视沃科技

大牛直播 SDK

Windows 端调用说明

官网：https://daniulive.com

Github：https://github.com/daniulive/SmarterStreaming

声　明

非常感谢您选用我们的 SDK，您的支持将激励我们持续进步。

随着产品的迭代，产品手册会在每个版本发布后不定期更新，最新版本请以官方网站(https://daniulive.com)为准。

本手册中内容仅为开发者提供参考指导作用，具体调用请以 SDK 示例为准。

目 录

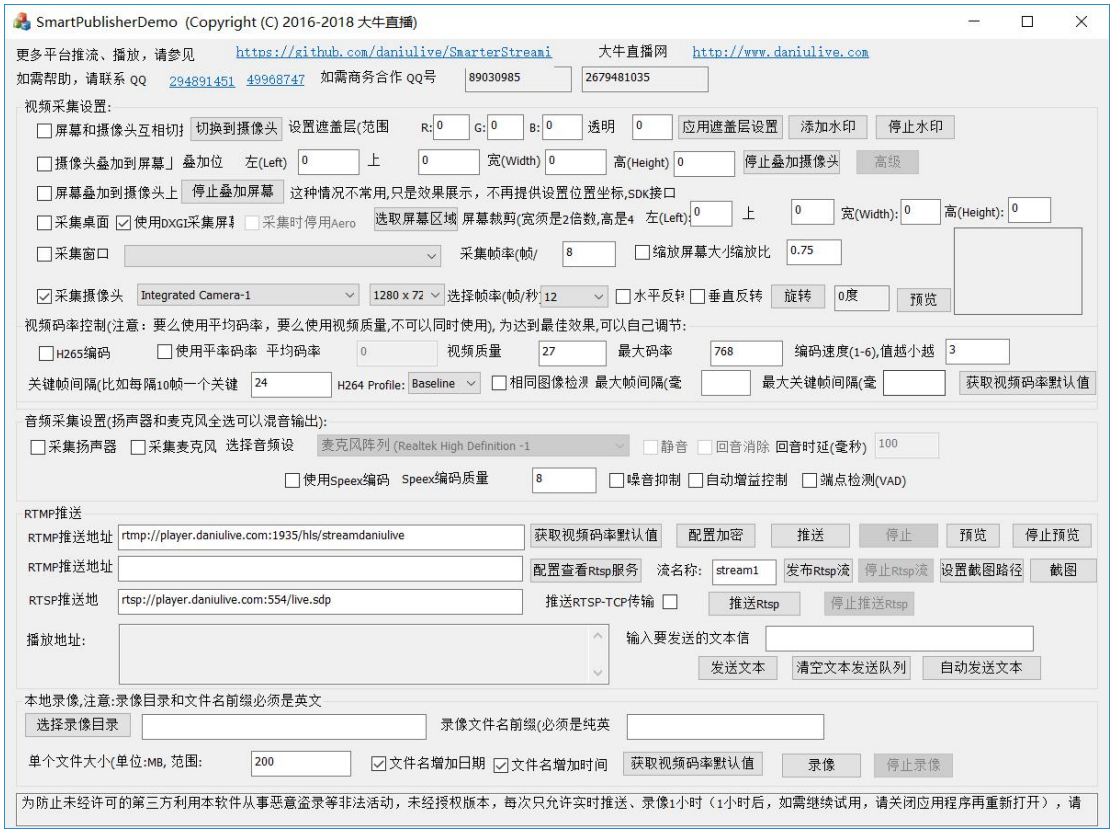
1. Windows 推送端 SDK 说明

- 大牛直播 SDK 自有框架，易于扩展，自适应算法让延迟更低、采集编码传输效率更高；
- 所有功能以 SDK 接口形式提供，所有状态，均有 event 回调，完美支持断网自动重连；
- SDK 模块化，可和大牛直播播放器 SDK 组合实现流媒体数据转发、连麦、一对一互动等场景；
- 推送叠加以层级模式提供，开发者可以自行组合数据源(如多摄像头/屏幕/水印叠加)；
- 支持外部 YUV/RGB/H.264/AAC/SPEEX/PCMA/PCMU 数据源接入；
- 所有参数均可通过 SDK 接口单独设置，亦可通过默认参数，傻瓜式设置；
- 推送、录像模块完全分离，可单独使用亦可组合使用；
- 业内甚至很难找到效果接近的 SDK。

1.1 demo 说明

- 大牛直播 SDK 提供 C++/C#两套接口，对外提供 32/64 位库，C++和 C#接口一一对应，C#接口比 C++接口增加前缀 NT_PB_。
- WIN-PublisherSDK-CPP-Demo：推送端 SDK 对应的 C++接口的 demo；
- WIN-PublisherSDK-CSharp-Demo：推送端 SDK 对应的 C#接口的 demo；
- 大牛直播推送端 SDK 支持 Win7 及以上系统。
- 本 demo 基于 VS2013 开发。

1.2 功能说明



功能列表：

- [视频采集处理]Windows 平台涵盖“Windows 视频采集处理 SDK”功能；
- [音频采集处理]Windows 平台涵盖“Windows 音频采集处理 SDK”功能；
- [本地预览]支持摄像头/屏幕/合成数据实时预览功能；
- [摄像头反转/旋转]支持摄像头水平反转、垂直反转、0°/90°/180°/270°旋转；
- [摄像头采集]除常规 YUV 格式外，还支持 MJPEG 格式的摄像头采集；
- [RTMP 推流]超低延时的 RTMP 协议直播推流 SDK（Windows 64 位库支持 RTMP 扩展 H.265 推送）；
- [音视频加密]RTMP 支持 AES128/AES192/AES256/SM4(国密)逐帧数据加密；
- [音视频加密]支持 RTMP H.264/H.265 加密；
- [音视频加密]支持 RTMP AAC/Speex/G711 加密；
- [视频格式]Windows 支持 H.264/H.265 编码；
- [音频格式]支持 AAC 编码和 Speex 编码；
- [音频编码]支持 Speex 推送、Speex 编码质量设置；
- [软硬编码参数配置]支持 gop 间隔、帧率、bit-rate 设置；
- [软编码参数配置]支持软编码 profile、软编码速度、可变码率设置；
- [多实例推送]支持多实例推送(如同时推送屏幕/摄像头和外部数据)；
- [RTMP 扩展 H.265]Windows/Android 推送 SDK 支持 RTMP 扩展 H.265 推送，Windows 针

对摄像头采集软编码，使用 H.265 可变码率，带宽大幅节省，效果直逼传统 H.265 编码摄像头；

- [多分辨率支持]支持摄像头或屏幕多种分辨率设置；
- [Windows 推屏]支持屏幕裁剪、窗口采集、屏幕/摄像头数据合成等多种模式推送；
- [事件回调]支持各种状态实时回调；
- [水印]Windows 平台支持文字水印、png 水印、实时遮挡；
- [复杂网络处理]支持断网重连等各种网络环境自动适配；
- [动态码率]支持根据网络情况自动调整推流码率；
- [实时静音]支持推送过程中，实时静音/取消静音；
- [实时快照]支持推流过程中，实时快照；
- [纯音频推流]支持仅采集音频流并发起推流功能；
- [纯视频推流]支持特殊场景下的纯视频推流功能；
- [降噪]支持环境音、手机干扰等引起的噪音降噪处理、自动增益、VAD 检测；
- [外部编码前视频数据对接]支持 YUV 数据对接；
- [外部编码前音频数据对接]支持 PCM 对接；
- [外部编码后视频数据对接]支持外部 H.264 数据对接；
- [外部编码后音频数据对接]外部 AAC/PCMA/PCMU/SPEEX 数据对接；
- [扩展录像功能]完美支持和录像 SDK 组合使用；
- [服务器兼容]支持支持自建服务器(如 Nginx、SRS)或 CDN。

1.3 集成说明

C++头文件：

- [类型定义]nt_type_define.h
- [Log 定义]smart_log.h
- [Log 定义]smart_log_define.h
- [音视频类型定义]nt_common_media_define.h
- [base code 定义]nt_base_code_define.h
- [publisher 接口]nt_smart_publisher_define.h
- [publisher 接口]nt_smart_publisher_sdk.h

C#头文件：

- [Log 定义]smart_log.cs
- [Log 定义]smart_log_define.cs
- [base code 定义]nt_base_code_define.cs
- [player 接口]nt_smart_publisher_define.cs
- [player 参数定义]nt_smart_publisher_sdk.cs

相关 Lib：

- **SmartLog.dll**
- **SmartLog.lib**
- **SmartPublisherSDK.dll**
- **SmartPublisherSDK.lib**

- avcodec-56.dll
- avdevice-56.dll
- avfilter-5.dll
- avformat-56.dll
- avutil-54.dll
- postproc-53.dll
- swresample-1.dll
- swscale-3.dll

集成步骤：

1. 把 lib 目录下 debug/release 库拷贝到需要集成的工程对应的 debug 或 release 目录下（确保 32 位/64 位库 debug/release 目录一一对应）；

   lib 目录如下：

   (1) 32 位 debug 库：debug

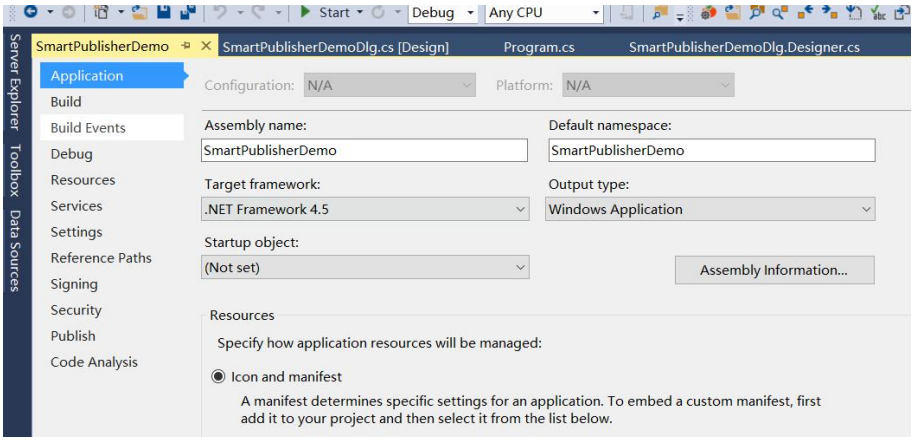
   (2) 32 位 release 库：release

   (3) 64 位 debug 库：x64\debug

   (4) 64 位 release 库：x64\release

2. 相关 cs 头文件，加入需要集成的工程；

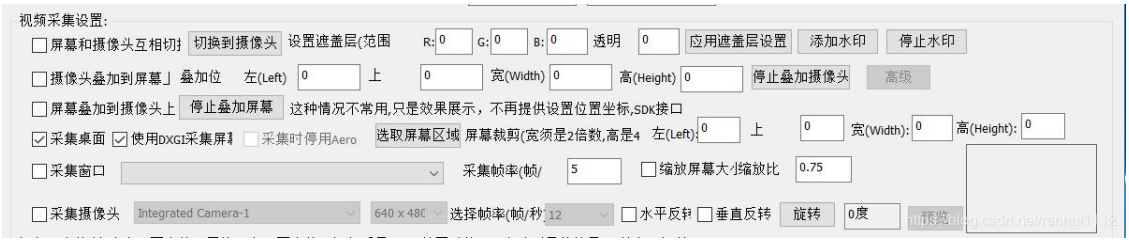
3. 在需要集成的工程，右键->Properties->Application->Assembly name，大牛直播按照 app 名称授权，未授权版本，此处请改成“SmartPulisherDemo”，如需授权，可直接联系商务：



1.4 功能详解

考虑到大牛直播推送端 SDK 功能相对复杂，以问答式：

1 视频采集设置



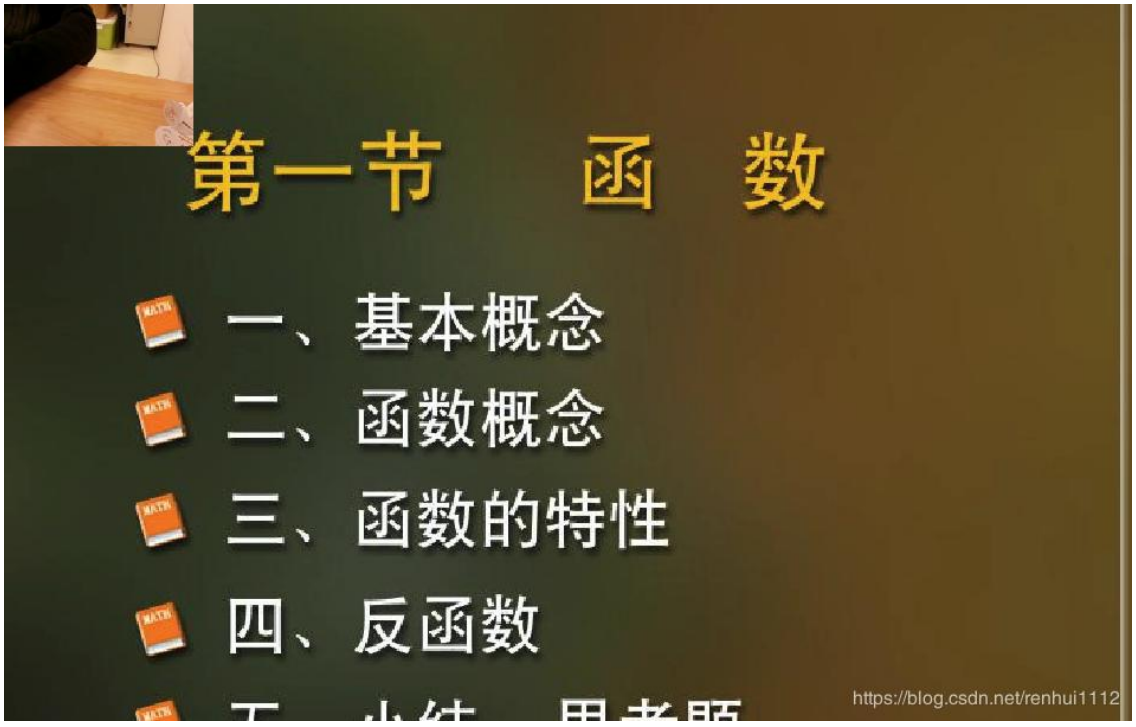
说明：

1. 屏幕和摄像头相互切换：用于在线教育或者无纸化等场景，推送或录像过程中，随时切换屏幕或摄像头数据（切换数据源），如需实时切换，点击页面“切换到摄像头”按钮即可；

2. 设置遮盖层，用于设定一个长方形或正方形区域（可自指定区域大小），遮盖不想给用户展示的部分；

3. 水印：添加 PNG 水印，支持推送或录像过程中，随时添加、取消水印；

4. 摄像头叠加到屏幕：意在用于同屏过程中，主讲人摄像头悬浮于屏幕之上（可指定叠加坐标），实现双画面展示，推送或录像过程中，可以随时取消摄像头叠加；



5. 屏幕叠加到摄像头：同 4，效果展示，实际根据需求实现；

6. 采集桌面：可以通过点击“选择屏幕区域”获取采集区域，并可在采集过程中，随时切

换区域位置，如不设定，默认全屏采集；

7. 使用 DXGI 采集屏幕，采集时停用 Aero；

8. 采集窗口：可设定需要采集的窗口，窗口放大或缩小，推送端会自适应码率和分辨率；

9. 采集帧率(帧/秒)：默认屏幕采集 5 帧，可根据实际场景需求设定到 10-25 帧；

10. 缩放屏幕大小缩放比：用于高清或超高清屏，通过设定一定的比例因子，缩放屏幕采集分辨率；

11. 采集摄像头：可选择需要采集的摄像头、采集分辨率、帧率、是否需要水平或者垂直反转、是否需要旋转；

追加提问：

问题[确认数据源]：采集桌面还是摄像头？如果桌面，全屏还是部分区域？

回答：

如果是摄像头：可以选择摄像头列表，然后分辨率、帧率。

如果是屏幕：默认帧率是 5 帧，可以根据实际场景调整，选取屏幕区域，可以实时拉取选择需要采集或录像区域；

如果是叠加模式：可选择摄像头叠加到屏幕，还是屏幕叠加到摄像头；

更高需求的用户，可以设置水印或应用层遮盖。

问题：如果是摄像头，采集到的摄像头角度不对怎么办？

回答：我们支持摄像头镜像和翻转设置，摄像头可通过 SDK 接口轻松实现水平/垂直翻转、镜像效果。

2 视频码率控制

我选可变码率还是平均码率？

回答：可变码率的优势在于，如果屏幕或摄像头变化不大，码率超低，特别是 H.265 编码，平均码率，码率比较均匀，需设置平均码率+最大码率，一般摄像头采集建议选择可变码率，屏幕采集选择平均码率，如需采用可变码率，请取消“使用平均码率”选项。

265 编码还是 H.264 编码？

回答：Windows 64 位库支持 H.265 编码，如果推 RTMP 流，需要服务器支持 RTMP H.265 扩展，播放器 SDK，也需要同步支持 RTMP H.265 扩展播放。

如果是轻量级 RTSP 服务 SDK 对接的话，只需要播放器支持 RTSP H.265 即可。

如果推摄像头数据，建议采用可变码率+H.265 编码。

如何设置码率参数更合理？

回答：

关键帧间隔：一般来说，设置到帧率的 2-4 倍，比如帧率 20，关键帧间隔可以设置到 40-80；

平均码率：可以点击“获取视频码率默认值”，最大码率是平均码率的 2 倍；

视频质量：如果使用可变码率，建议采用大牛直播 SDK 默认推荐视频质量值；

编码速度：如高分辨率，建议 1-3，值越小，编码速度越快；

H.264 Profile：默认 baseline profile，可根据需要，酌情设置 High profile；

NOTE：点击“推送”或“录像”或启动内置 RTSP 服务 SDK 之前，请务必设置视频码率，如不想手动设置，请点击“获取视频码率默认值”！！！

3 音频采集设置

问答式：采集音频吗？如果采集，采集麦克风还是扬声器的，亦或混音？

回答：

如果想采集电脑输出的音频（比如音乐之类），可以选择“采集扬声器”；

如果想采集麦克风音频，可以选择“采集麦克风”，并选择相关设备；

如果两个都想采集，可以两个都选择，混音输出。

4 音频编码

问题：是 AAC 还是 SPEEX？

回答：我们默认是 AAC 编码模式，如果需要码率更低，可以选择 SPEEX 编码模式，当然我们的 AAC 编码码率也不高。

5 音频处理

问题：我想过滤背景噪音怎么办？

回答：选中“噪音抑制”，“噪音抑制“请和“自动增益控制”组合使用，“端点检测(VAD)”可选设置。

问题：我想做一对一互动怎么办？

回答：选中“回音消除”，可以和“噪音抑制”、“自动增益控制”组合使用。

问题：我推送或者录像过程中，随时静音怎么办？

回答：推送过程中，随时选择或取消选择“静音”功能。

6 多路推送

问题：我想同时推送到多个 url 怎么办(比如一个内网服务器，一个外网服务器)？

回答：同时填写多个 url，然后点推送即可。

7 截图(快照)

问题：我想推送或者录像过程中，截取当前图像怎么办？

回答：那就设置好截图路径，推送或录像过程中，随时点击“截图”。

8 录像

问题：我还想录像，怎么办？

回答：设置录像文件存放目录，文件前缀、单个文件大小，是否加日期、时间，随时录制即可。

9 实时预览

问题：我还想看看视频特别是合成后的效果，怎么办？

回答：点击页面的“预览”按钮，就可以看到。

10 音视频加密

问题：我想我的数据走标准协议，但是加密流，怎么办？

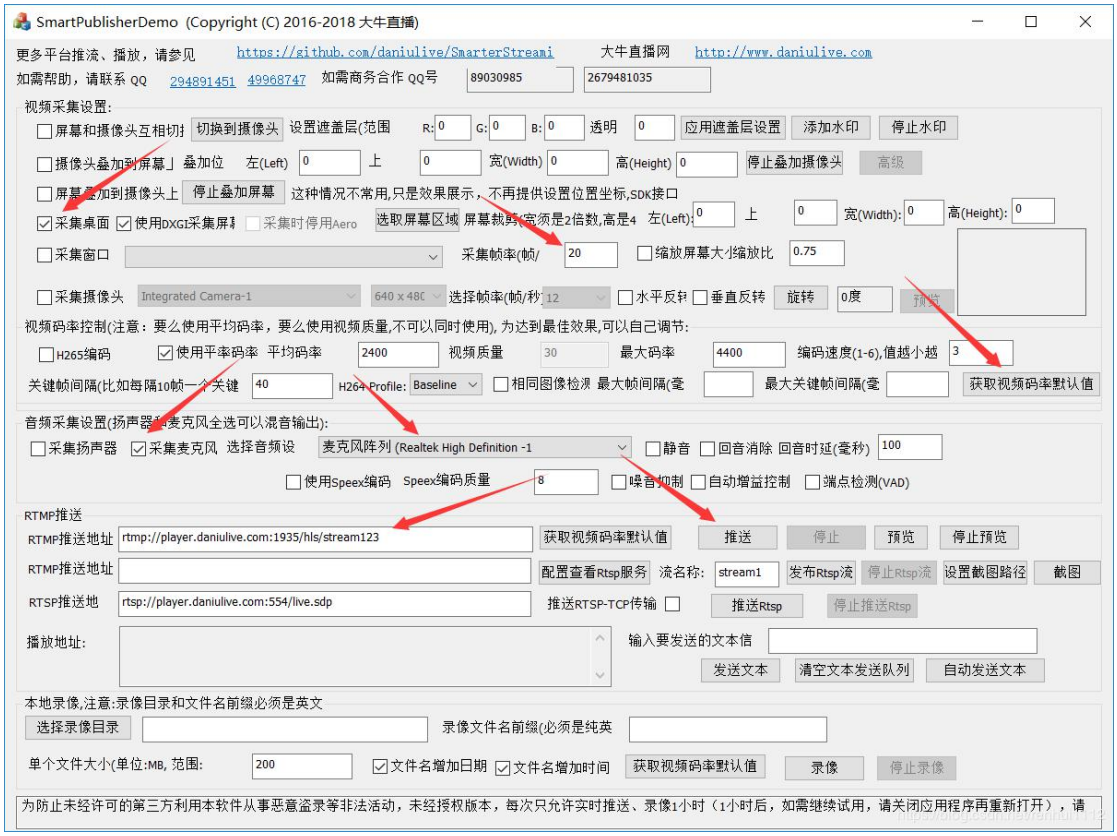
回答：大牛直播 SDK 的 RTMP 推流模块，支持 AES（AES128/AES192/AES256）和 SM4 加密。

11 如何快速测试

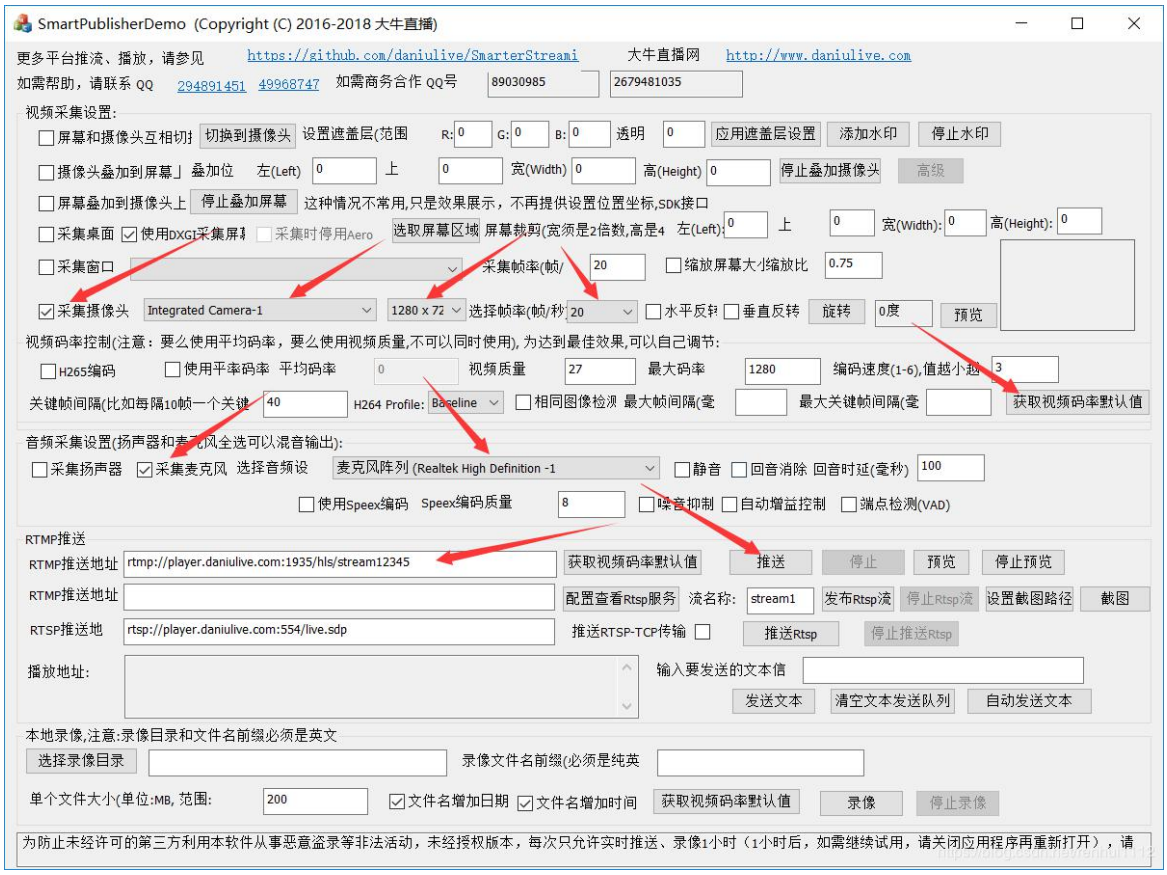
我是外行，我想快速测试推屏或推摄像头怎么办？

回答：

1. 推屏幕：


SmartPublisherDemo (Copyright (C) 2016-2018 大牛直播)
更多平台推流、播放，请参见
https://github.com/daniulive/SmarterStreami
大牛直播网
http://www.daniulive.com
如需帮助，请联系 QQ
294891451
49968747
加需商务合作 QQ号
89030985
2679481035
视频采集设置:
切换到摄像头
设置遮盖层(范围
应用遮盖层设置
添加水印
停止水印
停止叠加摄像头
高级
停止叠加屏幕
这种情况不常用,只是效果展示，不再提供设置位置坐标,SDK接口
选取屏幕区域
采集桌面
采集窗口
采集帧率(帧/
缩放屏幕大小缩放比
采集摄像头
Integrated Camera-1
旋转
预览
视频码率控制(注意：要么使用平均码率，要么使用视频质量,不可以同时使用), 为达到最佳效果,可以自己调节:
H265编码
使用平率码率
平均码率
2400
视频质量
最大码率
4400
编码速度(1-6),值越小越
关键帧间隔(比如每隔10帧一个关键
H264 Profile:
Baseline
相同图像检测
最大帧间隔(毫
最大关键帧间隔(毫
获取视频码率默认值
音频采集设置(扬声器和麦克风全选可以混音输出):
采集扬声器
采集麦克风
选择音频设
麦克风阵列 (Realtek High Definition -1
静音
回音消除
回音时延(毫秒)
100
使用Speex编码
Speex编码质量
噪音抑制
自动增益控制
端点检测(VAD)
RTMP推送
RTMP推送地址
rtmp://player.daniulive.com:1935/hls/stream123
获取视频码率默认值
推送
停止
预览
停止预览
RTMP推送地址
配置查看Rtsp服务
流名称:
stream1
发布Rtsp流
停止Rtsp流
设置截图路径
截图
RTSP推送地
rtsp://player.daniulive.com:554/live.sdp
推送RTSP-TCP传输
推送Rtsp
停止推送Rtsp
播放地址:
输入要发送的文本信
发送文本
清空文本发送队列
自动发送文本
本地录像,注意:录像目录和文件名前缀必须是英文
选择录像目录
录像文件名前缀(必须是纯英
单个文件大小(单位:MB, 范围:
200
文件名增加日期
文件名增加时间
获取视频码率默认值
录像
停止录像
为防止未经许可的第三方利用本软件从事恶意盗录等非法活动，未经授权版本，每次只允许实时推送、录像1小时（1小时后，如需继续试用，请关闭应用程序再重新打开），请

2. 推摄像头：



1.5 接口调用时序(以 C#为例)

1.5.1 初始化

NT_PB_Init

如需配置 log 路径，请在 NT_PB_Init 之前，做如下设置（目录可自行指定）：

```
// 设置日志路径(请确保目录存在)
//String log_path = "D:\\pulisherlog";
//NTSmartLog.NT_SL_SetPath(log_path);
```

1.5.2 Open

NT_PB_Open

1.5.3 设置回调事件

① NT_PB_SetEventCallBack：设置事件回调，如果想监听事件的话，建议调用 Open 成功后，就调用这个接口

② NT_PB_SetVideoPacketTimestampCallBack：设置视频包时间戳回调

③ NT_PB_SetPublisherStatusCallBack：设置推送状态回调

1.5.4 设置需要推送的 rtmp url

NT_PB_SetURL：rtmp 推送 url 设置

1.5.5 设置屏幕裁剪

① NT_PB_SetScreenClip：设置屏幕裁剪

② NT_PB_MoveScreenClipRegion：移动屏幕剪切区域，这个接口只能推送或者录像中调用

1.5.6 屏幕选取工具

① NT_PB_OpenScreenRegionChooseTool：打开一个屏幕选取工具的 toolHandle

② NT_PB_MoveScreenClipRegion：移动屏幕剪切区域，这个接口只能推送或者录像中调用

③ NT_PB_AllocateImage：分配 Image，分配后，SDK 内部会初始化这个结构体，失败的话返回 NULL

④ NT_PB_FreeImage：释放 Image，注意一定要调用这个接口释放内存，如果在你自己的模块中释放，Windows 会出问题的

⑤ NT_PB_CloneImage：克隆一个 Image， 失败返回 NULL

⑥ NT_PB_CopyImage：拷贝 Image，会先释放 dst 的资源，然后再拷贝

⑦ NT_PB_SetImagePlane： 给图像一个面设置数据，如果这个面已经有数据，将会释放掉再设置

⑧ NT_PB_LoadImage：加载 PNG 图片

1.5.7 设置屏幕采集参数

① NT_PB_EnableDXGIScreenCapturer：允许使用 DXGI 屏幕采集方式，这种方式需要 win8 及以上系统才支持

② NT_PB_DisableAeroScreenCapturer：采集屏幕时停用 Aero，这个只对 win7 有影响，win8 及以上系统，微软已经抛弃了 Aero Glass 效果

③ NT_PB_CheckCapturerWindow：判断顶层窗口能否能被捕获，如果不能被捕获的话返回 NT_ERC_FAILED（采集窗口）

④ NT_PB_SetCaptureWindow：设置要捕获的窗口的句柄（采集窗口）

1.5.8 设置摄像头采集参数

① NT_PB_StartGetVideoCaptureDeviceImage：获取句柄，且保存句柄
② NT_PB_FlipVerticalVideoCaptureDeviceImage：上下反转设备图像
③ NT_PB_FlipHorizontalVideoCaptureDeviceImage：水平反转设备图像
④ NT_PB_RotateVideoCaptureDeviceImage：旋转设备图像，顺时针旋转
⑤ NT_PB_GetVideoCaptureDeviceNumber：获取摄像头数量
⑥ NT_PB_GetVideoCaptureDeviceInfo：返回摄像头设备信息
⑦ NT_PB_GetVideoCaptureDeviceCapabilityNumber：返回摄像头能力数
⑧ NT_PB_GetVideoCaptureDeviceCapability:返回摄像头能力
⑨ NT_PB_DisableVideoCaptureResolutionSetting:

在多个实例推送多路时，对于一个摄像头来说,所有实例只能共享摄像头，那么只有一个实例可以改变摄像头分辨率，其他实例使用这个缩放后的图像；

在使用多实例时，调用这个接口禁止掉实例的分辨率设置能力.只留一个实例能改变分辨，如果不设置，行为未定义；

这个接口必须在 SetLayersConfig，AddLayerConfig 之前调用。

⑩ NT_PB_StartVideoCaptureDevicePreview：启动摄像头预览
⑪ NT_PB_FlipVerticalCameraPreview：上下反转摄像头预览图像
⑫ NT_PB_FlipHorizontalCameraPreview：水平反转摄像头预览图像
⑬ NT_PB_RotateCameraPreview：旋转摄像头预览图像，顺时针旋转
⑭ NT_PB_VideoCaptureDevicePreviewWindowSizeChanged：告诉 SDK 预览窗口大小改变
⑮ NT_PB_StopVideoCaptureDevicePreview：停止摄像头预览
⑯ NT_PB_GetVideoCaptureDeviceImage：调用这个接口可以获取摄像头图像
⑰ NT_PB_StopGetVideoCaptureDeviceImage：停止获取摄像头图像
⑱ NT_PB_SetVideoCaptureDeviceBaseParameter：设置摄像头信息
⑲ NT_PB_FlipVerticalCamera 上下反转摄像头图像
⑳ NT_PB_FlipHorizontalCamera：水平反转摄像头图像
㉑ NT_PB_RotateCamera：旋转摄像头图像，顺时针旋转

1.5.9 视频合成图层类型

```
public enum NT_PB_E_LAYER_TYPE : int
{
    NT_PB_E_LAYER_TYPE_SCREEN = 1,                  // 屏幕层
    NT_PB_E_LAYER_TYPE_CAMERA = 2,                  // 摄像头层
    NT_PB_E_LAYER_TYPE_RGBA_RECTANGLE = 3,          // RGBA 矩形
    NT_PB_E_LAYER_TYPE_IMAGE = 4,                   // 图片层
    NT_PB_E_LAYER_TYPE_EXTERNAL_VIDEO_FRAME = 5,    // 外部视频数据层
    NT_PB_E_LAYER_TYPE_WINDOW = 6,                    // 窗口层
}
```

1.5.10 音视频源类型

```
        /*定义 Video 源选项*/
        public enum NT_PB_E_VIDEO_OPTION : uint
        {
            NT_PB_E_VIDEO_OPTION_NO_VIDEO = 0x0,
            NT_PB_E_VIDEO_OPTION_SCREEN = 0x1, // 采集屏幕
            NT_PB_E_VIDEO_OPTION_CAMERA = 0x2, // 摄像头采集
            NT_PB_E_VIDEO_OPTION_LAYER = 0x3,  // 视频合并，比如桌面叠加摄像头等
            NT_PB_E_VIDEO_OPTION_ENCODED_DATA = 0x4, // 已经编码的视频数据，目前支持
H264
            NT_PB_E_VIDEO_OPTION_WINDOW = 0x5, // 采集窗口
        }

        /*定义 Auido 源选项*/
        public enum NT_PB_E_AUDIO_OPTION : uint
        {
            NT_PB_E_AUDIO_OPTION_NO_AUDIO = 0x0,
            NT_PB_E_AUDIO_OPTION_CAPTURE_MIC = 0x1,           // 采集麦克风音频
            NT_PB_E_AUDIO_OPTION_CAPTURE_SPEAKER = 0x2,           // 采集扬声器
            NT_PB_E_AUDIO_OPTION_CAPTURE_MIC_SPEAKER_MIXER = 0x3,    // 麦克风扬声
器混音
            NT_PB_E_AUDIO_OPTION_ENCODED_DATA = 0x4, // 编码后的音频数据，目前支持
AAC, speex 宽带(wideband mode)
        }
```

1.5.11 视频编码接口

① NT_PB_SetVideoEncoderType：设置编码类型，当前支持 h264 和 h265(注意:h265 只有 64 位 sdk 库支持，在 32 位库上设置会失败)；

② NT_PB_SetVideoQuality：设置视频质量，范围[0-20]，默认是 10，值越小质量越好，但码率会越大

③ NT_PB_SetVideoQualityV2：设置视频质量，范围[1-50]，值越小视频质量越好,但码率会越大．请优先考虑默认值；

④ NT_PB_SetFrameRate：设置帧率

⑤ NT_PB_SetVideoMaxBitRate：设置最大视频码率，单位 kbps

⑥ NT_PB_AddVideoEncoderBitrateGroupItem：

* 在一些特殊场景下，视频分辨率会改变，如果设置一个固定码率的的话,当视频分辨率变大的时候会变的模糊,变小的话又会浪费码率

* 所以提供可以设置一组码率的接口,满足不同分辨率切换的需求

* 规则：比如设置两组分辨率 640*360，640*480，那么当分辨率小于等于 640*360 时都使用 640*360 的码率，

* 当分辨率大于 640*360 且小于等于 640*480 时，就使用 640*480 的码率，如果分辨率大于 640*480 那就使用 640*480 的分辨率

* 为了设置的更准确，建议多划分几组，让区间变小

* 调用这个接口每次设置一组，设置多组就调用多次

* item 对应 NT_PB_VideoEncoderBitrateGroupItem

⑦ NT_PB_ClearVideoEncoderBitrateGroup：清除视频码率组

⑧ NT_PB_SetVideoKeyFrameInterval：设置关键帧间隔，比如 1 表示所有帧都是关键帧，10 表示每 10 帧里面一个关键帧，25 表示每 25 帧一个关键帧

⑨ NT_PB_SetVideoEncoderProfile：设置 H264 profile，1：H264 baseline(默认值)。2：H264 main。3。H264 high

⑩ NT_PB_SetVideoEncoderSpeed：设置 H264 编码速度，speed：范围是 1 到 6， 值越小，速度越快，质量也越差

⑪ NT_PB_SetVideoCompareSameImage：设置是否对图像进行相同比较，相同图像比较一般在采集桌面时有一定好处，可能能降低码率

⑫ NT_PB_SetVideoMaxKeyFrameInterval：设置视频最大关键帧间隔，这个接口一般不使用，这里是用来配合 SetVideoCompareSameImage 接口的，比如开启图像比较后，SDK 发现连续 20s 图像都是相同的，但播放端需要收到关键帧才能解码播放，所以需要一个限制

1.5.12 音频编码接口

① NT_PB_GetAuidoInputDeviceNumber：获取系统音频输入设备数

② NT_PB_GetAuidoInputDeviceName：获取音频输入设备名称

③ NT_PB_SetPublisherAudioCodecType：设置推送音频编码类型，type：1：使用 AAC 编码， 2：使用 speex 编码，其他值返回错误

④ NT_PB_SetPublisherSpeexEncoderQuality：设置推送 Speex 编码质量

⑤ NT_PB_SetAuidoInputDeviceId：设置音频输入设备 ID

⑥ NT_PB_IsCanCaptureSpeaker：检查是否能捕获扬声器音频

1.5.13 音频处理接口

① NT_PB_SetEchoCancellation：设置回音消除

② NT_PB_SetNoiseSuppression：设置音频噪音抑制

③ NT_PB_SetAGC：设置音频自动增益控制

④ NT_PB_SetVAD：设置端点检测(Voice Activity Detection (VAD))

1.5.14 图层合成等接口

① NT_PB_SetLayersConfig：设置视频合成层， 传入的是一个数组， 请正确填充每一层

② NT_PB_ClearLayersConfig：清除所有层配置，注意这个接口只能在推送或者录像之前调用，否则结果未定义

③ NT_PB_AddLayerConfig： 增加层配置，注意这个接口只能在推送或者录像之前调用，否则结果未定义

④ NT_PB_EnableLayer：动态禁止或者启用层

⑤ NT_PB_UpdateLayerConfigV2：更新层相关配置，注意不是层的所有字段都可以更新，只是部分可以更新，并且有些层没有字段可以更新，传入的参数，SDK 只选择能更新的字段更新，不能更新的字段会被忽略

⑥ NT_PB_UpdateLayerRegion：修改图层

⑦ NT_PB_PostLayerImage：给 index 层投递 Image 数据，目前主要是用来把 rgb 和 yuv 视频数据传给相关层

⑧ NT_PB_SetParam：万能接口，设置参数， 大多数问题， 这些接口都能解决

⑨ NT_PB_GetParam：万能接口，得到参数， 大多数问题，这些接口都能解决

1.5.15 RTMP 推送加密

① NT_PB_SetRtmpEncryptionOption：设置 rtmp 推送加密选项，可单独加密音频或视频

② NT_PB_SetRtmpEncryptionAlgorithm：设置 rtmp 加密算法，encryption_algorithm：加密算法，当前支持 aes 和国标 sm4．1 为 aes，2 为 sm4，默认为 aes

③ NT_PB_SetRtmpEncryptionKey：设置 rtmp 推送加密密钥，key：加密密钥，key_size：如果加密算法是 aes，key_size 必须是 16，24，32 这三个值，其他返回错误；如果加密算法是 sm4，key_size 必须是 16，其他值返回错误．

④ NT_PB_SetRtmpEncryptionIV：设置 rtmp 推送加密 IV(初始化向量)，这个接口不调用的话，将使用默认 IV，iv：初始化向量，iv_size：当前必须是 16，其他值返回错误

1.5.16 开始推送

NT_PB_StartPublisher

1.5.17 录像

⑤ NT_PB_SetRecorderDirectory：设置本地录像目录，必须是英文目录，否则会失败

⑥ NT_PB_SetRecorderFileMaxSize：设置单个录像文件最大大小，当超过这个值的时候，将切割成第二个文件

⑦ NT_PB_SetRecorderFileNameRuler：设置录像文件名生成规则

⑧ NT_PB_StartRecorder：启动录像

⑨ NT_PB_StopRecorder：停止录像

1.5.18 拉流接口(转发之用)

① NT_PB_PostVideoEncodedData：投递编码过的视频数据给 SDK(老接口)

② NT_PB_PostVideoEncodedDataV2：投递编码过的视频数据给 SDK V2 版，V2 版增加了 pts 参数，建议用 V2 接口

③ NT_PB_PostAudioEncodedData：投递编码过的音频数据给 SDK

1.5.19 实时静音(实时调用)

NT_PB_SetMute：设置推送静音

1.5.20 快照(实时调用)

NT_PB_CaptureImage：捕获图片

1.5.21 停止推送

NT_PB_Close

1.5.22 Uninit

NT_PB_UnInit

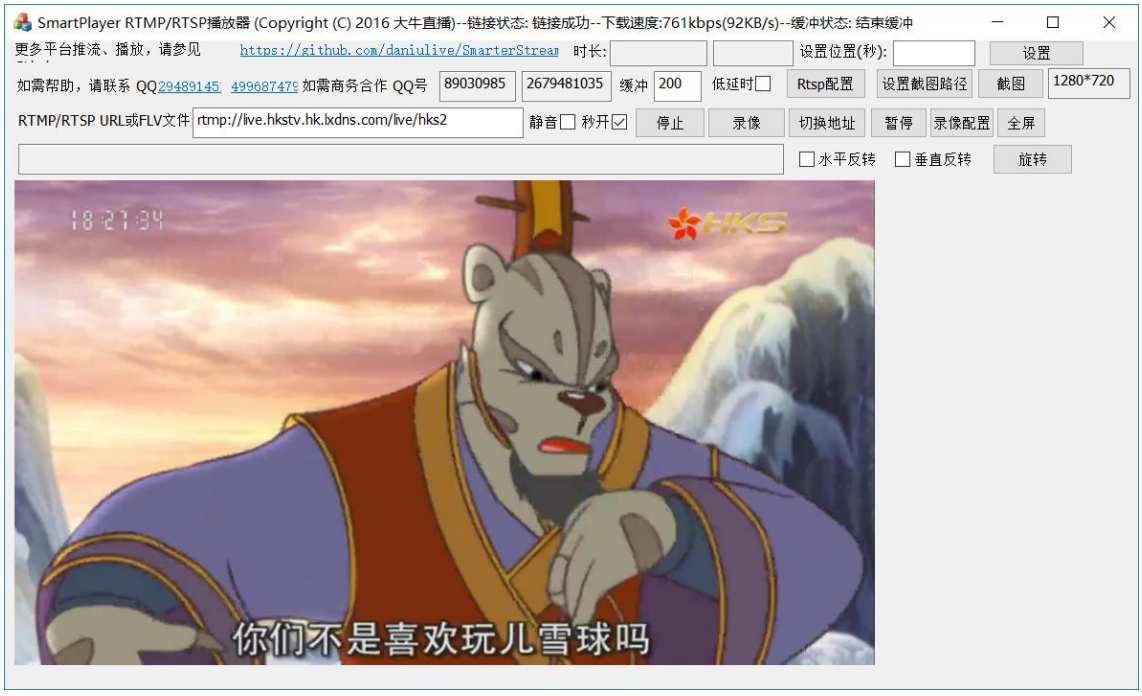
2. Windows 播放端 SDK 说明

高稳定、超低延迟（一秒内，行业内几无效果接近的播放端）、业内首屈一指的 RTMP/RTSP 直播播放器 SDK。

2.1 demo 说明

- 大牛直播 SDK 提供 C++/C#两套接口，对外提供 32/64 位库，C++和 C#接口一一对应，C#接口比 C++接口增加前缀 NT_PB_。
- WIN-PlayerSDK-CPP-Demo：播放端 SDK 对应的 C++接口的 demo；
- WIN-PlayerSDK-CSharp-Demo：播放端 SDK 对应的 C#接口的 demo；
- 大牛直播播放端 SDK 支持 Win7 及以上系统。
- 本 demo 基于 VS2013 开发。

2.2 功能说明



功能支持：

- [支持播放协议]RTSP、RTMP；
- [多实例播放]支持多实例播放（如同时播放多路 RTMP/RTSP 流）；
- [事件回调]支持网络状态、buffer 状态等回调；
- [音视频加密]Windows 平台支持 RTMP 推送端加密(AES/SM4(国密))音视频数据正常播放；

- [视频格式]支持 RTSP H.265、RTMP 扩展 H.265，RTSP/RTMP H.264；
- [音频格式]RTMP/RTSP 支持 AAC/PCMA/PCMU，此外 RTMP 还支持 Speex；
- [H.264/H.265 软解码]支持 H.264/H.265 软解；
- [RTSP 模式设置]支持 RTSP TCP/UDP 模式设置；
- [RTSP TCP/UDP 自动切换]支持 RTSP TCP、UDP 模式自动切换；
- [RTSP 超时设置]支持 RTSP 超时时间设置，单位：秒；
- [RTSP 401 认证处理]支持上报 RTSP 401 事件，如 URL 携带鉴权信息，会自动处理；
- [缓冲时间设置]支持 buffer time 设置；
- [首屏秒开]支持首屏秒开模式；
- [低延迟模式]支持类似于线上娃娃机等直播方案的超低延迟模式设置(公网 200~400ms)；
- [复杂网络处理]支持断网重连等各种网络环境自动适配；
- [快速切换 URL]支持播放过程中，快速切换其他 URL，内容切换更快；
- [实时静音]支持播放过程中，实时静音/取消静音；
- [实时快照]支持播放过程中截取当前播放画面；
- [渲染角度]支持 0°，90°，180° 和 270° 四个视频画面渲染角度设置；
- [渲染镜像]支持水平反转、垂直反转模式设置；
- [实时下载速度更新]支持当前下载速度实时回调(支持设置回调时间间隔)；
- [解码前视频数据回调]支持 H.264/H.265 数据回调；
- [解码后视频数据回调]支持解码后 YUV/RGB 数据回调；
- [解码前音频数据回调]支持 AAC/PCMA/PCMU/SPEEX 数据回调；
- [音视频自适应]支持播放过程中，音视频信息改变后自适应；
- [扩展录像功能]完美支持和录像 SDK 组合使用；
- [Windows 本地 FLV 播放器]支持本地 FLV 文件播放(支持获取 FLV 文件的 duration(时长)；支持显示当前播放位置；
- [Windows 本地 FLV 播放器]支持开始播放或播放过程中 seek(跳转播放位置)，也许是行业内 seek 最快的 flv 点播播放器)。

2.3 集成说明

C++头文件：

- [类型定义]nt_type_define.h
- [Log 定义]smart_log.h
- [Log 定义]smart_log_define.h
- [base code 定义]nt_base_code_define.h
- [player 接口]smart_player_define.h
- [player 参数定义]smart_player_sdk.h

C#头文件：

- [base code 定义]nt_base_code_define.cs
- [player 接口]smart_player_define.cs
- [player 参数定义]smart_player_sdk.cs

相关 Lib：

- **SmartLog.dll**
- **SmartLog.lib**
- **SmartPlayerSDK.dll**
- **SmartPlayerSDK.lib**
- avcodec-56.dll
- avdevice-56.dll
- avfilter-5.dll
- avformat-56.dll
- avutil-54.dll
- postproc-53.dll
- swresample-1.dll
- swscale-3.dll

集成步骤：

3. 把 lib 目录下 debug/release 库拷贝到需要集成的工程对应的 debug 或 release 目录下（确保 32 位/64 位库 debug/release 目录一一对应）；

lib 目录如下：

(1) 32 位 debug 库：debug

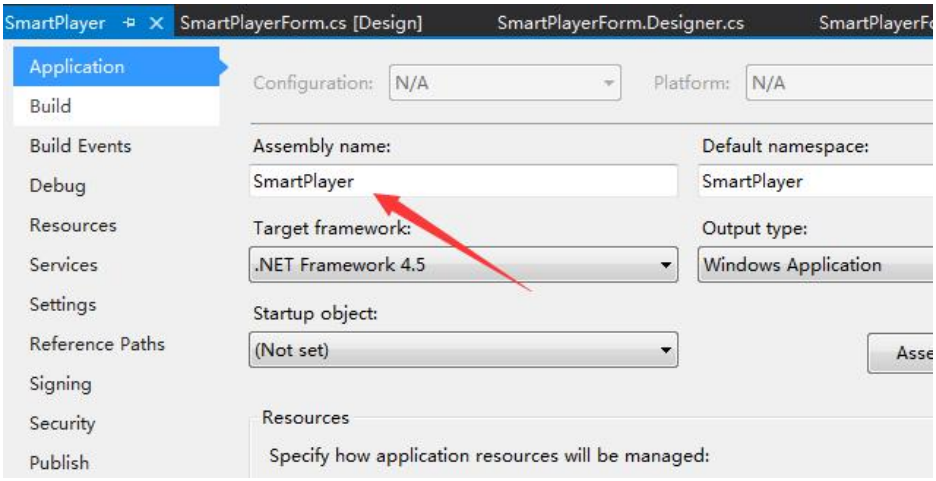
(2) 32 位 release 库：release

(3) 64 位 debug 库：x64\debug

(4) 64 位 release 库：x64\release

2. 相关 cs 头文件，加入需要集成的工程；

3. 在需要集成的工程，右键->Properties->Application->Assembly name，大牛直播按照 app 名称授权，未授权版本，此处请改成“SmartPlayer”，如需授权，可直接联系商务；



2.4 功能详解

2.4.1 常规功能

1. URL 播放地址

输入 rtmp 或 rtsp 播放 url。

2. Buffer time

设置缓冲时间。

3. RTSP-TCP

此设置只对 RTSP 流有效，默认 RTSP UDP 模式。

4. RTSP-TCP/UDP 自动切换

此设置只对 RTSP 流有效，对于 RTSP 来说，有些可能支持 rtp over udp 方式，有些可能支持使用 rtp over tcp 方式， 为了方便使用，有些场景下可以开启自动尝试切换开关，打开后如果 udp 无法播放，sdk 会自动尝试 tcp，如果 tcp 方式播放不了,sdk 会自动尝试 udp.

5. RTSP 超时时间设置

设置 RTSP 超时时间，timeout 单位为秒，必须大于 0。

6. 快速启动

快速 render 视频数据。

7. 静音

播放过程中，可以实时静音、取消静音。

8. 分辨率

点击播放后，如有视频数据，返回视频分辨率。

9. 当前状态

实时显示链接状态、实时下载速度统计、buffer 开始、buffer 比例、buffer 结束状态，**注意：如选择快速启动，buffer 比例显示不再精准。**

10. 视频 view 旋转

支持播放端视频 view 角度旋转，支持 0° 90° 180° 270° 旋转。

11. 视频 view 反转

支持播放端视频 view 水平反转、垂直反转。

2.4.2 录像

1. 选择本地录像

设置本地录像路径。

2. 录像文件名前缀

指定录像文件名前缀信息。

3．单个文件大小

设置单个文件最大录制大小。

4．文件名增加日期

在录制文件名，添加日期信息

5．文件名添加时间

在录制文件名，添加时间信息

6．设置录制音频、视频

大牛直播 SDK 支持设置只录制音频或者视频，如不设置，默认音视频均录制

7．开始录像

设置完音视频采集选项和录像参数后，随时开启录像。

8．停止录像

停止录制，生成录像文件。

NOTE：播放和录像，完全分离，播放过程中，可随时录像，反之、录像过程中，可随时开启播放。

2.4.3 截图

1．设置截图保存路径

设置本地截图保存路径。

2．实时截图

实时截取当前帧，并保存 png 文件到指定目录。

2.4.4 转发

完美支持播放端 SDK 和推送端 SDK 结合，通过调用 NT_SP_StartPullStream/NT_SP_StopPullStream 完成转发。

2.5 接口调用时序(以 C#为例)

2.5.1 初始化

NT_SP_Init

如需配置 log 路径，请在 NT_SP_Init 之前，做如下设置（目录可自行指定）：

```
// 设置日志路径(请确保目录存在)
```

```
//String log_path = "D:\\playerlog";
//NTSmartLog.NT_SL_SetPath(log_path);
```

2.5.2 Open

NT_SP_Open

2.5.3 设置回调事件

① NT_SP_SetEventCallBack：用于回调网络链接状态、buffer 状态(开始、buffer 比例、结束)、实时带宽等

② NT_SP_SetVideoSizeCallBack：设置视频分辨率回调

③ NT_SP_SetVideoFrameCallBack：设置 YUV/RGB32 数据回调

④ NT_SP_SetRenderVideoFrameTimestampCallBack：设置绘制视频帧时，视频帧时间戳回调

⑤ NT_SP_SetAudioPCMFrameCallBack：设置音频 PCM 帧回调，吐 PCM 数据出来，目前每帧大小是 10ms.

2.5.4 设置需要播放的 url

NT_SP_SetURL：支持 rtsp/rtmp/本地 FLV 文件(全路径)

2.5.5 查看是不是支持 D3DRender

设置绘制窗口句柄(可选接口)

① NT_SP_IsSupportD3DRender

② NT_SP_SetRenderWindow

③ NT_SP_GDIDrawRGB32：使用 GDI 绘制 RGB32 数据

2.5.6 回调 PCM

NT_SP_SetIsOutputAudioDevice：设置是否播放出声音，这个和静音接口是有区别的，这个接口的主要目的是为了用户设置了外部 PCM 回调接口后，又不想让 SDK 播放出声音时使用。

2.5.7 参数设置

① NT_SP_SetBuffer：设置 buffer 大小 ；

② NT_SP_SetMute：实时静音、取消静音；

③ NT_SP_SetRTSPTcpMode：设置 RTSP TCP 模式，1 为 TCP，0 为 UDP，仅 RTSP 有效；

④ NT_SP_SetRtspTimeout：设置 RTSP 超时时间，timeout 单位为秒，必须大于 0；

NT_SP_SetRtspAutoSwitchTcpUdp：对于 RTSP 来说，有些可能支持 rtp over udp 方式，有些可能支持使用 rtp over tcp 方式．为了方便使用，有些场景下可以开启自动尝试切换开关，打开后如果 udp 无法播放，sdk 会自动尝试 tcp，如果 tcp 方式播放不了，sdk 会自动尝试 udp，

is_auto_switch_tcp_udp： 如果设置 1 的话，sdk 将在 tcp 和 udp 之间尝试切换播放，如果设置为 0，则不尝试切换．

⑤ NT_SP_SetFastStartup：设置秒开，1 为秒开，0 为不秒开；

⑥ NT_SP_SetLowLatencyMode：设置低延时播放模式，默认是正常播放模式，mode：1 为低延时模式， 0 为正常模式；

⑦ NT_SP_SetRotation：设置旋转，顺时针旋转，degress： 设置 0， 90， 180， 270 度有效，其他值无效，注意：除了 0 度，其他角度播放会耗费更多 CPU；

⑧ NT_SP_SetFlipVertical：上下反转(垂直反转)；

⑨ NT_SP_SetFlipHorizontal：水平反转；

⑩ NT_SP_SetReportDownloadSpeed：设置下载速度上报，默认不上报下载速度

* is_report：上报开关，1：表上报．0：表示不上报．其他值无效．

*report_interval： 上报时间间隔（上报频率），单位是秒，最小值是 1 秒 1 次．如果小于 1 且设置了上报，将调用失败

* 注意：如果设置上报的话，请设置 SetEventCallBack，然后在回调函数里面处理这个事件．

* 上报事件是：NT_SP_E_EVENT_ID_DOWNLOAD_SPEED

⑪ NT_SP_GetDownloadSpeed：主动获取下载速度，speed： 返回下载速度，单位是 Byte/s；

⑫ NT_SP_SetParam：万能接口，设置参数， 大多数问题， 这些接口都能解决；

⑬ NT_SP_GetParam：万能接口，得到参数， 大多数问题，这些接口都能解决；

⑭ NT_SP_SetUserDataCallBack：设置用户数据回调；

⑮ NT_SP_SetSEIDataCallBack：设置视频 sei 数据回调；

⑯ NT_SP_SetKey:设置 RTMP 加密流的解密 key，目前只用来解密 rtmp 加密流；

⑰ NT_SP_SetDecryptionIV：设置 RTMP 加密流的解密向量，目前只用来解密 rtmp 加密流；

⑱ NT_SP_SetSDKClientKey：**设置授权 Key，正式授权 Key 请联系官方。**

2.5.7 FLV 本地文件点播特有接口

① NT_SP_GetDuration：获取视频时长，对于直播的话，没有时长，调用结果未定义

② NT_SP_GetPlaybackPos：获取当前播放时间戳，单位是毫秒(ms)，注意:这个时间戳是视频源的时间戳，只支持点播．直播不支持

③ NT_SP_GetPullStreamPos： 获取当前拉流时间戳，单位是毫秒(ms)，注意:这个时间戳是视频源的时间戳，只支持点播．直播不支持(用于转发模块)

④ NT_SP_SetPos：设置播放位置,单位是毫秒(ms)，注意:直播不支持,这个接口用于点播

⑤ NT_SP_Pause：暂停播放，isPause：1 表示暂停，0 表示恢复播放，其他错误，注意:直播不存在暂停的概念，所以直播不支持,这个接口用于点播

2.5.8 开始播放

NT_SP_StartPlay

2.5.9 绘制窗口大小改变

NT_SP_OnWindowSize
如播放窗口大小调整，需调用此接口。

2.5.10 录像(和播放/内置轻量级 RTSP 服务/转发功能逻辑完全分离)

① NT_SP_SetRecorderDirectory：设置录像目录
② NT_SP_SetRecorderFileMaxSize：设置单个文件最大大小
③ NT_SP_SetRecorderFileNameRuler：设置录像文件名生成规则
④ NT_SP_SetRecorderCallBack：设置录像回调接口
⑤ NT_SP_SetRecorderAudioTranscodeAAC：设置录像时音频转 AAC 编码的开关，aac 比较通用，sdk 增加其他音频编码(比如 speex，pcmu，pcma 等)转 aac 的功能
⑥ NT_SP_SetRecorderVideo：设置是否录视频，默认的话，如果视频源有视频就录，没有就没得录，但有些场景下可能不想录制视频，只想录音频，所以增加个开关
⑦ NT_SP_SetRecorderAudio：设置是否录音频，默认的话，如果视频源有音频就录，没有就没得录，但有些场景下可能不想录制音频，只想录视频，所以增加个开关
⑧ NT_SP_StartRecorder：启动录像
⑨ NT_SP_StopRecorder：停止录像

2.5.11 拉流接口(转发之用，和播放/录像功能逻辑完全分离)

① NT_SP_StartPullStream：启动拉流
② NT_SP_StopPullStream：停止拉流
③ NT_SP_SetPullStreamAudioTranscodeAAC：设置拉流时音频转 AAC 编码的开关，aac 比较通用，sdk 增加其他音频编码(比如 speex，pcmu，pcma 等)转 aac 的功能.

2.5.12 快照(实时调用)

NT_SP_CaptureImage：捕获图片

2.5.13 快速切换 URL(实时调用)

NT_SP_SwitchURL：切换 URL，其中：switch_pos：切换到新 url 以后，设置的播放位置，默认请填 0，这个只对设置播放位置的点播 url 有效，直播 url 无效

2.5.14 用户数据回调

NT_SP_SetUserDataCallBack：设置用户数据回调，用于接收扩展 SEI 模块发送的用户数据信息

2.5.15 SEI 数据回调

NT_SP_SetSEIDataCallBack：设置视频 sei 数据回调，用于接收 SEI 数据回调

2.5.16 停止播放

NT_SP_StopPlay

2.5.17 关闭播放实例

NT_SP_Close

2.5.18 Uninit

NT_SP_UnInit

2.5.19 Event 回调

```
/*事件 ID*/
       public enum NT_SP_E_EVENT_ID : uint
       {
           NT_SP_E_EVENT_ID_BASE = NTBaseCodeDefine.NT_EVENT_ID_SMART_PLAYER_SDK,

            NT_SP_E_EVENT_ID_CONNECTING             = NT_SP_E_EVENT_ID_BASE | 0x2,    /*连接中
*/
            NT_SP_E_EVENT_ID_CONNECTION_FAILED  = NT_SP_E_EVENT_ID_BASE | 0x3,    /*连接失败*/
            NT_SP_E_EVENT_ID_CONNECTED          = NT_SP_E_EVENT_ID_BASE | 0x4,    /*已连接*/
```

```
        NT_SP_E_EVENT_ID_DISCONNECTED       = NT_SP_E_EVENT_ID_BASE | 0x5,    /*断开连接*/
        NT_SP_E_EVENT_ID_NO_MEDIADATA_RECEIVED = NT_SP_E_EVENT_ID_BASE | 0x8, /*收不到 RTMP
数据*/
        NT_SP_E_EVENT_ID_RTSP_STATUS_CODE    = NT_SP_E_EVENT_ID_BASE | 0xB,
/*rtsp status code 上报，目前只上报 401，param1 表示 status code*/
         NT_SP_E_EVENT_ID_NEED_KEY           = NT_SP_E_EVENT_ID_BASE | 0xC,  /*
需要输入解密 key 才能播放*/
         NT_SP_E_EVENT_ID_KEY_ERROR          = NT_SP_E_EVENT_ID_BASE | 0xD,  /*
解密 key 不正确*/

        /* 接下来请从 0x81 开始*/
        NT_SP_E_EVENT_ID_START_BUFFERING = NT_SP_E_EVENT_ID_BASE | 0x81, /*开始缓冲*/
        NT_SP_E_EVENT_ID_BUFFERING      = NT_SP_E_EVENT_ID_BASE | 0x82, /*缓冲中，param1 表
示百分比进度*/
        NT_SP_E_EVENT_ID_STOP_BUFFERING  = NT_SP_E_EVENT_ID_BASE | 0x83, /*停止缓冲*/

        NT_SP_E_EVENT_ID_DOWNLOAD_SPEED  = NT_SP_E_EVENT_ID_BASE | 0x91, /*下载速度， param1
表示下载速度，单位是(Byte/s)*/

       NT_SP_E_EVENT_ID_PLAYBACK_REACH_EOS = NT_SP_E_EVENT_ID_BASE | 0xa1,    /*播放结束，直
播流没有这个事件，点播流才有*/
        NT_SP_E_EVENT_ID_RECORDER_REACH_EOS = NT_SP_E_EVENT_ID_BASE | 0xa2,    /*录像结束，直
播流没有这个事件，点播流才有*/
        NT_SP_E_EVENT_ID_PULLSTREAM_REACH_EOS = NT_SP_E_EVENT_ID_BASE | 0xa3,  /*拉流结束，直
播流没有这个事件，点播流才有*/

        NT_SP_E_EVENT_ID_DURATION = NT_SP_E_EVENT_ID_BASE | 0xa8, /*视频时长，如果是直播，则不
上报，如果是点播的话，若能从视频源获取视频时长的话，则上报， param1 表示视频时长，单位是毫秒(ms)*/
    }
```

2.5.20 拉流时，音视频数据回调

```
[StructLayoutAttribute(LayoutKind.Sequential)]
   public struct NT_SP_PullStreamVideoDataInfo
   {
       public Int32 is_key_frame_; /* 1:表示关键帧，0：表示非关键帧 */
       public UInt64 timestamp_; /* 解码时间戳，单位是毫秒 */
       public Int32 width_; /* 一般是 0 */
       public Int32 height_; /* 一般也是 0 */
       public IntPtr parameter_info_; /* 一般是 NULL */
       public UInt32 parameter_info_size_; /* 一般是 0 */
```

```
        public UInt64 presentation_timestamp_; /*显示时间戳，这个值要大于或等于 timestamp_， 单位是毫秒*/
}

    /*
     *拉流吐音频数据时，一些相关的数据
     */
  [StructLayoutAttribute(LayoutKind.Sequential)]
   public struct NT_SP_PullStreamAuidoDataInfo
   {
        public Int32 is_key_frame_; /* 1:表示关键帧，0：表示非关键帧 */
        public UInt64 timestamp_; /* 单位是毫秒 */
        public Int32 sample_rate_;     /* 一般是 0 */
        public Int32 channel_; /* 一般是 0 */
        public IntPtr parameter_info_; /* 如果是 AAC 的话，这个是有值的，其他编码一般忽略 */
        public UInt32 parameter_info_size_; /*如果是 AAC 的话，这个是有值的，其他编码一般忽略 */
        public UInt64 reserve_; /* 保留  */
    }
```

3 Windows 内置轻量级 RTSP 服务 SDK 说明

为满足内网无纸化/电子教室等内网超低延迟需求，避免让用户配置单独的服务器，大牛直播 SDK 在推送端发布了轻量级 RTSP 服务 SDK。

简单来说，之前推送端 SDK 支持的功能，内置轻量级 RTSP 服务 SDK 后，功能继续支持。

内置轻量级 RTSP 服务后，延迟更低，体验更好。

以下是接口详解：

Windows 内置轻量级 RTSP 服务 SDK 接口详解		
调用描述	接口	接口描述
SmartRTSPServerSDK		
创建一个 rtsp server	NT_PB_OpenRtspServer	创建一个 rtsp server，返回 rtsp server 句柄
设置端口	NT_PB_SetRtspServerPort	设置 rtsp server 监听端口，在 StartRtspServer 之前必须要设置端口
设置鉴权用户名、密码	NT_PB_SetRtspServerUserNamePassword	设置 rtsp server 鉴权用户名和密码，这个可以不设置，只有需要鉴权的再设置
获取 rtsp server 当前会话数	NT_PB_GetRtspServerClientSessionNumbers	获取 rtsp server 当前的客户会话数，这个接口必须在 StartRtspServer 之后再调用
启动 rtsp server	NT_PB_StartRtspServer	启动 rtsp server
停止 rtsp server	NT_PB_StopRtspServer	停止 rtsp server
关闭 rtsp server	NT_PB_CloseRtspServer	关闭 rtsp server
SmartRTSPServerSDK 供 Publisher 调用的接口		
设置 rtsp 的流名称	NT_PB_SetRtspStreamName	设置 rtsp 的流名称
给要发布的 rtsp 流设置 rtsp server	NT_PB_AddRtspStreamServer	给要发布的 rtsp 流设置 rtsp server，一个流可以发布到多个 rtsp server 上，rtsp server 的创建启动请参考 OpenRtspServer 和 StartRtspServer 接口
清除设置的 rtsp server	NT_PB_ClearRtspStreamServer	清除设置的 rtsp server
启动 rtsp 流	NT_PB_StartRtspStream	启动 rtsp 流
停止 rtsp 流	NT_PB_StopRtspStream	停止 rtsp 流

4 Windows 多路流媒体转发 SDK 说明

Windows 转发 SDK，简单来说，播放端 SDK 拉取 RTSP/RTMP 流，并回调编码后的音视频数据到上层，然后，调用我们的推送端 SDK，通过推送端 SDK 扩展数据接口，完成 RTMP 数据转发，整个过程由于不涉及解码、重编码，支持多路转发，超低延迟和低资源占用。

4.1 功能说明

Windows 多路流媒体转发 SDK	
功能	功能描述
标准功能	通过“**Windows 播放端 SDK**”拉流，回调 H.264/AAC/SPEEX/PCMA/PCMU 数据，调用“**Windows 推送端 SDK**”外置数据接口，实现转发
拉流音频转码	支持拉取的 RTMP/RTSP 的 PCMA/PCMU/SPEEX 音频格式**转 AAC** 后再转发到 RTMP 服务器
多实例	支持同时转发多路 RTMP/RTSP 音视频流
本地预览	因支持“**Windows 播放端 SDK**”功能，支持转发过程中，随时本地预览
拉流音频调节	支持拉取的 RTMP/RTSP 流实时静音
切换转发源	支持转发过程中，随时切换拉流的数据源(源 URL 变化，推流 URL 不变)，播放端 SDK 无感知(还是同一个拉流 URL)低延迟播放切换后数据源
逻辑分离	支持播放、录像、转发逻辑完全分离，三者可随意组合或单独使用
H.265(HEVC)支持	业内为数不多**支持 RTSP/RTMP H.265 转 RTMP 推送**的 SDK(提供配套 RTMP 扩展 H.265 服务器)；
RTMP 加密	支持转发的 RTMP 数据流加密(AES/SM4)后推送

4.2 接口详解

Windows 转发 SDK 接口详解		
调用描述	接口	接口描述
创建推流实例和播放实例	从播放端拉取 RTSP/RTMP 流，并回调到上层，调用推送端数据转	注意：推送端调用 NT_PB_Open 时： Audio_opt：

	发接口，实现转发逻辑	NT_PB_E_AUDIO_OPTION_ENCODED_DATA video_opt: NT_PB_E_VIDEO_OPTION_ENCODED_DATA.
播放端开始拉流	NT_SP_StartPullStream	播放端开始拉流，用于音视频数据转发
播放端停止拉流	NT_SP_StopPullStream	播放端停止拉流，用于音视频数据转发
播放端视频回调	NT_SP_SetPullStreamVideoDataCallBack	回调视频数据
播放端音频回调	NT_SP_SetPullStreamAudioDataCallBack	回调音频数据
音频转码	NT_SP_SetPullStreamAudioTranscodeAAC	设置拉流时音频转 AAC 编码的开关(PCMA/PCMU/SPEEX 转 AAC)
视频转发	NT_PB_PostVideoEncodedDataV2	设置编码后视频数据(H.264)
音频转发	NT_PB_PostAudioEncodedData	设置音频数据(AAC/PCMA/PCMU/SPEEX)

4.3 转发 demo 样例

以 C#为例，核心调用参见 nt_stream_relay_wrapper.cs

```
public bool StartPull(String url, bool is_rtsp_tcp_mode)
 {
      if ( is_pulling_ )
           return false;

     if (!OpenPullHandle(url, is_rtsp_tcp_mode, is_mute_))
           return false;

     pull_stream_video_data_call_back_ = new
SP_SDKPullStreamVideoDataCallBack(OnVideoDataHandle);
     pull_stream_audio_data_call_back_ = new
SP_SDKPullStreamAudioDataCallBack(OnAudioDataHandle);

     NTSmartPlayerSDK.NT_SP_SetPullStreamVideoDataCallBack(pull_handle_, IntPtr.Zero,
pull_stream_video_data_call_back_);
     NTSmartPlayerSDK.NT_SP_SetPullStreamAudioDataCallBack(pull_handle_, IntPtr.Zero,
pull_stream_audio_data_call_back_);
```

```
            int is_transcode_aac = 1;   //PCMA/PCMU/Speex 格式转 AAC 后 再转发
            NTSmartPlayerSDK.NT_SP_SetPullStreamAudioTranscodeAAC(pull_handle_,
is_transcode_aac);

            UInt32 ret = NTSmartPlayerSDK.NT_SP_StartPullStream(pull_handle_);

            if (NTBaseCodeDefine.NT_ERC_OK != ret)
             {
                  if ( !is_playing_ )
                  {
                   NTSmartPlayerSDK.NT_SP_Close(pull_handle_);
                       pull_handle_ = IntPtr.Zero;
                  }

                  return false;
             }

             is_pulling_ = true;

             return true;
        }

        public void StopPull()
        {
             if ( !is_pulling_ )
                  return;

            NTSmartPlayerSDK.NT_SP_StopPullStream(pull_handle_);

             if ( !is_playing_ )
             {
               NTSmartPlayerSDK.NT_SP_Close(pull_handle_);
                  pull_handle_ = IntPtr.Zero;
             }

             is_pulling_ = false;
        }
public bool OpenPullHandle(String url, bool is_rtsp_tcp_mode, bool is_mute)
        {
             if ( pull_handle_ != IntPtr.Zero )
                  return true;
```

```
            if ( String.IsNullOrEmpty(url) )
                return false;

            duration_ = 0;

            IntPtr pull_handle = IntPtr.Zero;

            if (NTBaseCodeDefine.NT_ERC_OK != NTSmartPlayerSDK.NT_SP_Open(out pull_handle,
render_wnd_, 0, IntPtr.Zero))
            {
                return false;
            }

            if (pull_handle == IntPtr.Zero)
            {
                return false;
            }

            pull_event_call_back_ = new SP_SDKEventCallBack(SDKPullEventCallBack);
            NTSmartPlayerSDK.NT_SP_SetEventCallBack(pull_handle, IntPtr.Zero,
pull_event_call_back_);

            NTSmartPlayerSDK.NT_SP_SetBuffer(pull_handle, 0);
            NTSmartPlayerSDK.NT_SP_SetFastStartup(pull_handle, 1);
            NTSmartPlayerSDK.NT_SP_SetRTSPTcpMode(pull_handle, is_rtsp_tcp_mode ? 1 : 0);

            //RTSP timeout 设置
            Int32 rtsp_timeout = 10;
            NTSmartPlayerSDK.NT_SP_SetRtspTimeout(pull_handle, rtsp_timeout);

            //RTSP TCP/UDP 自动切换设置
            Int32 is_auto_switch_tcp_udp = 1;
            NTSmartPlayerSDK.NT_SP_SetRtspAutoSwitchTcpUdp(pull_handle, is_auto_switch_tcp_udp);

            NTSmartPlayerSDK.NT_SP_SetMute(pull_handle, is_mute ? 1 : 0);

            if (NTBaseCodeDefine.NT_ERC_OK != NTSmartPlayerSDK.NT_SP_SetURL(pull_handle, url))
            {
                NTSmartPlayerSDK.NT_SP_Close(pull_handle);
                pull_handle = IntPtr.Zero;
                return false;
            }
```

```
            if ( setting_pos_ >= 0L )
             {
                NTSmartPlayerSDK.NT_SP_SetPos(pull_handle, setting_pos_);
             }

            pull_handle_ = pull_handle;

            return true;
          }
         private void OnVideoDataHandle(IntPtr handle, IntPtr user_data,
         UInt32 video_codec_id, IntPtr data, UInt32 size,
        IntPtr info, IntPtr reserve)
         {
            if (!is_pushing_ && !is_started_rtsp_stream_)
                return;

            if (data == IntPtr.Zero)
                return;

            if (size < 1)
                return;

            if (info == IntPtr.Zero)
                return;

            NT_SP_PullStreamVideoDataInfo video_info =
(NT_SP_PullStreamVideoDataInfo)Marshal.PtrToStructure(info,
typeof(NT_SP_PullStreamVideoDataInfo));

            lock (push_handle_mutex_)
            {
                if (!is_pushing_ && !is_started_rtsp_stream_)
                    return;

                if (push_handle_ == IntPtr.Zero)
                    return;
                //新接口
                NTSmartPublisherSDK.NT_PB_PostVideoEncodedDataV2(push_handle_, video_codec_id,
                data, size, video_info.is_key_frame_, video_info.timestamp_,
video_info.presentation_timestamp_);
            }
```

```
        }

        private void OnAudioDataHandle(IntPtr handle, IntPtr user_data,
                        UInt32 audio_codec_id, IntPtr data, UInt32 size,
                        IntPtr info, IntPtr reserve)
        {
            if (!is_pushing_ && !is_started_rtsp_stream_)
                return;

            if (data == IntPtr.Zero)
                return;

            if (size < 1)
                return;

            if (info == IntPtr.Zero)
                return;

            NT_SP_PullStreamAuidoDataInfo audio_info =
(NT_SP_PullStreamAuidoDataInfo)Marshal.PtrToStructure(info,
typeof(NT_SP_PullStreamAuidoDataInfo));

            lock (push_handle_mutex_)
            {
                if (!is_pushing_ && !is_started_rtsp_stream_)
                    return;

                if (push_handle_ == IntPtr.Zero)
                    return;

                NTSmartPublisherSDK.NT_PB_PostAudioEncodedData(push_handle_, audio_codec_id,
data, size,
                audio_info.is_key_frame_, audio_info.timestamp_,
                audio_info.parameter_info_, audio_info.parameter_info_size_);
            }
        }
```

5 商务合作/技术支持

公司：上海视沃信息科技有限公司
地址：上海市浦东新区碧波路 690 号张江微电子港 7 号楼 6 楼
官网：https://daniulive.com
Github 地址：https://github.com/daniulive/SmarterStreaming

商务合作：
手机：130-7210-2209 或 135-6452-9354
商务合作 QQ ：89030985 或 2679481035

技术支持：
技术支持 QQ: 2679481035

获取更多技术支持：
QQ 群 1：499687479
QQ 群 2：294891451