Agora Rtc Engine SDK for Windows API 参考文档

support@agora

1.1.0 版本　2015 年 11 月

目录

所需库

- **Agora Rtc Engine SDK** 需要 Visual C++ 2013 x86 runtime 库。
- 将 AgoraRtcEngineSDK/include 目录添加到项目的 INCLUDE 目录下。
- 将 AgoraRtcEngineSDK/lib 目录放入项目的 LIB 目录下，并确保 mediasdk.lib 与项目有连接。
- 将 AgoraRtcEngineSDK/dll 下的 dll 文件复制到你的可执行文件所在的目录下。

Agora Rtc Engine SDK 接口方法

1. agora::rtc::IRtcEngine 接口类

agora::rtc::IRtcEngine 是 **Agora Media SDK** 的基本接口，通过生成 agora::rtc::IRtcEngine 对象，调用该对象的方法使用 **Agora Media SDK** 的通话能力。在之前的版本中，该类名为 IAgoraAudio，从 1.0 版本更改为 agora::rtc::IRtcEngine。

创建 agora::rtc::IRtcEngine 对象

IRtcEngine*createAgoraRtcEngine(agora::rtc::IRtcEngineEventHandler* pEventHandler)

创建 IRtcEngine 对象。IRtcEngine 对象提供的接口方法，如无特殊说明，都是异步调用，对接口的调用建议在同一个线程进行。所有返回值为 int 型的 API，如无特殊说明，返回值 0 为调用成功，返回值小于 0 为调用失败。

名称	描述
pEventHandler	**Agora Media SDK** 所触发的事件通过 agora::rtc::IRtcEngineEventHandler 通知应用程序。应用程序必须继承该接口类，根据需要实现相应的回调方法，在 createAgoraRtcEngine 创建 IRtcEngine 对象时传递进去。所有回调接口都有缺省实现，因此只要继承应用程序需要的回调方法即可，不需要继承所有方法。另外，在继承的回调方法中，应该避免阻塞操作（如调用 SendMessage API），否则可能会影响 SDK 的运行。详细信息请查阅本文档中该接口类的详细说明部分。
Return Value	agora::rtc::IRtcEngine 对象

初始化(initialize)

int initialize(const char* vendorKey)

该方法用来进行初始化 Agora Media 服务。传入 Agora 为开发者签发的厂商秘钥进行初始化。在创建 agora::rtc::IRtcEngine 对象后，必须先调用该方法进行初始化，才能使用其他方法。初始化成功后，默认处于语音通话模式。使用视频功能需要额外调用一次 enableVideo API 启用视频服务。

名称	描述
vendorKey	Agora 为应用程序开发者签发的厂商秘钥。如果没有，请向

	Agora 申请。
Return Value	0：方法调用成功 <0：方法调用失败 ERR_INVALID_VENDOR_KEY(-101)：传入的 vendorKey 无效。

开启视频模式 (enableVideo)

int enableVideo()

该方法用于开启视频模式。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。关闭视频模式使用 disablevideo 方法。

名称	描述
Return Value	0：方法调用成功 <0：方法调用失败

开启纯音频模式 (disableVideo)

int disableVideo()

该方法用于关闭视频，开启纯音频模式。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。开启视频模式使用 enablevideo 方法。

名称	描述
Return Value	0：方法调用成功 <0：方法调用失败

开启视频预览(startPreview)

int startPreview()

该方法用于启动本地视频预览关闭视频。在开启预览前，必须先调用 setupLocalVideo 设置预览窗口及属性，且必须调用 enableVideo 开启视频功能。

名称	描述
Return Value	0：方法调用成功 <0：方法调用失败

停止视频预览(stopPreview)

int stopPreview()

该方法用于停止本地视频预览关闭视频。

名称	描述
Return Value	0：方法调用成功 <0：方法调用失败

加入频道(joinChannel)

int joinChannel(const char* key, const char* channel, const char* info, uid_t uid)

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 vendor key 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。

名称	描述
key	此为程序动态生成的 Token； 当用户使用静态 Key 也即只使用 vendor key 时，该参数是可选的，传 NULL 即可； 当用户使用动态 Key 时，Agora 为应用程序开发者额外签发一个签名秘钥 sign key，开发者通过 Agora 提供的算法和秘钥生成此用户 Token，用于服务器端用户验证。 一般来说使用静态 Key 即可，对于安全有极高要求的使用者请联系 Agora 客服或销售人员获得动态 Key 使用支持。
channel	频道名称。可以是任何描述性的名称，如“游戏 1”或“通话 2”。Channel 的最大长度为 128 个字符。
info	可选。一般可设置为空字符串，或者频道相关信息。
uid	可选。用户 id：32 位无符号整数。如果不指定（即设成 0），SDK 会自动分配一个，并在 onJoinChannelSuccess 方法中返回；如果指定，必须保证不同用户分配到的 uid 是唯一的。
Return Value	0：方法调用成功 <0：方法调用失败 ERR_INVALID_ARGUMENT (-2)：传递的参数无效。 ERR_NOT_READY (-3)：没有成功初始化。 ERR_REFUSED (-5)：SDK 不能发起通话，可能是因为处于另一个通话中，或者创建频道失败。

离开频道(leaveChannel)

int leaveChannel ()

离开频道，即挂断或退出通话。joinChannel 后，必须调用 leaveChannel 以结束通话，否则不能进行下一次通话。不管当前是否在通话中，都可以调用 leaveChannel，没有副作用。如果成功，则返回值为 0。leaveChannel 会把会话相关的所有资源释放掉。

名称	描述
Return Value	0：方法调用成功 <0：方法调用失败 ERR_REFUSED (-5)：离开频道失败，当前状态不在通话中，

	或者正在离开频道。

更新频道动态 key (renewChannelDynamicKey)

int renewChannelDynamicKey(const char* key)

该方法用于更新 token。如果启用了 token 机制，过一段时间后 token 会失效。当 onError 回调报告 ERR_DYNAMIC_KEY_TIMEOUT(109)时，APP 应重新获取 token，然后调用该 API 更新 token，否则 SDK 无法和服务器建立连接。

名称	描述
key	指定要更新的 token。
Return Value	0：方法调用成功 <0：方法调用失败

设置频道通话 Profile(setChannelProfile)

int setChannelProfile (CHANNEL_PROFILE_TYPE profile)

该方法用于设置频道 Profile。为了更好的优化，Agora RtcEngine 需要知道 APP 的使用场景（比如群聊模式还是主播模式），从而使用不同的优化手段。目前支持四种 Profile：自由、主播、听众、WEB SDK 兼容。

自由模式用于常见的一对一或者群聊，频道中的任何用户都可以自由说话，这是默认 Profile。

主播和听众 Profile 需要配合使用，主要用于主播模式。主播 Profile 的用户可以发送和接收音视频；听众 Profile 只能接收音视频，不能发送。

WEB SDK 兼容模式用于需要和 Agora Web SDK 互通的场景。默认的 FREE 模式是不能互通的。

注意：在同一个频道中，自由 Profile 不能和主播、听众 Profile 混用。

注意：该方法必选在进入频道前设置，在频道中设置无效。

名称	描述
Profile	指定频道 Profile，目前支持三种： • CHANNEL_PROFILE_FREE（默认） • CHANNEL_PROFILE_BROADCASTER • CHANNEL_PROFILE_AUDIENCE • CHANNEL_PROFILE_WEBSDK_COMPATIBLE
Return Value	0：方法调用成功 <0：方法调用失败

获取其它接口(queryInterface)

int queryInterface(INTERFACE_ID_TYPE iid, void inter)**

该方法用于其它接口，一般不需要 APP 直接调用。

名称	描述

iid	接口 ID。
inter	指定接口 ID 返回的接口指针。
Return Value	0：方法调用成功 <0：方法调用失败

获取通话 ID (getCallId)

int getCallId(agora::util::AString& callId)

获取当前的通话 ID，客户端在每次 joinChannel 后会生成一个对应的 CallId，标识该客户端的此次通话。有些方法如 rate()，complain()需要在通话结束后调用，向 SDK 提交反馈，这些方法必须指定 CallId 参数。使用这些反馈方法，需要在通话过程中调用 getCallId()方法获取 CallId，在通话结束后在反馈方法中作为参数传入。

名称	描述
callId	返回当前的通话 ID
Return Value	0：方法调用成功 <0：方法调用失败

给通话评分(rate)

int rate(const char* callId, int rating, const char* description)

该方法让用户给通话评分。一般在通话结束后调用。

名称	描述
callId	通过 getCallId 方法获取的通话 ID
rating	给通话的评分，最低 1 分，最高 10 分。
Description	给通话的描述，可选，长度应小于 800 字节。
Return Value	0：方法调用成功 <0：方法调用失败 ERR_INVALID_ARGUMENT (-2)：传入的参数无效，比如 callId 无效。 ERR_NOT_READY (-3)：SDK 不在正确的状态，可能是因为没有成功初始化。

投诉通话质量(complain)

int complain(const char* callId, const char* description)

该方法让用户就通话质量进行投诉。一般在通话结束后调用。

名称	描述
callId	通过 getCallId 函数获取的通话 ID
description	给通话的描述，可选，长度应小于 800 字节。
Return Value	0：方法调用成功 <0：方法调用失败

	ERR_INVALID_ARGUMENT (-2)：传入的参数无效，比如 callId 无效。 ERR_NOT_READY (-3)：SDK 不在正确的状态，可能是因为没有成功初始化。

启动语音通话测试(startEchoTest)

int startEchoTest()

该方法启动回声测试，目的是测试系统的音频设备（耳麦、扬声器等）和网络连接是否正常。在测试过程中，用户先说一段话，在 10 秒后，声音会回放出来。如果 10 秒后用户能正常听到自己刚才说的话，就表示系统音频设备和网络连接都是正常的。

注：调用 startEchoTest 后必须调用 stopEchoTest 以结束测试，否则不能进行下一次回声测试，或者调用 joinChannel 进行通话。

名称	描述
Return Value	0：方法调用成功 <0：方法调用失败 ERR_REFUSED (-5)：不能启动测试，可能没有成功初始化。

终止应答测试(stopEchoTest)

int stopEchoTest()

该方法结束回声测试。

名称	描述
Return Value	0：方法调用成功 <0：方法调用失败 ERR_REFUSED(-5)：停止 EchoTest 失败，可能是因为不在进行 EchoTest。

启用网络测试(enableNetworkTest)

int enableNetworkTest()

该方法启用网络连接质量测试，用于检测用户网络接入质量。启用后，SDK 通过 onNetworkQuality 回调方法（不定期）通知应用程序当前的网络质量。

注：启用后，在没有通话时也会消耗一些网络流量。建议的使用方法一般如下：

✓ 应用程序在前台时，调用 enableNetworkTest 启用网络接入检测；应用程序被切换到后台后，调用 disableNetworkTest 禁用检测以节省流量。网络测试是默认关闭的。

名称	描述
Return Value	0：方法调用成功 <0：方法调用失败

禁用网络测试 (disableNetworkTest)

int disableNetworkTest()

该方法禁用网络连接质量测试。

名称	描述
Return Value	0：方法调用成功 <0：方法调用失败

注册 Packet 检测器(registerPacketObserver)

int registerPacketObserver(IPacketObserver* observer)

该方法注册 Packet 检测器。在 Agora SDK 发送/接收的（语音、视频）网络包时，会回调 IPacketObserver 定义的接口，APP 可用此接口对数据做处理，比如加解密。

注意：处理后发送到网络的包大小不应超过 1200 字节，否则有可能发送失败。

名称	描述
observer	发送/接收封包的回调接口。
Return Value	0：方法调用成功 <0：方法调用失败

设置视频 Profile(setVideoProfile)

int setVideoProfile(VIDEO_PROFILE_TYPE profile)

该方法设置视频 Profile。每个 Profile 对应一套视频参数，如分辨率、帧率、码率等。

注意：应该在调用 joinChannel 进入频道前设置视频 Profile。

名称	描述
profile	视频 Profile。细节见 VIDEO_PROFILE_TYPE 的定义。
Return Value	0：方法调用成功 <0：方法调用失败

视频 Profile 定义

视频 Profile	枚举值	分辨率(宽 x 高)	帧率(fps)	码率(kbps)
120P	0	160x120	15	80
120P_2	1	120x160	15	80
120P_3	2	120x120	15	60
180P	10	320x180	15	160

180P_2	11	180x320	15	160
180P_3	12	180x180	15	120
240P	20	320x240	15	200
240P_2	21	240x320	15	200
240P_3	22	240x240	15	160
360P	30	640x360	15	400
360P_2	31	360x640	15	400
360P_3	32	360x360	15	300
360P_4	33	640x360	30	800
360P_5	34	360x640	30	800
360P_6	35	360x360	30	600
480P	40	640x480	15	500
480P_2	41	480x640	15	500
480P_3	42	480x480	15	400
480P_4	43	640x480	30	1000
480P_5	44	480x640	30	1000
480P_6	45	480x480	30	800
720P	50	1280x720	15	1000
720P_2	51	720x1080	15	1000
720P_3	52	1280x720	30	2000
720P_4	53	720x1280	30	2000
1080P	60	1920x1080	15	1500
1080P_2	61	1080x1920	15	1500
1080P_3	62	1920x1080	30	3000
1080P_4	63	1080x1920	30	3000
1080P_5	64	1920x1080	60	6000
1080P_6	65	1080x1920	60	6000
4K	70	3840x2160	30	8000

4K_2	71	2160x3840	30	8000
4K_3	72	3840x2160	60	16000
4K_4	73	2160x3840	60	16000

设置本地视频视图 (setupLocalVideo)

int setupLocalVideo(const VideoCanvas& canvas)

该方法设置本地视频的显示相关属性。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。

名称	描述

canvas	视频显示相关信息。
	`struct VideoCanvas {`
	`view_t view;`
	`int renderMode;`
	`uid_t uid;`
	`};`
	• view：视频显示视窗。在 Windows 上，view 的类型是 Windows 的窗口句柄（HWND）。应用程序应保证窗口句柄在通话中是有效的，否则可能会导致 SDK 崩溃。窗口句柄可以在 leaveChannel 调用返回后安全销毁。由于窗口句柄的值会缓存在 SDK 中，切换窗口或者销毁窗口前，应调用 setupLocalVideo 指定 view 为 NULL 清除缓存。
	• renderMode：视频显示模式
	RENDER_MODE_HIDDEN(1)：如果视频尺寸与显示视窗尺寸不一致，则视频流会按照显示视窗的比例进行周边裁剪或图像拉伸后填满视窗。
	RENDER_MODE_FIT(2)：如果视频尺寸与显示视窗尺寸不一致，则视频流会按照显示视窗的比例进行缩放后填满视窗。
	RENDER_MODE_ADAPTIVE(3)：如果自己和对方都是竖屏，或者如果自己和对方都是横屏，使用 RENDER_MODE_HIDDEN；如果对方和自己一个竖屏一个横屏，则使用 RENDER_MODE_FIT。
	• uid：用户 ID，设置为 0，代表本地视频流。

Return Value	0：方法调用成功 <0：方法调用失败

设置远程视频显示视图 (setupRemoteVideo)

int setupRemoteVideo(const VideoCanvas& canvas)

该方法绑定远程用户和显示视图，即设定 uid 指定的用户用哪个视图显示。调用该接口时需要指定远程视频的 uid，一般可以在进频道前提前设置好，如果应用程序不能事先知道对方的 uid，可以在 APP 收到 onUserJoined 事件时设置。解除某个用户的绑定视图可以把 view 设置为空。

名称	描述

canvas	视频显示相关信息。
	struct VideoCanvas {
	view_t view;
	int renderMode;
	uid_t uid;
	};
	• view：视频显示视窗。应用程序应保证窗口句柄在通话中是有效的，否则可能会导致 SDK 崩溃。窗口句柄可以在 leaveChannel 调用返回后安全销毁。由于窗口句柄的值会缓存在 SDK 中，切换窗口或者销毁窗口前，应调用 setupRemoteVideo 指定 view 为 NULL 清除缓存。
	• renderMode：视频显示模式
	RENDER_MODE_HIDDEN(1)：如果视频尺寸与显示视窗尺寸不一致，则视频流会按照显示视窗的比例进行周边裁剪或图像拉伸后填满视窗。
	RENDER_MODE_FIT(2)：如果视频尺寸与显示视窗尺寸不一致，则视频流会按照显示视窗的比例进行缩放后填满视窗。
	RENDER_MODE_ADAPTIVE(3)：如果自己和对方都是竖屏，或者如果自己和对方都是横屏，使用 RENDER_MODE_HIDDEN；如果对方和自己一个竖屏一个横屏，则使用 RENDER_MODE_FIT。
	• uid：用户 ID，指定远程视频流来自哪个用户。

Return Value	0：方法调用成功 <0：方法调用失败

报告通话质量 url (makeQualityReportUrl)

int makeQualityReportUrl(const char* channel, uid_t listenerUid, uid_t speakerUid, int format, agora::util::AString& url)

该方法生成报告通话质量的 url。通过返回的 url，应用程序可以获取详细的通话质量报告数据。使用时需要知道频道名和双方用户 ID。比如一个频道中有 A 和 B，可以分别获取 A 说话、B 收听的报告，以及 B 说话、A 收听的报告。

名称	描述

channel	在 joinChannel 方法中指定的频道名
listenerUid	听话方用户 ID
speakerUid	说话方用户 ID
format	报告格式类型 QUALITY_REPORT_JSON(0)： JSON。返回质量报告数据，应用程序根据需要展示给用户。 QUALITY_REPORT_HTML(1)： HTML。返回 HTML 格式报告，可以直接通过浏览器或 WebView 组件直接展示给用户。
url	返回生成的 URL。
Return Value	0：方法调用成功 <0：方法调用失败 ERR_INVALID_ARGUMENT (-2)：参数无效。 ERR_BUFFER_TOO_SMALL (-6)：指定缓冲区的缓冲区长度不够。

销毁 IRtcEngine 对象(release)

void release()

该方法会销毁 IRtcEngine 对象。

2. agora::rtc::RtcEngineParameters 接口类

该辅助类用于对 SDK 进行参数设置，如下为该类的具体方法说明。

将自己静音(muteLocalAudioStream)

int muteLocalAudioStream(bool mute)

静音/取消静音。该方法用于允许/禁止往网络发送本地音频流。

注：该方法不影响录音状态，并没有禁用麦克风。

名称	描述
mute	True：麦克风静音 False：取消静音
Return Value	0：方法调用成功 <0：方法调用失败

静音所有远端音频流(muteAllRemoteAudioStreams)

int muteAllRemoteAudioStreams(bool mute)

将所有远端用户静音/取消静音。本方法用于允许/禁止播放所有远端用户的音频流

注：该方法不影响音频数据流的接收，只是不播放音频流。

名称	描述

mute	True：停止播放接收到的所有音频流 False：允许播放接收到的所有音频流
Return Value	0：方法调用成功 <0：方法调用失败

将指定用户静音(muteRemoteAudioStream)

```
int muteRemoteAudioStream(uid_t uid, bool mute);
```

静音/取消静音指定用户的音频。本方法用于允许/禁止播放指定用户的音频流。

注：该方法不影响音频数据流的接收，只是不播放音频流。

名称	描述
uid	想要将其静音的用户 ID
mute	True：停止播放指定用户的音频流 False：允许播放指定用户的音频流
Return Value	0：方法调用成功 <0：方法调用失败

暂停本地视频流(muteLocalVideoStream)

```
int muteLocalVideoStream (bool mute)
```

暂停/恢复本地视频流。该方法用于允许/禁止往网络发送本地视频流。

注：该方法不影响本地视频流获取，没有禁用摄像头。

名称	描述
mute	True：不发送本地视频流 False：发送本地视频流
Return Value	0：方法调用成功 <0：方法调用失败

暂停所有远端视频流(muteAllRemoteVideoStreams)

```
int muteAllRemoteVideoStreams (bool mute)
```

暂停/恢复所有人视频流。本方法用于允许/禁止播放所有人的视频流。

注：该方法不影响视频数据流的接收，只是不播放视频流。

名称	描述
mute	True：停止播放接收到的所有视频流 False：允许播放接收到的所有视频流
Return Value	0：方法调用成功 <0：方法调用失败

暂停显示指定用户视频流(muteRemoteVideoStream)

```
int muteRemoteVideoStream(uid_t uid, bool mute);
```

暂停/恢复显示指定用户的视频。本方法用于允许/禁止显示指定用户的视频流。

名称	描述
uid	指定的用户 ID
mute	True：停止显示指定用户的视频流 False：允许播放指定用户的视频流
Return Value	0：方法调用成功 <0：方法调用失败

设定扬声器音量(setPlaybackDeviceVolume)

```
int setPlaybackDeviceVolume(int volume);
```

该方法设定扬声器音量。

名称	描述
volume	设定扬声器音量，最小 0，最大 255。
Return Value	0：方法调用成功 <0：方法调用失败

设定本地视频显示模式(setLocalRenderMode)

```
int setLocalRenderMode (int mode);
```

该方法设置本地视频显示模式。应用程序可以多次调用此方法更改显示模式。

名称	描述
mode	设置视频显示模式。 • RENDER_MODE_HIDDEN (1)：如果视频尺寸与显示视窗尺寸不一致，则视频流会按照显示视窗的比例进行周边裁剪或图像拉伸后填满窗口。 • RENDER_MODE_FIT (2)：如果视频尺寸与显示视窗尺寸不一致，在保持长宽比的前提下，将视频进行缩放后填满视窗。 • RENDER_MODE_ADAPTIVE(3)：如果自己和对方都是竖屏，或者如果自己和对方都是横屏，使用 RENDER_MODE_HIDDEN；如果对方和自己一个竖屏一个横屏，则使用 RENDER_MODE_FIT。
Return Value	0：方法调用成功 <0：方法调用失败

设定远端视频显示模式(setRemoteRenderMode)

int setRemoteRenderMode (uid_t uid, int mode);

该方法设置远端视频显示模式。应用程序可以多次调用此方法更改显示模式。

名称	描述
uid	用户 ID。
mode	设置视频显示模式。 • RENDER_MODE_HIDDEN (1)：如果视频尺寸与显示视窗尺寸不一致，则视频流会按照显示视窗的比例进行周边裁剪或图像拉伸后填满窗口。 • RENDER_MODE_FIT (2)：如果视频尺寸与显示视窗尺寸不一致，在保持长宽比的前提下，将视频进行缩放后填满视窗。 • RENDER_MODE_ADAPTIVE(3)：如果自己和对方都是竖屏，或者如果自己和对方都是横屏，使用 RENDER_MODE_HIDDEN；如果对方和自己一个竖屏一个横屏，则使用 RENDER_MODE_FIT。
Return Value	0：方法调用成功 <0：方法调用失败

启用说话者音量提示(enableAudioVolumeIndication)

int enableAudioVolumeIndication (int interval, int smooth)

该方法允许 SDK 定期向应用程序反馈当前谁在说话以及说话者的音量。

名称	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	平滑系数。默认可以设置为 3。
Return Value	0：方法调用成功 <0：方法调用失败

开始客户端录音(startAudioRecording)

int startAudioRecording (const char* filePath)

开始录音。SDK 支持在通话中进行录音，录音文件的格式为 wav。应用程序必须保证指定的目录存在而且可写。该接口需要在 joinChannel 之后调用；在 leaveChannel 时如果还在录音，会自动停止。

名称	描述
filePath	录音文件的全路径名
Return Value	0：方法调用成功 <0：方法调用失败

停止客户端录音(stopAudioRecording)

int stopAudioRecording ()

停止录音。该接口需要在 leaveChannel 之前调用，如果没有调用，在 leaveChannel 时会自动停止。

名称	描述
Return Value	0：方法调用成功 <0：方法调用失败

设置日志文件(setLogFile)

int setLogFile(const char* filePath)

设置 SDK 的输出 log 文件。SDK 运行时产生的所有 log 将写入该文件。应用程序必须保证指定的目录存在而且可写。

名称	描述
filePath	log 文件的全路径名
Return Value	0：方法调用成功 <0：方法调用失败

设置日志过滤器(setLogFilter)

int setLogFilter(unsigned int filter)

设置 SDK 的输出日志过滤器。不同的过滤器可以用或组合。

名称	描述
filter	过滤器： • 1: INFO • 2: WARNING • 4: ERROR • 8: FATAL • 0x800: DEBUG
Return Value	0：方法调用成功 <0：方法调用失败

3. agora::rtc::IRtcEngineEventHandler 接口类

agora::rtc::IRtcEngineEventHandler 接口类用于 SDK 向应用程序发送回调事件通知，应用程序通过继承该接口类的方法获取 SDK 的事件通知。接口类的所有方法都有缺省（空）实现，应用程序可以根据需要只继承关心的事件。在回调方法中，应用程序不应该做耗时或者调用可能会引起阻塞的 API（如 SendMessage），否则可能影响 SDK 的运行。

加入频道回调(onJoinChannelSuccess)

virtual void onJoinChannelSuccess (const char* channel, uid_t uid, int elapsed)

该回调方法表示该客户端成功加入了指定的频道。

名称	描述
channel	频道名
uid	用户 ID。如果 joinChannel 中指定了 uid，则此处返回该 ID；否则使用 Agora 服务器自动分配的 ID。
elapsed	从 joinChannel 开始到该事件产生的延迟（毫秒）

重新加入频道回调(onRejoinChannelSuccess)

virtual void onRejoinChannelSuccess(const char* channel, uid_t uid, int elapsed)

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。

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

发生警告回调(onWarning)

virtual void onWarning(int warn, const char* msg)

该回调方法表示 SDK 运行时出现了（网络或媒体相关的）警告。通常情况下，SDK 上报的警告信息 APP 可以忽略，SDK 会自动恢复。比如和服务器失去连接时，SDK 可能会上报 ERR_OPEN_CHANNEL_TIMEOUT 警告，同时自动尝试重连。

名称	描述
warn	警告代码
msg	警告描述

发生错误回调(onError)

virtual void onError(int err, const char* msg)

该回调方法表示 SDK 运行时出现了（网络或媒体相关的）错误。通常情况下，SDK 上报的错误意味着 SDK 无法自动恢复，需要 APP 干预或提示用户。比如启动通话失败时，SDK 会上报 ERR_START_CALL 错误。APP 可以提示用户启动通话失败，并调用 leaveChannel 退出频道。

名称	描述
err	错误代码： ERR_INVALID_VENDOR_KEY(101):无效的厂商 Key。

	ERR_INVALID_CHANNEL_NAME(102)：无效的频道名。 ERR_LOOKUP_CHANNEL_REJECTED(105)：查找频道失败，意味着服务器主动拒绝了请求。 ERR_OPEN_CHANNEL_REJECTED(107)：加入频道失败，意味着媒体服务器主动拒绝了请求。 ERR_LOAD_MEDIA_ENGINE(1001)：加载媒体引擎失败。 ERR_START_CALL（1002）：打开本地音视频设备、启动通话失败。 ERR_START_CAMERA(1003)：打开本地摄像头失败。
msg	错误描述

声音质量回调(onAudioQuality)

virtual void onAudioQuality(uid_t uid, int quality, unsigned short delay, unsigned short lost)

在通话中，该回调方法每两秒触发一次，报告当前通话的（嘴到耳）音频质量。

名称	描述
uid	说话方的用户 ID
quality	语音质量评分： • QUALITY_EXCELLENT(1) • QUALITY_GOOD(2) • QUALITY_POOR(3) • QUALITY_BAD(4) • QUALITY_VBAD(5)
delay	延迟（毫秒）
lost	丢包率

说话声音音量提示回调(onAudioVolumeIndication)

virtual void onAudioVolumeIndication (const AudioVolumeInfo* speakers, unsigned int speakerNumber, int totalVolume)

提示谁在说话及其音量，默认禁用。可以通过 enableAudioVolumeIndication()方法设置。

名称	描述
speakers	说话者（数组）。每个 speaker()： • uid：说话者的用户 ID • volume：说话者的音量（0~255）
speakerNumber	Speakers 数组的大小
totalVolume	（混音后的）总音量（0~255）

离开频道回调(onLeaveChannel)

virtual void onLeaveChannel(const RtcStats& stat)

应用程序调用 leaveChannel()方法时，SDK 提示应用程序离开频道成功。在该回调方法中，应用程序可以得到此次通话的总通话时长、SDK 收发数据的流量等信息。

名称	描述

stat	通话相关的统计信息。
	struct RtcStats {
	unsigned int duration;
	unsigned int txBytes;
	unsigned int rxBytes;
	unsigned short txKBitRate;
	unsigned short rxKBitRate;
	unsigned short lastmileQuality;
	};
	• duration：通话时长（秒）
	• txBytes：发送字节数（bytes）
	• rxBytes：接收字节数（bytes）
	• txKBitRate：发送码率（kbps）
	• rxKBitRate：接收码率（kbps）

其他用户加入当前频道回调(onUserJoined)

virtual void onUserJoined(uid_t uid, int elapsed)

提示有用户加入了频道。如果该客户端加入频道时已经有人在频道中，SDK 也会向应用程序上报这些已在频道中的用户。

注意：该回调只在 Channel Profile 为自由 Profile 时有效。

名称	描述
uid	用户 ID
elapsed	joinChannel 开始到该回调触发的延迟（毫秒）

其他用户离开当前频道回调(onUserOffline)

virtual void onUserOffline(uid_t uid, USER_OFFLINE_REASON_TYPE reason)

提示用户离线（主动离开或掉线）。

注意：SDK 判断用户离开频道（或掉线）的依据是超时：在一定时间内（15 秒）没有收到对方的任何数据包，判定为对方掉线。在网络较差的情况下，可能会有误报。建议可靠的掉线检测应该由信令来做。

注意：该回调只在 Channel Profile 为自由 Profile 时有效。

名称	描述
uid	用户 ID
reason	离线原因： • USER_OFFLINE_QUIT：用户主动离开

	• USER_OFFLINE_DROPPED：因过长时间收不到对方数据包，超时掉线。注意：由于 SDK 使用的是不可靠通道，也有可能对方主动离开本方没收到对方离开消息而误判为超时掉线。

当前通话统计回调(onRtcStats)

virtual void onRtcStats(const RtcStats& stat)

SDK 定期向应用程序报告当前通话的统计信息，每两秒触发一次。

名称	描述

stat	通话相关的统计信息。
	struct RtcStats {
	unsigned int duration;
	unsigned int txBytes;
	unsigned int rxBytes;
	unsigned short txKBitRate;
	unsigned short rxKBitRate;
	double cpuAppUsage;
	double cpuTotalUsage;
	};
	• duration：通话时长（秒），累计值
	• txBytes：发送字节数（bytes），累计值
	• rxBytes：接收字节数（bytes），累计值
	• txKBitRate：发送码率（kbps），瞬时值
	• rxKBitRate：接收码率（kbps），瞬时值
	• lastmileQuality：客户端当前接入的网络质量，瞬时值

本地视频流上传统计信息回调(onLocalVideoStat)

virtual void onLocalVideoStat(int sentBitrate, int sentFrameRate)

SDK 定期向应用程序报告本地视频流的统计信息，每两秒触发一次。

名称	描述
sentBitrate	（上次统计后）发送码率(kbps)
sentFrameRate	（上次统计后）发送帧率(fps)

接收远程视频流统计信息回调(onRemoteVideoStat)

virtual void onRemoteVideoStat(uid_t uid, int delay, int receivedBitrate, int receivedFrameRate)

SDK 定期向应用程序报告远程视频流的统计信息，每两秒触发一次。

名称	描述
uid	用户 ID，指定是哪个用户的视频流
delay	延时（毫秒）

receivedBitrate	（上次统计后）接收到的码率(kbps)
receivedFrameRate	（上次统计后）接收帧率(fps)

接收到本地视频并完成显示回调(onFirstLocalVideoFrame)

```
virtual void onFirstLocalVideoFrame(int width, int height, int elapsed)
```

第一帧本地视频显示在视图上时，触发此调用。

名称	描述
width	视频流宽(像素)
height	视频流高(像素)
elapsed	joinChannel 开始到该回调触发的延迟（毫秒）

接收到远程视频并完成解码回调(onFirstRemoteVideoDecoded)

```
virtual void onFirstRemoteVideoDecoded(uid_t uid, int width, int
height, int elapsed)
```

收到第一帧远程视频流并解码成功时，触发此调用。应用程序可以在此回调中设置该用户的 view。

名称	描述
uid	用户 ID，指定是哪个用户的视频流
width	视频流宽(像素)
height	视频流高(像素)
elapsed	joinChannel 开始到该回调触发的延迟（毫秒）

接收到远程视频并完成显示(onFirstRemoteVideoFrame)

```
virtual void onFirstRemoteVideoFrame(uid_t uid, int width, int height,
int elapsed, int elapsed)
```

第一帧远程视频显示在视图上时，触发此调用。应用程序可在此调用中获知出图时间（elapsed）。

名称	描述
uid	用户 ID，指定是哪个用户的视频流
width	视频流宽(像素)
height	视频流高(像素)
elapsed	joinChannel 开始到该回调触发的延迟（毫秒）

音频设备变化回调(onAudioDeviceStateChanged)

```
virtual void onAudioDeviceStateChanged(const char* deviceId, int
deviceType, int deviceState)
```

提示系统音频设备状态发生改变，比如耳机被拔出。

名称	描述
deviceId	设备 ID

deviceType	• UNKNOWN_AUDIO_DEVICE(-1) • PLAYOUT_DEVICE(0) • RECORDING_DEVICE(1)
deviceState	• AUDIO_DEVICE_STATE_ACTIVE(1) • AUDIO_DEVICE_STATE_DISABLED(2) • AUDIO_DEVICE_STATE_NOT_PRESENT(4) • AUDIO_DEVICE_STATE_UNPLUGGED(8)

网络质量报告回调(onNetworkQuality)

virtual void onNetworkQuality(int quality)

报告网络质量。在通话中，该回调方法每两秒触发一次。不在通话中时，如果启用了网络质量测试功能（enableNetworkTest），该回调方法会被不定期触发，向应用程序上报当前网络连接质量。

名称	描述
quality	• QUALITY_EXCELLENT(1) • QUALITY_GOOD(2) • QUALITY_POOR(3) • QUALITY_BAD(4) • QUALITY_VBAD(5) • QUALITY_DOWN(6)

用户静音回调(onUserMuteAudio)

virtual void onUserMuteAudio(uid_t uid, bool muted)

提示有其他用户已将其音频流静音（或取消静音）。

注意：该回调只在 Channel Profile 为自由 Profile 时有效。

名称	描述
uid	用户 ID
muted	True：该用户已将音频静音 False：该用户取消了音频静音

用户停止/重新发送视频(onUserMuteVideo)

virtual void onUserMuteVideo(uid_t uid, bool muted)

提示有其他用户暂停发送/恢复发送其视频流。

注意：该回调只在 Channel Profile 为自由 Profile 时有效。

名称	描述
uid	用户 ID
muted	True: 该用户已暂停发送视频流 False: 该用户已恢复发送视频流

摄像头就绪回调(onCameraReady)

virtual void onCameraReady()

提示已成功打开摄像头，可以开始捕获视频。如果摄像头打开失败，可在 onError()中处理相应错误。

视频功能停止回调(onVideoStopped)

virtual void onVideoStopped()

提示视频功能已停止。APP 如需在停止视频后对 view 做其他处理（比如显示其他画面），可以在这个回调中进行。

连接丢失回调(onConnectionLost)

virtual void onConnectionLost()

该回调方法表示 SDK 和服务器失去了网络连接。

4. agora::rtc::IAudioDeviceManager 接口类

IAudioDeviceManager 接口类提供用于测试音频设备的相关接口。可通过实例化 AAudioDeviceManager 类来获取 IAudioDeviceManager 接口。

获取系统中所有的播放设备列表(enumeratePlaybackDevices)

IAudioDeviceCollection* enumeratePlaybackDevices()

该方法返回一个 IAudioDeviceCollection 对象，包含系统中所有的播放设备。通过 IAudioDeviceCollection 对象，应用程序可以枚举播放设备。在使用结束后，应用程序需调用 IAudioDeviceCollection::release()方法销毁返回的对象。

名称	描述
Return Value	调用成功时，IAudioDeviceCollection 对象，包含所有的播放设备； 调用失败时，返回 NULL

获取系统中所有的录音设备列表(enumerateRecordingDevices)

IAudioDeviceCollection* enumerateRecordingDevices()

该方法返回一个 IAudioDeviceCollection 对象，包含系统中所有的录音设备。通过 IAudioDeviceCollection 对象，应用程序可以枚举录音设备。在使用结束后，应用程序需调用 IAudioDeviceCollection::release()方法销毁返回的对象。

名称	描述
Return Value	调用成功时，IAudioDeviceCollection 对象，包含所有的录音设备； 调用失败时，返回 NULL

（根据 deviceID）指定播放设备 (setPlaybackDevice)

int setPlaybackDevice(const char deviceId[MAX_DEVICE_ID_LENGTH])

根据 deviceID 指定播放设备。

名称	描述
deviceId	播放设备的 Device ID。可通过 enumeratePlaybackDevices()获取。插拔设备不会影响 deviceId。
Return Value	0：方法调用成功 <0：方法调用失败

（根据 deviceID）指定录音设备(setRecordingDevice)

int setRecordingDevice(const char deviceId[MAX_DEVICE_ID_LENGTH])

通过 device ID 指定录音设备。

名称	描述
deviceId	录音设备的 Device ID。可通过 enumerateRecordingDevices()获取。插拔设备不会影响 deviceId。
Return Value	0：方法调用成功 <0：方法调用失败

启动播放设备测试(startPlaybackDeviceTest)

int startPlaybackDeviceTest(const char* testAudioFilePath)

该方法测试播放设备是否能正常工作。SDK 播放指定的音频文件，测试者如果能听到声音，说明播放设备能正常工作。

名称	描述
testAudioFilePath	指定音频文件
Return Value	0：方法调用成功 <0：方法调用失败

停止播放设备测试(stopPlaybackDeviceTest)

int stopPlaybackDeviceTest()

该方法停止播放设备测试。调用 startPlaybackDeviceTest()后，必须调用该方法停止测试。

名称	描述
Return Value	0：方法调用成功 <0：方法调用失败

启动麦克风测试(startRecordingDeviceTest)

int startRecordingDeviceTest(int indicationInterval)

该方法测试麦克风是否能正常工作。启动测试后，SDK 通 onAudioVolumeIndication 回调方法向应用程序上报音量信息。

名称	描述
indicationInterval	麦克风音量报告的时间间隔（毫秒）
Return Value	0：方法调用成功 <0：方法调用失败

停止麦克风测试(stopRecordingDeviceTest)

int stopRecordingDeviceTest()

该方法停止麦克风测试。调用 startRecordingDeviceTest()后，必须调用该方法停止测试。

名称	描述
Return Value	0：方法调用成功 <0：方法调用失败

5. IAudioDeviceCollection 接口类

获取播放或录音设备数量(getCount)

int getCount()

可通过 enumeratePlaybackDevices 和 enumerateRecordingDevices 获得当前系统可以的音频设备。

名称	描述
Return Value	>0:音频设备数量 <=0:调用失败

获取指定 index 的设备信息(getDevice)

int getDevice(int index, char deviceName[MAX_DEVICE_ID_LENGTH], char deviceId[MAX_DEVICE_ID_LENGTH])

该方法获取指定 index 的音频设备信息。

名称	描述
index	输入参数，指定想查询的设备信息
deviceName	输出参数，设备名称
deviceId	输出参数，设备 ID
Return Value	0：方法调用成功 <0：方法调用失败

（根据 deviceID）指定设备(setDevice)

int setDevice(const char deviceId[MAX_DEVICE_ID_LENGTH])

该方法通过 deviceId 设置指定设备。建议使用。

名称	描述
deviceId	设备 ID
Return Value	0：方法调用成功 <0：方法调用失败

释放接口对象(release)

`void release()`

该方法用于销毁 `enumeratePlaybackDevices` 或 `enumerateRecordingDevices` 返回的 `IAudioDeviceCollection` 对象。应用程序在使用结束后，必须调用该方法销毁对象。