，比例和填充之间的妥协。2，比例优先，不保证视图完整性。3，填充优先，不保证原视频比例。-1，忽略，不设置渲染模式
mirrorMode 镜像模式。0：关闭，1开启

返回值 0表示成功，不等于0表示失败

14 enableVideo

启用视频模块

public int enableVideo()

返回值 0表示成功，不等于0表示失败。

直播水印

1 addVideoWatermark

添加本地水印

public int addVideoWatermark(String watermarkUrl, WatermarkOptions options)

参数	参数	说明
watermarkUrl	待添加的水印图片的本地路径	
options	待添加的水印图片的设置选项	

返回值 0表示成功，不等于0表示失败

2 clearVideoWatermarks

清除水印

public int clearVideoWatermarks()

返回值：0表示成功，不等于0表示失败

摄像头控制

1 switchCamera

切换前置/后置摄像头

public int switchCamera()

返回值：0表示成功，不等于0表示失败

2 isCameraZoomSupported

检测设备是否支持摄像头缩放功能

public boolean isCameraZoomSupported()

返回值 true：设备支持相机缩放功能 false：设备不支持相机缩放功能

3 isCameraTorchSupported

检测设备是否支持闪光灯常开

public boolean isCameraTorchSupported()

返回值 true：设备支持闪光灯常开；false：设备不支持闪光灯常开

4 isCameraFocusSupported

检测设备是否支持手动对焦功能

public boolean isCameraFocusSupported()

返回值：true：设备支持手动对焦功能 false：设备不支持手动对焦功能。

5 isCameraExposurePositionSupported

检测设备是否支持手动曝光功能。

public boolean isCameraExposurePositionSupported()

返回值、true：设置支持手动曝光功能 false：设备不支持手动曝光功能

6 isCameraAutoFocusFaceModeSupported

检测设备是否支持人脸对焦功能

public boolean isCameraAutoFocusFaceModeSupported()

返回值 true：设备支持人脸对焦功能 false：设备不支持人脸对焦功能

7 setCameraZoomFactor

设置摄像头缩放比例

注意：相机缩放比例，有效范围从 1.0 到最大缩放

public int setCameraZoomFactor(float camZoomf) 参数	参数	说明
camZoomf	缩放比例	

返回值 方法调用成功：返回设置的 Factor 值 方法调用失败：返回值 < 0

8 getCameraMaxZoomFactor

获取摄像头支持最大缩放比例

public float getCameraMaxZoomFactor()

返回值：该相机支持的最大缩放比例

9 setCameraFocusPositionInPreview

置手动对焦位置，并触发对焦

public int setCameraFocusPositionInPreview(float positionX, float positionY, boolean var3)

参数	参数	说明
positionX	触摸点相对于视图的横坐标	
positionY	触摸点相对于视图的纵坐标	
var3	是否开启此功能	

返回值 0：方法调用成功 < 0：方法调用失败

10 setCameraExposurePosition

设置手动曝光位置

```
public int setCameraExposurePosition(float positionXinView, float positionYinView)
```

参数	参数	说明
positionXinView	触摸点相对于视图的横坐标	
positionYinView	触摸点相对于视图的纵坐标	

返回值 0：方法调用成功 < 0：方法调用失败

11 setCameraTorchOn

设置是否打开闪光灯。

```
public int setCameraTorchOn(boolean isOn)
```

参数	参数	说明
isOn	true：打开 false：关闭	

返回值 0：方法调用成功 < 0：方法调用失败。

12 setCameraAutoFocusFaceModeEnabled

设置是否开启人脸对焦功能

```
public int setCameraAutoFocusFaceModeEnabled(boolean enabled)
```

参数	参数	说明
enabled	true：打开 false：关闭	

返回值 false：（默认）关闭人脸对焦功能

其他SDK通用

1 getSdkVersion

获取SDK版本号。createKRtcEngine（）之后，无需等待结果即可调用

```
public String getSdkVersion()
```

返回值：SDK版本号

日志文件

1 setLogFilter

设置日志输出等级

注意：此接口必须在createKRtcEngine()方法之前调用

```
public static int setLogFilter(int filter)
```

参数	参数	说明
filter	日志输出等级。最小可关闭日志。	

返回值 0表示成功，不等于0表示失败。

2 setLogFileSize

设置 SDK 输出的日志文件大小

注意：此接口必须在createKRtcEngine()方法之前调用

```
public static int setLogFileSize(int fileSizeInKBytes)
```

参数	参数	说明
fileSizeInKBytes	单个日志文件的大小，单位为 KB。默认值为 1024 KB。如果你将 fileSizeInKByte 设为 1024 KB，SDK 会最多输出 5 MB 的日志文件。	

返回值 0表示成功，不等于0表示失败。

核心方法

1 joinChannel

加入频道

```
public int joinChannel(String token, String channelID, String userName, String userId)
```

参数	参数	说明
token	客户签名	
channelID	频道ID。当前频道ID，格式校验规则为：英文、数字及下划线。且开头不为下划线，长度不超过64个字符	
userName	用户名。不为空	
userId	用户ID。不为空	

返回值 0表示方法调用成功，不等于0表示失败。注意，此返回值仅代表接口调用是否成功，实际加入频道是否成功，仍需要通过IKRtcEngineEventHandler的onJoinChannelSuccess()回调来进行判断。

2 leaveChannel

离开频道

```
public int leaveChannel(String channelId)
```

参数	参数	说明
channelId	频道ID。当前频道ID，格式校验规则为：英文、数字及下划线。且开头不为下划线，长度不超过64个字符	

返回值 0表示成功，不等于0表示失败。注意，此返回值仅代表接口调用是否成功，实际离开频道是否成功，仍需要通过IKRtcEngineEventHandler的onLeaveChannel()回调来进行判断。

3 getConnectionState

获取当前频道连接状态

public int getConnectionState()

返回值 频道连接状态。1 - 网络连接断开。网络连接断开。2 - 建立网络连接中。3 - 网络已连接。4 - 重新建立网络连接中。5 - 网络连接失败

4 renewToken

更新 Token核心

public int renewToken(String token)

5 destroy

销毁SDK实例。

注意：此方法为同步方法，调用需一定耗时，建议在非UI线程调用。

public static void destroy()

6 switchChannel

快速切换直播频道。

public int switchChannel(String token, String channelId)

参数	参数	说明
token	客户签名	
channelId	频道号	

返回值 0表示成功，不等于0表示失败。

7 setClientRole

设置用户角色。

注意：SDK加入频道前调用，则为设置默认角色。SDK加入频道后调用，则为切换角色。默认为观众，即只拉流不能推流。

public int setClientRole(int role)

参数	参数	说明
role	角色	

返回值 0表示成功，不等于0表示失败。

8 createKRtcEngine

创建SDK单例并初始化。初始化完毕之后，会调用IKRtcEngineEventHandler的onSDKInitFinish(bool result)方法，返回SDK初始化结果。

public static KRtcEngine createKRtcEngine(Context context, String appId, IKRtcEngineEventHandler handler)

参数	参数	说明
context	上下文环境	
appId	用户ID	
handler	事件回调handler	

返回值 KRtcEngine单例

IKRtcEngineEventHandler

本地媒体-事件

1 onAudioPublishStateChanged

音频发布状态改变回调。

public void onAudioPublishStateChanged(String channel, int oldState, int newState)

参数	参数	说明
channel	频道ID	
oldState	旧状态	
newState	新状态	

2 onVideoPublishStateChanged

视频发布状态改变回调。

public void onVideoPublishStateChanged(String channel, int oldState, int newState)

参数	参数	说明
channel	频道ID	
oldState	旧状态	
newState	新状态	

3 onLocalAudioStateChanged

本地音频状态发生改变回调。

public void onLocalAudioStateChanged(int state, int error)

参数	参数	说明
state	当前的本地音频状态	
error	本地音频出错原因	

4 onLocalVideoStateChanged 本地视频状态发生改变回调。

public void onLocalVideoStateChanged(int localVideoState, int error)

参数	参数	说明
localVideoState	当前的本地视频状态	
error	本地视频出错原因	

远端媒体-事件

1 onRemoteVideoStateChanged

远端用户视频流状态已变化回调。

public void onRemoteVideoStateChanged(String uid, int state, int reason)

参数	参数	说明
uid	发生视频状态改变的远端用户 ID	
state	远端视频流状态	
reason	远端视频流状态改变的具体原因	

2 onRemoteAudioStateChanged

远端用户音频流状态已变化回调。

public void onRemoteAudioStateChanged(String uid, int state, int reason)

参数	参数	说明
uid	发生视频状态改变的远端用户 ID	
state	远端视频流状态	
reason	远端视频流状态改变的具体原因	

3 onAudioSubscribeStateChanged

音频订阅状态发生改变回调。

public void onAudioSubscribeStateChanged(String channel, String uid, int oldState, int newState)

参数	参数	说明
uid	用户ID	
oldState	旧状态	
newState	新状态	

4 onVideoSubscribeStateChanged

视频订阅状态发生改变回调。

public void onVideoSubscribeStateChanged(String channel, String uid, int oldState, int newState)

参数	参数	说明
uid	用户ID	
oldState	旧状态	
newState	新状态	

数据统计-事件

1 onRtcStats

当前通话统计回调。 该回调在通话中每两秒触发一次。 public void onRtcStats(IKRtcEngineEventHandler.RtcStats stats)

参数	参数	说明
stats	统计信息	

2 onRemoteAudioStats

通话中远端音频流的统计信息回调。该回调描述远端用户在通话中端到端的音频流统计信息，针对每个远端用户/主播每 2 秒触发一次。如果远端同时存在多个用户/主播，该回调每 2 秒会被触发多次。 public void onRemoteAudioStats(IKRtcEngineEventHandler.RemoteAudioStats stats)

参数	参数	说明
stats	统计信息	

3 onRemoteVideoStats

通话中远端视频流的统计信息回调。该回调描述远端用户在通话中端到端的音频流统计信息，针对每个远端用户/主播每 2 秒触发一次。如果远端同时存在多个用户/主播，该回调每 2 秒会被触发多次。 public void onRemoteVideoStats(IKRtcEngineEventHandler.RemoteVideoStats stats)

参数	参数	说明
stats	统计信息	

4 onLocalAudioStats

通话中本地音频流的统计信息回调。该回调描述本地设备发送音频流的统计信息，每 2 秒触发一次。 public void onLocalAudioStats(IKRtcEngineEventHandler.LocalAudioStats stats)

参数	参数	说明
stats	统计信息	

5 onLocalVideoStats

通话中本地视频流的统计信息回调。该回调描述本地设备发送视频流的统计信息，每 2 秒触发一次。 public void onLocalVideoStats(IKRtcEngineEventHandler.LocalVideoStats stats)

参数	参数	说明
stats	统计信息	

CDN推流

1 onRtmpStreamingStateChanged

回调当前推流状态

public void onRtmpStreamingStateChanged(String url, int state, int errCode)

参数	参数	说明
url	推流状态发生改变的 URL 地址	
state	详细的推流错误信息	

2 onTranscodingUpdated

旁路推流设置被更新回调。调用setLiveTranscoding方法更新转码参数 LiveTranscoding 时，SDK 会触发 onTranscodingUpdated 回调。 首次调用该方法设置转码参数时，不会触发

onTranscodingUpdated 回调。

public void onTranscodingUpdated()

音频管理

1、onAudioVolumeIndication

【获取用户音量（回调）】回调报告频道内瞬时音量最高的几个用户（即说话者）的用户 ID 及他们的音量

注意：*该回调默认禁用。可以通过 enableAudioVolumeIndication 方法开启。开启后，只要频道内有发流用户，SDK 会在加入频道后按 enableAudioVolumeIndication 中设置的时间间隔触发 onAudioVolumeIndication 回调。* 每次会触发两个 onAudioVolumeIndication 回调，一个报告本地发流用户的音量相关信息，另一个报告瞬时音量最高的远端用户（最多 3 位）的音量相关信息。 *本地用户静音后，SDK 立即停止报告本地用户的音量提示回调。* 瞬时音量最高的远端用户静音后 20 秒，远端的音量提示回调中将不再包含该用户；如果远端所有用户都将自己静音，20 秒后 SDK 停止报告远端用户的音量提示回调。

public void onAudioVolumeIndication(IKRtcEngineEventHandler.AudioVolumeInfo[] speakers, int totalVolume)

参数	参数	说明
speakers	用户音量信息，详见 AudioVolumeInfo 数组。如果 speakers 为空，则表示远端用户不发流或没有远端用户。	
totalVolume	混音后的总音量，取值范围为 [0,255]	

2、onActiveSpeaker

回调报告特定时间段内累积音量最高的用户 ID

public void onActiveSpeaker(String uid)

参数	参数	说明
uid	远端最活跃用户的ID	

其他SDK通用

1 onWarning

SDK 发生警告回调

public void onWarning(int warn)

参数	参数	说明
warn	警告码	