移动直播 SDK

Windows端集成

产品文档

腾讯云

文档目录

Windows端集成
C++
工程配置（Visual Studio）

最近更新时间：2018-10-10 11:35:24

SDK信息

您可以在腾讯云官网更新直播SDK，目前Windows端版本仅推出精简版功能：

版本类型	功能
直播精简版	支持推流、直播

下载的SDK解压后有以下几个部分：

文件名	说明
SDK	带有详细接口说明的 SDK include 及 lib、dll 文件
Demo	基于隐式链接动态库方式的简化 Demo，包含简单的 UI 界面和 SDK 的主要功能演示，使用 VS2015 可以快速导入并体验。

Visual Studio工程设置

环境要求

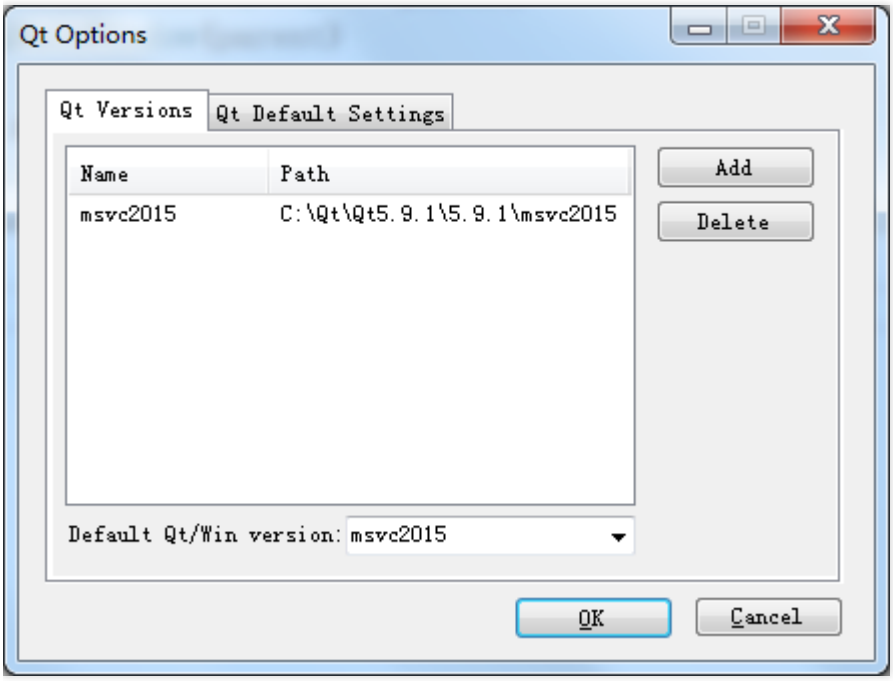
操作系统最低要求是Windows 7

Visual Studio 最低版本要求是2008

注：CustomServiceDemo 的工程，Visual Studio 最低版本要求是2015，以及依赖 Qt 5.9 或更高的版本

Qt工程设置

安装Qt 5.9 32位 和 Visual Studio Add-in For Qt 插件后，在 Visual Studio 中的 Qt VS Tools-Qt Options 中增加 Qt Versions，如下图：

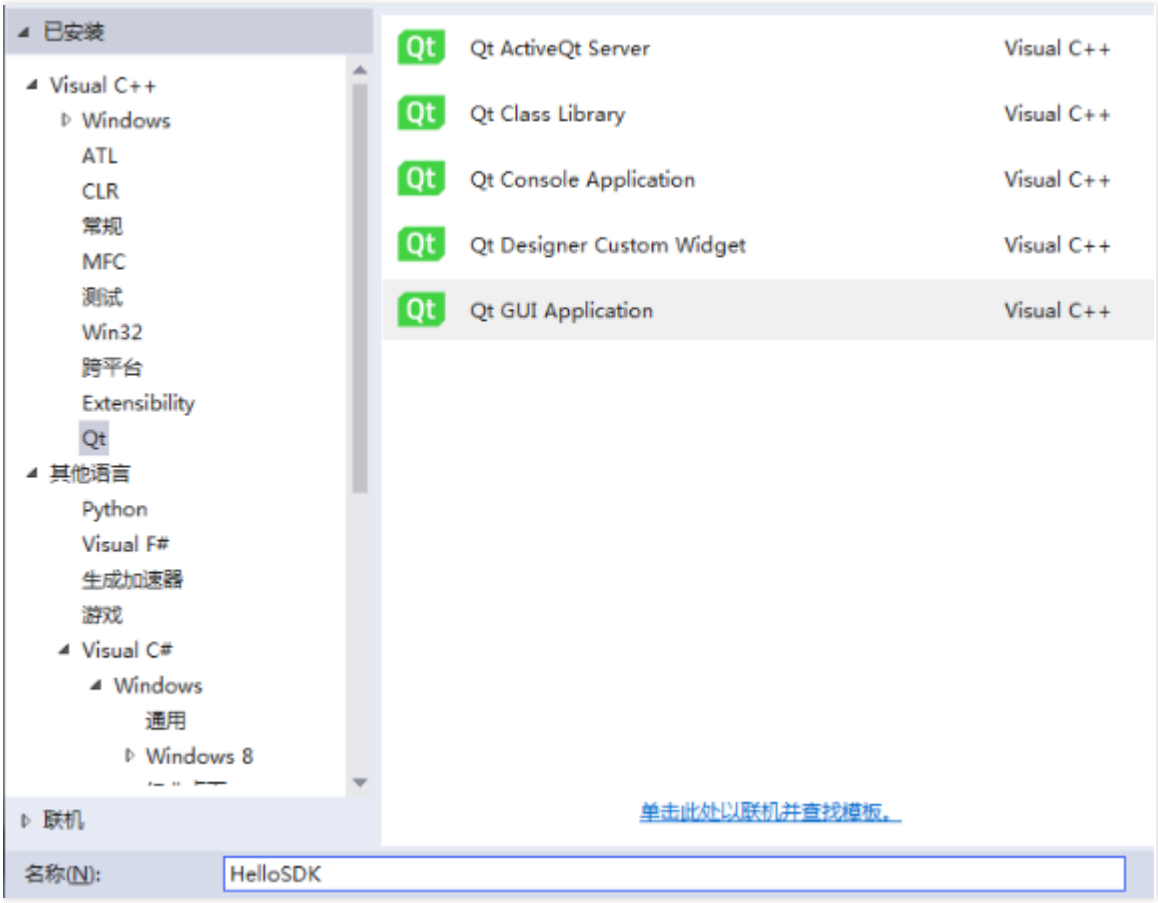


工程设置

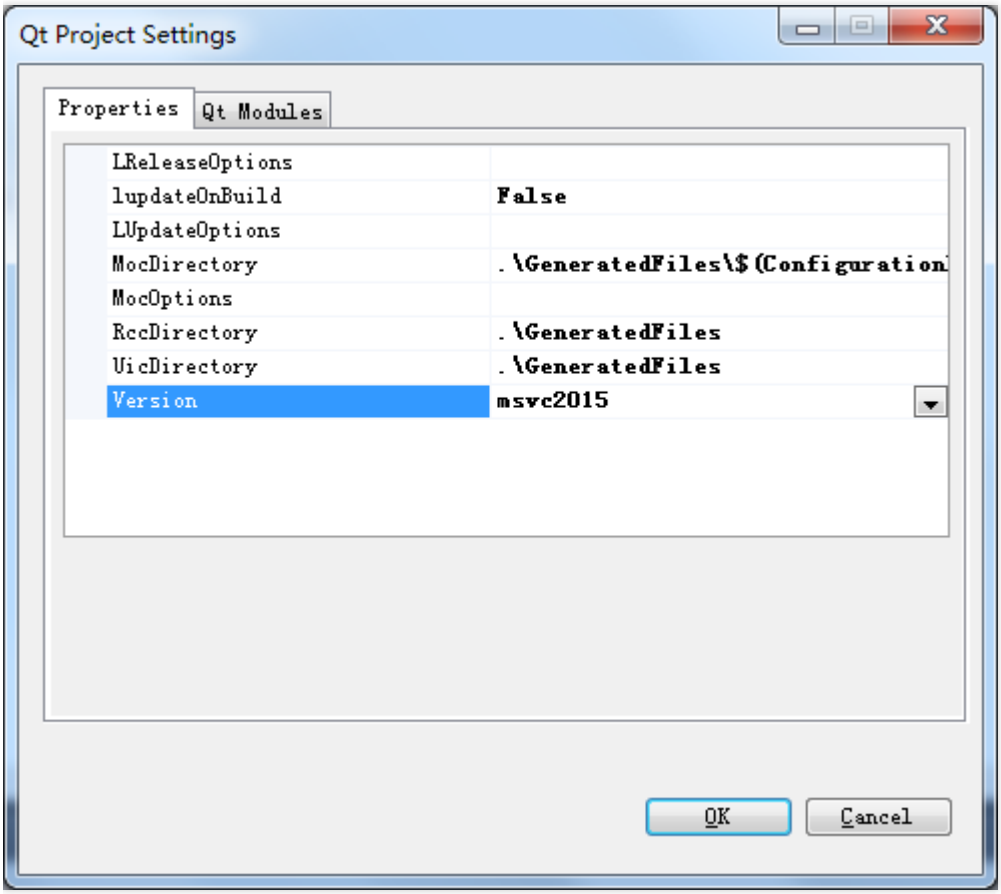
下面通过一个简单的 Qt GUI application 工程，说明如何在 Visual Studio 工程中配置 SDK。

1. 新建 Qt 工程

在本例中，新建一个名字叫做 HelloSDK 的 Qt GUI application，如下图：

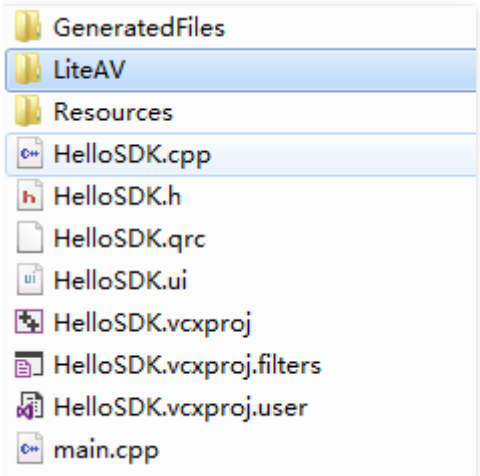


右键工程，选择 Qt Project Settings，设置 Version。如下图：



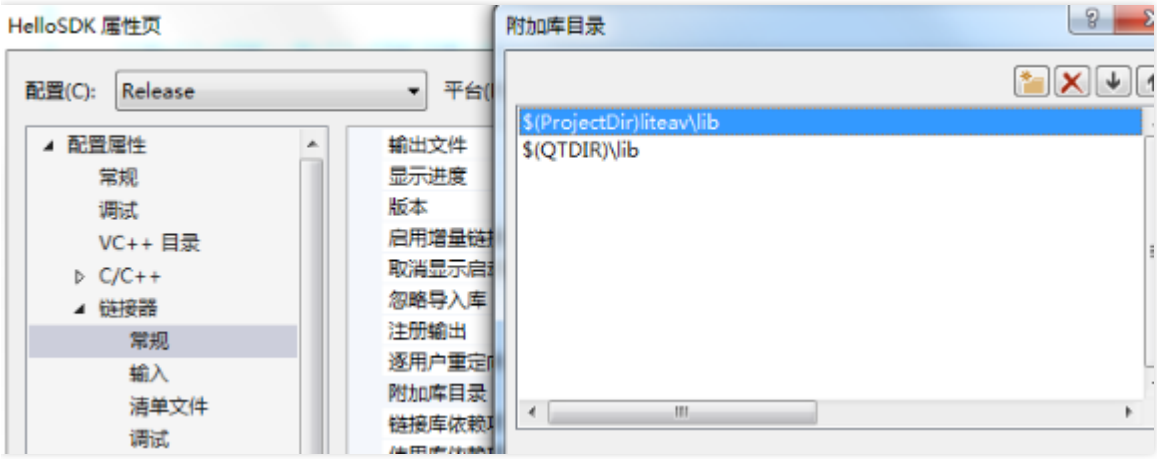
2. 拷贝SDK文件

工程目录中新建 LiteAV 文件夹，将下载的 SDK 文件夹内容拷贝至工程目录。目录结构如下图所示：

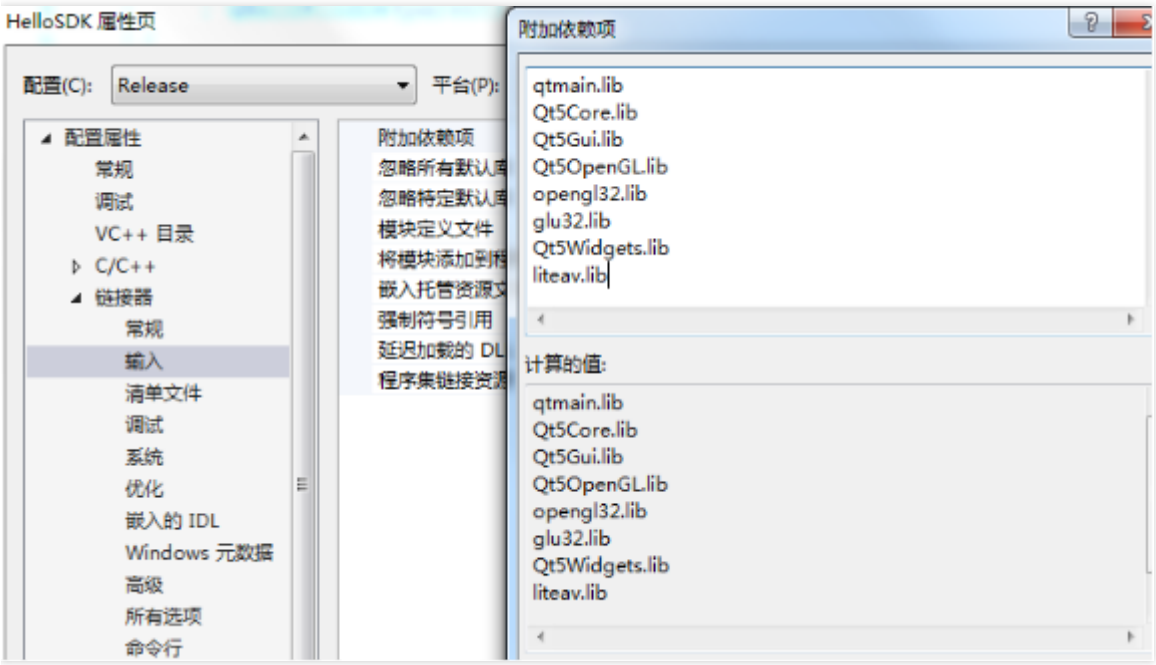


3. 添加库依赖

在工程属性页-链接器-常规添加附加库目录，如图：

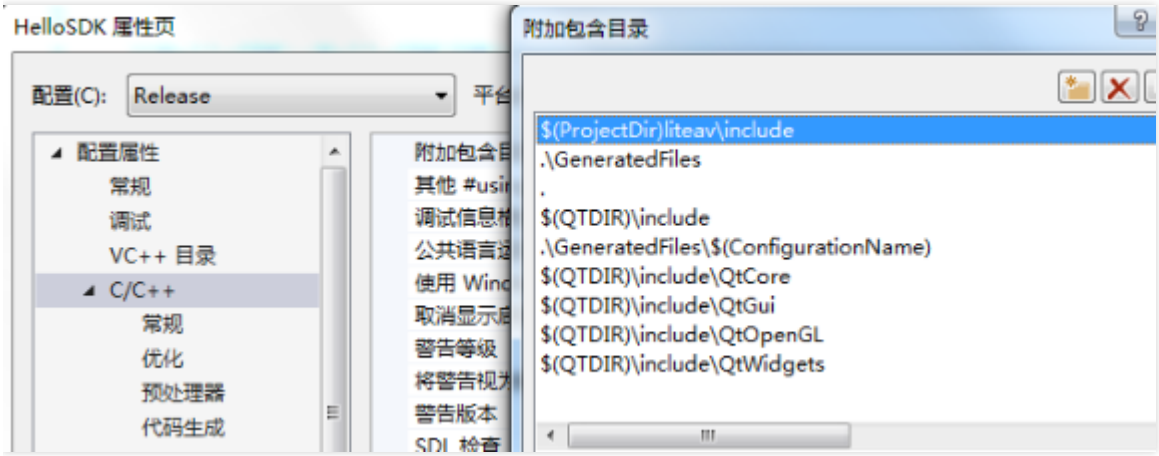


在工程属性页-链接器-输入-附加依赖项添加liteav.lib，如图：



4. 添加头文件

在工程属性页-C/C++-常规添加附加包含目录，如图：



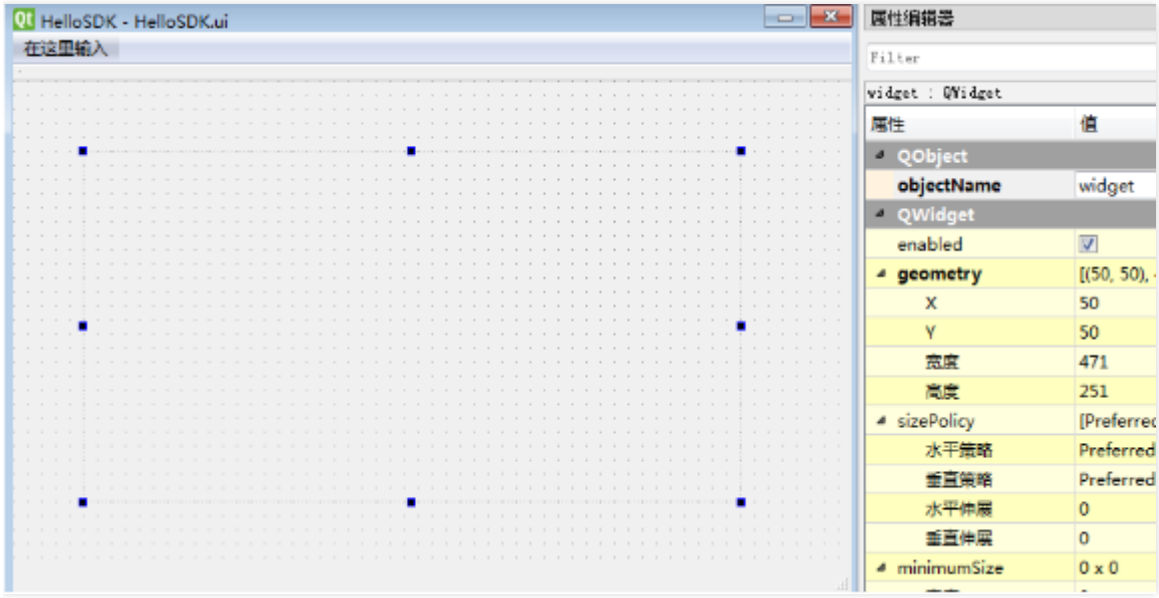
注意此项不是必须的，如果您没有添加LiteAV/include的头文件搜索路径，则在引用SDK的相关头文件时，需要在头文件前增加"LiteAV/include"，如下所示：

```
#include "LiteAV\include\TXLivePusher.h"
```

验证

1.添加渲染控件

打开工程From Files中的HelloSDK.ui，往窗体中拖入一个Widget控件，如图：



2.引用头文件

在HelloSDK.cpp开头引用SDK的头文件：

```
#include "TXLivePusher.h"
```

3.添加调用代码

声明

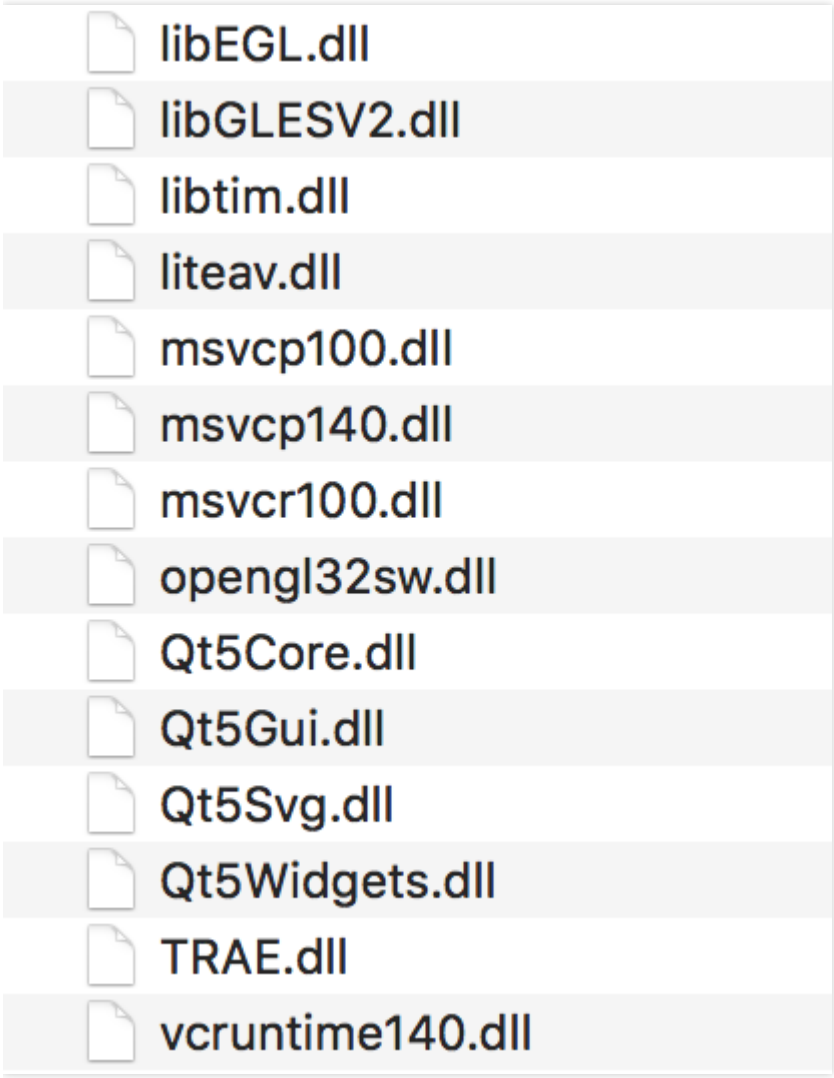
```
#pragma once

#include <QtWidgets/QMainWindow>
#include "ui_HelloSDK.h"
#include "TXLivePusher.h"

class HelloSDK : public QMainWindow
{
Q_OBJECT

public:
HelloSDK(QWidget *parent = Q_NULLPTR);

private:
Ui::HelloSDKClass ui;
TXLivePusher m_pusher;
};
```

实现

```
#include "HelloSDK.h"

HelloSDK::HelloSDK(QWidget *parent)
: QMainWindow(parent)
{
ui.setupUi(this);
m_pusher.startPreview((HWND)ui.widget->winId(), RECT{ 0, 0, ui.widget->width(), ui.widget->height()
}, 0);
}
```

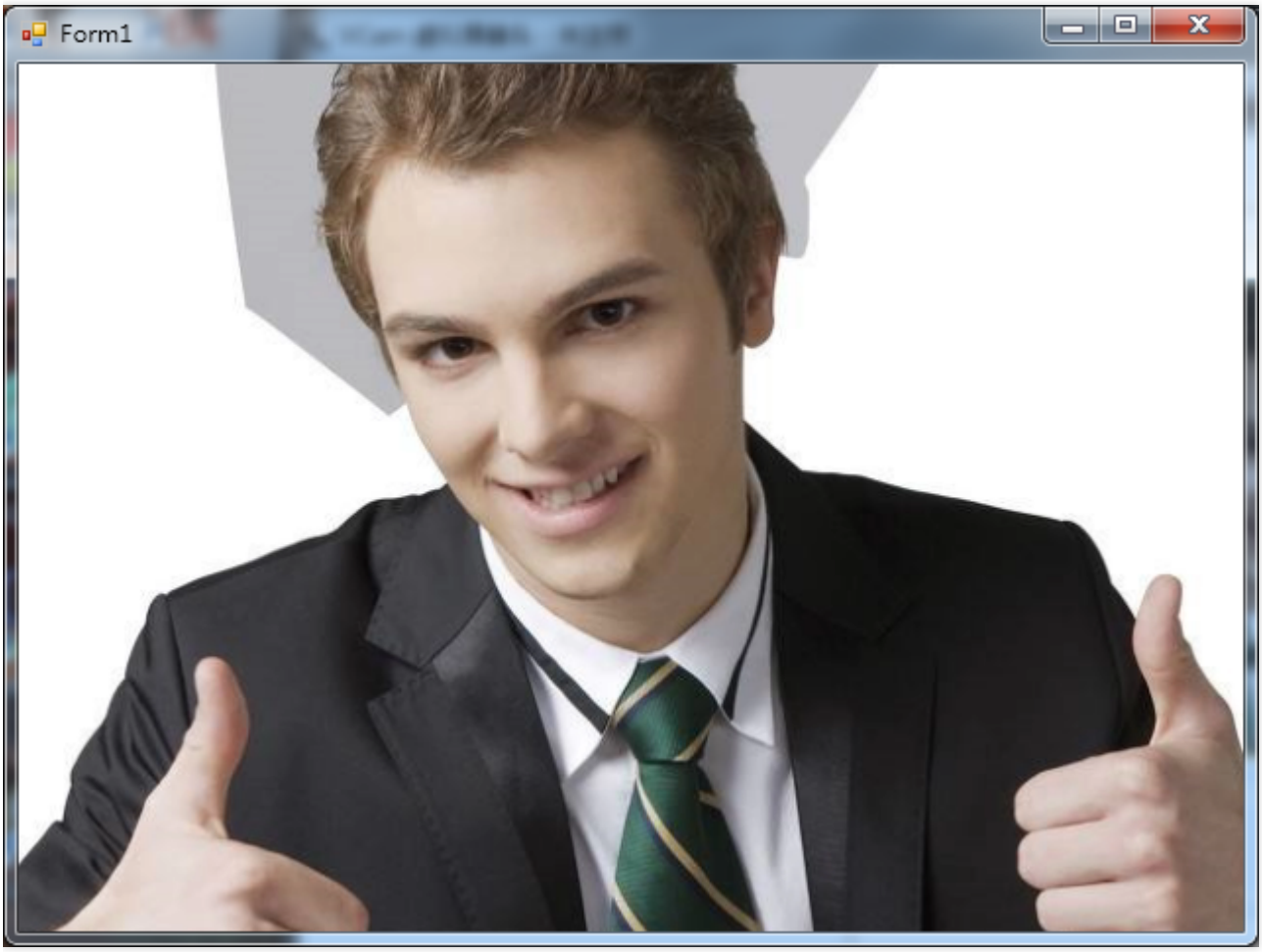
4.复制dll

将程序所需dll从LiteAV复制到release路径下，如图：



5.编译运行

如果前面各个步骤都操作正确的话，HelloSDK工程应该可以顺利编译通过。在Release下启动工程，可以看到摄像头所捕获画面已经在程序窗体上显示：


Form1

基础功能
API（C++）

最近更新时间：2018-07-10 11:36:57

API 接口列表

接口文件

在腾讯云官网 下载 SDK 开发包，可于SDK\include中可以看到详细的接口注释。

下面是腾讯视频云Windows SDK（C++）的接口列表，分为TXLivePusher和TXLivePlayer两个类，以及若干枚举enum和全局常量。

TXLivePusher

API列表

名称	描述
setCallback(callback, pUserData)	设置TXLivePusher 的回调代理
enumCameras(camerasName, capacity)	枚举当前的摄像头
micDeviceCount()	查询可用的麦克风设备的数量
micDeviceName(index, name[])	查询麦克风设备的名称
selectMicDevice(index)	选择指定的麦克风作为录音设备
micVolume()	查询已选择麦克风的音量
setMicVolume(volume)	设置已选择麦克风的音量
enableMic(bEnable)	麦克风开关，一般和混音一起使用
openSystemVoiceInput(szPlayerPath)	打开系统声音采集
closeSystemVoiceInput()	关闭系统声音采集

名称	描述
setSystemVoiceInputVolume(value)	设置系统声音采集的音量
startAudioCapture(srcType)	启动音频采集
stopAudioCapture()	关闭音频采集
startPreview(rendHwnd, rect, cameraIndex)	启动摄像头预览
updatePreview(rendHwnd, rect)	重设摄像头预览区域
stopPreview()	关闭摄像头预览
startPreview(srcType,rendHwnd,rect,userType)	启动视频预览,包括录屏、摄像头、外部源数据
setScreenCaptureParam(captureHwnd, rect)	屏幕区域捕抓参数设置接口
enumCaptureWindow(windowArray, capacity)	枚举当前的可以捕抓的窗口
captureVideoSnapShot(filePath, length)	截图当前推流的图像到本地
startPush(url)	启动推流 s
stopPush()	停止推流
switchCamera(cameraIndex)	切换摄像头，支持在推流中动态切换
setMute(mute)	静音接口
setRenderMode(mode)	设置图像的渲染（填充）模式
setRotation(rotation)	设置图像的顺时针旋转角度
setVideoQualityParamPreset(paramType)	设置推流的画面质量预设选项
setVideoResolution(resolution)	设置视频分辨率
setBeautyStyle(beautyStyle , beautyLevel, whitenessLevel)	设置美颜和美白效果
setRenderYMirror(bMirror)	设置预览渲染的镜像效果
setOutputYMirror(bMirror)	设置推流画面的镜像效果
setVideoBitRate(bitrate)	设置视频码率
setAutoAdjustStrategy(adjuststrategy)	设置流控策略

名称	描述
setVideoBitRateMin(videoBitrateMin)	允许 SDK 输出的最小视频码率，配合 setAutoAdjustStrategy 使用
setVideoBitRateMax(videoBitrateMax)	允许 SDK 输出的最大视频码率，配合 setAutoAdjustStrategy 使用
setVideoFPS(fps)	设置视频帧率
setPauseVideo(bPause)	设置视频暂停

TXLivePlayer

API列表

名称	描述
setCallback(callback, pUserData)	设置TXLivePlayer 的回调代理
speakerDeviceCount()	查询可用的扬声器设备的数量
speakerDeviceName(index, name)	查询扬声器设备的名称
selectSpeakerDevice(index)	选择指定的扬声器作为音频播放的设备
speakerVolume()	查询SDK播放的音量
setSpeakerVolume(volume)	设置SDK播放的音量
setRenderFrame(hWnd, rect)	挂接视频图像渲染
updateRenderFrame(hWnd, rect)	重设图像渲染区域
closeRenderFrame()	关闭图像渲染
startPlay(url, type)	开始播放
stopPlay()	停止播放
pause()	暂停播放
resume()	恢复播放
isPlaying()	是否正在播放
setMute(mute)	静音接口

名称	描述
setRenderMode(mode)	设置图像的渲染（填充）模式
setRotation(TXEVideoRotation)	设置图像的顺时针旋转角度
setRenderYMirror(mirror)	设置渲染的镜像效果
setOutputVideoFormat(format)	设置视频编码格式
captureVideoSnapShot(filePath,length)	截图当前拉流的图像到本地

TXLivePusher对象接口详情

1.setCallback(callback, pUserData)

接口详情：void setCallback(callback, pUserData)
设置 TXLivePusher 的回调代理，监听推流事件

- **参数说明**

参数	类型	说明
callback	TXLivePusherCallback *	代理指针
pUserData	void *	透传用户数据到 TXLivePusherCallback 的回调函数

- **示例代码**：

```
pusher.setCallback(this, reinterpret_cast<void*>(index));
```

2.enumCameras(camerasName, capacity)

接口详情：int enumCameras(camerasName, capacity)

枚举当前的摄像头，如果一台Windows同时安装了多个摄像头，那么此函数获取可用的摄像头数量和名称

- **参数说明**

参数	类型	说明
camerasName	wchar_t **	每个摄像头的名字

参数	类型	说明
capacity	size_t	camerasName 数组的大小

- **示例代码**：

```
m_cameraCount = pusher.enumCameras();
wchar_t **camerasName = new wchar_t *[m_cameraCount];
for (int i = 0; i < m_cameraCount; ++i)
{
camerasName[i] = new wchar_t[256];
}
pusher.enumCameras(camerasName, m_cameraCount);
```

3. micDeviceCount()

接口详情： int micDeviceCount() const

查询可用的麦克风设备的数量

- **返回值说明**

参数	类型	说明
vRet	int	麦克风数量

- **示例代码**：

```
int ret = pusher.micDeviceCount();
```

4. micDeviceName(index,name)

接口详情： bool micDeviceName(unsigned int index, char name[SPEAKER_DEVICE_NAME_MAX_SIZE])

查询麦克风设备的名称

- **参数说明**

参数	类型	说明
index	int	麦克风设备的索引

参数	类型	说明
name	char[]	用于存放麦克风设备的名称的字符串数组

- **示例代码**：

```
int index = 0;
char name[SPEAKER_DEVICE_NAME_MAX_SIZE];
pusher.micDeviceName(index, name);
```

5. selectMicDevice(index)

接口详情： void selectMicDevice(unsigned int index)

选择指定的麦克风作为音频录制的设备，不调用该接口时，默认选择索引为0的麦克风

- **参数说明**

参数	类型	说明
index	int	麦克风设备的索引

- **示例代码**：

```
int index = 0 ;
pusher.selectMicDevice(index);
```

6. micVolume()

接口详情：unsigned int micVolume()

查询已选择麦克风的音量

- **返回值说明**

参数	类型	说明
vRet	int	音量大小，范围是[0, 65535]

- **示例代码**：

```
int ret = pusher.micVolume();
```

7. setMicVolume(volume)

接口详情： void setMicVolume(unsigned int volume)

设置已选择麦克风的音量

- **参数说明**

参数	类型	说明
volume	int	设置的音量大小，范围是[0, 65535]

- **示例代码**：

```
int volume = 100 ;
pusher.setMicVolume(volume);
```

8. enableMic(bEnable)

接口详情： void enableMic(bool bEnable)

麦克风开关

- **参数说明**

参数	类型	说明
bEnable	bool	false为关闭麦克风采集，true为启用麦克风采集

- **示例代码**：

```
pusher.enableMic(true);
```

9. openSystemVoiceInput(szPlayerPath)

接口详情：void openSystemVoiceInput(const char* szPlayerPath)

打开系统声音或进程声音和麦克风混音

- **参数说明**

参数	类型	说明
szPlayerPath	string	播放器地址;如果用户此参数填空或不填，表示采集系统中的所有声音;如果填入exe程序(如:酷狗、QQ音乐)所在路径,将会启动此程序,并只采集此程序的声音; \

- **示例代码**：

```
pusher.openSystemVoiceInput(null);
```

10. closeSystemVoiceInput()

接口详情：void closeSystemVoiceInput()

关闭系统声音或进程声音混音

- **示例代码**：

```
pusher.closeSystemVoiceInput();
```

11. setSystemVoiceInputVolume(value)

接口详情：void setSystemVoiceInputVolume(int value)

设置系统声音采集的音量

- **参数说明**

参数	类型	说明
value	int	设置目标音量,取值范围[0,100].

- **示例代码**：

```
pusher.setSystemVoiceInputVolume(50);
```

12. startAudioCapture(srcType)

接口详情：void startAudioCapture(TXEAudioCaptureSrcType srcType = TXE_AudioSrc_Sdk_Data)

启动音频采集，SDK内部采用48K采样率，单声道，16位宽，实现很低延迟的实时音频通话的效果

- **参数说明**

参数	类型	说明
srcType	enum	音频数据源：参考定义TXEAudioCaptureSrcType

- **示例代码**：

```
pusher.startAudioCapture(TXE_AudioSrc_Sdk_Data);
```

13. stopAudioCapture()

接口详情：void stopAudioCapture()

关闭音频采集

- **示例代码**：

```
pusher.stopAudioCapture();
```

14.startPreview(rendHwnd, rect, cameraIndex)

接口详情：bool startPreview(rendHwnd, rect, cameraIndex)
启动摄像头预览，接口调用成功（true）或者失败（false）

- **参数说明**

参数	类型	说明
rendHwnd	HWND	承载预览画面的 HWND
rect	const RECT &	指定视频图像在 HWND 上的渲染区域
cameraIndex	int	指定要开启哪个摄像头的预览

- **示例代码**：

```
pusher.startPreview(m_pushRender, m_pushRect, 0);
```

15.updatePreview(rendHwnd, rect)

接口详情：void updatePreview(rendHwnd, rect)
重设摄像头预览区域，当您指定的 HWND 的窗口尺寸发生变化时，可以通过这个函数重新调整视频渲染区域

- **参数说明**

参数	类型	说明
rendHwnd	HWND	承载预览画面的 HWND
rect	const RECT &	指定视频图像在 HWND 上的渲染区域

- **示例代码**：

```
pusher.updatePreview(m_pushRender, m_pushRect);
```

16.stopPreview()

接口详情：void stopPreview()

关闭摄像头预览，stopPush 之前调用此函数并不会停止推流，而会导致 SDK 只推送音频数据

- **示例代码**：

```
pusher.stopPreview();
```

17.startPreview(srcType,rendHwnd,rect,userType)

接口详情：bool startPreview(TXEVideoCaptureSrcType srcType, HWND rendHwnd, const RECT &rect, TXEVideoUserDataSrcType userType)
启动视频源预览，SDK支持预览视频来源有多种，如摄像头、屏幕录制的视频等，接口调用成功（true）或者失败（false）

- **参数说明**

参数	类型	说明
srcType	enum	视频源类型，参考TXEVideoCaptureSrcType定义。
rendHwnd	HWND	承载预览画面的 HWND
rect	const RECT &	指定视频图像在 HWND 上的渲染区域
userType	enum	预留参数

- **示例代码**：

```
pusher.startPreview(TXE_VideoSrc_Sdk_Camera， m_pushRender, m_pushRect, TXE_UserData_Undefined);
```

18.setScreenCaptureParam(captureHwnd,rect)

接口详情：void setScreenCaptureParam(HWND captureHwnd, const RECT &rect)

屏幕区域捕抓参数设置接口，startPreview(srcType = TXE_VideoSrc_Sdk_Screen..)前调用，默认捕抓整个桌面

- **参数说明**

参数	类型	说明
captureHwnd	HWND	指定捕抓窗口，如果captureHwnd不为NULL时，捕抓整个captureHwnd窗口大小,此时captureRect设置会失效。
rect	const RECT	指定捕抓主屏幕区域RC，captureHwnd为NULL时生效，captureRect为{0}时，则捕抓整个主屏幕。

- **示例代码**：

```
pusher.setScreenCaptureParam(TXE_AudioSrc_Sdk_Data);
```

19.enumCaptureWindow(windowArray, capacity)

接口详情：static int enumCaptureWindow(HWND* windowArray,size_t capacity)

枚举当前的可以捕抓的窗口，如果桌面同时存在多个窗口，那么此函数获取可采集的窗口句柄和名称

- **参数说明**

参数	类型	说明
windowArray	HWND*	每个可采集的窗口的句柄和名字
capacity	size_t	windowArray 数组的大小

- **示例代码**：

```
int iCntWnd = m_pusher.enumCaptureWindow();
HWND *hwndList = new HWND[iCntWnd];
pusher.enumCaptureWindow(hwndList, iCntWnd);
```

20. captureVideoSnapShot(filePath,length)

接口详情：int captureVideoSnapShot(wchar_t * filePath, unsigned int length)

截图当前推流的图像到本地

- **参数说明**

参数	类型	说明
filePath	wchar_t *	文件路径，目前只支持 .jpg格式
length	unsigned int	filePath 的长度

- **示例代码**：

```
std::wstring path = L"D:\\132.jpg";
pusher.captureVideoSnapShot(path.c_str(), path.size());
```

21.startPush(url)

接口详情：bool startPush(url)

启动推流 (在 startPush 之前需要先 startPreview 启动摄像头预览，否则推送出去的数据流里只有音频)

注意推流 url 有排他性，也就是一个推流 Url 同时只能有一个推流端向上推流

接口调用成功（true）或者失败（false）。内存分配、资源申请失败等原因可能会导致返回失败

- **参数说明**

参数	类型	说明
url	const char *	一个合法的推流地址，支持 rtmp 协议（URL 以 “rtmp://” 打头 ，腾讯云推流 URL 的获取方法见 DOC ）

- **示例代码**：

```
pusher.startPush(m_pushUrl.c_str());
```

22.stopPush()

接口详情：void stopPush()

停止推流

- **示例代码**：

```
pusher.stopPush();
```

23.switchCamera(cameraIndex)

接口详情：void switchCamera(cameraIndex)

切换摄像头，支持在推流中动态切换

- **参数说明**

参数	类型	说明
cameraIndex	int	摄像头需要，取值返回： 0 ~ (摄像头个数 - 1)

- **示例代码**：

```
pusher.switchCamera(0);
```

24.setMute(mute)

接口详情：void setMute(mute)

设置静音接口

- **参数说明**

参数	类型	说明
mute	bool	是否静音

- **示例代码**：

```
pusher.setMute(true);
```

25.setRenderMode(mode)

接口详情：void setRenderMode(mode)

设置图像的渲染（填充）模式

- **参数说明**

参数	类型	说明
mode	TXERenderMode	参考 TXLiveSDKTypeDef.h 中定义的 TXERenderMode 枚举值

- **示例代码**：

```
pusher.setRenderMode(TXE_RENDER_MODE_ADAPT);
```

26.setRotation(rotation)

接口详情：void setRotation(rotation)

设置图像的顺时针旋转角度

- **参数说明**

参数	类型	说明
rotation	TXEVideoRotation	参考 TXLiveSDKTypeDef.h 中定义的 TXEVideoRotation 枚举值

- **示例代码**：

```
pusher.setRotation(TXE_VIDEO_ROTATION_90);
```

27.setVideoQualityParamPreset(paramType)

接口详情：void setVideoQualityParamPreset(TXEVideoQualityParamPreset paramType)

设置推流的画面质量预设选项

- **参数说明**

参数	类型	说明
paramType	TXEVideoQualityParamPreset	参考 TXLiveSDKTypeDef.h 中定义的 TXEVideoQualityParamPreset 枚举值

- **示例代码**：

```
pusher.setVideoQualityParamPreset(TXE_VIDEO_QUALITY_STANDARD_DEFINITION);
```

28.setVideoResolution(resolution)

接口详情：void setVideoResolution(resolution)

设置视频分辨率

- **参数说明**

参数	类型	说明
resolution	TXEVideoResolution	参考 TXLiveSDKTypeDef.h 中定义的 TXEVideoResolution 枚举值

- **示例代码**：

```
pusher.setVideoResolution(TXE_VIDEO_RESOLUTION_640x480);
```

29.setBeautyStyle(beautyStyle , beautyLevel, whitenessLevel)

接口详情：void setBeautyStyle(beautyStyle, beautyLevel, whitenessLevel)

设置美颜和美白效果

- **参数说明**

参数	类型	说明
beautyStyle	TXEBeautyStyle	参考 TXLiveSDKTypeDef.h 中定义的 TXEBeautyStyle 枚举值
beautyLevel	int	美颜级别取值范围 1 ~ 9； 0 表示关闭，1 ~ 9值越大，效果越明显
whitenessLevel	Int	美白级别取值范围 1 ~ 9； 0 表示关闭，1 ~ 9值越大，效果越明显

- **示例代码**：

```
pusher.setBeautyStyle(TXE_BEAUTY_STYLE_NATURE, 5, 5);
```

30.setRenderYMirror(mirror)

接口详情：void setRenderYMirror(mirror)

设置预览渲染的镜像效果

- **参数说明**

参数	类型	说明
mirror	bool	true表示画面左右反转，false表示保持原样

- **示例代码**：

```
pusher.setRenderYMirror(true);
```

31.setOutputYMirror(mirror)

接口详情：void setOutputYMirror(mirror)

设置推流画面的镜像效果

- **参数说明**

参数	类型	说明
mirror	bool	true表示画面左右反转，false表示保持原样

- **示例代码**：

```
pusher.setOutputYMirror(true);
```

32.setVideoBitRate(bitrate)

接口详情：void setVideoBitRate(bitrate)

设置视频码率，注意，不是分辨率越高画面越清晰，而是码率越高画面越清晰

- **参数说明**

参数	类型	说明
bitrate	unsigned long	视频码率，单位 kbps， 比如 640x360 分辨率需要配合 800kbps 的视频码率

- **示例代码**：

```
pusher.setVideoBitRate(800);
```

33.setAutoAdjustStrategy(strategy)

接口详情：void setAutoAdjustStrategy(strategy)

设置流控策略，即是否允许 SDK 根据当前网络情况调整视频码率，以避免网络上传速度不足导致的画面卡顿

- **参数说明**

参数	类型	说明
strategy	TXEAutoAdjustStrategy	参考 TXLiveSDKTypeDef.h 中定义的 TXEAutoAdjustStrategy 枚举值

- **示例代码**：

```
pusher.setAutoAdjustStrategy(TXE_AUTO_ADJUST_REALTIME_VIDEOCHAT_STRATEGY);
```

34.setVideoBitRateMin(videoBitrateMin) && setVideoBitRateMax(videoBitrateMax)

接口详情：void setVideoBitRateMin(videoBitrateMin)

void setVideoBitRateMax(videoBitrateMax)

配合 setAutoAdjustStrategy 使用，当 AutoAdjust 策略指定为 TXE_AUTO_ADJUST_NONE 时，以上的两个函数调用均视为无效

- **参数说明**

参数	类型	说明
videoBitrateMin	int	允许 SDK 输出的最小视频码率，比如 640x360 分辨率下这个值适合设置为 300kbps
videoBitrateMax	int	允许 SDK 输出的最大视频码率, 比如 640x360 分辨率下这个值适合设置为 1000kbps

- **示例代码**：

```
pusher.setVideoBitRateMin(300);
pusher.setVideoBitRateMax(1000);
```

35.setVideoFPS(fps)

接口详情：void setVideoFPS(fps)

设置视频帧率

- **参数说明**

参数	类型	说明
fps	unsigned long	视频帧率，默认值为15，重启后生效

- **示例代码**：

```
pusher.setVideoFPS(15);
```

36.setPauseVideo(bPause)

接口详情：void setPauseVideo(bool bPause)

设置视频帧率

- **参数说明**

参数	类型	说明
bPause	bool	设置视频是否暂停

- **示例代码**：

```
pusher.setPauseVideo(true);
```

##

TXLivePlayer对象接口列表详情

1.setCallback(callback, pUserData)

接口详情：void setCallback(callback, pUserData)
设置 TXLivePlayer 的回调代理，监听播放事件

- **参数说明**

参数	类型	说明
callback	TXLivePlayerCallback *	代理指针
pUserData	void *	透传用户数据到 TXLivePlayerCallback 的回调函数

- **示例代码**：

```
player.setCallback(this, reinterpret_cast<void*>(index));
```

2. speakerDeviceCount()

接口详情： int speakerDeviceCount() const

查询可用的扬声器设备的数量

- **返回值说明**

参数	类型	说明
vRet	int	扬声器数量

- **示例代码**：

```
int ret = player.speakerDeviceCount();
```

3. speakerDeviceName(index,name)

接口详情： bool speakerDeviceName(unsigned int index, char name[SPEAKER_DEVICE_NAME_MAX_SIZE])

查询扬声器设备的名称

- **参数说明**

参数	类型	说明
index	int	扬声器设备的索引

参数	类型	说明
name	string	用于存放扬声器设备的名称的字符串数组

- **示例代码**：

```
int index = 0;
char name[SPEAKER_DEVICE_NAME_MAX_SIZE];
player.speakerDeviceName(index, name);
```

4. selectSpeakerDevice(index)

接口详情： void selectSpeakerDevice(unsigned int index)

选择指定的扬声器作为音频播放的设备，不调用该接口时，默认选择索引为0的扬声器

- **参数说明**

参数	类型	说明
index	int	扬声器设备的索引

- **示例代码**：

```
int index = 0 ;
player.selectSpeakerDevice(index);
```

5. speakerVolume()

接口详情：unsigned int speakerVolume()

查询SDK播放的音量,音量值，范围是[0, 65535]

- **返回值说明**

参数	类型	说明
vRet	int	音量值，范围是[0, 65535]

- **示例代码**：

```
int ret = player.speakerVolume();
```

6. setSpeakerVolume(volume)

接口详情： void setSpeakerVolume(unsigned int volume);

设置SDK播放的音量，注意设置的不是系统扬声器的音量大小

- **参数说明**

参数	类型	说明
volume	int	设置的音量大小，范围是[0, 65535]

- **示例代码**：

```
int volume = 100 ;
player.setSpeakerVolume(volume);
```

7.setRenderFrame(hWnd, rect)

接口详情：void setRenderFrame(hWnd, rect)

挂接视频图像渲染

- **参数说明**

参数	类型	说明
hWnd	HWND	承载视频画面的 HWND
rect	const RECT &	指定视频图像在 HWND 上的渲染区域

- **示例代码**：

```
player.setRenderFrame(rendHwnd, rect);
```

8.updateRenderFrame(hWnd, rect)

接口详情：void updateRenderFrame(hWnd, rect)

重设图像渲染区域，当您指定的 HWND 的窗口尺寸发生变化时，可以通过这个函数重新调整视频渲染区域

- **参数说明**

参数	类型	说明
hWnd	HWND	承载视频画面的 HWND
rect	const RECT &	指定视频图像在 HWND 上的渲染区域

- **示例代码**：

```
player.updateRenderFrame(rendHwnd, rect);
```

9.closeRenderFrame()

接口详情：void closeRenderFrame()

关闭图像渲染

- **示例代码**：

```
player.closeRenderFrame();
```

10.startPlay(url, type)

接口详情：void startPlay(url, type)
开始播放，请在 startPlay 之前 setRenderFrame

- **参数说明**

参数	类型	说明
url	const char *	一个合法的拉流地址，视频播放 URL（URL 以 “rtmp://” 打头 ，腾讯云拉流 URL 的获取方法见 DOC ）
type	TXEPlayType	播放类型，参考 TXLiveSDKTypeDef.h 中定义的 TXEPlayType 枚举值

- **示例代码**：

```
player.startPlay(playUrl, PLAY_TYPE_LIVE_RTMP_ACC);
```

11.stopPlay()

接口详情：void stopPlay()

停止播放

- **示例代码**：

```
player.stopPlay();
```

12.pause()

接口详情：void pause()

暂停播放

- **示例代码**：

```
player.pause()
```

13.resume()

接口详情：void resume()

恢复播放

- **示例代码**：

```
player.resume()
```

14.isPlaying()

接口详情：bool isPlaying()

是否正在播放

正在播放（true）或者尚未播放（false）

- **示例代码**：

```
bool isPlaying = player.isPlaying();
```

15.setMute(mute)

接口详情：void setMute(mute)

- **参数说明**

参数	类型	说明
mute	bool	是否静音

- **示例代码**：

```
player.setMute(true);
```

16.setRenderMode(mode)

接口详情： void setRenderMode(mode)

设置图像的渲染（填充）模式

- **参数说明**

参数	类型	说明
mode	TXERenderMode	参考 TXLiveSDKTypeDef.h 中定义的 TXERenderMode 枚举值

- **示例代码**：

```
player.setRenderMode(AX_TXE_RENDER_MODE_ADAPT);
```

17.setRotation(rotation)

接口详情：void setRotation(rotation)

设置图像的顺时针旋转角度

- **参数说明**

参数	类型	说明
rotation	TXEVideoRotation	参考 TXLiveSDKTypeDef.h 中定义的 TXEVideoRotation 枚举值

- **示例代码**：

```
player.setRotation(TXE_VIDEO_ROTATION_90);
```

18.setRenderYMirror(mirror)

接口详情：void setRenderYMirror(mirror)

设置预览渲染的镜像效果

- **参数说明**

参数	类型	说明
mirror	bool	true表示画面左右反转，false表示保持原样

- **示例代码**：

```
player.setRenderYMirror(true);
```

19. setOutputVideoFormat(format)

接口详情： void setOutputVideoFormat(format)

设置视频编码格式，默认格式是TXE_OUTPUT_VIDEO_WITHOUT_OUTPUT

- **参数说明**

参数	类型	说明
format	TXEOutputVideoFormat	视频编码格式，参考 TXLiveSDKTypeDef.h 中定义的 TXEOutputVideoFormat 枚举值

- **示例代码**：

```
player.setOutputVideoFormat(TXE_OUTPUT_VIDEO_FORMAT_YUV420);
```

20. captureVideoSnapShot(filePath,length)

接口详情：int captureVideoSnapShot(wchar_t * filePath, unsigned int length)

截图当前拉流的图像到本地，

- **参数说明**

参数	类型	说明
filePath	wchar_t *	文件路径，目前只支持 .jpg格式
length	unsigned int	filePath 的长度

- **示例代码**：

```
std::wstring path = L"D:\\132.jpg";
player.captureVideoSnapShot(path.c_str(), path.size());
```

枚举类型定义-接口传参

TXEAudioCaptureSrcType

音频数据源类型

```
TXE_AudioSrc_Sdk_Data = 1, 音频采集源来自LiteAvSDK的包括本地麦克风，本地播放器、系统等声音
```

TXEVideoCaptureSrcType

视频数据源类型

```
TXE_VideoSrc_Undefined = 0, 无视频采集源
TXE_VideoSrc_Sdk_Camera = 1, 视频采集源来自LiteAvSDK地摄像头数据
TXE_VideoSrc_Sdk_Screen = 2, 视频采集源来自LiteAvSDK屏幕录屏数据
TXE_VidoeSrc_User_Data = 1001, 预留参数
```

TXEVideoUserDataSrcType

预留参数定义，尚未启用

```
TXE_UserData_Undefined = 0,
```

TXEVideoResolution

推流视频分辨率

```
// 普屏 4:3
TXE_VIDEO_RESOLUTION_320x240 = 1,
```

```
TXE_VIDEO_RESOLUTION_480x360 = 2,
TXE_VIDEO_RESOLUTION_640x480 = 3,
TXE_VIDEO_RESOLUTION_960x720 = 4,

// 宽屏16:9
TXE_VIDEO_RESOLUTION_320x180 = 101,
TXE_VIDEO_RESOLUTION_480x272 = 102,
TXE_VIDEO_RESOLUTION_640x360 = 103,
TXE_VIDEO_RESOLUTION_960x540 = 104,

// 宽屏9:16
TXE_VIDEO_RESOLUTION_180x320 = 201,
TXE_VIDEO_RESOLUTION_272x480 = 202,
TXE_VIDEO_RESOLUTION_360x640 = 203,
TXE_VIDEO_RESOLUTION_540x960 = 204,
```

TXERenderMode

视频渲染到窗口方式

```
TXE_RENDER_MODE_ADAPT = 1, // 适应，此模式下会显示整个画面的全部内容，但可能有黑边的存在
TXE_RENDER_MODE_FILLSCREEN = 2, // 填充，此模式下画面无黑边，但是会裁剪掉一部分超出渲染区域的部分，裁剪模式为居中裁剪
```

TXEVideoRotation

渲染视频旋转

```
TXE_VIDEO_ROTATION_NONE = 1, // 保持原图像的角度
TXE_VIDEO_ROTATION_90 = 2, // 顺时针旋转90度，最终图像的宽度和高度互换
TXE_VIDEO_ROTATION_180 = 3, // 顺时针旋转180度，最终图像颠倒
TXE_VIDEO_ROTATION_270 = 4, // 顺时针旋转270度，最终图像的宽度和高度互换
```

TXEAutoAdjustStrategy

推流视频流控策略

```
TXE_AUTO_ADJUST_NONE = -1, // 无流控，恒定使用 setVideoBitRate 指定的视频码率
TXE_AUTO_ADJUST_LIVEPUSH_STRATEGY = 0, // 适用于普通直播推流的流控策略，该策略敏感度比较低，会缓慢适应带宽变化，有利于在带宽波动时保持画面的清晰度。
TXE_AUTO_ADJUST_LIVEPUSH_RESOLUTION_STRATEGY = 1, // 适用于普通直播推流的流控策略，是对 LIV
```

```
EPUSH_STRATEGY 的升级版本，差别是该模式下 SDK 会根据当前码率自动调整出适合的分辨率
TXE_AUTO_ADJUST_REALTIME_VIDEOCHAT_STRATEGY = 5, // 适用于实时音视频通话的流控策略，也就是
VIDEO_QUALITY_REALTIME_VIDEOCHAT 所使用流控策略,
//该策略敏感度比较高，网络稍有风吹草动就会进行自适应调整
```

TXEVideoQualityParamPreset

SDK推流画质预设选项

```
TXE_VIDEO_QUALITY_STANDARD_DEFINITION = 1, //标清：建议追求流畅性的客户使用该选项
TXE_VIDEO_QUALITY_HIGH_DEFINITION = 2, //高清：建议对清晰度有要求的客户使用该选项
TXE_VIDEO_QUALITY_SUPER_DEFINITION = 3, //超清：如果不是大屏观看，不推荐使用
TXE_VIDEO_QUALITY_LINKMIC_MAIN_PUBLISHER = 4, //连麦大主播
TXE_VIDEO_QUALITY_LINKMIC_SUB_PUBLISHER = 5, //连麦小主播
TXE_VIDEO_QUALITY_REALTIME_VIDEOCHAT = 6, //实时音视频
TXE_VIDEO_QUALITY_STILLIMAGE_DEFINITION = 7, //静态画质场景，一般用户截屏推流，画面动态变化比
较小。内部有自适应码率和分辨率算法，不建议做自定义设置。
```

TXEOutputVideoFormat

设置输出的视频格式

```
TXE_OUTPUT_VIDEO_WITHOUT_OUTPUT = 1, // 不输出数据
TXE_OUTPUT_VIDEO_FORMAT_YUV420 = 2, // yuv420格式
TXE_OUTPUT_VIDEO_FORMAT_RGBA = 3, // RBGA格式
```

TXEBeautyStyle

美颜风格

```
TXE_BEAUTY_STYLE_SMOOTH = 0, // 光滑
TXE_BEAUTY_STYLE_NATURE = 1, // 自然
TXE_BEAUTY_STYLE_BLUR = 2, // 朦胧
```

TXEPlayType

播放类型

```
PLAY_TYPE_LIVE_RTMP = 0, RTMP直播
PLAY_TYPE_LIVE_RTMP_ACC, RTMP直播加速播放
```

CallBack Event 回调参数定义

PushEvent

推流事件列表

```
TXE_STATUS_UPLOAD_EVENT : 200001, //推流相关数据

PUSH_EVT_CONNECT_SUCC = 1001, // 已经连接推流服务器
PUSH_EVT_PUSH_BEGIN = 1002, // 已经与服务器握手完毕,开始推流
PUSH_EVT_OPEN_CAMERA_SUCC = 1003, // 打开摄像头成功
PUSH_EVT_CHANGE_RESOLUTION = 1005, // 推流动态调整分辨率
PUSH_EVT_CHANGE_BITRATE = 1006, // 推流动态调整码率
PUSH_EVT_FIRST_FRAME_AVAILABLE = 1007, // 首帧画面采集完成
PUSH_EVT_START_VIDEO_ENCODER = 1008, // 编码器启动
PUSH_EVT_CAMERA_REMOVED = 1009, // 摄像头设备已被移出（PC版SDK专用）
PUSH_EVT_CAMERA_AVAILABLE = 1010, // 摄像头设备重新可用（PC版SDK专用）
PUSH_EVT_CAMERA_CLOSED = 1011, // 关闭摄像头完成（PC版SDK专用）
PUSH_EVT_SNAPSHOT_RESULT = 1012, // 截图快照返回码

PUSH_ERR_OPEN_CAMERA_FAIL = -1301, // 打开摄像头失败
PUSH_ERR_OPEN_MIC_FAIL = -1302, // 打开麦克风失败
PUSH_ERR_VIDEO_ENCODE_FAIL = -1303, // 视频编码失败
PUSH_ERR_AUDIO_ENCODE_FAIL = -1304, // 音频编码失败
PUSH_ERR_UNSUPPORTED_RESOLUTION = -1305, // 不支持的视频分辨率
PUSH_ERR_UNSUPPORTED_SAMPLERATE = -1306, // 不支持的音频采样率
PUSH_ERR_NET_DISCONNECT = -1307, // 网络断连,且经多次重连抢救无效,可以放弃治疗,更多重试请自行重启推流
PUSH_ERR_CAMERA_OCCUPY = -1308, // 摄像头正在被占用中，可尝试打开其他摄像头（PC版SDK专用）

PUSH_WARNING_NET_BUSY = 1101, // 网络状况不佳：上行带宽太小，上传数据受阻
PUSH_WARNING_RECONNECT = 1102, // 网络断连, 已启动自动重连 (自动重连连续失败超过三次会放弃)
PUSH_WARNING_HW_ACCELERATION_FAIL = 1103, // 硬编码启动失败，采用软编码
PUSH_WARNING_VIDEO_ENCODE_FAIL = 1104, // 视频编码失败,非致命错,内部会重启编码器
PUSH_WARNING_BEAUTYSURFACE_VIEW_INIT_FAIL = 1105, // 视频编码码率异常，警告
PUSH_WARNING_VIDEO_ENCODE_BITRATE_OVERFLOW = 1106, // 视频编码码率异常，警告
PUSH_WARNING_DNS_FAIL = 3001, // RTMP -DNS解析失败
PUSH_WARNING_SEVER_CONN_FAIL = 3002, // RTMP服务器连接失败
PUSH_WARNING_SHAKE_FAIL = 3003, // RTMP服务器握手失败
PUSH_WARNING_SERVER_DISCONNECT = 3004, // RTMP服务器主动断开，请检查推流地址的合法性或防
```

```
盗链有效期
PUSH_WARNING_SERVER_NO_DATA = 3005, // 超过30s没有数据发送，主动断开连接。
```

PlayEvent

播放事件列表

```
TXE_STATUS_DOWNLOAD_EVENT : 200002, //拉流相关数据

PLAY_EVT_CONNECT_SUCC : 2001, // 已经连接服务器
PLAY_EVT_RTMP_STREAM_BEGIN : 2002, // 已经连接服务器，开始拉流
PLAY_EVT_RCV_FIRST_I_FRAME : 2003, // 渲染首个视频数据包(IDR)
PLAY_EVT_PLAY_BEGIN : 2004, // 视频播放开始
PLAY_EVT_PLAY_PROGRESS : 2005, // 视频播放进度
PLAY_EVT_PLAY_END : 2006, // 视频播放结束
PLAY_EVT_PLAY_LOADING : 2007, // 视频播放loading
PLAY_EVT_START_VIDEO_DECODER : 2008, // 解码器启动
PLAY_EVT_CHANGE_RESOLUTION : 2009, // 视频分辨率改变
PLAY_EVT_SNAPSHOT_RESULT : 2010, // 截图快照返回码

PLAY_ERR_NET_DISCONNECT : -2301, // 网络断连,且经多次重连抢救无效,可以放弃治疗,更多重试请自行重启播放
PLAY_ERR_GET_RTMP_ACC_URL_FAIL : -2302, // 获取加速拉流地址失败

PLAY_WARNING_VIDEO_DECODE_FAIL : 2101, // 当前视频帧解码失败
PLAY_WARNING_AUDIO_DECODE_FAIL : 2102, // 当前音频帧解码失败
PLAY_WARNING_RECONNECT : 2103, // 网络断连, 已启动自动重连 (自动重连连续失败超过三次会放弃)
PLAY_WARNING_RECV_DATA_LAG : 2104, // 网络来包不稳：可能是下行带宽不足，或由于主播端出流不均匀
PLAY_WARNING_VIDEO_PLAY_LAG : 2105, // 当前视频播放出现卡顿（用户直观感受）
PLAY_WARNING_HW_ACCELERATION_FAIL : 2106, // 硬解启动失败，采用软解
PLAY_WARNING_VIDEO_DISCONTINUITY : 2107, // 当前视频帧不连续，可能丢帧
PLAY_WARNING_FIRST_IDR_HW_DECODE_FAIL : 2108, // 当前流硬解第一个I帧失败，SDK自动切软解
PLAY_WARNING_DNS_FAIL : 3001, // RTMP -DNS解析失败
PLAY_WARNING_SEVER_CONN_FAIL : 3002, // RTMP服务器连接失败
PLAY_WARNING_SHAKE_FAIL : 3003, // RTMP服务器握手失败
PLAY_WARNING_SERVER_DISCONNECT : 3004, // RTMP服务器主动断开
```

TXE_STATUS_UPLOAD_EVENT && TXE_STATUS_DOWNLOAD_EVENT

状态键名定义

```
#define NET_STATUS_CPU_USAGE "CPU_USAGE" // cpu使用率
#define NET_STATUS_CPU_USAGE_D "CPU_USAGE_DEVICE" // 设备总CPU占用
#define NET_STATUS_VIDEO_BITRATE "VIDEO_BITRATE" // 当前视频编码器输出的比特率，也就是编
```

```
码器每秒生产了多少视频数据，单位 kbps
#define NET_STATUS_AUDIO_BITRATE "AUDIO_BITRATE" // 当前音频编码器输出的比特率，也就是编码器每秒生产了多少音频数据，单位 kbps
#define NET_STATUS_VIDEO_FPS "VIDEO_FPS" // 当前视频帧率，也就是视频编码器每条生产了多少帧画面
#define NET_STATUS_VIDEO_GOP "VIDEO_GOP" // 当前视频I帧间隔 ，也就是视频编码器每个I帧之间的间隔，单位s
#define NET_STATUS_NET_SPEED "NET_SPEED" // 当前的发送速度
#define NET_STATUS_NET_JITTER "NET_JITTER" // 网络抖动情况，抖动越大，网络越不稳定
#define NET_STATUS_CACHE_SIZE "CACHE_SIZE" // 缓冲区大小，缓冲区越大，说明当前上行带宽不足以消费掉已经生产的视频数据
#define NET_STATUS_DROP_SIZE "DROP_SIZE"
#define NET_STATUS_VIDEO_WIDTH "VIDEO_WIDTH"
#define NET_STATUS_VIDEO_HEIGHT "VIDEO_HEIGHT"
#define NET_STATUS_SERVER_IP "SERVER_IP"
#define NET_STATUS_CODEC_CACHE "CODEC_CACHE" //编解码缓冲大小
#define NET_STATUS_CODEC_DROP_CNT "CODEC_DROP_CNT" //编解码队列DROPCNT
#define NET_STATUS_SET_VIDEO_BITRATE "SET_VIDEO_BITRATE"
```

TXLivePusherCallback

监听推流事件

接口定义	功能说明
onEventCallback(eventId, paramCount, paramKeys, paramValues, pUserData)	TXLivePusher的推流事件通知

TXLivePlayerCallback

监听播放事件

接口定义	功能说明
onEventCallback(eventId, paramCount, paramKeys, paramValues, pUserData)	TXLivePlayer 的播放事件通知

推流和播放

最近更新时间：2018-07-11 11:18:25

推流和播放

腾讯视频云 SDK 是集 **标准推拉流** 和 **实时视频通话** 于一体的音视频 SDK 组件，本文档主要介绍了其 C++ 版本的**标准推拉流**集成方案。

产品介绍

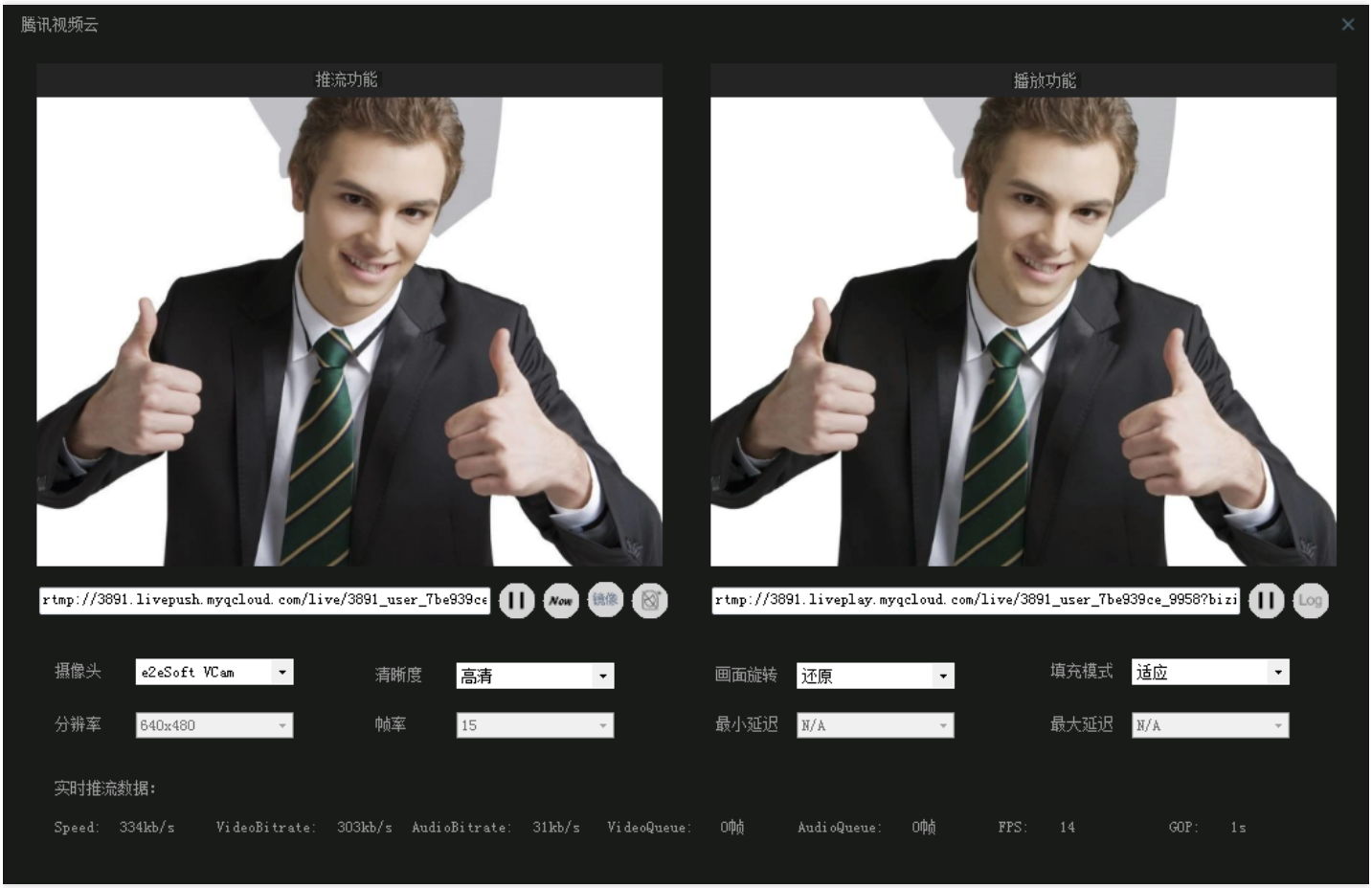
相比于 Obs Studio （常用的一款推流软件）和 Flash 播放器（PC 浏览器上常用的播放插件），视频云 SDK 最大的优势在于：

- 低延迟方面的优化 ： 可以实现实时音视频通话。
- UDP 协议加速 ： 可以获得更好的推流稳定性。

产品体验

下载 Windows SDK+Demo ，打开 CustomServiceDemo.exe 程序，首页包含3种模式，**标准直播**、**双人视频**和**多人视频**。

选择**标准直播**后，您仅需要点击 **New** 按钮即可以获取一个新的测试推流地址。开启推流及播放后，Demo 效果如下：



对接攻略

- step1：进入直播控制台，开通腾讯云直播服务，开通方法可参考 DOC。
- step2：进入直播控制台>>直播码接入>>接入配置 配置推流防盗链KEY 和 API鉴权KEY， 配置方法可参考 DOC。最终这些 URL 都是由您的后台业务服务器自动生成的。
- step3：进入直播控制台>>直播码接入>>推流生成器 可以生成测试用的 推流URL 。播放URL的拼接跟推流URL一样简单，只是需要把子域名从 **livepush** 改成 **liveplay**。
- step4：下载 SDK 开发包，并按照 工程配置 将 SDK 嵌入您的 APP 开发工程。之后参考本文档的**代码对接**进行推流及播放的相关设置与接入。
- step5：您也可以在 Windows SDK+Demo 里下载 Demo 的源码进行参考。

推流功能

代码对接

step 1: 创建 TXLivePusher 对象

先创建一个 **TXLivePusher** 对象，我们后面主要用它来完成推流工作。

```
TXLivePusher pusher;
```

创建完成后，在开始推流之前，可以调用 TXLivePusher 的相关 API 接口，设置镜像效果、码率、分辨率和填充模式等。具体 API 可参考 接口列表。

```
pusher.setRenderYMirror(true);
pusher.setOutputYMirror(true);
pusher.setVideoBitRate(900);
pusher.setVideoBitRateMin(300);
pusher.setVideoBitRateMax(1000);
pusher.setVideoResolution(TXE_VIDEO_RESOLUTION_640x480);
pusher.setRenderMode(TXE_RENDER_MODE_FILLSCREEN);
...
```

step 2: 设置预览区域

接下来我们要给摄像头的影像画面找个地方来显示，Windows 使用 HWND 窗口句柄作为基本的界面渲染单位，在 Qt 窗体应用程序中，每个控件可以调用它的 WinID() 获取得到 HWND。您只需要准备一个 HWND 和渲染区域传给 TXLivePusher 对象的 **startPreview** 接口就可以了。

```
// 设置回调，用于获取推流事件通知
pusher.setCallback(this, reinterpret_cast<void*>(index));
// 打开摄像头，摄像头索引值可通过 TXLivePusher 对象的 enumCameras 接口获取得到
// 当仅接入一个摄像头时，摄像头索引值默认为0
pusher.startPreview((HWND)ui.widget_video_push->winId(), RECT{ 0, 0, ui.widget_video_push->width
(), ui.widget_video_push->height() }, m_cameraIndex);
```

step 3: 启动推流

经过 step1 和 step2 的准备之后，用下面这段代码就可以启动推流了：

```
//rtmpURL 为对接攻略中生成的推流URL。您也可以先使用 Demo 的 New 功能产生一个测试URL进行体验。
Qstring rtmpURL = "rtmp://3891.livepush.myqcloud.com/live/xxxxxx";
pusher.startPush(rtmpURL.toLocal8Bit());
```

step 4: 控制摄像头

一台电脑中如果存在多个摄像头，可以在不同摄像头之间进行切换，只需要调用 TXLivePusher 对象的 **switchCamera** 接口，传入摄像头的索引值，即可完成切换。

```
// 传入要切换的摄像头索引值
int index = ui.cmb_cameras->currentIndex();
pusher.switchCamera(index);
```

TXLivePusher 的 enumCameras 函数可以获取摄像头的索引值。具体示例可参考 接口列表。

- Windows 下开启一个 USB 摄像头需要很长的电路和驱动启动时间，推荐的做法是：调用 TXLivePusher 的 startPreview 时指定 cameraIndex 为 -1 来打开全部摄像头 ，再用 switchCamera 实现瞬间切换摄像头
- SDK 支持虚拟摄像头，启用用法和普通摄像头一致，如果您的 PC 还没有物理摄像头可用，可以安装 VCam 这样的虚拟摄像头软件来辅助调试。

step 5: 结束推流

结束推流很简单，但也很重要（例如我们需要释放占用的摄像头硬件等），若清理工作不当可能会导致下次推流出现问题。

```
// 回调置空
pusher.setCallback(NULL, NULL);
// 停止摄像头预览
pusher.stopPreview();
// 停止推流
pusher.stopPush();
```

step 6: 更多功能

本地摄像头的画面镜像、静音、清晰度设置等功能，可以参考 接口列表，获取更多详细信息。

事件处理

SDK 通过 TXLivePusher 对象的 setCallback 接口来监听推流相关的事件。

1. 常规事件

即成功的推流都可能会通知的事件，比如收到1003就意味着摄像头的画面已经开始渲染了。

事件ID	数值	含义说明
PUSH_EVT_CONNECT_SUCC	1001	已经连接推流服务器
PUSH_EVT_PUSH_BEGIN	1002	已经与服务器握手完毕,开始推流

事件ID	数值	含义说明
PUSH_EVT_OPEN_CAMERA_SUCC	1003	打开摄像头成功
PUSH_EVT_CHANGE_RESOLUTION	1005	推流动态调整分辨率
PUSH_EVT_CHANGE_BITRATE	1006	推流动态调整码率
PUSH_EVT_FIRST_FRAME_AVAILABLE	1007	首帧画面采集完成
PUSH_EVT_START_VIDEO_ENCODER	1008	编码器启动
PUSH_EVT_CAMERA_REMOVED	1009	摄像头设备已被移出
PUSH_EVT_CAMERA_AVAILABLE	1010	摄像头设备重新可用
PUSH_EVT_CAMERA_CLOSED	1011	关闭摄像头完成
PUSH_EVT_SNAPSHOT_RESULT	1012	截图快照返回码

2. 错误通知

SDK发现了一些严重问题，推流无法继续了，比如 Camera 已被其他程序占用导致摄像头打不开。

事件ID	数值	含义说明
PUSH_ERR_OPEN_CAMERA_FAIL	-1301	打开摄像头失败
PUSH_ERR_OPEN_MIC_FAIL	-1302	打开麦克风失败
PUSH_ERR_VIDEO_ENCODE_FAIL	-1303	视频编码失败
PUSH_ERR_AUDIO_ENCODE_FAIL	-1304	音频编码失败
PUSH_ERR_UNSUPPORTED_RESOLUTION	-1305	不支持的视频分辨率
PUSH_ERR_UNSUPPORTED_SAMPLERATE	-1306	不支持的音频采样率
PUSH_ERR_NET_DISCONNECT	-1307	网络断连,且经多次重连抢救无效,可以放弃治疗,更多重试请自行重启推流
PUSH_ERR_CAMERA_OCCUPY	-1308	摄像头正在被占用中，可尝试打开其他摄像头

3. 警告事件

相比于错误信息，WARNING 事件所代表的问题通常不会打断推流进程，而且 SDK 内部往往会启动自动修复逻辑，所以大多数 WARNING 事件您都无需关心，不过下面两个事件较为重要和有用：

WARNING_NET_BUSY

上行的网速不佳，它代表用户当前的音视频数据不能很流畅的推给服务器，这时一般可以给用户一些 UI 上的提示，比如“您的网速不太给力”，让用户对于另一端的卡顿有一个心理预期。

WARNING_SERVER_DISCONNECT

推流请求被后台拒绝了，出现这个问题一般是由于推流地址里的 txSecret 计算错了，或者是推流地址被其他人占用了（推流 URL 是排他的，一个推流 URL 同时只能有一个用户去使用）。

事件ID	数值	含义说明
PUSH_WARNING_NET_BUSY	1101	网络状况不佳：上行带宽太小，上传数据受阻
PUSH_WARNING_RECONNECT	1102	网络断连, 已启动自动重连 (自动重连连续失败超过三次会放弃)
PUSH_WARNING_HW_ACCELERATION_FAIL	1103	硬编码启动失败，采用软编码
PUSH_WARNING_VIDEO_ENCODE_FAIL	1104	视频编码失败,非致命错,内部会重启编码器
PUSH_WARNING_BEAUTYSURFACE_VIEW_INIT_FAIL	1105	视频编码码率异常，警告
PUSH_WARNING_VIDEO_ENCODE_BITRATE_OVERFLOW	1106	视频编码码率异常，警告
PUSH_WARNING_DNS_FAIL	3001	RTMP -DNS解析失败（会触发重试流程）
PUSH_WARNING_SEVER_CONN_FAIL	3002	RTMP服务器连接失败（会触发重试流程）
PUSH_WARNING_SHAKE_FAIL	3003	RTMP服务器握手失败（会触发重试流程）
PUSH_WARNING_SERVER_DISCONNECT	3004	RTMP服务器主动断开，请检查推流地址的合法性或防盗链有效期
PUSH_WARNING_SERVER_NO_DATA	3005	超过30s没有数据发送，主动断开连接

播放功能

代码对接

step 1: 创建 TXLivePlayer 对象

先创建一个 **TXLivePlayer** 对象，我们后面主要用它来完成推流工作。

```
TXLivePlayer player;
```

step 2: 设置渲染区域

和推流相似，接下来我们要给摄像头的影像画面找个地方来显示，Windows 使用 HWND 窗口句柄作为基本的界面渲染单位，在 Qt 窗体应用程序中，每个控件可以调用它的 WinID() 获取得到 HWND。您只需要准备一个 HWND 和渲染区域传给 TXLivePlayer 对象的 **setRenderFrame** 接口就可以了。

```
// 设置回调，用于获取播放事件通知
player.setCallback(this, reinterpret_cast<void*>(index));
// 传入HWND窗口句柄，以及渲染位置和大小
player.setRenderFrame((HWND)ui.widget_video_play->winId(), RECT{ 0, 0, ui.widget_video_play->widt
h(), ui.widget_video_play->height() });
```

step 3: CDN播放

启动播放的功能，只需调用 TXLivePlayer 的 startPlay 接口即可，其中 PLAY_TYPE_LIVE_RTMP 是常规 CDN 的 RTMP 地址，优点是带宽价格很低，但延迟一般比较大。

```
//rtmpURL 为对接攻略中生成的播放URL。
string rtmpURL = "rtmp://3891.liveplay.myqcloud.com/live/xxxxxx";
player.startPlay(rtmpURL，PLAY_TYPE_LIVE_RTMP);
```

step 3': 超低延迟播放

超低延迟播放可以将端到端的延迟拉低到 400ms 左右，该模式下 SDK 的内部机制和常规模式完全不同，于此同时， SDK 所使用的音视频线路也不是常规 CDN 线路，所以单价较 CDN 偏高，一般仅适用于音视频通话和对延迟要求比较苛刻的场景。

```
//rtmpURL 为对接攻略中生成的带防盗链签名的播放URL。您也可以先使用 Demo 的 New 功能产生一个测试URL进行体验。
string rtmpURL = "rtmp://3891.liveplay.myqcloud.com/live/xxxxxxbizid=2157&
txSecret=xxxxxx&txTime=xxxxxx";
player.startPlay(rtmpURL，PLAY_TYPE_LIVE_RTMP_ACC);
```

注意事项

- 该功能并不需要提前开通，但是要求直播流必须位于腾讯云，跨云商实现低延时链路的难度不仅仅是技术层面的。
- 播放URL 不能用普通的 CDN URL， 必须要带防盗链签名。

- 在调用 startPlay 接口时，指定 type 为 **PLAY_TYPE_LIVE_RTMP_ACC**，SDK 会拉取超低延迟的直播流。
- 最多同时 10 路 并发播放，所以您只能在互动场景中使用（比如连麦主播或者夹娃娃直播中的操作者这一路）。
- Obs Studio 推出的 RTMP 流在客户端就引入了 1s 以上的延迟。因此建议使用视频云 SDK 进行推流。

- **updateRenderFrame**
  当您指定的 HWND 的窗口尺寸发生变化时，可以通过 **updateRenderFrame** 函数重新调整视频渲染区域。
- **setRenderMode**

step 4: 结束播放

结束播放时，需调用 TXLivePlayer 的 stopPlay 接口进行资源的释放（例如网络的下载等）。

```
// 回调置空
player.setCallback(NULL, NULL);
// 停止拉流播放
player.stopPlay();
```

step 5: 更多功能

可以参考 接口列表，获取更多详细信息。

事件处理

通过 TXLivePlayer 对象的 setCallback 接口来监听播放相关的事件。

1. 常规事件

即成功的播放都可能会通知的事件。

事件ID	数值	含义说明
PLAY_EVT_CONNECT_SUCC	2001	已经连接服务器
PLAY_EVT_RTMP_STREAM_BEGIN	2002	已经连接服务器，开始拉流
PLAY_EVT_RCV_FIRST_I_FRAME	2003	渲染首个视频数据包(IDR)
PLAY_EVT_PLAY_BEGIN	2004	视频播放开始
PLAY_EVT_PLAY_PROGRESS	2005	视频播放进度
PLAY_EVT_PLAY_END	2006	视频播放结束
PLAY_EVT_PLAY_LOADING	2007	视频播放loading
PLAY_EVT_START_VIDEO_DECODER	2008	解码器启动

事件ID	数值	含义说明
PLAY_EVT_CHANGE_RESOLUTION	2009	视频分辨率改变
PLAY_EVT_SNAPSHOT_RESULT	2010	截图快照返回码

2. 错误通知

SDK 遭遇了一些严重错误，比如网络断开等等，这些错误会导致播放无法继续，因此您的代码需要对这些错误进行相应的处理。

事件ID	数值	含义说明
PLAY_ERR_NET_DISCONNECT	-2301	网络断连，且重试亦不能恢复，将导致播放失败
PLAY_ERR_GET_RTMP_ACC_URL_FAIL	-2302	获取加速拉流地址失败，会导致播放失败

3. 警告事件

SDK 发现了一些非严重错误，一般不会导致播放停止，所以大多数 WARNING 事件您都无需关心，不过下面两个事件较为重要和有用：

PLAY_WARNING_VIDEO_PLAY_LAG

SDK 对外通知播放卡顿的事件信号，它指的是视频画面的卡顿（两帧画面的刷新时间）超过 500ms。

PLAY_WARNING_SERVER_DISCONNECT

播放请求被后台拒绝了，出现这个问题需要检查一下播放URL是否合法。

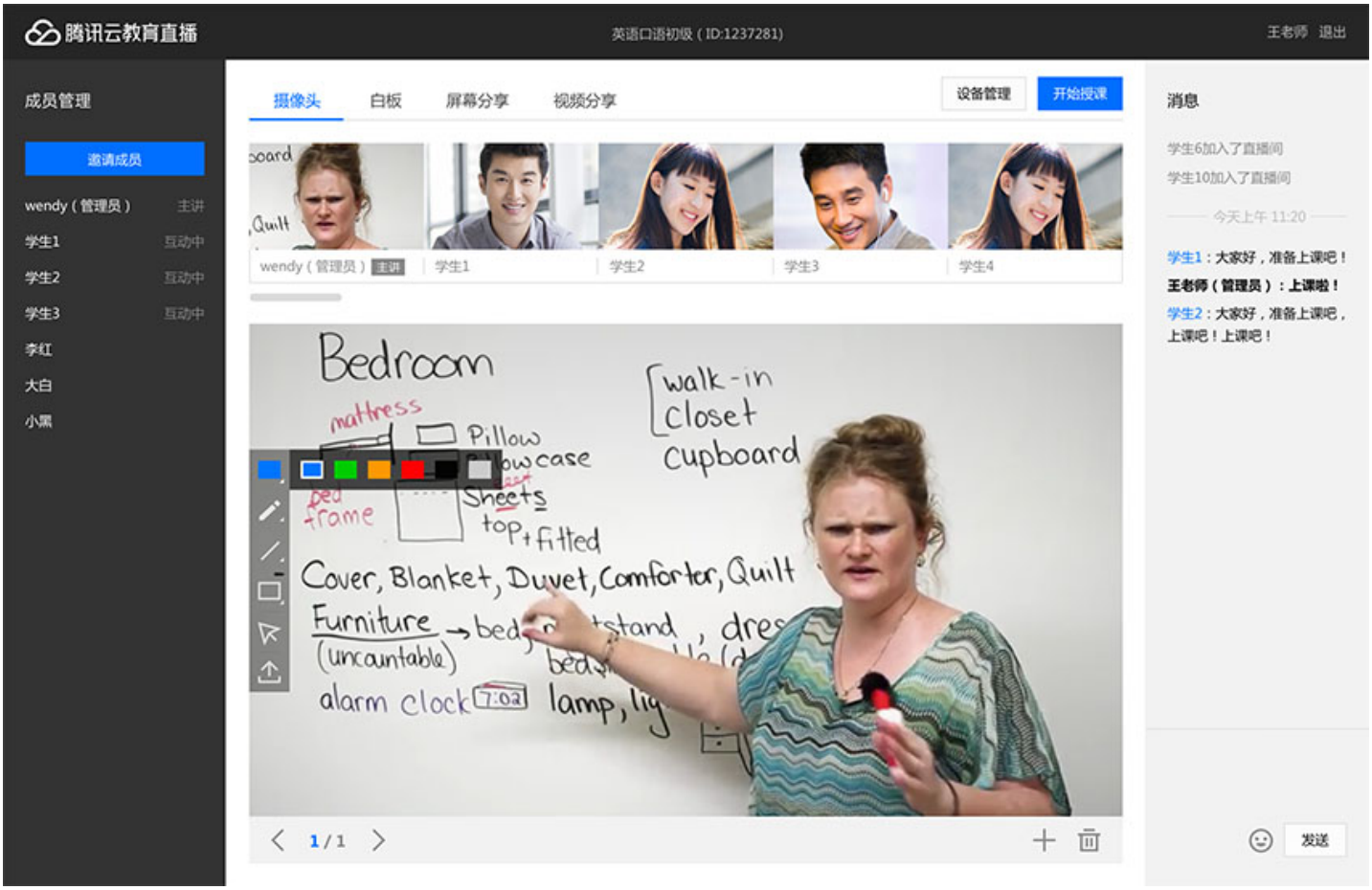
事件ID	数值	含义说明
PLAY_WARNING_VIDEO_DECODE_FAIL	2101	当前视频帧解码失败
PLAY_WARNING_AUDIO_DECODE_FAIL	2102	当前音频帧解码失败
PLAY_WARNING_RECONNECT	2103	网络断连, 已启动自动重连 (自动重连连续失败超过三次会放弃)
PLAY_WARNING_RECV_DATA_LAG	2104	网络来包不稳：可能是下行带宽不足，或由于主播端出流不均匀
PLAY_WARNING_VIDEO_PLAY_LAG	2105	当前视频播放出现卡顿（用户直观感受）
PLAY_WARNING_HW_ACCELERATION_FAIL	2106	硬解启动失败，采用软解（暂不支持）
PLAY_WARNING_VIDEO_DISCONTINUITY	2107	当前视频帧不连续，可能丢帧

事件ID	数值	含义说明
PLAY_WARNING_FIRST_IDR_HW_DECODE_FAIL	2108	当前流硬解第一个I帧失败，SDK自动切软解
PLAY_WARNING_DNS_FAIL	3001	RTMP -DNS解析失败
PLAY_WARNING_SEVER_CONN_FAIL	3002	RTMP服务器连接失败
PLAY_WARNING_SHAKE_FAIL	3003	RTMP服务器握手失败
PLAY_WARNING_SERVER_DISCONNECT	3004	RTMP服务器主动断开

实时连麦
直播连麦（LiveRoom）

最近更新时间：2018-07-11 11:18:50

直播+连麦 是在 **秀场直播** 和 **在线教育** 场景中经常使用的直播模式，它既能支持高并发和低成本的在线直播，又能通过连麦实现主播和观众之间的视频通话互动，具有极强的场景适用性。



腾讯云基于 **LiveRoom** 组件实现"直播 + 连麦"功能，它分成 Client 和 Server 两个部分（都是开源的），对接攻略请参考 DOC，本文档主要是详细列出了 Client 端的 API 列表：

> 在腾讯云官网 下载 SDK 开发包，在SDK\Rooms\LiveRoom中，包括LiveRoom相关的头文件和源码文件。

LiveRoom

名称	描述

名称	描述
instance()	获取LiveRoom单例，通过单例调用LiveRoom的接口
setCallback(ILiveRoomCallback * callback)	设置回调 LiveRoom 的回调代理，监听 LiveRoom 的内部状态和接口的执行结果
login(const std::string & serverDomain, const LRAuthData & authData, ILoginCallback* callback)	登录业务服务器RoomService，登录后才能够正常使用其他接口和使用IM功能
logout()	登出业务服务器RoomService，请注意在leaveRoom调用后，再调用logout，否则leaveRoom会调用失败
getRoomList(int index, int count, IGetRoomListCallback* callback)	获取房间列表，房间数量比较多时，可以分页获取
void getAudienceList(const std::string& roomID)	获取观众列表，只返回最近进入房间的 30 位观众
createRoom(const std::string& roomID, const std::string& roomInfo)	创建房间，后台的房间列表中会添加一个新房间，同时主播端会进入推流模式
enterRoom(const std::string& roomID)	进入房间
leaveRoom()	离开房间，如果是大主播，这个房间会被后台销毁，如果是小主播或者观众，不影响其他人继续观看
sendRoomTextMsg(const char * msg)	发送普通文本消息，比如直播场景中，发送弹幕
sendRoomCustomMsg(const char *cmd, const char* msg)	发送自定义消息，比如直播场景中，发送点赞、送花等消息
startLocalPreview(HWND rendHwnd, const RECT & rect)	启动默认的摄像头预览
updateLocalPreview(HWND rendHwnd, const RECT &rect)	重设摄像头预览区域，当您指定的本地 HWND 的窗口尺寸发生变化时，可以通过这个函数重新调整视频渲染区域
stopLocalPreview()	关闭摄像头预览
startScreenPreview(HWND rendHwnd, HWND captureHwnd, const RECT & renderRect, const RECT & captureRect)	启动屏幕分享
stopScreenPreview()	关闭屏幕分享

名称	描述
addRemoteView(HWND rendHwnd, const RECT & rect, const char * userID)	播放房间内其他主播的视频
removeRemoteView(const char * userID)	停止播放其他主播的视频
setMute(bool mute)	静音接口
setVideoQuality(LRVideoQuality quality, LRAspectRatio ratio)	设置画面质量预设选项
setBeautyStyle(TXEBeautyStyle beautyStyle, int beautyLevel, int whitenessLevel)	设置美颜和美白效果
requestJoinPusher()	观众端发起请求连麦
acceptJoinPusher(const std::string& userID)	大主播接受连麦请求，并通知给连麦发起方
rejectJoinPusher(const std::string& userID, const std::string& reason)	大主播拒绝连麦请求，并通知给连麦发起方
kickoutSubPusher(const std::string& userID)	大主播踢掉某一个小主播

ILoginCallback

名称	描述
onLogin(const LRResult& res, const LRAuthData& authData)	login登录结果的回调

IGetRoomListCallback

名称	描述
onGetRoomList(const LRResult& res, const std::vector& rooms)	获取房间列表的回调

ILiveRoomCallback

名称	描述

名称	描述
onCreateRoom(const LRResult& res, const std::string& roomID)	创建房间的回调
onEnterRoom(const LRResult& res)	进入房间的回调
onUpdateRoomData(const LRResult& res, const LRRoomData& roomData)	房间信息变更的回调
void onGetAudienceList(const LRResult& res, const std::vector& audiences)	查询观众列表的回调
onPusherJoin(const LRMemberData& member)	连麦观众进入房间的回调
void onPusherQuit(const LRMemberData& member)	连麦观众退出房间的回调
onRoomClosed(const std::string& roomID)	房间解散的回调
onRecvRoomTextMsg(const char *roomID, const char* userID, const char *userName, const char* userAvatar, const char * msg)	收到普通文本消息
onRecvRoomCustomMsg(const char *roomID, const char* userID, const char *userName, const char* userAvatar, const char *cmd, const char* message)	收到自定义消息
onError(const LRResult& res, const std::string& userID)	LiveRoom内部发生错误的通知
onRecvJoinPusherRequest(const std::string& roomID, const std::string& userID, const std::string& userName, const std::string& userAvatar)	接收到连麦请求
onRecvAcceptJoinPusher(const std::string& roomID, const std::string& msg)	接收到接受连麦请求的回复
onRecvRejectJoinPusher(const std::string& roomID, const std::string& msg)	接收到拒绝连麦请求的回复
onRecvKickoutSubPusher(const std::string& roomID)	接收大主播踢小主播的通知

LiveRoom对象接口的详情

1. instance

- 定义：static LiveRoom* instance()
- 说明：获取LiveRoom单例，通过单例调用LiveRoom的接口
- 示例：

```
LiveRoom* m_liveRoom = NULL;
m_liveRoom = LiveRoom::instance();
```

2. setCallback

- 定义：void setCallback(ILiveRoomCallback * callback)
- 说明：设置回调 LiveRoom 的回调代理，监听 LiveRoom 的内部状态和接口的执行结果
- 参数：

参数	类型	描述
callback	ILiveRoomCallback *	ILiveRoomCallback 类型的代理指针

- 示例：

```
class MainDialog : public ILiveRoomCallback
{
public:
MainDialog();
virtual ~MainDialog();

virtual void onCreateRoom(const LRResult& res, const std::string& roomID);
virtual void onEnterRoom(const LRResult& res);
virtual void onUpdateRoomData(const LRResult& res, const LRRoomData& roomData);
...
};

MainDialog::MainDialog()
{
m_liveRoom->setCallback(this); // 设置回调
}

...
```

3. login

- 定义：void login(const std::string & serverDomain, const LRAuthData & authData, ILoginCallback* callback)
- 说明：登录业务服务器RoomService，登录后才能够正常使用其他接口和使用IM功能
- 参数：

参数	类型	描述
serverDomain	const std::string &	RoomService的URL地址，安全起见，建议访问https加密链接， 参考 DOC
authData	const LRAuthData &	RoomService提供的登录信息，包括IM相关的配置字段，在login成功后，获取到token字段，参考 DOC
callback	ILoginCallback*	ILoginCallback 类型的代理指针，回调login的结果

- 示例：

```
std::string serverDomain = "https://roomtest.qcloud.com/weapp/live_room";

m_authData.sdkAppID = sdkAppID;
m_authData.accountType = accountType;
m_authData.userID = userID;
m_authData.userSig = userSig;
m_authData.userName = userName;
m_authData.userAvatar = userAvatar;
m_authData.token = token;

// 该类实现ILoginCallback接口，获取login结果
m_liveRoom->login(serverDomain, m_authData, this);
```

4. logout

- 定义：void logout();

- 说明：登出业务服务器RoomService，请注意在leaveRoom调用后，再调用logout，否则leaveRoom会调用失败
- 示例：

```
m_liveRoom->logout();
```

5. getRoomList

- 定义：void getRoomList(int index, int count, IGetRoomListCallback* callback)
- 说明：获取房间列表，房间数量比较多时，可以分页获取
- 参数：

参数	类型	描述
index	int	分页获取，初始默认可设置为0，后续获取为起始房间索引（如第一次设置index=0, cnt=5,获取第二页时可用index=5）
count	int	每次调用，最多返回房间个数；0表示所有满足条件的房间
callback	IGetRoomListCallback*	IGetRoomListCallback 类型的代理指针，查询结果的回调

- 示例：

```
// 此处从index=0开始，最多查询20个房间，该类实现IGetRoomListCallback接口，获取查询结果
m_liveRoom->getRoomList(0, 20, this);
```

6. getAudienceList

- 定义：void getAudienceList(const std::string& roomID)
- 说明：获取观众列表，只返回最近进入房间的 30 位观众
- 参数：

参数	类型	描述
roomID	const std::string&	房间ID，在 getRoomList 接口房间列表中查询得到

- 示例：

```
m_liveRoom->getRoomList(roomID);
```

7. createRoom

- 定义：void createRoom(const std::string& roomID, const std::string& roomInfo)
- 说明：创建房间，后台的房间列表中会添加一个新房间，同时主播端会进入推流模式
- 参数：

参数	类型	描述
roomID	const std::string&	房间ID，若传入空字符串，则后台会为您分配roomID，否则，传入的roomID作为这个房间的ID
roomInfo	const std::string&	自定义数据，该字段包含在房间信息中，推荐您将 roomInfo 定义为 json 格式，这样可以有很强的扩展性

- 示例：

```
// 后台会为您分配roomID
m_liveRoom->createRoom("", "{\"roomName\": \"My first room\"}");
```

8. enterRoom

- 定义：void enterRoom(const std::string& roomID)
- 说明：进入房间
- 参数：

参数	类型	描述
roomID	const std::string&	roomID - 要进入的房间ID，在 getRoomList 接口房间列表中查询得到

- 示例：

```
m_liveRoom->enterRoom(roomID);
```

9. leaveRoom

- 定义：void leaveRoom()
- 说明：离开房间，如果是大主播，这个房间会被后台销毁，如果是小主播或者观众，不影响其他人继续观看
- 示例：

```
m_liveRoom->leaveRoom();
```

10. sendRoomTextMsg

- 定义：void sendRoomTextMsg(const char * msg)
- 说明：发送普通文本消息，比如直播场景中，发送弹幕
- 参数：

参数	类型	描述
msg	const char *	文本消息

- 示例：

```
m_liveRoom->sendRoomTextMsg("Hello LiveRoom");
```

11. sendRoomCustomMsg

- 定义：void sendRoomCustomMsg(const char *cmd, const char* msg)
- 说明：发送自定义消息，比如直播场景中，发送点赞、送花等消息
- 参数：

参数	类型	描述
cmd	const char *	自定义cmd，收发双方协商好的cmd
msg	const char *	自定义消息

- 示例：

```
// 假设收发双方协商好的cmd有Smile, Anger
m_liveRoom->sendRoomCustomMsg("Smile", "I am hungry");
```

12. startLocalPreview

- 定义：void startLocalPreview(HWND rendHwnd, const RECT & rect)
- 说明：启动默认的摄像头预览
- 参数：

参数	类型	描述
rendHwnd	HWND	承载预览画面的 HWND，目前 SDK 是采用 OpenGL 向 HWND 上绘制图像的,rendHwnd = null时无需预览视频
rect	const RECT &	指定视频图像在 HWND 上的渲染区域

- 示例：

```
RECT rect = { 0 };
::GetClientRect(hwnd, &rect); // 在hwnd整个窗口中渲染
m_liveRoom->startLocalPreview(hwnd, rect);
```

13. updateLocalPreview

- 定义：void updateLocalPreview(HWND rendHwnd, const RECT &rect)
- 说明：重设摄像头预览区域，当您指定的本地 HWND 的窗口尺寸发生变化时，可以通过这个函数重新调整视频渲染区域

- 参数：

参数	类型	描述
rendHwnd	HWND	承载预览画面的 HWND，目前 SDK 是采用 OpenGL 向 HWND 上绘制图像的,rendHwnd = NULL时无需预览视频
rect	const RECT &	指定视频图像在 HWND 上的渲染区域

- 示例：

```
RECT rect = { 0 };
::GetClientRect(hwnd, &rect); // 在hwnd整个窗口中渲染
m_liveRoom->updateLocalPreview(hwnd, rect);
```

14. stopLocalPreview

- 定义：void stopLocalPreview()
- 说明：关闭摄像头预览
- 示例：

```
m_liveRoom->stopLocalPreview();
```

15. startScreenPreview

- 定义：bool startScreenPreview(HWND rendHwnd, HWND captureHwnd, const RECT & renderRect, const RECT & captureRect)
- 说明：启动屏幕分享
- 参数：

参数	类型	描述
rendHwnd	HWND	承载预览画面的 HWND，目前 SDK 是采用 OpenGL 向 HWND 上绘制图像的,rendHwnd = NULL时无需预览视频

参数	类型	描述
captureHwnd	HWND	指定录取窗口，若为NULL，则 captureRect 不起效，并且录取整个屏幕；若不为NULL，则录取这个窗口的画面
renderRect	const RECT &	指定视频图像在 rendHwnd 上的渲染区域
captureRect	const RECT &	指定录取窗口客户区的区域

- 返回值：成功 or 失败
- 示例：

```
RECT renderRect = { 0 };
::GetClientRect(rendHwnd, &renderRect); // 在hwnd整个窗口中渲染

RECT captureRect = { 0 };
::GetClientRect(captureHwnd, &captureRect); // 录取整个窗口的画面

m_liveRoom->startScreenPreview(rendHwnd, captureHwnd, renderRect, captureRect);
```

16. stopScreenPreview

- 定义：void stopScreenPreview()
- 说明：关闭屏幕分享
- 示例：

```
m_liveRoom->stopScreenPreview();
```

17. addRemoteView

- 定义：void addRemoteView(HWND rendHwnd, const RECT & rect, const char * userID)
- 说明：播放房间内其他主播的视频
- 参数：

参数	类型	描述
rendHwnd	HWND	承载预览画面的 HWND
rect	const RECT &	指定视频图像在 HWND 上的渲染区域
userID	const char *	用户ID

- 示例：

```
RECT rect = { 0 };
::GetClientRect(hwnd, &rect); // 在hwnd整个窗口中渲染

// 打开用户的音视频播放
m_liveRoom->addRemoteView(hwnd ， rect, userID);
```

18. removeRemoteView

- 定义：void removeRemoteView(const char * userID)
- 说明：停止播放其他主播的视频
- 参数：

参数	类型	描述
userID	const char *	用户ID

- 示例：

```
m_liveRoom->removeRemoteView(userID);
```

19. setMute

- 定义：void setMute(bool mute)
- 说明：静音接口
- 参数：

参数	类型	描述
mute	bool	是否静音

- 示例：

```
m_liveRoom->setMute(true); // 设置为静音
```

20. setVideoQuality

- 定义：void setVideoQuality(LRVideoQuality quality, LRAspectRatio ratio)
- 说明：设置推流的画面质量选项
- 参数：

参数	类型	描述
quality	LRVideoQuality	画质，参考 LiveRoomUtil.h 中定义的 LRVideoQuality 枚举值
ratio	LRAspectRatio	宽高比，参考 LiveRoomUtil.h 中定义的 LRAspectRatio 枚举值

- 示例：

```
// 设置画质为高清，宽高比为4:3
m_liveRoom->setVideoQuality(LIVEROOM_VIDEO_QUALITY_HIGH_DEFINITION, LIVEROOM_ASPECT_RATIO_4_3);
```

21. setBeautyStyle

- 定义：void setBeautyStyle(TXEBeautyStyle beautyStyle, int beautyLevel, int whitenessLevel)
- 说明：设置美颜和美白效果
- 参数：

参数	类型	描述

参数	类型	描述
beautyStyle	TXEBeautyStyle	参考 TXLiveSDKTypeDef.h 中定义的 TXEBeautyStyle 枚举值
beautyLevel	int	美颜级别取值范围 1 ~ 9； 0 表示关闭，1 ~ 9值越大，效果越明显
whitenessLevel	int	美白级别取值范围 1 ~ 9； 0 表示关闭，1 ~ 9值越大，效果越明显

- 示例：

```
// 自然，美颜度数5，美白度数5
m_liveRoom->setBeautyStyle(TXE_BEAUTY_STYLE_NATURE, 5, 5);
```

22. requestJoinPusher

- 定义：void requestJoinPusher()
- 说明：观众端发起请求连麦
- 示例：

```
m_liveRoom->requestJoinPusher();
```

23. acceptJoinPusher

- 定义：void acceptJoinPusher(const std::string& userID)
- 说明：大主播接受连麦请求，并通知给连麦发起方
- 参数：

参数	类型	描述
userID	const std::string&	用户ID

- 示例：

```
m_liveRoom->acceptJoinPusher(userID);
```

24. rejectJoinPusher

- 定义：void rejectJoinPusher(const std::string& userID, const std::string& reason)
- 说明：大主播拒绝连麦请求，并通知给连麦发起方
- 参数：

参数	类型	描述
userID	const std::string&	用户ID
reason	const std::string&	拒绝的原因

- 示例：

```
m_liveRoom->acceptJoinPusher(userID, reason);
```

25. kickoutSubPusher

- 定义：void kickoutSubPusher(const std::string& userID)
- 说明：大主播踢掉某一个小主播
- 参数：

参数	类型	描述
userID	const std::string&	用户ID

- 示例：

```
m_liveRoom->kickoutSubPusher(userID);
```

ILoginCallback回调接口的详情

1. onLogin

- 定义：virtual void onLogin(const LRResult& res, const LRAuthData& authData)
- 说明：login登录结果的回调
- 参数：

参数	类型	描述
res	const LRResult&	执行结果，参考 LiveRoomUtil.h 中定义的 LRResult 结构体
authData	const LRAuthData&	RoomService提供的登录信息，包括IM相关的配置字段，在login成功后，获取到token字段

- 示例：

```
class LoginDialog : public ILoginCallback
{
public:
virtual void onLogin(const LRResult& res, const LRAuthData& authData)
{
if (LIVEROOM_SUCCESS != res.ec)
{
// 登录失败
}
else
{
// 登录成功，保存authData信息
}
}
};

// 参见LiveRoom::login接口，了解回调的使用
...
```

IGetRoomListCallback回调接口的详情

1. onGetRoomList

- 定义：virtual void onGetRoomList(const LRResult& res, const std::vector& rooms)

- 说明：获取房间列表的回调
- 参数：

参数	类型	描述
res	const LRResult&	执行结果，参考 LiveRoomUtil.h 中定义的 LRResult 结构体
rooms	const std::vector&	房间信息的列表

- 示例：

```
class RoomListDialog : public IGetRoomListCallback
{
public:
virtual void onGetRoomList(const LRResult& res, const std::vector<LRRoomData>& rooms)
{
if (LIVEROOM_SUCCESS != res.ec)
{
// 查询失败
}
else
{
// 查询成功，处理rooms数据
}
}
};

// 参见LiveRoom::getRoomList接口，了解回调的使用
...
```

ILiveRoomCallback回调接口的详情

参见LiveRoom::setCallback接口，了解如何设置ILiveRoomCallback回调。

1. onCreateRoom

- 定义：virtual void onCreateRoom(const LRResult& res, const std::string& roomID)
- 说明：创建房间的回调

- 参数：

参数	类型	描述
res	const LRResult&	执行结果，参考 LiveRoomUtil.h 中定义的 LRResult 结构体
roomID	const std::string&	房间ID

- 示例：

```
class MainDialog : public ILiveRoomCallback
{
public:
virtual void onCreateRoom(const LRResult& res, const std::string& roomID)
{
if (LIVEROOM_SUCCESS != res.ec)
{
// 创建房间失败，弹框提示
}
else
{
// 创建房间成功，其他观众可以观看直播
// 紧接着处理onUpdateRoomData回调
}
}

...
};
```

2. onEnterRoom

- 定义：virtual void onEnterRoom(const LRResult& res)
- 说明：进入房间的回调
- 参数：

参数	类型	描述
res	const LRResult&	执行结果，参考 LiveRoomUtil.h 中定义的 LRResult 结构体

- 示例：

```
class MainDialog : public ILiveRoomCallback
{
public:
virtual void onEnterRoom(const LRResult& res)
{
if (LIVEROOM_SUCCESS != res.ec)
{
// 进入房间失败，弹框提示
}
else
{
// 进入房间成功，紧接着处理onUpdateRoomData回调
}
}

...
};
```

3. onUpdateRoomData

- 定义：virtual void onUpdateRoomData(const LRResult& res, const LRRoomData& roomData)
- 说明：房间信息变更的回调
- 参数：

参数	类型	描述
res	const LRResult&	执行结果，参考 LiveRoomUtil.h 中定义的 LRResult 结构体
roomData	const LRRoomData&	房间信息，参考 LiveRoomUtil.h 中定义的 LRRoomData 结构体

- 示例：

```
class MainDialog : public ILiveRoomCallback
{
public:
virtual void onUpdateRoomData(const LRResult& res, const LRRoomData& roomData)
{
if (LIVEROOM_SUCCESS != res.ec)
```

```
{
// 更新失败，弹框提示
}
else
{
// 更新成功，根据createRoom还是enterRoom导致这个接口的触发来做出相应的处理
}
}

...
};
```

4. onGetAudienceList

- 定义：virtual void onGetAudienceList(const LRResult& res, const std::vector& audiences)
- 说明：查询观众列表的回调
- 参数：

参数	类型	描述
res	const LRResult&	执行结果，参考 LiveRoomUtil.h 中定义的 LRResult 结构体
audiences	const std::vector&	观众信息的列表

- 示例：

```
class MainDialog : public ILiveRoomCallback
{
public:
virtual void onGetAudienceList(const LRResult& res, const std::vector<LRAudienceData>& audience
s)
{
if (LIVEROOM_SUCCESS != res.ec)
{
// 查询失败，弹框提示
}
else
{
// 查询成功，列表更新到UI
}
}
```

```
...
};
```

5. onPusherJoin

- 定义：virtual void onPusherJoin(const LRMemberData& member)
- 说明：连麦观众进入房间的回调
- 参数：

参数	类型	描述
member	const LRMemberData&	成员信息，参考 LiveRoomUtil.h 中定义的 LRMemberData 结构体

- 示例：

```
class MainDialog : public ILiveRoomCallback
{
public:
virtual void onPusherJoin(const LRMemberData& member)
{
// 通常连麦观众请求连麦，打开连麦观众的画面和音频
m_liveRoom->addRemoteView(rendHwnd, rect, member.userID);
...
}

...
};
```

6. onPusherQuit

- 定义：virtual void onPusherQuit(const LRMemberData& member)
- 说明：连麦观众退出房间的回调
- 参数：

参数	类型	描述

参数	类型	描述
member	const LRMemberData&	成员信息，参考 LiveRoomUtil.h 中定义的 LRMemberData 结构体

- 示例：

```
class MainDialog : public ILiveRoomCallback
{
public:
virtual void onPusherQuit(const LRMemberData& member)
{
// 通常连麦观众关闭连麦，关闭连麦观众的画面和音频
m_liveRoom->removeRemoteView(member.userID);
...
}

...
};
```

7. onRoomClosed

- 定义：virtual void onRoomClosed(const std::string& roomID)
- 说明：房间解散的回调
- 参数：

参数	类型	描述
roomID	const std::string&	房间ID

- 示例：

```
class MainDialog : public ILiveRoomCallback
{
public:
virtual void onRoomClosed(const std::string& roomID)
{
// 提示房间解散，关闭已打开的音视频
...
}
```

```
...
};
```

8. onRecvRoomTextMsg

- 定义：virtual void onRecvRoomTextMsg(const char *roomID, const char* userID, const char *userName, const char* userAvatar, const char * msg)
- 说明：收到普通文本消息
- 参数：

参数	类型	描述
roomID	const char *	房间ID
userID	const char *	发送者ID
userName	const char *	发送者昵称
userAvatar	const char *	发送者头像
message	const char *	文本消息内容

- 示例：

```
class MainDialog : public ILiveRoomCallback
{
public:
virtual void onRecvRoomTextMsg(const char * roomID, const char * userID, const char * userName, const char * userAvatar, const char * msg)
{
// IM面板显示文本消息
}

...
};
```

9. onRecvRoomCustomMsg

- 定义：virtual void onRecvRoomCustomMsg(const char *roomID, const char* userID, const char *userName, const char* userAvatar, const char *cmd, const char* message)
- 说明：收到自定义消息
- 参数：

参数	类型	描述
roomID	const char *	房间ID
userID	const char *	发送者ID
userName	const char *	发送者昵称
userAvatar	const char *	发送者头像
cmd	const char *	自定义cmd
message	const char *	自定义消息内容

- 示例：

```
class MainDialog : public ILiveRoomCallback
{
public:
virtual void onRecvRoomCustomMsg(const char * roomID, const char * userID, const char * user
Name, const char * userAvatar, const char * cmd, const char * message)
{
// 解析和处理自定义消息，例如点赞、礼物等
}

...
};
```

10. onError

- 定义：virtual void onError(const LRResult& res, const std::string& userID)
- 说明：LiveRoom内部发生错误的通知
- 参数：

参数	类型	描述
res	const LRResult&	执行结果，参考 LiveRoomUtil.h 中定义的 LRResult 结构体
userID	const std::string&	用户ID

- 示例：

```
class MainDialog : public ILiveRoomCallback
{
public:
virtual void onError(const LRResult& res, const std::string& userID)
{
// 弹框提示错误，根据错误严重程序，是否关闭直播
}

...
};
```

11. onRecvJoinPusherRequest

- 定义：virtual void onRecvJoinPusherRequest(const std::string& roomID, const std::string& userID, const std::string& userName, const std::string& userAvatar)
- 说明：接收到连麦请求
- 参数：

参数	类型	描述
roomID	const std::string&	房间ID
userID	const std::string&	发送者ID
userName	const std::string&	发送者昵称
userAvatar	const std::string&	发送者头像

- 示例：

```
class MainDialog : public ILiveRoomCallback
{
public:
virtual void onRecvJoinPusherRequest(const std::string& roomID, const std::string& userID, const std::string& userName, const std::string& userAvatar)
{
// 大主播接收到观众的连麦请求
// 先判断roomID是否合法
// UI显示谁发起请求的提示
// 大主播可以通过acceptJoinPusher接受连麦，或者rejectJoinPusher拒绝连麦
}

...
};
```

12. onRecvAcceptJoinPusher

- 定义：virtual void onRecvAcceptJoinPusher(const std::string& roomID, const std::string& msg)
- 说明：接收到接受连麦请求的回复
- 参数：

参数	类型	描述
roomID	const std::string&	房间ID
msg	const std::string&	消息

- 示例：

```
class MainDialog : public ILiveRoomCallback
{
public:
virtual void onRecvAcceptJoinPusher(const std::string& roomID, const std::string& msg)
{
// 连麦观众接收到大主播的接收连麦的回复
// 先判断roomID是否合法
// 处理msg
}
```

```
...
};
```

13. onRecvRejectJoinPusher

- 定义：virtual void onRecvRejectJoinPusher(const std::string& roomID, const std::string& msg)
- 说明：接收到拒绝连麦请求的回复
- 参数：

参数	类型	描述
roomID	const std::string&	房间ID
msg	const std::string&	消息

- 示例：

```
class MainDialog : public ILiveRoomCallback
{
public:
virtual void onRecvRejectJoinPusher(const std::string& roomID, const std::string& msg)
{
// 连麦观众接收到大主播的拒绝连麦的回复
// 先判断roomID是否合法
// 处理msg
}

...
};
```

14. onRecvKickoutSubPusher

- 定义：virtual void onRecvKickoutSubPusher(const std::string& roomID)

- 说明：接收大主播踢小主播的通知

- 参数：

参数	类型	描述
roomID	const std::string&	房间ID

- 示例：

```
class MainDialog : public ILiveRoomCallback
{
public:
virtual void onRecvRejectJoinPusher(const std::string& roomID, const std::string& msg)
{
// 大主播踢掉某个正在连麦的观众
// 先判断roomID是否合法
// 处理msg
}

...
};
```

视频通话（RTCRoom）

最近更新时间：2018-07-11 11:19:35

实时音视频（RTCRoom ）接口（C++）

RTCRoom

名称	描述
instance()	获取RTCRoom单例，通过单例调用RTCRoom的接口
setCallback(IRTCRoomCallback * callback)	设置回调 RTCRoom 的回调代理，监听 RTCRoom 的内部状态和接口的执行结果
login(const std::string & serverDomain, const RTCAuthData & authData, ILoginRTCCallback* callback)	登录业务服务器RoomService，登录后才能够正常使用其他接口和使用IM功能
logout()	登出业务服务器RoomService，请注意在leaveRoom调用后，再调用logout，否则leaveRoom会调用失败
getRoomList(int index, int count, IGetRTCRoomListCallback* callback)	获取房间列表，房间数量比较多时，可以分页获取
createRoom(const std::string& roomID, const std::string& roomInfo)	创建房间，后台的房间列表中会添加一个新房间，同时会议创建者端会进入推流模式
enterRoom(const std::string& roomID, HWND rendHwnd, const RECT & rect)	进入房间
leaveRoom()	离开房间，如果是会议创建者，这个房间会被后台销毁，如果是会议参与者，不影响其他人继续通话
sendRoomTextMsg(const char * msg)	发送普通文本消息
sendRoomCustomMsg(const char *cmd, const char* msg)	发送自定义消息
startLocalPreview(HWND rendHwnd, const RECT & rect)	启动默认的摄像头预览

名称	描述
updateLocalPreview(HWND rendHwnd, const RECT &rect)	重设摄像头预览区域，当您指定的本地 HWND 的窗口尺寸发生变化时，可以通过这个函数重新调整视频渲染区域
stopLocalPreview()	关闭摄像头预览
startScreenPreview(HWND rendHwnd, HWND captureHwnd, const RECT & renderRect, const RECT & captureRect)	启动屏幕分享
stopScreenPreview()	关闭屏幕分享
addRemoteView(HWND rendHwnd, const RECT & rect, const char * userID)	播放房间内其他会议参与者的视频
updateRemotePreview(HWND rendHwnd, const RECT &rect, const char * userID)	重设指定userID的视频预览区域，当您指定的本地 HWND 的窗口尺寸发生变化时，可以通过这个函数重新调整视频渲染区域
removeRemoteView(const char * userID)	停止播放其他会议参与者的视频
setMute(bool mute)	静音接口
setBeautyStyle(TXEBeautyStyle beautyStyle, int beautyLevel, int whitenessLevel)	设置美颜和美白效果

ILoginRTCCallback

名称	描述
onLogin(const RTCResult& res, const RTCAuthData& authData)	login登录结果的回调

IGetRTCRoomListCallback

名称	描述
onGetRoomList(const RTCResult& res, const std::vector& rooms)	获取房间列表的回调

IRTCRoomCallback

名称	描述
onCreateRoom(const RTCResult& res, const std::string& roomID)	创建房间的回调
onEnterRoom(const RTCResult& res)	进入房间的回调
onUpdateRoomData(const RTCResult& res, const RTCRoomData& roomData)	房间信息变更的回调
onPusherJoin(const RTCMemberData& member)	成员进入房间的回调
void onPusherQuit(const RTCMemberData& member)	成员退出房间的回调
onRoomClosed(const std::string& roomID)	房间解散的回调
onRecvRoomTextMsg(const char *roomID, const char* userID, const char *userName, const char* userAvatar, const char * msg)	收到普通文本消息
onRecvRoomCustomMsg(const char *roomID, const char* userID, const char *userName, const char* userAvatar, const char *cmd, const char* message)	收到自定义消息
onError(const RTCResult& res, const std::string& userID)	RTCRoom内部发生错误的通知

RTCRoom对象接口的详情

1. instance

- 定义：static RTCRoom* instance()
- 说明：获取RTCRoom单例，通过单例调用RTCRoom的接口
- 示例：

```
RTCRoom* m_RTCRoom = NULL;
m_RTCRoom = RTCRoom::instance();
```

2. setCallback

- 定义：void setCallback(IRTCRoomCallback * callback)
- 说明：设置回调 RTCRoom 的回调代理，监听 RTCRoom 的内部状态和接口的执行结果
- 参数：

参数	类型	描述
callback	IRTCRoomCallback *	IRTCRoomCallback 类型的代理指针

- 示例：

```
class MainDialog : public IRTCRoomCallback
{
public:
MainDialog();
virtual ~MainDialog();

virtual void onCreateRoom(const RTCResult& res, const std::string& roomID);
virtual void onEnterRoom(const RTCResult& res);
virtual void onUpdateRoomData(const RTCResult& res, const LRRoomData& roomData);
...
};

MainDialog::MainDialog()
{
m_RTCRoom->setCallback(this); // 设置回调
}

...
```

3. login

- 定义：void login(const std::string & serverDomain, const RTCAuthData & authData, ILoginRTCCallback* callback)
- 说明：登录业务服务器RoomService，登录后才能够正常使用其他接口和使用IM功能
- 参数：

参数	类型	描述
serverDomain	const std::string &	RoomService的URL地址，安全起见，建议访问https加密链接，参考 DOC
authData	const RTCAuthData &	RoomService提供的登录信息，包括IM相关的配置字段，在login成功后，获取到token字段，参考 DOC
callback	ILoginRTCCallback*	ILoginRTCCallback 类型的代理指针，回调login的结果

- 示例：

```
std::string serverDomain = "https://roomtest.qcloud.com/weapp/double_room";

m_authData.sdkAppID = sdkAppID;
m_authData.accountType = accountType;
m_authData.userID = userID;
m_authData.userSig = userSig;
m_authData.userName = userName;
m_authData.userAvatar = userAvatar;
m_authData.token = token;

// 该类实现ILoginCallback接口，获取login结果
m_RTCRoom->login(serverDomain, m_authData, this);
```

4. logout

- 定义：void logout();
- 说明：登出业务服务器RoomService，请注意在leaveRoom调用后，再调用logout，否则leaveRoom会调用失败
- 示例：

```
m_RTCRoom->logout();
```

5. getRoomList

- 定义：void getRoomList(int index, int count, IGetRTCRoomListCallback* callback)

- 说明：获取房间列表，房间数量比较多时，可以分页获取
- 参数：

参数	类型	描述
index	int	分页获取，初始默认可设置为0，后续获取为起始房间索引（如第一次设置index=0, cnt=5,获取第二页时可用index=5）
count	int	每次调用，最多返回房间个数；0表示所有满足条件的房间
callback	IGetRTCRoomListCallback*	IGetRTCRoomListCallback 类型的代理指针，查询结果的回调

- 示例：

```
// 此处从index=0开始，最多查询20个房间，该类实现IGetRoomListCallback接口，获取查询结果
m_RTCRoom->getRoomList(0, 20, this);
```

6. createRoom

- 定义：void createRoom(const std::string& roomID, const std::string& roomInfo)
- 说明：创建房间，后台的房间列表中会添加一个新房间，同时会议创建者会进入推流模式
- 参数：

参数	类型	描述
roomID	const std::string&	房间ID，若传入空字符串，则后台会为您分配roomID，否则，传入的roomID作为这个房间的ID
roomInfo	const std::string&	自定义数据，该字段包含在房间信息中，推荐您将 roomInfo 定义为 json 格式，这样可以有很强的扩展性

- 示例：

```
// 后台会为您分配roomID
m_RTCRoom->createRoom("", "{\"roomName\": \"My first room\"}");
```

7. enterRoom

- 定义：void enterRoom(const std::string& roomID)
- 说明：进入房间
- 参数：

参数	类型	描述
roomID	const std::string&	roomID - 要进入的房间ID，在 getRoomList 接口房间列表中查询得到

- 示例：

```
m_RTCRoom->enterRoom(roomID);
```

8. leaveRoom

- 定义：void leaveRoom()
- 说明：离开房间，如果是会议创建者，这个房间会被后台销毁，如果是会议参与者，不影响其他人继续通话
- 示例：

```
m_RTCRoom->leaveRoom();
```

9. sendRoomTextMsg

- 定义：void sendRoomTextMsg(const char * msg)
- 说明：发送普通文本消息
- 参数：

参数	类型	描述
msg	const char *	文本消息

- 示例：

```
m_RTCRoom->sendRoomTextMsg("Hello RTCRoom");
```

10. sendRoomCustomMsg

- 定义：void sendRoomCustomMsg(const char *cmd, const char* msg)
- 说明：发送自定义消息
- 参数：

参数	类型	描述
cmd	const char *	自定义cmd，收发双方协商好的cmd
msg	const char *	自定义消息

- 示例：

```
// 假设收发双方协商好的cmd有Smile, Anger
m_RTCRoom->sendRoomCustomMsg("Smile", "I am hungry");
```

11. startLocalPreview

- 定义：void startLocalPreview(HWND rendHwnd, const RECT & rect)
- 说明：启动默认的摄像头预览
- 参数：

参数	类型	描述
rendHwnd	HWND	承载预览画面的 HWND，目前 SDK 是采用 OpenGL 向 HWND 上绘制图像的,rendHwnd = null时无需预览视频
rect	const RECT &	指定视频图像在 HWND 上的渲染区域

- 示例：

```
RECT rect = { 0 };
::GetClientRect(hwnd, &rect); // 在hwnd整个窗口中渲染
m_RTCRoom->startLocalPreview(hwnd, rect);
```

12. updateLocalPreview

- 定义：void updateLocalPreview(HWND rendHwnd, const RECT &rect)
- 说明：重设摄像头预览区域，当您指定的本地 HWND 的窗口尺寸发生变化时，可以通过这个函数重新调整视频渲染区域
- 参数：

参数	类型	描述
rendHwnd	HWND	承载预览画面的 HWND，目前 SDK 是采用 OpenGL 向 HWND 上绘制图像的,rendHwnd = NULL时无需预览视频
rect	const RECT &	指定视频图像在 HWND 上的渲染区域

- 示例：

```
RECT rect = { 0 };
::GetClientRect(hwnd, &rect); // 在hwnd整个窗口中渲染
m_RTCRoom->updateLocalPreview(hwnd, rect);
```

13. stopLocalPreview

- 定义：void stopLocalPreview()
- 说明：关闭摄像头预览
- 示例：

```
m_RTCRoom->stopLocalPreview();
```

14. startScreenPreview

- 定义：bool startScreenPreview(HWND rendHwnd, HWND captureHwnd, const RECT & renderRect, const RECT & captureRect)
- 说明：启动屏幕分享
- 参数：

参数	类型	描述
rendHwnd	HWND	承载预览画面的 HWND，目前 SDK 是采用 OpenGL 向 HWND 上绘制图像的,rendHwnd = NULL时无需预览视频
captureHwnd	HWND	指定录取窗口，若为NULL，则 captureRect 不起效，并且录取整个屏幕；若不为NULL，则录取这个窗口的画面
renderRect	const RECT &	指定视频图像在 rendHwnd 上的渲染区域
captureRect	const RECT &	指定录取窗口客户区的区域

- 返回值：成功 or 失败
- 示例：

```
RECT renderRect = { 0 };
::GetClientRect(rendHwnd, &renderRect); // 在hwnd整个窗口中渲染

RECT captureRect = { 0 };
::GetClientRect(captureHwnd, &captureRect); // 录取整个窗口的画面

m_RTCRoom->startScreenPreview(rendHwnd, captureHwnd, renderRect, captureRect);
```

15. stopScreenPreview

- 定义：void stopScreenPreview()
- 说明：关闭屏幕分享
- 示例：

```
m_RTCRoom->stopScreenPreview();
```

16. addRemoteView

- 定义：void addRemoteView(HWND rendHwnd, const RECT & rect, const char * userID)
- 说明：播放房间内其他会议参与者的视频
- 参数：

参数	类型	描述
rendHwnd	HWND	承载预览画面的 HWND
rect	const RECT &	指定视频图像在 HWND 上的渲染区域
userID	const char *	用户ID

- 示例：

```
RECT rect = { 0 };
::GetClientRect(hwnd, &rect); // 在hwnd整个窗口中渲染

// 打开用户的音视频播放
m_RTCRoom->addRemoteView(hwnd， rect, userID);
```

17. updateRemotePreview

- 定义：void updateRemotePreview(HWND rendHwnd, const RECT &rect, const char * userID)
- 说明：重设指定userID的视频预览区域，当您指定的本地 HWND 的窗口尺寸发生变化时，可以通过这个函数重新调整视频渲染区域
- 参数：

参数	类型	描述
rendHwnd	HWND	承载预览画面的 HWND
rect	const RECT &	指定视频图像在 HWND 上的渲染区域
userID	const char *	用户ID

- 示例：

```
RECT rect = { 0 };
::GetClientRect(hwnd, &rect); // 在hwnd整个窗口中渲染

// 更新用户的音视频播放
m_RTCRoom->updateRemotePreview(hwnd ， rect, userID);
```

18. removeRemoteView

- 定义：void removeRemoteView(const char * userID)
- 说明：停止播放其他会议参与者的视频
- 参数：

参数	类型	描述
userID	const char *	用户ID

- 示例：

```
m_RTCRoom->removeRemoteView(userID);
```

19. setMute

- 定义：void setMute(bool mute)
- 说明：静音接口
- 参数：

参数	类型	描述
mute	bool	是否静音

- 示例：

```
m_RTCRoom->setMute(true); // 设置为静音
```

20. setBeautyStyle

- 定义：void setBeautyStyle(TXEBeautyStyle beautyStyle, int beautyLevel, int whitenessLevel)
- 说明：设置美颜和美白效果
- 参数：

参数	类型	描述
beautyStyle	TXEBeautyStyle	参考 RTCRoomUtil.h 中定义的 RTCBeautyStyle 枚举值
beautyLevel	int	美颜级别取值范围 1 ~ 9； 0 表示关闭，1 ~ 9值越大，效果越明显
whitenessLevel	int	美白级别取值范围 1 ~ 9； 0 表示关闭，1 ~ 9值越大，效果越明显

- 示例：

```
// 自然，美颜度数5，美白度数5
m_RTCRoom->setBeautyStyle(TXE_BEAUTY_STYLE_NATURE, 5, 5);
```

ILoginRTCCallback回调接口的详情

1. onLogin

- 定义：virtual void onLogin(const RTCResult& res, const RTCAuthData& authData)
- 说明：login登录结果的回调
- 参数：

参数	类型	描述
res	const RTCResult&	执行结果，参考 RTCRoomUtil.h 中定义的 RTCResult 结构体

参数	类型	描述
authData	const RTCAuthData&	RoomService提供的登录信息，包括IM相关的配置字段，在login成功后，获取到token字段

- 示例：

```
class LoginDialog : public ILoginRTCCallback
{
public:
virtual void onLogin(const RTCResult& res, const RTCAuthData& authData)
{
if (RTCROOM_SUCCESS != res.ec)
{
// 登录失败
}
else
{
// 登录成功，保存authData信息
}
}
};

// 参见RTCRoom::login接口，了解回调的使用
...
```

IGetRTCRoomListCallback回调接口的详情

1. onGetRoomList

- 定义：virtual void onGetRoomList(const RTCResult& res, const std::vector& rooms)
- 说明：获取房间列表的回调
- 参数：

参数	类型	描述
res	const RTCResult&	执行结果，参考 RTCRoomUtil.h 中定义的 RTCResult 结构体

参数	类型	描述
rooms	const std::vector&	房间信息的列表

- 示例：

```
class RoomListDialog : public IGetRTCRoomListCallback
{
public:
virtual void onGetRoomList(const RTCResult& res, const std::vector<RTCRoomData>& rooms)
{
if (RTCROOM_SUCCESS != res.ec)
{
// 查询失败
}
else
{
// 查询成功，处理rooms数据
}
}
};

// 参见RTCRoom::getRoomList接口，了解回调的使用
...
```

IRTCRoomCallback回调接口的详情

参见RTCRoom::setCallback接口，了解如何设置IRTCRoomCallback回调。

1. onCreateRoom

- 定义：virtual void onCreateRoom(const RTCResult& res, const std::string& roomID)
- 说明：创建房间的回调
- 参数：

参数	类型	描述
res	const RTCResult&	执行结果，参考 RTCRoomUtil.h 中定义的 RTCResult 结构体

参数	类型	描述
roomID	const std::string&	房间ID

- 示例：

```
class MainDialog : public IRTCRoomCallback
{
public:
virtual void onCreateRoom(const RTCResult& res, const std::string& roomID)
{
if (RTCROOM_SUCCESS != res.ec)
{
// 创建房间失败，弹框提示
}
else
{
// 创建房间成功，其他观众可以参与会议
// 紧接着处理onUpdateRoomData回调
}
}

...
};
```

2. onEnterRoom

- 定义：virtual void onEnterRoom(const RTCResult& res)
- 说明：进入房间的回调
- 参数：

参数	类型	描述
res	const RTCResult&	执行结果，参考 RTCRoomUtil.h 中定义的 RTCResult 结构体

- 示例：

```
class MainDialog : public IRTCRoomCallback
{
```

```
public:
virtual void onEnterRoom(const RTCResult& res)
{
if (RTCROOM_SUCCESS != res.ec)
{
// 进入房间失败，弹框提示
}
else
{
// 进入房间成功，紧接着处理onUpdateRoomData回调
}
}

...
};
```

3. onUpdateRoomData

- 定义：virtual void onUpdateRoomData(const RTCResult& res, const RTCRoomData& roomData)
- 说明：房间信息变更的回调
- 参数：

参数	类型	描述
res	const RTCResult&	执行结果，参考 RTCRoomUtil.h 中定义的 RTCResult 结构体
roomData	const RTCRoomData&	房间信息，参考 RTCRoomUtil.h 中定义的 RTCRoomData 结构体

- 示例：

```
class MainDialog : public IRTCRoomCallback
{
public:
virtual void onUpdateRoomData(const RTCResult& res, const RTCRoomData& roomData)
{
if (RTCROOM_SUCCESS != res.ec)
{
// 更新失败，弹框提示
}
else
```

```
{
// 更新成功，根据createRoom还是enterRoom导致这个接口的触发来做出相应的处理
}
}

...
};
```

4. onPusherJoin

- 定义：virtual void onPusherJoin(const RTCMemberData& member)
- 说明：成员进入房间的回调
- 参数：

参数	类型	描述
member	const RTCMemberData&	成员信息，参考 RTCRoomUtil.h 中定义的 RTCMemberData 结构体

- 示例：

```
class MainDialog : public IRTCRoomCallback
{
public:
virtual void onPusherJoin(const RTCMemberData& member)
{
// 成员进入房间，打开成员的画面和音频
m_RTCRoom->addRemoteView(rendHwnd, rect, member.userID);
...
}

...
};
```

5. onPusherQuit

- 定义：virtual void onPusherQuit(const RTCMemberData& member)

- 说明：成员退出房间的回调
- 参数：

参数	类型	描述
member	const RTCMemberData&	成员信息，参考 RTCRoomUtil.h 中定义的 RTCMemberData 结构体

- 示例：

```
class MainDialog : public IRTCRoomCallback
{
public:
virtual void onPusherQuit(const RTCMemberData& member)
{
// 成员退出房间，关闭成员的画面和音频
m_RTCRoom->removeRemoteView(member.userID);
...
}

...
};
```

6. onRoomClosed

- 定义：virtual void onRoomClosed(const std::string& roomID)
- 说明：房间解散的回调
- 参数：

参数	类型	描述
roomID	const std::string&	房间ID

- 示例：

```
class MainDialog : public IRTCRoomCallback
{
```

```
public:
virtual void onRoomClosed(const std::string& roomID)
{
// 提示房间解散，关闭已打开的音视频
...
}

...
};
```

7. onRecvRoomTextMsg

- 定义：virtual void onRecvRoomTextMsg(const char *roomID, const char* userID, const char *userName, const char* userAvatar, const char * msg)
- 说明：收到普通文本消息
- 参数：

参数	类型	描述
roomID	const char *	房间ID
userID	const char *	发送者ID
userName	const char *	发送者昵称
userAvatar	const char *	发送者头像
message	const char *	文本消息内容

- 示例：

```
class MainDialog : public IRTCRoomCallback
{
public:
virtual void onRecvRoomTextMsg(const char * roomID, const char * userID, const char * userName, const char * userAvatar, const char * msg)
{
// IM面板显示文本消息
}
```

```
...
};
```

8. onRecvRoomCustomMsg

- 定义：virtual void onRecvRoomCustomMsg(const char *roomID, const char* userID, const char *userName, const char* userAvatar, const char *cmd, const char* message)
- 说明：收到自定义消息
- 参数：

参数	类型	描述
roomID	const char *	房间ID
userID	const char *	发送者ID
userName	const char *	发送者昵称
userAvatar	const char *	发送者头像
cmd	const char *	自定义cmd
message	const char *	自定义消息内容

- 示例：

```
class MainDialog : public IRTCRoomCallback
{
public:
virtual void onRecvRoomCustomMsg(const char * roomID, const char * userID, const char * user
Name, const char * userAvatar, const char * cmd, const char * message)
{
// 解析和处理自定义消息，例如点赞、礼物等
}

...
};
```

9. onError

- 定义：virtual void onError(const RTCResult& res, const std::string& userID)
- 说明：RTCRoom内部发生错误的通知
- 参数：

参数	类型	描述
res	const RTCResult&	执行结果，参考 RTCRoomUtil.h 中定义的 RTCResult 结构体
userID	const std::string&	用户ID

- 示例：

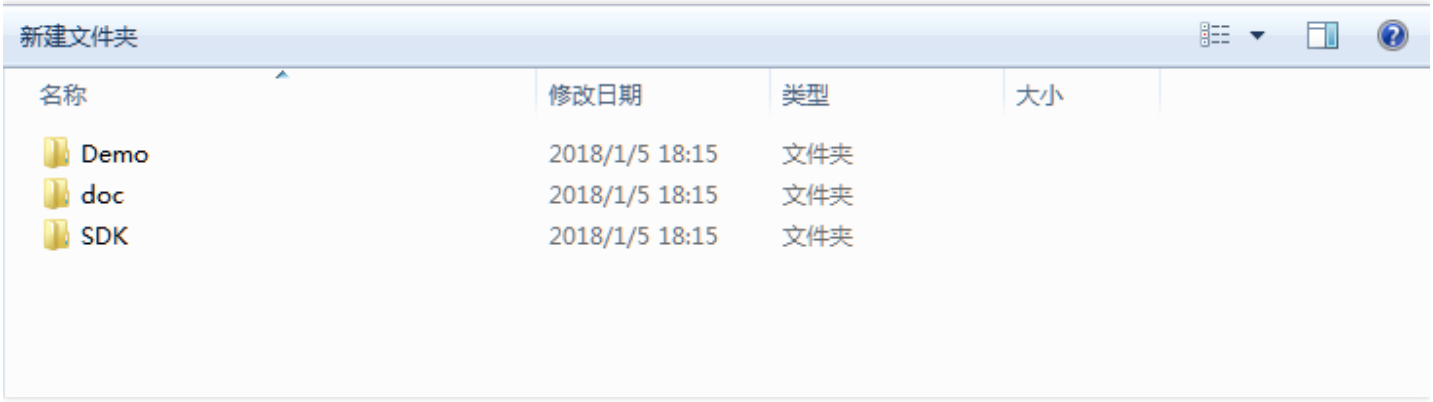
```
class MainDialog : public IRTCRoomCallback
{
public:
virtual void onError(const RTCResult& res, const std::string& userID)
{
// 弹框提示错误，根据错误严重程序，是否解散或者退出会议
}

...
};
```

C#
工程配置（Visual Studio）

最近更新时间：2019-03-29 11:57:59

SDK开发包

您可以在腾讯云官网获取到 视频云 SDK：



文件名	说明
Demo	包含 Demo 安装包和源码，Demo 安装包可以立即安装快速体验 SDK 的功能，可使用 Visual Studio 打开源码的工程
doc	包含介绍工程配置和 SDK 接入等文档
SDK	.dll 的形式，其中 C# 程序集 ManageLiteAV.dll 可导入 C# 工程中

Visual Studio 工程配置

开发环境

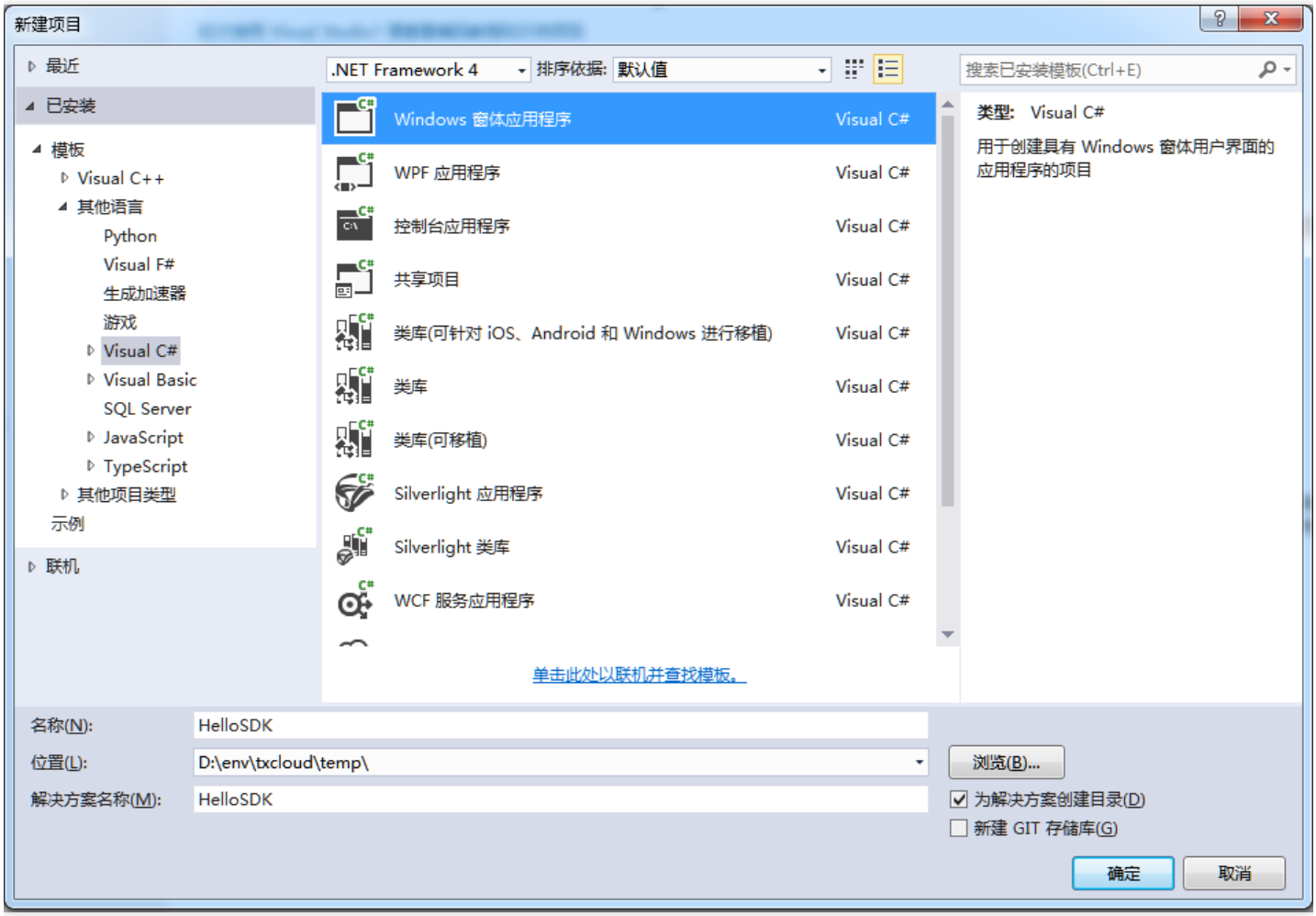
- Visual Studio 最低版本要求是2010
- .Net framework 最低版本要求是4.0
- Windows 最低版本要求是 Windows 7

C# 工程引用 SDK

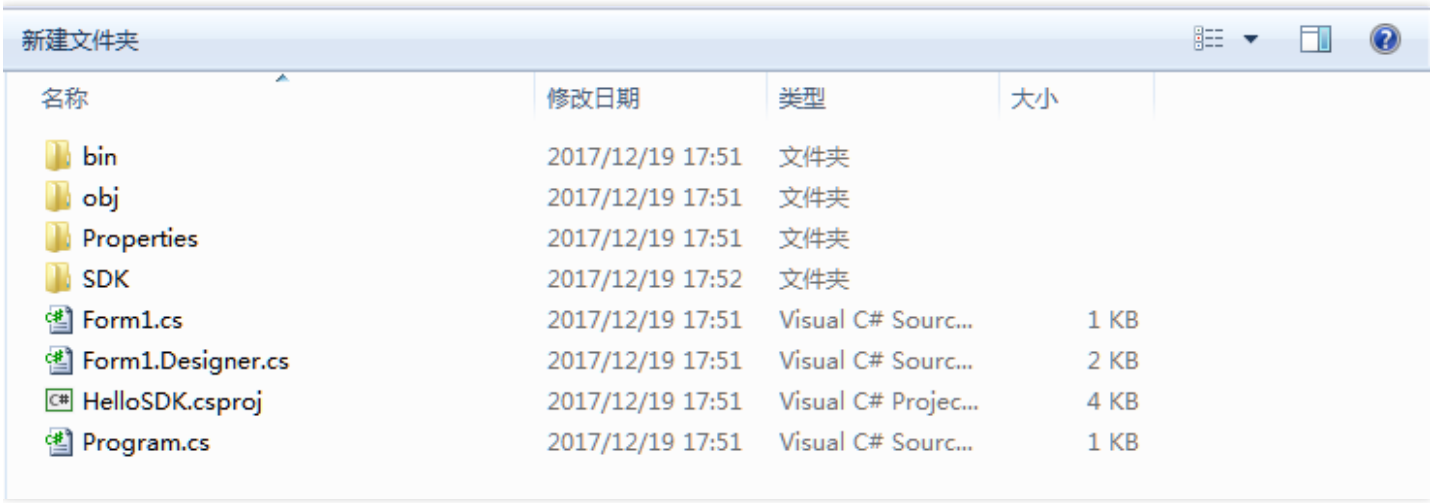
下面通过一个简单的 C# WinForm 工程，说明如何在 Visual Studio 工程中配置 SDK：

step1：拷贝 SDK 文件

创建一个名字叫 HelloSDK 的 C# 窗体应用程序的工程，注意选择 .NET Framework 4 或更高的版本。

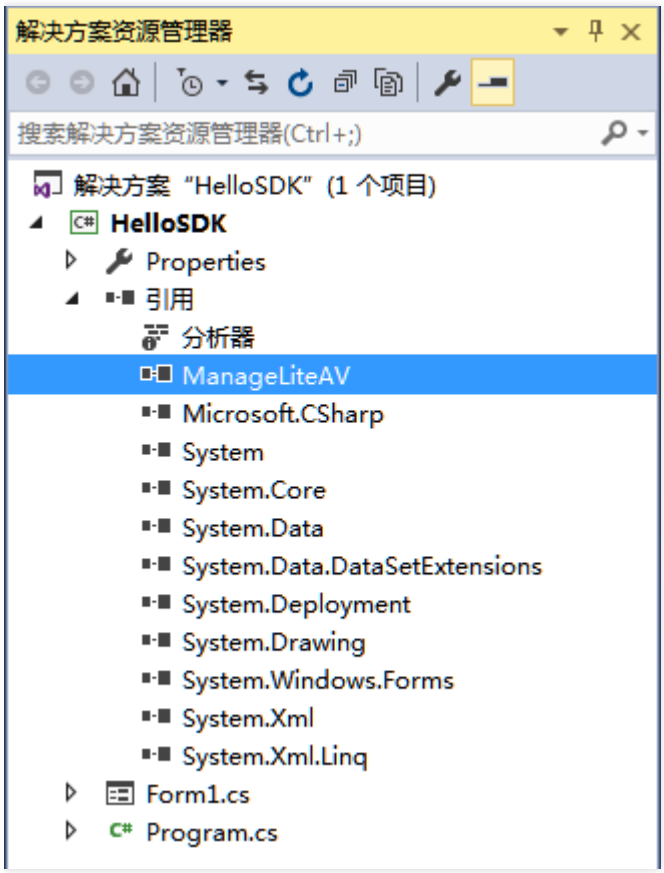


将下载得到的SDK文件拷贝到工程目录中。



在HelloSDK的工程中，打开引用管理器的页面，单击 浏览 按钮，添加引用ManageLiteAV.dll，添加完成后，如下

图所示：



step2：验证

下面在 HelloSDK 的代码中，调用 SDK 的接口，显示摄像头的预览，以验证工程设置是否正确。

添加代码

在 Form1 类中添加 ManageLiteAV.TXLivePusher 的实例。

```
private ManageLiteAV.TXLivePusher pusher = new ManageLiteAV.TXLivePusher();
```

在属性面板添加 Form1 的 Load 事件，添加代码打开 ManageLiteAV.TXLivePusher 摄像头预览的功能。

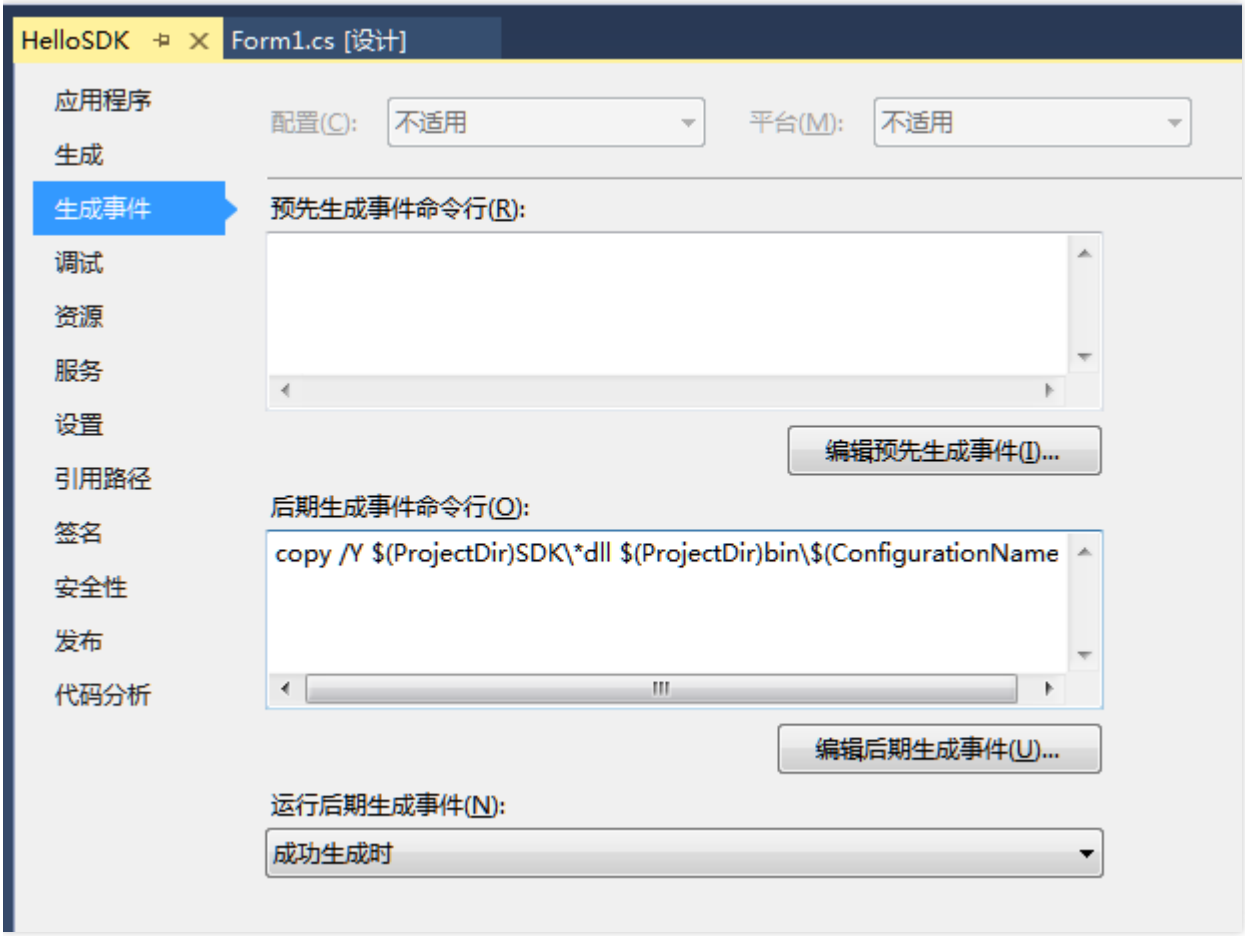
```
private void onLoaded(object sender, EventArgs e)
{
// 默认至少有一台可用的摄像头，故此处 cameraIndex 传入0
pusher.startPreview(ManageLiteAV.TXEVideoCaptureSrcType.TXE_VIDEO_SRC_SDK_CAMERA, this.Handle, 0, 0, this.Width, this.Height, ManageLiteAV.TXEVideoUserDataFormat.TXE_USER_DATA_FORMAT_UNDEFINED);
}
```

编译运行

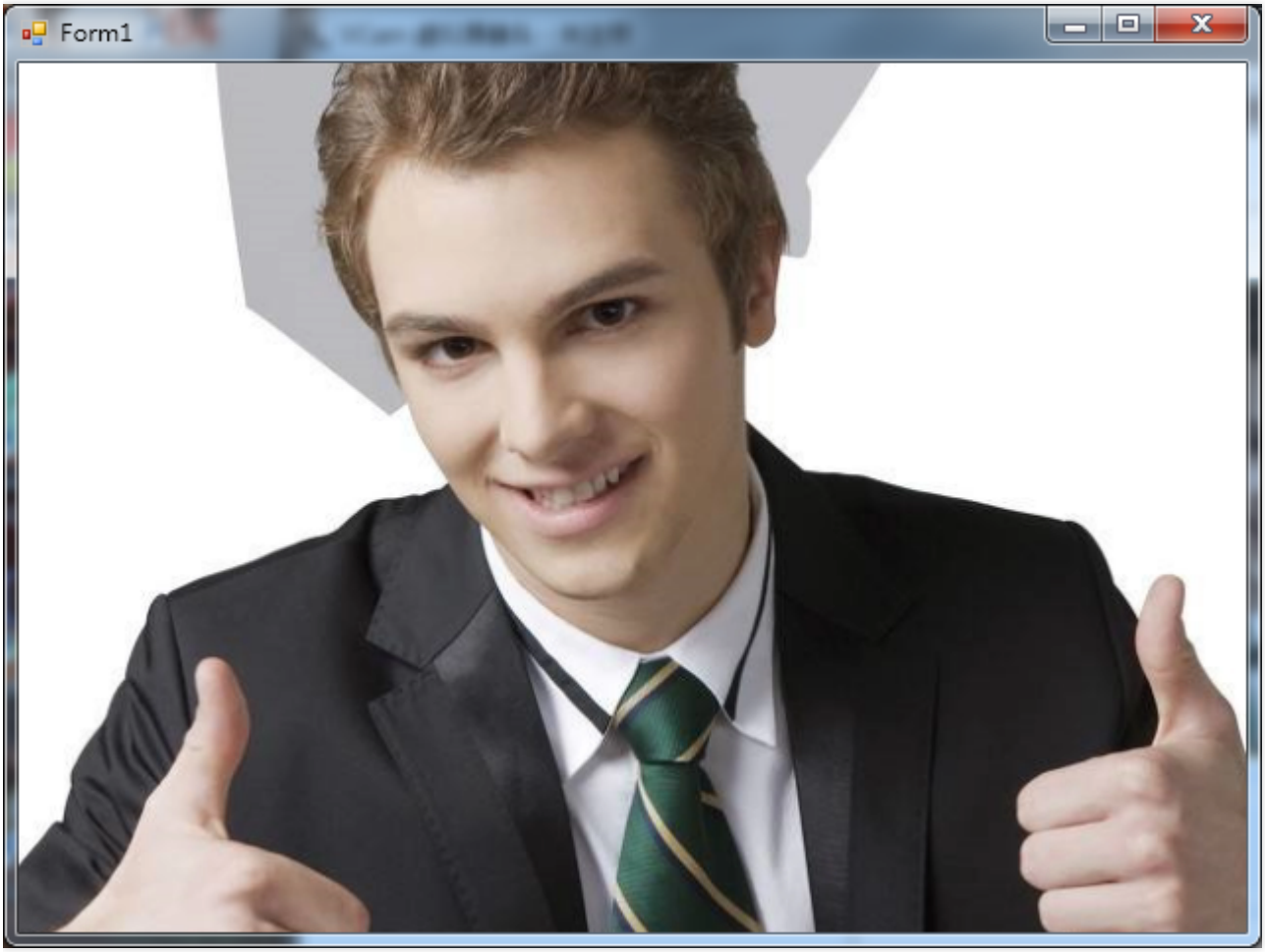
打开 HelloSDK 工程的属性页面，添加后期生成的命令行，用于将 SDK 的 dll 文件拷贝到 exe 可执行文件所在的目录中。

```
copy /Y "$(ProjectDir)SDK\liteav\*dll" "$(ProjectDir)bin\$(ConfigurationName)"
```

添加的命令行位置如下图所示：



执行编译（F7），编译通过后，运行调试（F5），可以看到窗口显示摄像头的画面：



至此，工程配置完成。

基础功能
API（C#）

最近更新时间：2018-07-11 11:17:28

API 接口列表

接口文件

在腾讯云官网 下载 SDK 开发包，该开发包包含有程序集ManageLiteAV.dll和接口注释文档 ManageLiteAV.xml，您的 C# 工程引用该程序集后，可以在Visual Studio中查看详细的接口注释。

下面是腾讯视频云Windows SDK（C#）的接口列表，分为TXLivePusher和TXLivePlayer两个类，以及若干枚举enum和全局常量。

TXLivePusher

成员函数	功能说明
enumCameras()	枚举当前的摄像头，如果一台Windows同时安装了多个摄像头，那么此函数用于获取可用的摄像头数量和名称
micDevices()	枚举当前可用的麦克风，如果一台Windows同时安装了多个麦克风，那么此函数获取可用的麦克风数量和名称
micVolume()	查询已选择麦克风的音量
selectMicDevice(uint)	选择指定的麦克风作为录音设备，不调用该接口时，默认选择索引为0的麦克风
setAutoAdjustStrategy(TXEAutoAdjustStrategy)	设置流控策略，即是否允许 SDK 根据当前网络情况调整视频码率，以避免网络上传速度不足导致的画面卡顿
setBeautyStyle(TXEBeautyStyle, int, int)	设置美颜和美白效果
setListener(ITXLivePusherListener, IntPtr)	设置回调 TXLivePusher 的回调代理，用于监听推流事件

成员函数	功能说明
setMicVolume(uint)	设置已选择麦克风的音量
setMute(bool)	静音接口
setOutputYMirror(bool)	设置推流画面的镜像效果
setRenderMode(TXERenderMode)	设置图像的渲染（填充）模式
setRenderYMirror(bool)	设置预览渲染的镜像效果
setRotation(TXEVideoRotation)	设置图像的顺时针旋转角度
setVideoBitRate(uint)	设置视频码率，注意，不是分辨率越高画面越清晰，是码率越高画面越清晰
setVideoBitRateMax(int)	配合 setAutoAdjustStrategy 使用，当 AutoAdjust 策略指定为 TXE_AUTO_ADJUST_NONE 时，该函数调用均视为无效
setVideoBitRateMin(int)	配合 setAutoAdjustStrategy 使用，当 AutoAdjust 策略指定为 TXE_AUTO_ADJUST_NONE 时，该函数调用均视为无效
setVideoFPS(uint)	设置视频帧率
setVideoResolution(TXEVideoResolution)	设置视频分辨率
startPreview(IntPtr, int, int, int, int, int)	启动摄像头预览
startPush(string)	启动推流 (在 startPush 之前需要先 startPreview 启动摄像头预览，否则推送出去的数据流里只有音频)
stopPreview()	关闭摄像头预览，stopPush 之前调用此函数并不会停止推流，会导致 SDK 只推送音频数据
stopPush()	停止推流，注意推流 url 有排他性，也就是一个推流 Url 同时只能有一个推流端向上推流
switchCamera(int)	切换摄像头，支持在推流中动态切换
updatePreview(IntPtr, int, int, int, int)	重设摄像头预览区域，当您指定的 HWND 的窗口尺寸发生变化时，可以通过这个函数重新调整视频渲染区域

TXLivePlayer

成员函数	功能说明
closeRenderFrame()	关闭图像渲染
isPlaying()	是否正在播放
pause()	暂停播放
resume()	恢复播放
setListener(ITXLivePlayerListener, IntPtr)	设置回调 TXLivePlayer 的回调代理，用于监听播放事件
setMute(bool)	静音接口
setOutputVideoFormat(TXEOutputVideoFormat)	设置视频编码格式，默认格式是 TXE_OUTPUT_VIDEO_WITHOUT_OUTPUT
setRenderFrame(IntPtr, int, int, int, int)	挂接视频图像渲染
setRenderMode(TXERenderMode)	设置图像的渲染（填充）模式
setRenderYMirror(bool)	设置渲染的镜像效果
setRotation(TXEVideoRotation)	设置图像的顺时针旋转角度
startPlay(string, TXEPlayType)	开始播放，请在 startPlay 之前 setRenderFrame
stopPlay()	停止播放
updateRenderFrame(IntPtr, int, int, int, int)	重设图像渲染区域，当您指定的 HWND 的窗口尺寸发生变化时，可以通过这个函数重新调整视频渲染区域

ITXLivePusherListener（interface）

主要用来监听推流事件。

接口定义	功能说明
onEventCallback(int, Dictionary, IntPtr)	TXLivePusher的推流事件通知

ITXLivePlayerListener（interface）

主要用来监听播放事件。

接口定义	功能说明
onEventCallback(int, Dictionary, IntPtr)	TXLivePlayer的播放事件通知

推流和播放

最近更新时间：2018-07-11 11:17:43

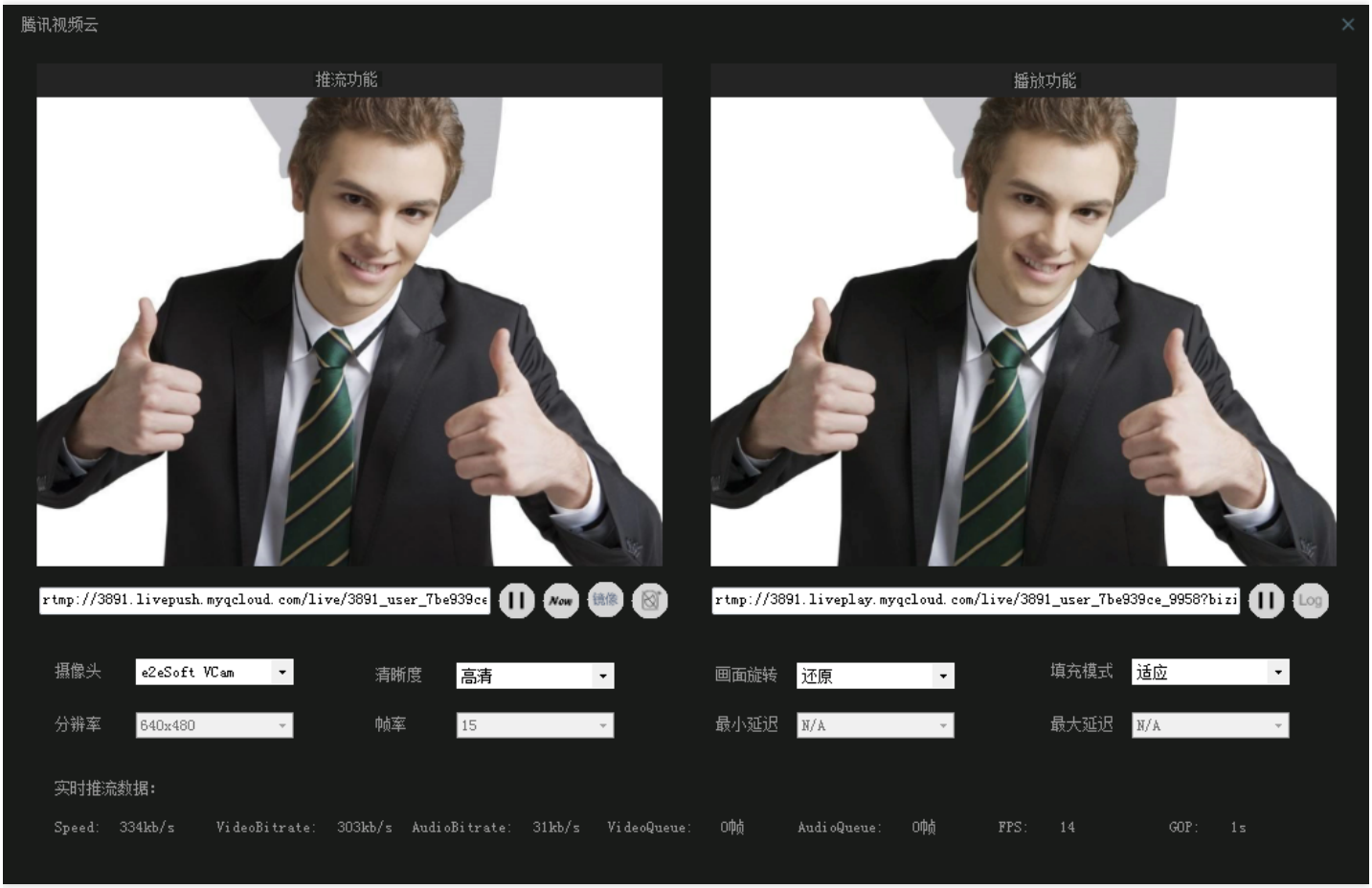
推流和播放

产品介绍

腾讯视频云 SDK 是集 **标准推拉流** 和 **实时视频通话** 于一体的音视频 SDK 组件，本文档主要介绍了其 C# 版本的集成方案。

相比于 Obs Studio （常用的一款推流软件）和 Flash 播放器（PC 浏览器上常用的播放插件），视频云 SDK 最大的优势在于：

- 低延迟方面的优化 ： 可以实现实时音视频通话。
- UDP 协议加速 ： 可以获得更好的推流稳定性。



准备工作

- **获取开发包**
  下载 SDK 开发包，并按照工程配置指引将 SDK 嵌入您的 APP 开发工程。

- **推流测试URL**
  Demo支持自动从后台获取到推流测试 URL，点击上图中的 **New** 圆形按钮即可获得推流地址。

  另外还有一种手动从腾讯云后台的控制台生成的方式，开通直播服务后，可以使用 直播控制台>>直播码接入>>推流生成器 生成推流地址，详细信息可以参考 获得推流播放URL。

- **播放测试URL**
  推流地址 `rtmp://8888.livepush.myqcloud.com/live/8888_teststream?bizid=8888&txSecret=6e18e8db0ff2070a339ab739ff46b957&txTime=5A3E7D7F`
  对应的播放地址即为： `rtmp://8888.liveplay.myqcloud.com/live/8888_teststream`

推流功能

代码对接

step 1: 创建Pusher对象

先创建一个 **ManageLiteAV.TXLivePusher** 对象，我们后面主要用它来完成推流工作。

创建完成后，在开始推流之前，可以调用ManageLiteAV.TXLivePusher的设置接口，设置镜像效果、码率、分辨率和填充模式等

```
pusher.setRenderYMirror(true);
pusher.setOutputYMirror(true);
pusher.setVideoBitRate(900);
pusher.setVideoBitRateMin(300);
pusher.setVideoBitRateMax(1200);
pusher.setVideoResolution(ManageLiteAV.TXEVideoResolution.TXE_VIDEO_RESOLUTION_640x480);
pusher.setRenderMode(ManageLiteAV.TXERenderMode.TXE_RENDER_MODE_FILLSCREEN);
...
```

step 2: 设置预览区域

接下来我们要给摄像头的影像画面找个地方来显示，Windows 使用 HWND 窗口句柄作为基本的界面渲染单位，在 C# 窗体应用程序中，每个控件可以调用它的Handle获取得到HWND，所以您只需要准备一个 HWND 传给 ManageLiteAV.TXLivePusher 对象的 **startPreview** 接口就可以了。

```
// 设置回调，用于获取音视频数据和SDK内部事件
pusher.setCallback(this, (IntPtr)UserDataFlag.PusherFlag);
// 打开摄像头，摄像头索引值可通过ManageLiteAV.TXLivePusher 对象的 enumCameras 接口获取得到
pusher.startPreview(pushRenderPanel.Handle, 0, 0, pushRenderPanel.Width, pushRenderPanel.Heigh
t, cameraIndex);
```

step 3: 启动推流

经过 step1 和 step2 的准备之后，用下面这段代码就可以启动推流了：

```
string rtmpURL = "rtmp://2157.livepush.myqcloud.com/live/xxxxxx";
pusher.startPush(rtmpURL);
```

step 4: 控制摄像头

一台电脑中如果存在多个摄像头，可以在不同摄像头之间进行切换，只需要调用ManageLiteAV.TXLivePusher 对象的 **switchCamera**接口，传入摄像头的索引值，即可完成切换。

```
// 传入要切换的摄像头索引值
pusher.switchCamera(cameraComboBox.SelectedIndex);
```

- TXLivePusher 的 enumCameras 函数可以获取摄像头的索引值。
- Windows 下开启一个 USB 摄像头需要很长的电路和驱动启动时间，TXLivePusher 的 startPreview 的最后一个参数为 -1 时 ，代表打开全部摄像头，这时再切换摄像头，反应时间就会很快。
- SDK 支持虚拟摄像头，启用用法和普通摄像头一致，如果您的 PC 还没有物理摄像头可用，可以安装 VCam 这样的虚拟摄像头软件来辅助调试。

step 5: 结束推流

结束推流很简单，不过要做好清理工作，因为用于推流的 ManageLiteAV.TXLivePusher 对象同一时刻只能有一个在运行，所以清理工作不当会导致下次直播遭受不良的影响。

```
// 回调置空
pusher.setCallback(null, (IntPtr)null);
// 停止摄像头预览
pusher.stopPreview();
// 停止推流
pusher.stopPush();
```

step 6: 更多功能

本地摄像头的画面镜像、静音、清晰度设置等功能，可以参考接口列表，获取更多详细信息。

事件处理

SDK 通过 ManageLiteAV.TXLivePusher 对象的 setListener 接口来监听推流相关的事件。

1. 常规事件

一次成功的推流都会通知的事件，比如收到1003就意味着摄像头的画面会开始渲染了。

事件ID	数值	含义说明
PUSH_EVT_CONNECT_SUCC	1001	已经连接推流服务器
PUSH_EVT_PUSH_BEGIN	1002	已经与服务器握手完毕,开始推流
PUSH_EVT_OPEN_CAMERA_SUCC	1003	打开摄像头成功
PUSH_EVT_CHANGE_RESOLUTION	1005	推流动态调整分辨率
PUSH_EVT_CHANGE_BITRATE	1006	推流动态调整码率
PUSH_EVT_FIRST_FRAME_AVAILABLE	1007	首帧画面采集完成

事件ID	数值	含义说明
PUSH_EVT_START_VIDEO_ENCODER	1008	编码器启动
PUSH_EVT_CAMERA_REMOVED	1009	摄像头设备已被移出
PUSH_EVT_CAMERA_AVAILABLE	1010	摄像头设备重新可用
PUSH_EVT_CAMERA_CLOSED	1011	关闭摄像头完成

2. 错误通知

SDK发现了一些严重问题，推流无法继续了，比如Camera已被其他程序占用导致摄像头打不开。

事件ID	数值	含义说明
PUSH_ERR_OPEN_CAMERA_FAIL	-1301	打开摄像头失败
PUSH_ERR_OPEN_MIC_FAIL	-1302	打开麦克风失败
PUSH_ERR_VIDEO_ENCODE_FAIL	-1303	视频编码失败
PUSH_ERR_AUDIO_ENCODE_FAIL	-1304	音频编码失败
PUSH_ERR_UNSUPPORTED_RESOLUTION	-1305	不支持的视频分辨率
PUSH_ERR_UNSUPPORTED_SAMPLERATE	-1306	不支持的音频采样率
PUSH_ERR_NET_DISCONNECT	-1307	网络断连,且经多次重连抢救无效,可以放弃治疗,更多重试请自行重启推流
PUSH_ERR_CAMERA_OCCUPY	-1308	摄像头正在被占用中，可尝试打开其他摄像头

3. 警告事件

相比于错误信息，WARNING 事件所代表的问题通常不会打断推流进程，而且SDK内部往往会启动自动修复逻辑，所以大多数 WARNING 事件您都无需关心，不过下面两个事件较为重要和有用：

WARNING_NET_BUSY

上行的网速不佳，它代表用户当前的音视频数据不能很流畅的推给服务器，这时一般可以给用户一些 UI 上的提示，比如 “您的网速不太给力”，让用户对于另一端的卡顿有一个心理预期。

WARNING_SERVER_DISCONNECT

推流请求被后台拒绝了，出现这个问题一般是由于推流地址里的 txSecret 计算错了，或者是推流地址被其他人占用了（推流 URL 是排他的，一个推流 URL 同时只能有一个用户去使用）。

事件ID	数值	含义说明
PUSH_WARNING_NET_BUSY	1101	网络状况不佳：上行带宽太小，上传数据受阻
PUSH_WARNING_RECONNECT	1102	网络断连, 已启动自动重连 (自动重连连续失败超过三次会放弃)
PUSH_WARNING_HW_ACCELERATION_FAIL	1103	硬编码启动失败，采用软编码
PUSH_WARNING_VIDEO_ENCODE_FAIL	1104	视频编码失败,非致命错,内部会重启编码器
PUSH_WARNING_BEAUTYSURFACE_VIEW_INIT_FAIL	1105	视频编码码率异常，警告
PUSH_WARNING_VIDEO_ENCODE_BITRATE_OVERFLOW	1106	视频编码码率异常，警告
PUSH_WARNING_DNS_FAIL	3001	RTMP -DNS解析失败（会触发重试流程）
PUSH_WARNING_SEVER_CONN_FAIL	3002	RTMP服务器连接失败（会触发重试流程）
PUSH_WARNING_SHAKE_FAIL	3003	RTMP服务器握手失败（会触发重试流程）
PUSH_WARNING_SERVER_DISCONNECT	3004	RTMP服务器主动断开，请检查推流地址的合法性或防盗链有效期
PUSH_WARNING_SERVER_NO_DATA	3005	超过30s没有数据发送，主动断开连接

播放功能

代码对接

step 1: 创建Player对象

先创建一个 **ManageLiteAV.TXLivePlayer** 对象，我们后面主要用它来完成推流工作。

```
player = new ManageLiteAV.TXLivePlayer();
```

step 2: 设置渲染区域

接下来我们要给播放器的视频画面找个地方来显示，Windows系统中使用 HWND 作为基本的界面渲染单位，在C#窗体应用程序中，每个控件可以调用它的Handle获取得到HWND。

```
// 设置回调
player.setListener(this, (IntPtr)UserDataFlag.PlayerFlag);
// 传入HWND窗口句柄，以及渲染位置和大小
player.setRenderFrame(pullRenderPanel.Handle, 0, 0, pullRenderPanel.Width, pullRenderPanel.Height);
```

step 3: CDN播放

启动播放的功能，只需调用 ManageLiteAV.TXLivePlayer 的 startPlay 接口即可，其中 PLAY_TYPE_LIVE_RTMP 是常规 CDN 的 RTMP 地址，优点是带宽价格很低，但延迟一般比较大。

```
string rtmpURL = "rtmp://2157.liveplay.myqcloud.com/live/xxxxxx";
player.startPlay(rtmpURL，ManageLiteAV.TXEPlayType.PLAY_TYPE_LIVE_RTMP);
```

step 3': 超低延迟播放

超低延迟播放可以将端到端的延迟拉低到 400ms 左右，该模式下 SDK 的内部机制和常规模式完全不同，于此同时， SDK 所使用的音视频线路也不是常规 CDN 线路，所以单价较 CDN 偏高，一般仅适用于音视频通话和对延迟要求比较苛刻的场景。

```
string rtmpURL = "rtmp://2157.liveplay.myqcloud.com/live/xxxxxxbizid=2157&txSecret=6e18e8db0ff2070a339ab739ff46b957&txTime=5A3E7D7F";
player.startPlay(rtmpURL，ManageLiteAV.TXEPlayType.PLAY_TYPE_LIVE_RTMP_ACC);
```

- 该功能并不需要提前开通，但是要求直播流必须位于腾讯云，跨云商实现低延时链路的难度不仅仅是技术层面的。
- 播放URL 不能用普通的 CDN URL， 必须要带防盗链签名，防盗链签名的计算方法见 **txTime&txSecret**。
- 在调用 startPlay 接口时，指定 type 为 **PLAY_TYPE_LIVE_RTMP_ACC**，SDK 会拉取超低延迟的直播流。
- 最多同时 10 路 并发播放，所以您只能在互动场景中使用（比如连麦主播 或者 夹娃娃直播中的操作者这一路）。
- Obs Studio 推出的 RTMP 流在客户端就引入了 1s 以上的延迟，请使用视频云 SDK 进行推流。

step 4: 画面调整

- **updateRenderFrame**
  当您指定的 HWND 的窗口尺寸发生变化时，可以通过 **updateRenderFrame** 函数重新调整视频渲染区域。
- **setRenderMode**

可选值	含义

可选值	含义
TXE_RENDER_MODE_ADAPT	适应，此模式下会显示整个画面的全部内容，但可能有黑边的存在
TXE_RENDER_MODE_FILLSCREEN	填充，此模式下画面无黑边，但是会裁剪掉一部分超出渲染区域的部分，裁剪模式为局中裁剪

step 5: 结束播放

结束播放时，如果要退出当前的UI界面，调用ManageLiteAV.TXLivePlayer的 stopPlay 接口。

```
// 回调置空
player.setCallback(null, (IntPtr)null);
// 停止拉流播放
player.stopPlay();
```

step 6: 更多功能

可以参考 接口列表，获取更多详细信息。

事件处理

通过 ManageLiteAV.TXLivePlayer 对象的 setListener 接口来监听播放相关的事件。

1. 常规事件

事件ID	数值	含义说明
PLAY_EVT_CONNECT_SUCC	2001	已经连接服务器
PLAY_EVT_RTMP_STREAM_BEGIN	2002	已经连接服务器，开始拉流
PLAY_EVT_RCV_FIRST_I_FRAME	2003	渲染首个视频数据包(IDR)
PLAY_EVT_PLAY_BEGIN	2004	视频播放开始
PLAY_EVT_PLAY_PROGRESS	2005	视频播放进度
PLAY_EVT_PLAY_END	2006	视频播放结束
PLAY_EVT_PLAY_LOADING	2007	视频播放loading
PLAY_EVT_START_VIDEO_DECODER	2008	解码器启动
PLAY_EVT_CHANGE_RESOLUTION	2009	视频分辨率改变

2. 错误通知

SDK 遭遇了一些严重错误，比如网络断开等等，这些错误会导致播放无法继续，因此您的代码需要对这些错误进行相应的处理。

事件ID	数值	含义说明
PLAY_ERR_NET_DISCONNECT	-2301	网络断连，且重试亦不能恢复，将导致播放失败
PLAY_ERR_GET_RTMP_ACC_URL_FAIL	-2302	获取加速拉流地址失败，会导致播放失败

3. 警告事件

SDK 发现了一些非严重错误，一般不会导致播放停止，所以您可以不关注如下事件，其中：

PLAY_WARNING_VIDEO_PLAY_LAG 是 SDK 对外通知播放卡顿的事件信号，它指的是视频画面的卡顿（两帧画面的刷新时间）超过 500ms。

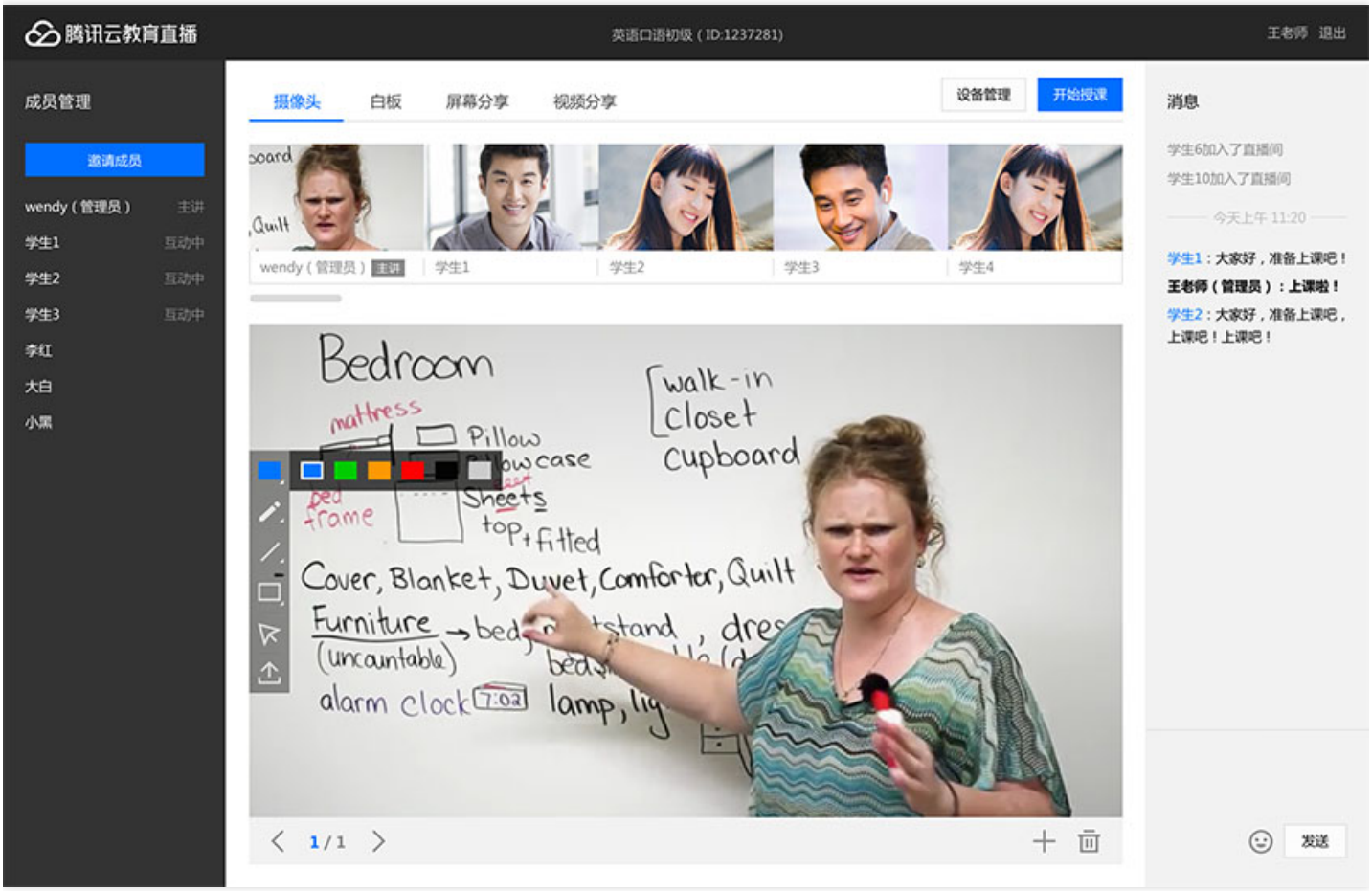
事件ID	数值	含义说明
PLAY_WARNING_VIDEO_DECODE_FAIL	2101	当前视频帧解码失败
PLAY_WARNING_AUDIO_DECODE_FAIL	2102	当前音频帧解码失败
PLAY_WARNING_RECONNECT	2103	网络断连, 已启动自动重连 (自动重连连续失败超过三次会放弃)
PLAY_WARNING_RECV_DATA_LAG	2104	网络来包不稳：可能是下行带宽不足，或由于主播端出流不均匀
PLAY_WARNING_VIDEO_PLAY_LAG	2105	当前视频播放出现卡顿（用户直观感受）
PLAY_WARNING_HW_ACCELERATION_FAIL	2106	硬解启动失败，采用软解（暂不支持）
PLAY_WARNING_VIDEO_DISCONTINUITY	2107	当前视频帧不连续，可能丢帧
PLAY_WARNING_FIRST_IDR_HW_DECODE_FAIL	2108	当前流硬解第一个I帧失败，SDK自动切软解
PLAY_WARNING_DNS_FAIL	3001	RTMP -DNS解析失败
PLAY_WARNING_SEVER_CONN_FAIL	3002	RTMP服务器连接失败
PLAY_WARNING_SHAKE_FAIL	3003	RTMP服务器握手失败
PLAY_WARNING_SERVER_DISCONNECT	3004	RTMP服务器主动断开

实时连麦
直播连麦（LiveRoom）

最近更新时间：2018-07-11 11:08:05

直播+连麦（LiveRoom ）

直播+连麦 是在 **秀场直播** 和 **在线教育** 场景中经常使用的直播模式，它既能支持高并发和低成本的在线直播，又能通过连麦实现主播和观众之间的视频通话互动，具有极强的场景适用性。



腾讯云基于 **LiveRoom** 组件实现"直播 + 连麦"功能，它分成 Client 和 Server 两个部分（都是开源的），对接攻略请参考 DOC，本文档主要是详细列出了 Client 端的 API 列表：

在腾讯云官网 下载 SDK 开发包，在SDK\Rooms\LiveRoom中，包括LiveRoom相关的头文件和源码文件。

LiveRoom

名称	描述
instance()	获取LiveRoom单例，通过单例调用LiveRoom的接口
setCallback(ILiveRoomCallback callback)	设置回调 LiveRoom 的回调代理，监听 LiveRoom 的内部状态和接口的执行结果
login(string serverDomain, LiveAuthData authData, ILoginLiveCallback callback)	登录业务服务器RoomService，登录后才能够正常使用其他接口和使用IM功能
logout()	登出业务服务器RoomService，请注意在leaveRoom调用后，再调用logout，否则leaveRoom会调用失败
getRoomList(int index, int cnt, IGetLiveRoomListCallback callback)	获取房间列表，房间数量比较多时，可以分页获取
getAudienceList(string roomID)	获取观众列表，只返回最近进入房间的 30 位观众
createRoom(string roomID, string roomInfo)	创建房间，后台的房间列表中会添加一个新房间，同时主播端会进入推流模式
enterRoom(string roomID, IntPtr rendHwnd, Rectangle rect)	进入房间
leaveRoom()	离开房间，如果是大主播，这个房间会被后台销毁，如果是小主播或者观众，不影响其他人继续观看
sendRoomTextMsg(string msg)	发送普通文本消息，比如直播场景中，发送弹幕
sendRoomCustomMsg(string cmd, string msg)	发送自定义消息，比如直播场景中，发送点赞、送花等消息
startLocalPreview(IntPtr rendHwnd, Rectangle rect)	启动默认的摄像头预览
updateLocalPreview(IntPtr rendHwnd, Rectangle rect)	重设摄像头预览区域，当您指定的本地 HWND 的窗口尺寸发生变化时，可以通过这个函数重新调整视频渲染区域
stopLocalPreview()	关闭摄像头预览
startScreenPreview(IntPtr rendHwnd, IntPtr captureHwnd, Rectangle renderRect, Rectangle captureRect)	启动屏幕分享

名称	描述
stopScreenPreview()	关闭屏幕分享
addRemoteView(IntPtr rendHwnd, Rectangle rect, string userID)	播放房间内其他主播的视频
updateRemotePreview(IntPtr rendHwnd, Rectangle rect, string userID)	重设指定userID的视频预览区域，当您指定的本地 HWND 的窗口尺寸发生变化时，可以通过这个函数重新调整视频渲染区域
removeRemoteView(string userID)	停止播放其他主播的视频
setMute(bool mute)	静音接口
setVideoQuality(LiveVideoQuality quality, LiveAspectRatio ratio)	设置画面质量预设选项
setBeautyStyle(LiveBeautyStyle beautyStyle, int beautyLevel, int whitenessLevel)	设置美颜和美白效果
requestJoinPusher()	观众端发起请求连麦
acceptJoinPusher(string userID)	大主播接受连麦请求，并通知给连麦发起方
rejectJoinPusher(string userID, string reason)	大主播拒绝连麦请求，并通知给连麦发起方
kickoutSubPusher(string userID)	大主播踢掉某一个小主播

ILoginLiveCallback

名称	描述
onLogin(LiveResult res, LiveAuthData authData)	login登录结果的回调

IGetLiveRoomListCallback

名称	描述
onGetRoomList(LiveResult res, List rooms)	获取房间列表的回调

ILiveRoomCallback

名称	描述
onCreateRoom(LiveResult res, string roomID)	创建房间的回调
onEnterRoom(LiveResult res)	进入房间的回调
onUpdateRoomData(LiveResult res, LiveRoomData roomData)	房间信息变更的回调
void onGetAudienceList(LiveResult res, List audiences)	查询观众列表的回调
onPusherJoin(LiveMemberData member)	连麦观众进入房间的回调
void onPusherQuit(LiveMemberData member)	连麦观众退出房间的回调
onRoomClosed(string roomID)	房间解散的回调
onRecvRoomTextMsg(string roomID, string userID, string userName, string userAvatar, string msg)	收到普通文本消息
onRecvRoomCustomMsg(string roomID, string userID, string userName, string userAvatar, string cmd, string message)	收到自定义消息
onError(LiveResult res, string userID)	LiveRoom内部发生错误的通知
onRecvJoinPusherRequest(string roomID, string userID, string userName, string userAvatar)	接收到连麦请求
onRecvAcceptJoinPusher(string roomID, string msg)	接收到接受连麦请求的回复
onRecvRejectJoinPusher(string roomID, string msg)	接收到拒绝连麦请求的回复
onRecvKickoutSubPusher(string roomID)	接收大主播踢小主播的通知

LiveRoom对象接口的详情

1. instance

- 定义：static LiveRoom instance()
- 说明：获取LiveRoom单例，通过单例调用LiveRoom的接口
- 示例：

```
LiveRoom m_liveRoom = null;
m_liveRoom = LiveRoom.instance();
```

2. setCallback

- 定义：void setCallback(ILiveRoomCallback callback)
- 说明：设置回调 LiveRoom 的回调代理，监听 LiveRoom 的内部状态和接口的执行结果
- 参数：

参数	类型	描述
callback	ILiveRoomCallback	ILiveRoomCallback 类型的代理接口

- 示例：

```
public class MainDialog : ILiveRoomCallback
{
MainDialog()
{
m_liveRoom.setCallback(this); // 设置回调
}

void ILiveRoomCallback.onCreateRoom(LiveResult res, string roomID)
{

}

void ILiveRoomCallback.onEnterRoom(LiveResult res)
{

}
```

```
void ILiveRoomCallback.onUpdateRoomData(LiveResult res, LiveRoomData roomData)
{

}

...
}
```

3. login

- 定义：void login(string serverDomain, LiveAuthData authData, ILoginLiveCallback callback)
- 说明：登录业务服务器RoomService，登录后才能够正常使用其他接口和使用IM功能
- 参数：

参数	类型	描述
serverDomain	string	RoomService的URL地址，安全起见，建议访问https加密链接，参考 DOC
authData	LiveAuthData	RoomService提供的登录信息，包括IM相关的配置字段，在login成功后，获取到token字段，参考 DOC
callback	ILoginLiveCallback	ILoginLiveCallback 类型的代理接口，回调login的结果

- 示例：

```
string serverDomain = "https://roomtest.qcloud.com/weapp/live_room";

m_authData.sdkAppID = sdkAppID;
m_authData.accountType = accountType;
m_authData.userID = userID;
m_authData.userSig = userSig;
m_authData.userName = userName;
m_authData.userAvatar = userAvatar;
m_authData.token = token;
```

```
// 该类实现ILoginCallback接口，获取login结果
m_liveRoom.login(serverDomain, m_authData, this);
```

4. logout

- 定义：void logout();
- 说明：登出业务服务器RoomService，请注意在leaveRoom调用后，再调用logout，否则leaveRoom会调用失败
- 示例：

```
m_liveRoom.logout();
```

5. getRoomList

- 定义：void getRoomList(int index, int cnt, IGetLiveRoomListCallback callback)
- 说明：获取房间列表，房间数量比较多时，可以分页获取
- 参数：

参数	类型	描述
index	int	分页获取，初始默认可设置为0，后续获取为起始房间索引（如第一次设置index=0, cnt=5,获取第二页时可用index=5）
count	int	每次调用，最多返回房间个数；0表示所有满足条件的房间
callback	IGetLiveRoomListCallback	IGetLiveRoomListCallback 类型的代理接口，查询结果的回调

- 示例：

```
// 此处从index=0开始，最多查询20个房间，该类实现IGetRoomListCallback接口，获取查询结果
m_liveRoom.getRoomList(0, 20, this);
```

6. getAudienceList

- 定义：void getAudienceList(string roomID)

- 说明：获取观众列表，只返回最近进入房间的 30 位观众
- 参数：

参数	类型	描述
roomID	string	房间ID，在 getRoomList 接口房间列表中查询得到

- 示例：

```
m_liveRoom.getRoomList(roomID);
```

7. createRoom

- 定义：void createRoom(string roomID, string roomInfo)
- 说明：创建房间，后台的房间列表中会添加一个新房间，同时主播端会进入推流模式
- 参数：

参数	类型	描述
roomID	string	房间ID，若传入空字符串，则后台会为您分配roomID，否则，传入的roomID作为这个房间的ID
roomInfo	string	自定义数据，该字段包含在房间信息中，推荐您将 roomInfo 定义为 json 格式，这样可以有很强的扩展性

- 示例：

```
// 后台会为您分配roomID
m_liveRoom.createRoom("", "{\"roomName\": \"My first room\"}");
```

8. enterRoom

- 定义：void enterRoom(string roomID, IntPtr rendHwnd, Rectangle rect)
- 说明：进入房间

- 参数：

参数	类型	描述
roomID	string	roomID - 要进入的房间ID，在 getRoomList 接口房间列表中查询得到
rendHwnd	IntPtr	承载预览画面的 HWND
rect	Rectangle	指定视频图像在 HWND 上的渲染区域

- 示例：

```
// 在form整个窗口中渲染
Rectangle rect = new Rectangle(0, 0, form.Width, form.Height);
m_liveRoom.enterRoom(roomID, form.Handle, rect);
```

9. leaveRoom

- 定义：void leaveRoom()
- 说明：离开房间，如果是大主播，这个房间会被后台销毁，如果是小主播或者观众，不影响其他人继续观看
- 示例：

```
m_liveRoom.leaveRoom();
```

10. sendRoomTextMsg

- 定义：void sendRoomTextMsg(string msg)
- 说明：发送普通文本消息，比如直播场景中，发送弹幕
- 参数：

参数	类型	描述
msg	string	文本消息

- 示例：

```
m_liveRoom.sendRoomTextMsg("Hello LiveRoom");
```

11. sendRoomCustomMsg

- 定义：void sendRoomCustomMsg(string cmd, string msg)
- 说明：发送自定义消息，比如直播场景中，发送点赞、送花等消息
- 参数：

参数	类型	描述
cmd	string	自定义cmd，收发双方协商好的cmd
msg	string	自定义消息

- 示例：

```
// 假设收发双方协商好的cmd有Smile, Anger
m_liveRoom.sendRoomCustomMsg("Smile", "I am hungry");
```

12. startLocalPreview

- 定义：void startLocalPreview(IntPtr rendHwnd, Rectangle rect)
- 说明：启动默认的摄像头预览
- 参数：

参数	类型	描述
rendHwnd	IntPtr	承载预览画面的 HWND，目前 SDK 是采用 OpenGL 向 HWND 上绘制图像的,rendHwnd = null时无需预览视频
rect	Rectangle	指定视频图像在 HWND 上的渲染区域

- 示例：

```
// 在form整个窗口中渲染
Rectangle rect = new Rectangle(0, 0, form.Width, form.Height);
m_liveRoom.startLocalPreview(form.Handle, rect);
```

13. updateLocalPreview

- 定义：void updateLocalPreview(IntPtr rendHwnd, Rectangle rect)
- 说明：重设摄像头预览区域，当您指定的本地 HWND 的窗口尺寸发生变化时，可以通过这个函数重新调整视频渲染区域
- 参数：

参数	类型	描述
rendHwnd	IntPtr	承载预览画面的 HWND，目前 SDK 是采用 OpenGL 向 HWND 上绘制图像的,rendHwnd = null时无需预览视频
rect	Rectangle	指定视频图像在 HWND 上的渲染区域

- 示例：

```
// 在form整个窗口中渲染
Rectangle rect = new Rectangle(0, 0, form.Width, form.Height);
m_liveRoom.updateLocalPreview(form.Handle, rect);
```

14. stopLocalPreview

- 定义：void stopLocalPreview()
- 说明：关闭摄像头预览
- 示例：

```
m_liveRoom.stopLocalPreview();
```

15. startScreenPreview

- 定义：bool startScreenPreview(IntPtr rendHwnd, IntPtr captureHwnd, Rectangle renderRect, Rectangle captureRect)
- 说明：启动屏幕分享
- 参数：

参数	类型	描述
rendHwnd	IntPtr	承载预览画面的 HWND，目前 SDK 是采用 OpenGL 向 HWND 上绘制图像的,rendHwnd = null时无需预览视频
captureHwnd	IntPtr	指定录取窗口，若为null，则 captureRect 不起效，并且录取整个屏幕；若不为null，则录取这个窗口的画面
renderRect	Rectangle	指定视频图像在 rendHwnd 上的渲染区域
captureRect	Rectangle	指定录取窗口客户区的区域

- 返回值：成功 or 失败
- 示例：

```
// 在form整个窗口中渲染
Rectangle renderRect = new Rectangle(0, 0, rendForm.Width, rendForm.Height);

// 录取整个窗口的画面
Rectangle captureRect = new Rectangle(0, 0, captureForm.Width, captureForm.Height);

m_liveRoom.startScreenPreview(rendForm.Handle, captureForm.Handle, renderRect, captureRect);
```

16. stopScreenPreview

- 定义：void stopScreenPreview()
- 说明：关闭屏幕分享
- 示例：

```
m_liveRoom.stopScreenPreview();
```

17. addRemoteView

- 定义：void addRemoteView(IntPtr rendHwnd, Rectangle rect, string userID)
- 说明：播放房间内其他主播的视频
- 参数：

参数	类型	描述
rendHwnd	IntPtr	承载预览画面的 HWND
rect	Rectangle	指定视频图像在 HWND 上的渲染区域
userID	string	用户ID

- 示例：

```
// 在form整个窗口中渲染
Rectangle renderRect = new Rectangle(0, 0, form.Width, form.Height);

// 打开用户的音视频播放
m_liveRoom.addRemoteView(form.Handle，rect, userID);
```

18. updateRemotePreview

- 定义：void updateRemotePreview(IntPtr rendHwnd, Rectangle rect, string userID)
- 说明：重设指定userID的视频预览区域，当您指定的本地 HWND 的窗口尺寸发生变化时，可以通过这个函数重新调整视频渲染区域
- 参数：

参数	类型	描述
rendHwnd	IntPtr	承载预览画面的 HWND
rect	Rectangle	指定视频图像在 HWND 上的渲染区域
userID	string	用户ID

- 示例：

```
// 在form整个窗口中渲染
Rectangle renderRect = new Rectangle(0, 0, form.Width, form.Height);

// 重设指定userID的视频预览区域
m_liveRoom.updateRemotePreview(form.Handle, rect, userID);
```

18. removeRemoteView

- 定义：void removeRemoteView(string userID)
- 说明：停止播放其他主播的视频
- 参数：

参数	类型	描述
userID	string	用户ID

- 示例：

```
m_liveRoom.removeRemoteView(userID);
```

19. setMute

- 定义：void setMute(bool mute)
- 说明：静音接口
- 参数：

参数	类型	描述
mute	bool	是否静音

- 示例：

```
m_liveRoom.setMute(true); // 设置为静音
```

20. setVideoQuality

- 定义：void setVideoQuality(LiveVideoQuality quality, LiveAspectRatio ratio)
- 说明：设置推流的画面质量选项
- 参数：

参数	类型	描述
quality	LiveVideoQuality	画质，参考 LiveRoomUtil.h 中定义的 LiveVideoQuality 枚举值
ratio	LiveAspectRatio	宽高比，参考 LiveRoomUtil.h 中定义的 LiveAspectRatio 枚举值

- 示例：

```
// 设置画质为高清，宽高比为4:3
m_liveRoom.setVideoQuality(LIVEROOM_VIDEO_QUALITY_HIGH_DEFINITION, LIVEROOM_ASPECT_RATIO_4_3);
```

21. setBeautyStyle

- 定义：void setBeautyStyle(LiveBeautyStyle beautyStyle, int beautyLevel, int whitenessLevel)
- 说明：设置美颜和美白效果
- 参数：

参数	类型	描述
beautyStyle	TXEBeautyStyle	参考 LiveRoomUtil.h 中定义的 LiveBeautyStyle 枚举值
beautyLevel	int	美颜级别取值范围 1 ~ 9； 0 表示关闭，1 ~ 9值越大，效果越明显
whitenessLevel	int	美白级别取值范围 1 ~ 9； 0 表示关闭，1 ~ 9值越大，效果越明显

- 示例：

```
// 自然，美颜度数5，美白度数5
m_liveRoom.setBeautyStyle(TXE_BEAUTY_STYLE_NATURE, 5, 5);
```

22. requestJoinPusher

- 定义：void requestJoinPusher()
- 说明：观众端发起请求连麦
- 示例：

```
m_liveRoom.requestJoinPusher();
```

23. acceptJoinPusher

- 定义：void acceptJoinPusher(string userID)
- 说明：大主播接受连麦请求，并通知给连麦发起方
- 参数：

参数	类型	描述
userID	string	用户ID

- 示例：

```
m_liveRoom.acceptJoinPusher(userID);
```

24. rejectJoinPusher

- 定义：void rejectJoinPusher(string userID, string reason)
- 说明：大主播拒绝连麦请求，并通知给连麦发起方
- 参数：

参数	类型	描述

参数	类型	描述
userID	string	用户ID
reason	string	拒绝的原因

- 示例：

```
m_liveRoom.acceptJoinPusher(userID, reason);
```

25. kickoutSubPusher

- 定义：void kickoutSubPusher(string userID)
- 说明：大主播踢掉某一个小主播
- 参数：

参数	类型	描述
userID	string	用户ID

- 示例：

```
m_liveRoom.kickoutSubPusher(userID);
```

ILoginLiveCallback回调接口的详情

1. onLogin

- 定义：void onLogin(LiveResult res, LiveAuthData authData)
- 说明：login登录结果的回调
- 参数：

参数	类型	描述

参数	类型	描述
res	LiveResult	执行结果，参考 LiveRoomUtil.h 中定义的 LiveResult 类
authData	LiveAuthData	RoomService提供的登录信息，包括IM相关的配置字段，在login成功后，获取到token字段

- 示例：

```
class LoginDialog : ILoginLiveCallback
{
void ILoginLiveCallback.onLogin(LiveResult res, LiveAuthData authData)
{
if (LIVEROOM_SUCCESS != res.ec)
{
// 登录失败
}
else
{
// 登录成功，保存authData信息
}
}
}

// 参见LiveRoom::login接口，了解回调的使用
...
```

IGetLiveRoomListCallback回调接口的详情

1. onGetRoomList

- 定义：void onGetRoomList(LiveResult res, List rooms)
- 说明：获取房间列表的回调
- 参数：

参数	类型	描述
res	LiveResult	执行结果，参考 LiveRoomUtil.h 中定义的 LiveResult 类

参数	类型	描述
rooms	List	房间信息的列表

- 示例：

```
class RoomListDialog : IGetLiveRoomListCallback
{
void IGetRoomListCallback.onGetRoomList(LiveResult res, List<LiveRoomData> rooms)
{
if (LIVEROOM_SUCCESS != res.ec)
{
// 查询失败
}
else
{
// 查询成功，处理rooms数据
}
}
}

// 参见LiveRoom::getRoomList接口，了解回调的使用
...
```

ILiveRoomCallback回调接口的详情

参见LiveRoom::setCallback接口，了解如何设置ILiveRoomCallback回调。

1. onCreateRoom

- 定义：void onCreateRoom(LiveResult res, string roomID)
- 说明：创建房间的回调
- 参数：

参数	类型	描述
res	LiveResult	执行结果，参考 LiveRoomUtil.h 中定义的 LiveResult 类

参数	类型	描述
roomID	string	房间ID

- 示例：

```
class MainDialog : ILiveRoomCallback
{
void ILiveRoomCallback.onCreateRoom(LiveResult res, string roomID)
{
if (LIVEROOM_SUCCESS != res.ec)
{
// 创建房间失败，弹框提示
}
else
{
// 创建房间成功，其他观众可以观看直播
// 紧接着处理onUpdateRoomData回调
}
}

...
}
```

2. onEnterRoom

- 定义：void onEnterRoom(LiveResult res)
- 说明：进入房间的回调
- 参数：

参数	类型	描述
res	LiveResult	执行结果，参考 LiveRoomUtil.h 中定义的 LiveResult 类

- 示例：

```
class MainDialog : ILiveRoomCallback
{
void ILiveRoomCallback.onEnterRoom(LiveResult res)
```

```
{
if (LIVEROOM_SUCCESS != res.ec)
{
// 进入房间失败，弹框提示
}
else
{
// 进入房间成功，紧接着处理onUpdateRoomData回调
}
}

...
}
```

3. onUpdateRoomData

- 定义：void onUpdateRoomData(LiveResult res, LiveRoomData roomData)
- 说明：房间信息变更的回调
- 参数：

参数	类型	描述
res	LiveResult	执行结果，参考 LiveRoomUtil.h 中定义的 LiveResult 类
roomData	LiveRoomData	房间信息，参考 LiveRoomUtil.h 中定义的 LiveRoomData 类

- 示例：

```
class MainDialog : ILiveRoomCallback
{
void ILiveRoomCallback.onUpdateRoomData(LiveResult res, LiveRoomData roomData)
{
if (LIVEROOM_SUCCESS != res.ec)
{
// 更新失败，弹框提示
}
else
{
// 更新成功，根据createRoom还是enterRoom导致这个接口的触发来做出相应的处理
}
```

```
}

...
}
```

4. onGetAudienceList

- 定义：void onGetAudienceList(LiveResult res, List audiences)
- 说明：查询观众列表的回调
- 参数：

参数	类型	描述
res	LiveResult	执行结果，参考 LiveRoomUtil.h 中定义的 LiveResult 类
audiences	List	观众信息的列表

- 示例：

```
class MainDialog : ILiveRoomCallback
{
void ILiveRoomCallback.onGetAudienceList(LiveResult res, List<LiveAudienceData> audiences)
{
if (LIVEROOM_SUCCESS != res.ec)
{
// 查询失败，弹框提示
}
else
{
// 查询成功，列表更新到UI
}
}

...
}
```

5. onPusherJoin

- 定义：void onPusherJoin(LiveMemberData member)
- 说明：连麦观众进入房间的回调

- 参数：

参数	类型	描述
member	LiveMemberData	成员信息，参考 LiveRoomUtil.h 中定义的 LiveMemberData 类

- 示例：

```
class MainDialog : ILiveRoomCallback
{
void ILiveRoomCallback.onPusherJoin(LiveMemberData member)
{
// 通常连麦观众请求连麦，打开连麦观众的画面和音频
m_liveRoom.addRemoteView(rendHwnd, rect, member.userID);
...
}

...
}
```

6. onPusherQuit

- 定义：void onPusherQuit(LiveMemberData member)
- 说明：连麦观众退出房间的回调
- 参数：

参数	类型	描述
member	LiveMemberData	成员信息，参考 LiveRoomUtil.h 中定义的 LiveMemberData 类

- 示例：

```
class MainDialog : ILiveRoomCallback
{
void ILiveRoomCallback.onPusherQuit(LiveMemberData member)
{
// 通常连麦观众关闭连麦，关闭连麦观众的画面和音频
m_liveRoom.removeRemoteView(member.userID);
```

```
...
}

...
}
```

7. onRoomClosed

- 定义：void onRoomClosed(string roomID)
- 说明：房间解散的回调
- 参数：

参数	类型	描述
roomID	string	房间ID

- 示例：

```
class MainDialog : ILiveRoomCallback
{
void ILiveRoomCallback.onRoomClosed(string roomID)
{
// 提示房间解散，关闭已打开的音视频
...
}

...
}
```

8. onRecvRoomTextMsg

- 定义：void onRecvRoomTextMsg(string roomID, string userID, string userName, string userAvatar, string msg)
- 说明：收到普通文本消息
- 参数：

参数	类型	描述
roomID	string	房间ID
userID	string	发送者ID
userName	string	发送者昵称
userAvatar	string	发送者头像
message	string	文本消息内容

- 示例：

```
class MainDialog : ILiveRoomCallback
{
void ILiveRoomCallback.onRecvRoomTextMsg(string roomID, string userID, string userName, string userAvatar, string msg)
{
// IM面板显示文本消息
}

...
}
```

9. onRecvRoomCustomMsg

- 定义：void onRecvRoomCustomMsg(string roomID, string userID, string userName, string userAvatar, string cmd, string message)
- 说明：收到自定义消息
- 参数：

参数	类型	描述
roomID	string	房间ID
userID	string	发送者ID
userName	string	发送者昵称

参数	类型	描述
userAvatar	string	发送者头像
cmd	string	自定义cmd
message	string	自定义消息内容

- 示例：

```
class MainDialog : ILiveRoomCallback
{
void ILiveRoomCallback.onRecvRoomCustomMsg(string roomID, string userID, string userName, string userAvatar, string cmd, string message)
{
// 解析和处理自定义消息，例如点赞、礼物等
}

...
}
```

10. onError

- 定义：void onError(LiveResult res, string userID)
- 说明：LiveRoom内部发生错误的通知
- 参数：

参数	类型	描述
res	LiveResult	执行结果，参考 LiveRoomUtil.h 中定义的 LiveResult 类
userID	string	用户ID

- 示例：

```
class MainDialog : ILiveRoomCallback
{
void ILiveRoomCallback.onError(LiveResult res, string userID)
{
```

```
// 弹框提示错误，根据错误严重程序，是否关闭直播
}

...
}
```

11. onRecvJoinPusherRequest

- 定义：void onRecvJoinPusherRequest(string roomID, string userID, string userName, string userAvatar)
- 说明：接收到连麦请求
- 参数：

参数	类型	描述
roomID	string	房间ID
userID	string	发送者ID
userName	string	发送者昵称
userAvatar	string	发送者头像

- 示例：

```
class MainDialog : ILiveRoomCallback
{
void ILiveRoomCallback.onRecvJoinPusherRequest(string roomID, string userID, string userName, string userAvatar)
{
// 大主播接收到观众的连麦请求
// 先判断roomID是否合法
// UI显示谁发起请求的提示
// 大主播可以通过acceptJoinPusher接受连麦，或者rejectJoinPusher拒绝连麦
}

...
}
```

12. onRecvAcceptJoinPusher

- 定义：void onRecvAcceptJoinPusher(string roomID, string msg)
- 说明：接收到接受连麦请求的回复
- 参数：

参数	类型	描述
roomID	string	房间ID
msg	string	消息

- 示例：

```
class MainDialog : ILiveRoomCallback
{
void ILiveRoomCallback.onRecvAcceptJoinPusher(string roomID, string msg)
{
// 连麦观众接收到大主播的接收连麦的回复
// 先判断roomID是否合法
// 处理msg
}

...
}
```

13. onRecvRejectJoinPusher

- 定义：void onRecvRejectJoinPusher(string roomID, string msg)
- 说明：接收到拒绝连麦请求的回复
- 参数：

参数	类型	描述
roomID	string	房间ID

参数	类型	描述
msg	string	消息

- 示例：

```
class MainDialog : ILiveRoomCallback
{
void ILiveRoomCallback.onRecvRejectJoinPusher(string roomID, string msg)
{
// 连麦观众接收到大主播的拒绝连麦的回复
// 先判断roomID是否合法
// 处理msg
}

...
}
```

14. onRecvKickoutSubPusher

- 定义：void onRecvKickoutSubPusher(string roomID)
- 说明：接收大主播踢小主播的通知
- 参数：

参数	类型	描述
roomID	string	房间ID

- 示例：

```
class MainDialog : ILiveRoomCallback
{
void ILiveRoomCallback.onRecvRejectJoinPusher(string roomID, string msg)
{
// 大主播踢掉某个正在连麦的观众
// 先判断roomID是否合法
```

```
// 处理msg
}

...
}
```

视频通话（RTCRoom）

最近更新时间：2018-07-10 11:25:15

实时音视频（RTCRoom ）接口（CSharp）

RTCRoom

名称	描述
instance()	获取RTCRoom单例，通过单例调用RTCRoom的接口
setCallback(IRTCRoomCallback callback)	设置回调 RTCRoom 的回调代理，监听 RTCRoom 的内部状态和接口的执行结果
login(string serverDomain, RTCAuthData authData, ILoginRTCCallback callback)	登录业务服务器RoomService，登录后才能够正常使用其他接口和使用IM功能
logout()	登出业务服务器RoomService，请注意在leaveRoom调用后，再调用logout，否则leaveRoom会调用失败
getRoomList(int index, int cnt, IGetRTCRoomListCallback callback)	获取房间列表，房间数量比较多时，可以分页获取
createRoom(string roomID, string roomInfo)	创建房间，后台的房间列表中会添加一个新房间，同时会议创建者端会进入推流模式
enterRoom(string roomID)	进入房间
leaveRoom()	离开房间，如果是会议创建者，这个房间会被后台销毁，如果是会议参与者，不影响其他人继续通话
sendRoomTextMsg(string msg)	在房间内，发送普通文本消息，比如发送弹幕
sendRoomCustomMsg(string cmd, string msg)	在房间内，发送普通自定义消息，比如发送点赞、送花等消息
startLocalPreview(IntPtr rendHwnd, Rectangle rect)	启动默认的摄像头预览

名称	描述
updateLocalPreview(IntPtr rendHwnd, Rectangle rect)	重设摄像头预览区域，当您指定的本地 HWND 的窗口尺寸发生变化时，可以通过这个函数重新调整视频渲染区域
stopLocalPreview()	关闭摄像头预览
startScreenPreview(IntPtr rendHwnd, IntPtr captureHwnd, Rectangle renderRect, Rectangle captureRect)	启动屏幕分享
stopScreenPreview()	关闭屏幕分享
addRemoteView(IntPtr rendHwnd, Rectangle rect, string userID)	播放房间内其他会议参与者的视频
updateRemotePreview(IntPtr rendHwnd, Rectangle rect, string userID)	重设指定userID的视频预览区域，当您指定的本地 HWND 的窗口尺寸发生变化时，可以通过这个函数重新调整视频渲染区域
removeRemoteView(string userID)	停止播放其他会议参与者的视频
setMute(bool mute)	静音接口
setBeautyStyle(RTCBeautyStyle beautyStyle, int beautyLevel, int whitenessLevel)	设置美颜和美白效果

ILoginRTCCallback

名称	描述
onLogin(RTCResult res, RTCAuthData authData)	login登录结果的回调

IGetRTCRoomListCallback

名称	描述
onGetRoomList(RTCResult res, List rooms)	获取房间列表的回调

IRTCRoomCallback

名称	描述
onCreateRoom(RTCResult res, string roomID)	创建房间的回调
onEnterRoom(RTCResult res)	进入房间的回调
onUpdateRoomData(RTCResult res, RTCRoomData roomData)	房间信息变更的回调
onPusherJoin(RTCMemberData member)	成员进入房间的回调
onPusherQuit(RTCMemberData member)	成员退出房间的回调
onRoomClosed(string roomID)	房间解散的回调
onRecvRoomTextMsg(string roomID, string userID, string userName, string userAvatar, string msg)	收到普通文本消息
onRecvRoomCustomMsg(string roomID, string userID, string userName, string userAvatar, string cmd, string message)	收到自定义消息
onError(RTCResult res, string userID)	RTCRoom内部发生错误的通知

RTCRoom对象接口的详情

1. instance

- 定义：static RTCRoom instance()
- 说明：获取RTCRoom单例，通过单例调用RTCRoom的接口
- 示例：

```
RTCRoom mRTCRoom = RTCRoom.instance();
```

2. setCallback

- 定义：void setCallback(IRTCRoomCallback callback)
- 说明：设置回调 RTCRoom 的回调代理，监听 RTCRoom 的内部状态和接口的执行结果
- 参数：

参数	类型	描述
callback	IRTCRoomCallback	IRTCRoomCallback 类型的接口

- 示例：

```
class MainDialog : IRTCRoomCallback
{
void IRTCRoomCallback.onCreateRoom(RTCResult res, string roomID){}
void IRTCRoomCallback.onEnterRoom(RTCResult res){}
void IRTCRoomCallback.onUpdateRoomData(RTCResult res, RTCRoomData roomData){}
...
};

MainDialog::MainDialog()
{
mRTCRoom.setCallback(this); // 设置回调
}

...
```

3. login

- 定义：void login(string serverDomain, RTCAuthData authData, ILoginRTCCallback callback)
- 说明：登录业务服务器RoomService，登录后才能够正常使用其他接口和使用IM功能
- 参数：

参数	类型	描述

参数	类型	描述
serverDomain	string	RoomService的URL地址，安全起见，建议访问https加密链接，参考 DOC
authData	RTCAuthData	RoomService提供的登录信息，包括IM相关的配置字段，在login成功后，获取到token字段，参考 DOC
callback	ILoginRTCCallback	ILoginRTCCallback 类型的接口，回调login的结果

- 示例：

```
string serverDomain = "https://roomtest.qcloud.com/weapp/double_room";

m_authData.sdkAppID = sdkAppID;
m_authData.accountType = accountType;
m_authData.userID = userID;
m_authData.userSig = userSig;
m_authData.userName = userName;
m_authData.userAvatar = userAvatar;
m_authData.token = token;

// 该类实现ILoginCallback接口，获取login结果
mRTCRoom.login(serverDomain, m_authData, this);
```

4. logout

- 定义：void logout()
- 说明：登出业务服务器RoomService，请注意在leaveRoom调用后，再调用logout，否则leaveRoom会调用失败
- 示例：

```
mRTCRoom.logout();
```

5. getRoomList

- 定义：void getRoomList(int index, int count, IGetRTCRoomListCallback callback)

- 说明：获取房间列表，房间数量比较多时，可以分页获取
- 参数：

参数	类型	描述
index	int	分页获取，初始默认可设置为0，后续获取为起始房间索引（如第一次设置index=0, cnt=5,获取第二页时可用index=5）
count	int	每次调用，最多返回房间个数；0表示所有满足条件的房间
callback	IGetRTCRoomListCallback	IGetRTCRoomListCallback 类型的接口，查询结果的回调

- 示例：

```
// 此处从index=0开始，最多查询20个房间，该类实现IGetRoomListCallback接口，获取查询结果
mRTCRoom.getRoomList(0, 20, this);
```

6. createRoom

- 定义：void createRoom(string roomID, string roomInfo)
- 说明：创建房间，后台的房间列表中会添加一个新房间，同时会议创建者会进入推流模式
- 参数：

参数	类型	描述
roomID	string	房间ID，若传入空字符串，则后台会为您分配roomID，否则，传入的roomID作为这个房间的ID
roomInfo	string	自定义数据，该字段包含在房间信息中，推荐您将 roomInfo 定义为 json 格式，这样可以有很强的扩展性

- 示例：

```
// 后台会为您分配roomID
mRTCRoom.createRoom("", "{\"roomName\": \"My first room\"}");
```

7. enterRoom

- 定义：void enterRoom(string roomID)
- 说明：进入房间
- 参数：

参数	类型	描述
roomID	string	roomID - 要进入的房间ID，在 getRoomList 接口房间列表中查询得到

- 示例：

```
mRTCRoom.enterRoom(roomID);
```

8. leaveRoom

- 定义：void leaveRoom()
- 说明：离开房间，如果是会议创建者，这个房间会被后台销毁，如果是会议参与者，不影响其他人继续通话
- 示例：

```
mRTCRoom.leaveRoom();
```

9. sendRoomTextMsg

- 定义：void sendRoomTextMsg(string msg)
- 说明：发送普通文本消息
- 参数：

参数	类型	描述
msg	string	文本消息

- 示例：

```
mRTCRoom.sendRoomTextMsg("Hello RTCRoom");
```

10. sendRoomCustomMsg

- 定义：void sendRoomCustomMsg(string cmd, string msg)
- 说明：发送自定义消息
- 参数：

参数	类型	描述
cmd	string	自定义cmd，收发双方协商好的cmd
msg	string	自定义消息

- 示例：

```
// 假设收发双方协商好的cmd有Smile, Anger
mRTCRoom.sendRoomCustomMsg("Anger", "I am hungry");
```

11. startLocalPreview

- 定义：void startLocalPreview(IntPtr rendHwnd, Rectangle rect)
- 说明：启动默认的摄像头预览
- 参数：

参数	类型	描述
rendHwnd	IntPtr	承载预览画面的 HWND，目前 SDK 是采用 OpenGL 向 HWND 上绘制图像的,rendHwnd = null时无需预览视频
rect	Rectangle	指定视频图像在 HWND 上的渲染区域

- 示例：

```
mRTCRoom.startLocalPreview(panel_main.Handle, new Rectangle(0, 0, panel_main.Width, panel_main.Height));
```

12. updateLocalPreview

- 定义：void updateLocalPreview(IntPtr rendHwnd, Rectangle rect)
- 说明：重设摄像头预览区域，当您指定的本地 HWND 的窗口尺寸发生变化时，可以通过这个函数重新调整视频渲染区域
- 参数：

参数	类型	描述
rendHwnd	IntPtr	承载预览画面的 HWND，目前 SDK 是采用 OpenGL 向 HWND 上绘制图像的,rendHwnd = null时无需预览视频
rect	Rectangle	指定视频图像在 HWND 上的渲染区域

- 示例：

```
mRTCRoom.updateLocalPreview(panel_main.Handle, new Rectangle(0, 0, panel_main.Width, panel_main.Height));
```

13. stopLocalPreview

- 定义：void stopLocalPreview()
- 说明：关闭摄像头预览
- 示例：

```
mRTCRoom.stopLocalPreview();
```

14. startScreenPreview

- 定义：bool startScreenPreview(IntPtr rendHwnd, IntPtr captureHwnd, Rectangle renderRect, Rectangle captureRect)
- 说明：启动屏幕分享
- 参数：

参数	类型	描述
rendHwnd	IntPtr	承载预览画面的 HWND，目前 SDK 是采用 OpenGL 向 HWND 上绘制图像的,rendHwnd = null时无需预览视频
captureHwnd	IntPtr	指定录取窗口，若为null，则 captureRect 不起效，并且录取整个屏幕；若不为null，则录取这个窗口的画面
renderRect	Rectangle	指定视频图像在 rendHwnd 上的渲染区域
captureRect	Rectangle	指定录取窗口客户区的区域

- 返回值：成功 or 失败
- 示例：

```
mRTCRoom.startScreenPreview(panel_main.Handle, null, new Rectangle(0, 0, panel_main.Width, panel_main.Height), null);
```

15. stopScreenPreview

- 定义：void stopScreenPreview()
- 说明：关闭屏幕分享
- 示例：

```
mRTCRoom.stopScreenPreview();
```

16. addRemoteView

- 定义：void addRemoteView(IntPtr rendHwnd, Rectangle rect, string userID)

- 说明：播放房间内其他会议参与者的视频
- 参数：

参数	类型	描述
rendHwnd	IntPtr	承载预览画面的 HWND
rect	Rectangle	指定视频图像在 HWND 上的渲染区域
userID	string	用户ID

- 示例：

```
// 打开用户的音视频播放
mRTCRoom.addRemoteView(panel_display.Handle, new Rectangle(0, 0, panel_display.Width, panel_display.Height), userID);
```

17. updateRemotePreview

- 定义：void updateRemotePreview(IntPtr rendHwnd, Rectangle rect, string userID)
- 说明：重设指定userID的视频预览区域，当您指定的本地 HWND 的窗口尺寸发生变化时，可以通过这个函数重新调整视频渲染区域
- 参数：

参数	类型	描述
rendHwnd	IntPtr	承载预览画面的 HWND
rect	Rectangle	指定视频图像在 HWND 上的渲染区域
userID	string	用户ID

- 示例：

```
// 更新用户的音视频播放
mRTCRoom.updateRemotePreview(panel_display.Handle, new Rectangle(0, 0, panel_display.Width, panel_display.Height), userID);
```

18. removeRemoteView

- 定义：void removeRemoteView(string userID)
- 说明：停止播放其他会议参与者的视频
- 参数：

参数	类型	描述
userID	string	用户ID

- 示例：

```
mRTCRoom.removeRemoteView(userID);
```

19. setMute

- 定义：void setMute(bool mute)
- 说明：静音接口
- 参数：

参数	类型	描述
mute	bool	是否静音

- 示例：

```
mRTCRoom.setMute(true); // 设置为静音
```

20. setBeautyStyle

- 定义：void setBeautyStyle(RTCBeautyStyle beautyStyle, int beautyLevel, int whitenessLevel)
- 说明：设置美颜和美白效果
- 参数：

参数	类型	描述
beautyStyle	RTCBeautyStyle	参考 RTCRoomUtil.cs 中定义的 RTCBeautyStyle 枚举值
beautyLevel	int	美颜级别取值范围 1 ~ 9； 0 表示关闭，1 ~ 9值越大，效果越明显
whitenessLevel	int	美白级别取值范围 1 ~ 9； 0 表示关闭，1 ~ 9值越大，效果越明显

- 示例：

```
// 自然，美颜度数5，美白度数5
mRTCRoom.setBeautyStyle(RTCBeautyStyle.RTCROOM_BEAUTY_STYLE_NATURE, 5, 5);
```

ILoginRTCCallback回调接口的详情

1. onLogin

- 定义：void onLogin(RTCResult res, RTCAuthData authData)
- 说明：login登录结果的回调
- 参数：

参数	类型	描述
res	RTCResult	执行结果，参考 RTCRoomUtil.cs中定义的 RTCResult 类
authData	RTCAuthData	RoomService提供的登录信息，包括IM相关的配置字段，在login成功后，获取到token字段

- 示例：

```
class LoginDialog : ILoginRTCCallback
{
void onLogin(RTCResult res, RTCAuthData authData)
{
if (RTCErrorCode.RTCROOM_SUCCESS != res.ec)
{
// 登录失败
}
else
```

```
{
// 登录成功，保存authData信息
}
}
}

// 参见RTCRoom.login接口，了解回调的使用
...
```

IGetRTCRoomListCallback回调接口的详情

1. onGetRoomList

- 定义：void onGetRoomList(RTCResult res, List rooms)
- 说明：获取房间列表的回调
- 参数：

参数	类型	描述
res	RTCResult	执行结果，参考 RTCRoomUtil.cs 中定义的 RTCResult 类
rooms	List	房间信息的列表

- 示例：

```
class RoomListDialog : IGetRTCRoomListCallback
{
void onGetRoomList(RTCResult res, List<RTCRoomData> rooms)
{
if (RTCErrorCode.RTCROOM_SUCCESS != res.ec)
{
// 查询失败
}
else
{
// 查询成功，处理rooms数据
}
}
}
```

```
// 参见RTCRoom.getRoomList接口，了解回调的使用
...
```

IRTCRoomCallback回调接口的详情

参见RTCRoom.setCallback接口，了解如何设置IRTCRoomCallback回调。

1. onCreateRoom

- 定义：void onCreateRoom(RTCResult res, string roomID)
- 说明：创建房间的回调
- 参数：

参数	类型	描述
res	RTCResult	执行结果，参考 RTCRoomUtil.cs 中定义的 RTCResult 类
roomID	string	房间ID

- 示例：

```
class MainDialog : IRTCRoomCallback
{
void onCreateRoom(RTCResult res, string roomID)
{
if (RTCErrorCode.RTCROOM_SUCCESS != res.ec)
{
// 创建房间失败，弹框提示
}
else
{
// 创建房间成功，其他观众可以参与会议
// 紧接着处理onUpdateRoomData回调
}
}

...
}
```

2. onEnterRoom

- 定义：void onEnterRoom(RTCResult res)
- 说明：进入房间的回调
- 参数：

参数	类型	描述
res	RTCResult	执行结果，参考 RTCRoomUtil.cs 中定义的 RTCResult 类

- 示例：

```
class MainDialog : IRTCRoomCallback
{
void onEnterRoom(RTCResult res)
{
if (RTCErrorCode.RTCROOM_SUCCESS != res.ec)
{
// 进入房间失败，弹框提示
}
else
{
// 进入房间成功，紧接着处理onUpdateRoomData回调
}
}

...
}
```

3. onUpdateRoomData

- 定义：void onUpdateRoomData(RTCResult res, RTCRoomData roomData)
- 说明：房间信息变更的回调
- 参数：

参数	类型	描述
res	RTCResult	执行结果，参考 RTCRoomUtil.cs 中定义的 RTCResult 类
roomData	RTCRoomData	房间信息，参考 RTCRoomUtil.cs 中定义的 RTCRoomData 类

- 示例：

```
class MainDialog : IRTCRoomCallback
{
void onUpdateRoomData(RTCResult res, RTCRoomData roomData)
{
if (RTCErrorCode.RTCROOM_SUCCESS != res.ec)
{
// 更新失败，弹框提示
}
else
{
// 更新成功，根据createRoom还是enterRoom导致这个接口的触发来做出相应的处理
}
}

...
}
```

4. onPusherJoin

- 定义：void onPusherJoin(RTCMemberData member)
- 说明：成员进入房间的回调
- 参数：

参数	类型	描述
member	RTCMemberData	成员信息，参考 RTCRoomUtil.cs 中定义的 RTCMemberData 类

- 示例：

```
class MainDialog : IRTCRoomCallback
{
```

```
void onPusherJoin(RTCMemberData member)
{
// 成员进入房间，打开成员的画面和音频
this.BeginInvoke(new Action(() =>
{
mRTCRoom.addRemoteView(panel_display.Handle, new Rectangle(0, 0, panel_display.Width, panel
_display.Height), userID);
...
}));
}

...
}
```

5. onPusherQuit

- 定义：void onPusherQuit(RTCMemberData member)
- 说明：成员退出房间的回调
- 参数：

参数	类型	描述
member	RTCMemberData	成员信息，参考 RTCRoomUtil.cs 中定义的 RTCMemberData 类

- 示例：

```
class MainDialog : IRTCRoomCallback
{
void onPusherQuit(RTCMemberData member)
{
// 成员退出房间，关闭成员的画面和音频
this.BeginInvoke(new Action(() =>
{
mRTCRoom.removeRemoteView(member.userID);
...
}));
...
}
```

```
...
}
```

6. onRoomClosed

- 定义：void onRoomClosed(string roomID)
- 说明：房间解散的回调
- 参数：

参数	类型	描述
roomID	string	房间ID

- 示例：

```
class MainDialog : IRTCRoomCallback
{
void onRoomClosed(string roomID)
{
// 提示房间解散，关闭已打开的音视频
...
}

...
}
```

7. onRecvRoomTextMsg

- 定义：void onRecvRoomTextMsg(string roomID, string userID, string userName, string userAvatar, string msg)
- 说明：收到普通文本消息
- 参数：

参数	类型	描述

参数	类型	描述
roomID	string	房间ID
userID	string	发送者ID
userName	string	发送者昵称
userAvatar	string	发送者头像
message	string	文本消息内容

- 示例：

```
class MainDialog : IRTCRoomCallback
{
void onRecvRoomTextMsg(string roomID, string userID, string userName, string userAvatar, string msg)
{
// IM面板显示文本消息
}

...
}
```

8. onRecvRoomCustomMsg

- 定义：void onRecvRoomCustomMsg(string roomID, string userID, string userName, string userAvatar, string cmd, string message)
- 说明：收到自定义消息
- 参数：

参数	类型	描述
roomID	string	房间ID
userID	string	发送者ID
userName	string	发送者昵称

参数	类型	描述
userAvatar	string	发送者头像
cmd	string	自定义cmd
message	string	自定义消息内容

- 示例：

```
class MainDialog : IRTCRoomCallback
{
void onRecvRoomCustomMsg(string roomID, string userID, string userName, string userAvatar, string cmd, string message)
{
// 解析和处理自定义消息，例如点赞、礼物等
}

...
}
```

9. onError

- 定义：void onError(RTCResult res, string userID)
- 说明：RTCRoom内部发生错误的通知
- 参数：

参数	类型	描述
res	RTCResult	执行结果，参考 RTCRoomUtil.cs 中定义的 RTCResult 类
userID	string	用户ID

- 示例：

```
class MainDialog : IRTCRoomCallback
{
void onError(RTCResult res, string userID)
{
```

```
    // 弹框提示错误，根据错误严重程序，是否解散或者退出会议
}

...
}
```