视沃科技

大牛直播 SDK(V2)

调用说明

官网：http://daniulive.com

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

目 录

1. Android 推流端 SDK 说明

1.1 demo 说明

- SmartPublisher：内置推前后摄像头 SDK 调用 demo(接口展示更全)；
- SmartServicePublisher：android 推屏和后台推摄像头 SDK 调用 demo；
- SmartEchoCancellation：android 一对一互动 demo，内置大牛直播推流端 SDK+播放端 SDK(支持回音消除)。

1.2 功能说明

标准接口：

- 音频编码：AAC；
- 视频编码：H.264；
- 推流协议：RTMP；
- 预览与推流分辨率设置；
- 支持横屏、竖屏推流；
- 推流过程中，前后摄像头动态切换；
- 码流自适应；
- 软、硬编码适配；
- 支持 rtmp 推送 live|record 模式设置；
- 支持后台 service 推送摄像头或屏幕(推送屏幕需要 5.0+版本)；
- 支持 gop 间隔、帧率、bierate 设置；
- 支持推送端镜像设置；
- 超低延迟推送；
- 断网自动重连。

增值接口：

- 实时录像；
- 水印：文字水印、logo 水印；
- 实时静音、取消静音；
- 编码前 YUV 接口对接；
- 编码后音频 AAC 对接；
- 编码后视频 H.264 对接；
- 实时快照；
- 音频处理(噪音抑制等)；
- 回音消除；
- 支持多实例推送。

1.3 集成说明

- 确保 SmartPublisherJni.java 放到 com.daniulive.smartpublisher 包名下(可在其他包名下调用)；
- Smartavengine.jar 加入到工程；
- 拷贝 SmartPublisher\libs\arm64-v8a 和 SmartPublisher\libs\armeabi 下 libSmartPublisher.so 到工程；
- Load 库：

```
static {
    System.loadLibrary("SmartPublisher");
}
```

- 如需集成到自己系统测试，请用大牛直播 SDK 的 app name(不然集成提示 license failed)，正式授权版按照授权 app name 正常使用即可：
- 如何改 app-name：
  strings.xml 做以下修改：
  <string name="app_name">SmartPublisherSDKDemo</string>

1.4 SDK 接口详解(V2 接口)

1. 【最先调用】SmartPublisherOpen()：初始化 SDK：

参数设置：

- ctx：上下文信息；
- Audio_opt:
  - 0：不发布 audio；
  - 1：发布 audio；
  - 2：对接外部编码后的 audio 数据（AAC）
- video_opt:
  - 0：不发布 video；
  - 1：发布 video；
  - 2：对接外部编码后的 video 数据（H.264）
- 宽高信息。

2. 【Event 回调】SetSmartPublisherEventCallbackV2()，设置 event callback；

3. 【硬编码设置】SetSmartPublisherVideoHWEncoder()，检测是否支持硬编码，如果返回 0，则支持，否则自动采用软编码；

4. 【水印】SmartPublisherSetFontWatermark()，设置文字水印；

5.【水印】SmartPublisherSetPictureWatermark()，设置图片水印；

6.【视频配置】SmartPublisherSetGopInterval，设置推送端 GOP 间隔，一般建议在帧率的 1~3 倍，如不设置，用底层计算的默认值；

7.【视频配置】SmartPublisherSetSWVideoBitRate,设置 software encode video bit-rate，最大码流一般是平均码流的 2 倍，如不设置，用底层计算的默认值；

8.【视频配置】SmartPublisherSetFPS，设置 fps，如不设置，用底层计算的默认值;

9.【视频配置】SmartPublisherSetSWVideoEncoderProfile，设置软编码模式下的 video encoder profile，默认 baseline profile;

10.【视频配置】SmartPublisherSetSWVideoEncoderSpeed，设置软编码编码速度，设置范围（1,6），1 最快，6 最慢，默认是 6；

11.【视频裁剪】SmartPublisherSetClippingMode，设置裁剪模式(仅用于 640*480 分辨率，裁剪主要用于移动端宽高适配)，如不设置，默认裁剪模式；

12.【音频配置】SmartPublisherSetAudioCodecType ,设置编码类型 ,默认 AAC 编码 ,type 设置为 2 时，启用 speex 编码（码率更低）；

13.【音频设置】SmartPublisherSetSpeexEncoderQuality，设置 speex 编码质量，数值越大，质量越高，范围（0,10），默认 8；

14.【音频处理】SmartPublisherSetNoiseSuppression，设置噪音抑制，噪音抑制开启后，去除采集端背景杂音；

15.【音频处理】SmartPublisherSetAGC，设置自动增益控制，保持声音稳定；

16. 【视频镜像】SmartPublisherSetMirror，镜像模式：播放端和推送端本地回显方向显示一致；

17. **【实时静音-可实时调用】**SmartPublisherSetMute()，设置实时静音、取消静音；

18. 【录像设置】SmartPublisherSetRecorder()，设置是否本地录像；

19. 【录像设置】如需本地录像，调用 SmartPublisherCreateFileDirectory()，创建录像文件目录；

20. 【录像设置】SmartPublisherSetRecorderDirectory()，设置录像文件目录；

21. 【录像设置】SmartPublisherSetRecorderFileMaxSize()，设置每个录像文件的大小，比如 100M，超过这个大小后，会自动生成下一个录像文件；

22. 【快照设置】SmartPublisherSaveImageFlag，设置是否启用快照；

23. **【实时快照-推送或录像后可实时调用】**SmartPublisherSaveCurImage，推送或录像过程中，根据设置路径和文件名，实时快照；

24. 【推送模式设置】SetRtmpPublishingType()，设置 rtmp publisher 类型，0：live，1：record，需服务器支持；

25. 【推送 URL 设置】SmartPublisherSetURL()，设置 publish stream url；

26. 【实时数据】SmartPublisherOnCaptureVideoData()，传递实时采集的 video 数据（编码前）；

27. 【第三方编码前数据】SmartPublisherOnCaptureVideoI420Data()，第三方 YUV（I420）接口；

28. 【第三方编码前数据】SmartPublisherOnCaptureVideoRGBAData()，第三方 RGBA 接口；

29. 【第三方编码前数据】SmartPublisherOnCaptureVideoABGRFlipVerticalData()，设置 ABGR flip vertical(垂直翻转) data；

30. 【第三方编码前数据】SmartPublisherOnFarEndPCMData，实时传递远端 PCM 数据(可用于互动级的回音消除处理)；

31. 【第三方编码后数据】SmartPublisherOnReceivingVideoEncodedData()，第三方编码后视频数据接口；

32. 【第三方编码后数据】SmartPublisherSetAudioSpecificConfig()，第三方音频参数设置接口；

33. 【第三方编码后数据】SmartPublisherOnReceivingAACData()，第三方编码后视频数据接口；

34. **【推流】**SmartPublisherStartPublisher，只推流；

35. **【推流】**SmartPublisherStopPublisher，关闭推流；

36. **【录像】**SmartPublisherStartRecorder，只录像；

37. **【录像】**SmartPublisherStopRecorder，停止录像；

38. **【最后调用】**SmartPublisherClose()，结束时必须调用 close 接口释放资源。

1.5 Event 回调

```
class EventHandeV2 implements NTSmartEventCallbackV2
{
    @Override
```

```
    public void onNTSmartEventCallbackV2(long handle, int id, long param1, long param2, String param3, String
param4, Object param5){

        Log.i(TAG, "EventHandeV2: handle=" + handle + " id:" + id);

        switch (id) {
          case NTSmartEventID.EVENT_DANIULIVE_ERC_PUBLISHER_STARTED:
             txt = "开始。。";
             break;
          case NTSmartEventID.EVENT_DANIULIVE_ERC_PUBLISHER_CONNECTING:
             txt = "连接中。。";
             break;
          case NTSmartEventID.EVENT_DANIULIVE_ERC_PUBLISHER_CONNECTION_FAILED:
             txt = "连接失败。。";
             break;
          case NTSmartEventID.EVENT_DANIULIVE_ERC_PUBLISHER_CONNECTED:
             txt = "连接成功。。";
             break;
          case NTSmartEventID.EVENT_DANIULIVE_ERC_PUBLISHER_DISCONNECTED:
             txt = "连接断开。。";
             break;
          case NTSmartEventID.EVENT_DANIULIVE_ERC_PUBLISHER_STOP:
             txt =  "关闭。。";
             break;
          case NTSmartEventID.EVENT_DANIULIVE_ERC_PUBLISHER_RECORDER_START_NEW_FILE:
             Log.i(TAG, "开始一个新的录像文件 : " + param3);
             txt = "开始一个新的录像文件。。";
             break;
          case    NTSmartEventID.EVENT_DANIULIVE_ERC_PUBLISHER_ONE_RECORDER_FILE_FINISHED:
             Log.i(TAG, "已生成一个录像文件 : " + param3);
             txt = "已生成一个录像文件。。";
             break;

          case NTSmartEventID.EVENT_DANIULIVE_ERC_PUBLISHER_SEND_DELAY:
             Log.i(TAG, "发送时延: " + param1 + " 帧数:" + param2);
             txt = "收到发送时延..";
            break;

          case NTSmartEventID.EVENT_DANIULIVE_ERC_PUBLISHER_CAPTURE_IMAGE:
             Log.i(TAG, "快照: " + param1 + " 路径: " + param3);
```

```
                if(param1 == 0)
                {
                      txt = "截取快照成功。.";
                }
                else
                {
                      txt = "截取快照失败。.";
                }
                break;
          }

          String str = "当前回调状态: " + txt;

          Log.i(TAG, str);

     }
}
```

2. Android 播放端 SDK 说明

2.1 demo 说明

- SmartPlayerV2：播放端 SDK 调用 demo(接口展示更全)；
- SmartEchoCancellation：android 一对一互动 demo，内置大牛直播推流端 SDK+播放端 SDK(支持回音消除)。

2.2 功能说明

标准接口：

- 音频：AAC/G.711/speex；
- 视频：H.264；
- 播放协议：RTMP/RTSP；
- 支持 RTSP TCP/UDP 模式切换；
- 支持纯音频、纯视频、音视频播放；
- 支持秒开模式；
- 音视频多种 render 机制；
- 支持 buffer 设置；
- 真正靠谱的超低延迟；
- 支持多实例播放；
- 支持播放 url 快速切换；
- 断网自动重连，支持视频追赶；
- 支持视频 video 实时旋转。

增值接口：

- 同时支持 rtsp、rtmp 播放；
- 播放过程中，实时静音、取消静音；
- 播放端出 YUV 或 RGB，供第三方 render 需求(如 unity3d)；
- 实时快照；
- 实时录像。

2.3 集成说明

- 确保 SmartPlayerJni.java 放到 com.daniulive.smartplayer 包名下(可在其他包名下调用)；
- Smartavengine.jar 加入到工程；

- 拷贝 SmartPlayer\libs\arm64-v8a 和 SmartPlayer\libs\armeabi 下 libSmartPlayer.so 到工程；
- Load 库：

```
static {
    System.loadLibrary("SmartPlayer");
}
```

- 如需集成到自己系统测试，请用大牛直播 SDK 的 app name(不然集成提示 license failed)，正式授权版按照授权 app name 正常使用即可：
- 如何改 app-name：
  strings.xml 做以下修改：
  <string name="app_name">SmartPlayerSDKDemo</string>

2.4 调用时序(V2)

1. 【最先调用】SmartPlayerOpen()，player 初始化，设置上下文信息，返回 player 句柄；

2. 【Event 回调】SetSmartPlayerEventCallbackV2()，设置 event callback；

3. 【设置硬解码】SetSmartPlayerVideoHWDecoder()，设置是否用硬解码播放，如硬解码不支持，自动适配到软解码；

4. 【设置 surface】SmartPlayerSetSurface()，设置播放的 surface，如果为 null，则播放纯音频；

5. 【外部数据接口】SmartPlayerSetExternalRender()，提供解码后 YUV/RGB 数据接口，供用户自己 render 或进一步处理；

6. 【外部数据接口】SmartPlayerSetExternalAudioOutput()，回调 audio 数据到上层(供二次处理之用)；

7. 【audio 输出类型】SmartPlayerSetAudioOutputType()，如果 use_audiotrack 设置为 0，将会自动选择输出设备，如果设置为 1，使用 audiotrack 模式；

8. 【Video 输出类型】参见上层接口 NTRenderer.CreateRenderer()，第二个参数，如果是 true，用 openGLES 绘制，false 则用默认 surfaceView；

9. 【缓冲设置】SmartPlayerSetBuffer()，设置播放端缓存数据 buffer，如不需 buffer，设置为 0；

10. 【RTSP TCP/UDP 设置】SmartPlayerSetRTSPTcpMode()，设置 TCP 播放模式，**注意：此接口仅用于 RTSP；**

11. **【实时静音-可实时调用】**SmartPlayerSetMute()，设置播放过程中，实时静音/取消静音；

12. 【快速启动】SmartPlayerSetFastStartup()，Set fast startup(快速启动)，设置快速启动后，如果 CDN 缓存 GOP，daniulive player 可快速出帧；

13. 【低延迟模式】SmartPlayerSetLowLatencyMode()，**针对类似于直播娃娃机等期待超低延迟的使用场景，超低延迟播放模式下，延迟可达到 200~400ms；**

14. **【视频显示角度设置-可实时调用】**SmartPlayerSetRotation()，**针对类似于安防摄像头或其他设备出来的图像倒置现象，支持视频播放 view 顺时针旋转，当前支持 0 度，90 度，180 度，270 度 旋转，注意除了 0 度之外， 其他角度都会额外消耗性能；**

15. 【下载速度回调设置】SmartPlayerSetReportDownloadSpeed()，设置下载速度上报，默认不上报下载速度；

16. 【横竖屏设置】SmartPlayerSetOrientation()，设置横屏竖屏；

17. 【快照设置】SmartPlayerSaveImageFlag()，设置是否启用快照；

18. **【快照-录像或播放后，可随时调用】**SmartPlayerSaveCurImage()，播放过程中，根据设置路径和文件名，实时快照；

19. **【快速切换 url-可实时调用】**SmartPlayerSwitchPlaybackUrl，快速切换播放 url，快速切换时，只换播放 source 部分，适用于不同数据流之间，快速切换（如娃娃机双摄像头切换或高低分辨率流切换）；

20. 【录像设置】SmartPlayerCreateFileDirectory()，创建文件路径；

21. 【录像设置】SmartPlayerSetRecorderDirectory()，设置文件路径；

22. 【录像设置】SmartPlayerSetRecorderFileMaxSize()，设置每个录像文件最大 size，以兆（M）为单位，范围(5M~500M)；

23. 【设置播放或录像 URL】SmartPlayerSetUrl()，设置播放/录像 url；

24. **【播放】**SmartPlayerStartPlay()，开始播放；

25. **【播放】**SmartPlayerStopPlay()，停止播放；

26. **【录像】**SmartPlayerStartRecorder()，开始录像；

27. **【录像】**SmartPlayerStopRecorder()，停止录像；

28. **【最后调用】**SmartPlayerClose()，关闭播放器实例。

2.5 Event 回调

```
class EventHandeV2 implements NTSmartEventCallbackV2 {
    @Override
    public void onNTSmartEventCallbackV2(long handle, int id, long param1,
            long param2, String param3, String param4, Object param5) {

        //Log.i(TAG, "EventHandeV2: handle=" + handle + " id:" + id);

        switch (id) {
```

```
            case NTSmartEventID.EVENT_DANIULIVE_ERC_PLAYER_STARTED:
                  Log.i(TAG, "开始。。");
                  break;
            case NTSmartEventID.EVENT_DANIULIVE_ERC_PLAYER_CONNECTING:
                  Log.i(TAG, "连接中。。");
                  break;
            case NTSmartEventID.EVENT_DANIULIVE_ERC_PLAYER_CONNECTION_FAILED:
                  Log.i(TAG, "连接失败。。");
                  break;
            case NTSmartEventID.EVENT_DANIULIVE_ERC_PLAYER_CONNECTED:
                  Log.i(TAG, "连接成功。。");
                  break;
            case NTSmartEventID.EVENT_DANIULIVE_ERC_PLAYER_DISCONNECTED:
                  Log.i(TAG, "连接断开。。");
                  break;
            case NTSmartEventID.EVENT_DANIULIVE_ERC_PLAYER_STOP:
                  Log.i(TAG, "停止播放。。");
                  break;
            case NTSmartEventID.EVENT_DANIULIVE_ERC_PLAYER_RESOLUTION_INFO:
                  Log.i(TAG, "分辨率信息: width: " + param1 + ", height: " + param2);
                  break;
            case NTSmartEventID.EVENT_DANIULIVE_ERC_PLAYER_NO_MEDIADATA_RECEIVED:
                  Log.i(TAG, "收不到媒体数据，可能是 url 错误。。");
                  break;
            case NTSmartEventID.EVENT_DANIULIVE_ERC_PLAYER_SWITCH_URL:
                  Log.i(TAG, "切换播放 URL。。");
                  break;
            case NTSmartEventID.EVENT_DANIULIVE_ERC_PLAYER_CAPTURE_IMAGE:
                  Log.i(TAG, "快照: " + param1 + " 路径: " + param3);

                  if (param1 == 0) {
                        Log.i(TAG, "截取快照成功。.");
                  } else {
                        Log.i(TAG, "截取快照失败。.");
                  }
                  break;

            case NTSmartEventID.EVENT_DANIULIVE_ERC_PLAYER_RECORDER_START_NEW_FILE:
            Log.i(TAG, "[record]开始一个新的录像文件 : " + param3);
        break;
    case NTSmartEventID.EVENT_DANIULIVE_ERC_PLAYER_ONE_RECORDER_FILE_FINISHED:
```

```
            Log.i(TAG, "[record]已生成一个录像文件 : " + param3);
              break;


                case NTSmartEventID.EVENT_DANIULIVE_ERC_PLAYER_START_BUFFERING:
                      Log.i(TAG, "Start_Buffering");
                      break;


                case NTSmartEventID.EVENT_DANIULIVE_ERC_PLAYER_BUFFERING:
                      Log.i(TAG, "Buffering:" + param1 + "%");
                      break;


                case NTSmartEventID.EVENT_DANIULIVE_ERC_PLAYER_STOP_BUFFERING:
                      Log.i(TAG, "Stop_Buffering");
                      break;


                case NTSmartEventID.EVENT_DANIULIVE_ERC_PLAYER_DOWNLOAD_SPEED:
                      Log.i(TAG, "download_speed:" + param1 + "Byte/s" + ", "
                                 + (param1 * 8 / 1000) + "kbps" + ", " + (param1 / 1024)
                                 + "KB/s");
                      break;
                }
            }
        }
```

3. iOS 推流端 SDK 说明

3.1 demo 说明

- SmartiOSPublisher：内置推前后摄像头 SDK 调用 demo(接口展示更全)；
- SmartiOSRelayDemo:iOS 平台播放、录像、转发 demo，内置大牛直播推流端 SDK+播放端 SDK(适用于推送+播放一起集成参考)。

3.2 功能说明

标准接口：

- 音频编码：AAC；
- 视频编码：H.264；
- 推流协议：RTMP；
- 预览与推流分辨率设置；
- 支持横屏、竖屏推流；
- 推流过程中，前后摄像头动态切换；
- 码流自适应；
- 软、硬编码适配；
- 支持 rtmp 推送 live|record 模式设置；
- 支持 gop 间隔、帧率、bierate 设置；
- 支持推送端镜像设置；
- 超低延迟推送；
- 断网自动重连。

增值接口：

- 实时录像；
- 实时静音、取消静音；
- 编码前 YUV 接口对接；
- 编码后音频 AAC 对接；
- 编码后视频 H.264 对接；
- 实时快照；
- 实时转发模块。

3.3 集成说明

- 相关库：libSmartPublisherSDK.a
- 相关头文件：

- nt_common_media_define.h(如需转发或第三方数据对接)
- nt_event_define.h
- SmartPublisherSDK.h

➢ 如集需要引入的 framework

- libz.tbd
- libc++.tbd
- libstdc++.tbd
- Libbz.tbd
- libiconv.tbd
- Accelerate.framework
- AudioToolBox.framework
- AssetsLibrary.framework
- AVFoundation.framework
- CoreMedia.framework
- Foundation.framework
- UIKit.framework
- VideoToolBox.framework

➢ 如需集成到自己系统测试，请用大牛直播的 app name:

Info.plist-->右键 Open As-->Source Code

添加或者编辑

<key>CFBundleName</key>

<string>SmartiOSPublisher</string>

➢ 快照添加到“照片”权限:

Info.plist-->右键 Open As-->Source Code

添加

<key>NSPhotoLibraryUsageDescription</key>

<string>1</string>

3.4 调用时序(V2)

1. **【最先调用】**SmartPublisherInit，初始化 publisher，参数设置如下：

➢ Audio_opt:

- 0：不发布 audio；
- 1：发布 audio；
- 2：对接外部编码后的 audio 数据（AAC）

➢ video_opt:

- 0：不发布 video；
- 1：发布 video；
- 2：对接外部编码后的 video 数据（H.264）

2. 【视频推送方向设置】SmartPublisherSetPublishOrientation，设置横竖屏推送模式（仅适用于内置非美颜模式）；

3. 【视频配置】SmartPublisherSetGopInterval，设置推送端 GOP 间隔，一般建议在帧率的 1~3 倍，如不设置，用底层计算的默认值；

4. 【视频配置】SmartPublisherSetVideoBitRate,设置软/硬编码 video bit-rate，最大码流一般是平均码流的 2 倍，如不设置，用底层计算的默认值；

5. 【视频配置】SmartPublisherSetFPS，设置 fps，如不设置，用底层计算的默认值;

6. 【视频裁剪】SmartPublisherSetClippingMode ,设置裁剪模式(仅用于 640*480 分辨率，裁剪主要用于移动端宽高适配)，如不设置，默认裁剪模式；

7. 【视频镜像】SmartPublisherSetMirror，镜像模式：播放端和推送端本地回显方向显示一致；

8. 【美颜相关】SmartPublisherSetBeauty，是否使用美颜：

- beautyType:
  - 0:不使用美颜;
  - 1:内部实现美颜;
  - 2:第三方美颜接口给数据。

9. 【美颜相关】SmartPublisherSetBeautyBrightness，内部美颜时使用，设置美颜效果。

10. 【美颜或外部视频】SmartPublisherSetExternalResolution，设置采集分辨率，美颜或外部视频采集时使用；

11. 【美颜或外部编码前视频数据对接】可选的第三方 video 数据接入方式(三选一)，YUV/BGRA/ARGB：

- SmartPublisherSetExternalYuvData，传递 YUV 数据；
- SmartPublisherSetExternalBGRAData，传递 BGRA 数据；
- SmartPublisherSetExternalARGBData，传递 ARGB 数据。

12. 【外部编码后 video 对接，如数据转发之用】SmartPublisherPostVideoEncodedData，目前 video 只支持 H.264；

13. 【外部编码后 audio 对接，如数据转发之用】SmartPublisherPostAudioEncodedData，目前 audio 支持 AAC/speex/pcma/pcmu；

14. 【实时静音】SmartPublisherSetMute()，设置实时静音、取消静音；

15. 【录像设置】SmartPublisherSetRecorder，设置是否边推流边本地存储；

16. 【录像设置】SmartPublisherSetRecorderDirectory，设置录像存放目录；

17. 【录像设置】SmartPublisherSetRecorderFileMaxSize，设置每个录像文件的大小(5~500M)，默认 200M；

18. 【快照设置】SmartPublisherSaveImageFlag，设置是否启用快照；

19. **【实时快照-推送或录像后可实时调用】**SmartPublisherSaveCurImage，推送或录像过程中，根据设置路径和文件名，实时快照；

20. 【推送模式设置】SmartPublisherSetRtmpPublishingType，设置 rtmp publisher 类型，0：live，1：record。

21. 【设置 video 显示 view】SmartPublisherSetVideoPreview，设置 video preview，此接口仅当用 daniulive 采集视频数据时设置，若视频来自外部美颜（DN_BEAUTY_ADDITIONAL_BEAUTY）或外部第三方数据，无需调用；

22. 【采集音视频】SmartPublisherStartCapture，设置分辨率，开始采集音视频数据；

23. **【前后摄像头切换-可实时调用】**SmartPublisherSwitchCamera，前后摄像头切换，此接口仅当用 daniulive 采集视频数据时设置；

24. 【推流】SmartPublisherStartPublisher，只推流；

25. 【推流】SmartPublisherStopPublisher，关闭推流；

26. 【录像】SmartPublisherStartRecorder，只录像；

27. 【录像】SmartPublisherStopRecorder，停止录像。

28. 【采集音视频】SmartPublisherStopCaputure，停止采集音视频数据，和 SmartPublisherStartCapture 配对使用；

29. 【最后调用】SmartPublisherUnInit，unInit 推流 SDK；

30. SmartPublisherGetSDKVersionID，获取当前 SDK 版本；

31. handleSmartPublisherEvent: Event callback 处理。

3.4 Event 回调

```
- (NSInteger) handleSmartPublisherEvent:(NSInteger)nID param1:(unsigned long long)param1 param2:(unsigned long long)param2
param3:(NSString*)param3 param4:(NSString*)param4 pObj:(void *)pObj;
{
    if (nID == EVENT_DANIULIVE_ERC_PUBLISHER_STARTED) {
        NSLog(@"[event]开始推流..");
    }
    else if (nID == EVENT_DANIULIVE_ERC_PUBLISHER_CONNECTING)
    {
        NSLog(@"[event]连接中..");
    }
    else if (nID == EVENT_DANIULIVE_ERC_PUBLISHER_CONNECTION_FAILED)
    {
        NSLog(@"[event]连接失败..");
    }
    else if (nID == EVENT_DANIULIVE_ERC_PUBLISHER_CONNECTED)
    {
        NSLog(@"[event]已连接..");
    }
    else if (nID == EVENT_DANIULIVE_ERC_PUBLISHER_DISCONNECTED)
    {
```

```
        NSLog(@"[event]断开连接..");
    }
    else if (nID == EVENT_DANIULIVE_ERC_PUBLISHER_STOP)
    {
        NSLog(@"[event]停止推流..");
    }
    else if (nID == EVENT_DANIULIVE_ERC_PUBLISHER_RECORDER_START_NEW_FILE)
    {
        NSLog(@"[event]录像写入新文件..文件名: %@", param3);
    }
    else if (nID == EVENT_DANIULIVE_ERC_PUBLISHER_ONE_RECORDER_FILE_FINISHED)
    {
        NSLog(@"[event]一个录像文件完成..文件名: %@", param3);
    }
    else if (nID == EVENT_DANIULIVE_ERC_PUBLISHER_CAPTURE_IMAGE)
    {
        NSLog(@"[event]推送快照..");
    }
    else
        NSLog(@"[event]nID:%lx", (long)nID);

    return 0;
}
```

4. iOS 播放端 SDK 说明

4.1 demo 说明

- SmartiOSPlayer：播放端 SDK 调用 demo(接口展示更全，不带录像)；
- SmartiOSRelayDemo:iOS 平台播放、录像、转发 demo，内置大牛直播推流端 SDK+播放端 SDK(适用于推送+播放一起集成参考)。

4.2 功能说明

标准接口：

- 音频：AAC/G.711/speex；
- 视频：H.264；
- 播放协议：RTMP/RTSP；
- 支持 RTSP TCP/UDP 模式切换；
- 支持纯音频、纯视频、音视频播放；
- 支持秒开模式；
- 音视频多种 render 机制；
- 支持 buffer 设置；
- 真正靠谱的超低延迟；
- 支持多实例播放；
- 支持播放 url 快速切换；
- 断网自动重连，支持视频追赶；
- 支持视频 video 实时旋转。

增值接口：

- 同时支持 rtsp、rtmp 播放；
- 播放过程中，实时静音、取消静音；
- 播放端出 YUV 或 RGB，供第三方 render 需求(如 unity3d)；
- 播放端出 H.264，供转发或第三方之用；
- 播放端出 AAC/speex/PCMA/PCMU，供转发或第三方之用；
- 实时快照；
- 实时录像。

4.3 集成说明

- 相关库：libSmartPlayerSDK.a
- 相关头文件：

- nt_common_media_define.h(如需转发或第三方数据对接)
- nt_event_define.h
- SmartPublisherSDK.h

➢ 如集需要引入的 framework

- libbz.tbd
- Libbz2.tbd
- libiconv.tbd
- libstdc++.tbd
- Accelerate.framework
- AssetsLibrary.framework
- AudioToolBox.framework
- AVFoundation.framework
- CoreMedia.framework
- Foundation.framework
- UIKit.framework
- VideoToolBox.framework

➢ 如需集成到自己系统测试，请用大牛直播的 app name:

Info.plist-->右键 Open As-->Source Code

添加或者编辑

<key>CFBundleName</key>

<string>SmartiOSPlayer</string>

➢ 快照添加到“照片”权限:

Info.plist-->右键 Open As-->Source Code

添加

<key>NSPhotoLibraryUsageDescription</key>

<string>1</string>

4.4 调用时序(V2)

1. **【最先调用】**SmartPlayerInitPlayer，初始化，创建 player 实例；

2. 【软硬解码设置】SmartPlayerSetVideoDecoderMode，设置视频解码模式，如不设置，默认软解码，0：软解码，1：硬解码；

3. 【创建 view】SmartPlayerCreatePlayView，创建播放 view；

4. 【释放 view】SmartPlayeReleasePlayView，释放 SmartPlayerCreatePlayView 创建的 view；

5. 【设置 view】SmartPlayerSetPlayView，设置播放 view；

6. 【设置 YUV 回调】SmartPlayerSetYuvBlock ,设置 YUV 数据回调(用于用户自己 render，如 unity3d 绘制 ）；

7. **【设置视频回调】**SmartPlayerSetPullStreamVideoDataBlock，用于用户自己处理，或数据转发；

8. **【设置音频回调】**SmartPlayerSetPullStreamAudioDataBlock，用于用户自己处理，或数据转发；

9. 【设置 buffer】SmartPlayerSetBuffer，设置播放端缓存数据 buffer；

10. 【RTSP TCP/UDP 设置】SmartPlayerSetRTSPTcpMode()，设置 TCP 播放模式，**注意：此接口仅用于 RTSP；**

11. 【快速启动】SmartPlayerSetFastStartup()，Set fast startup(快速启动)，设置快速启动后，如果 CDN 缓存 GOP，daniulive player 可快速出帧；

12. 【低延迟模式】SmartPlayerSetLowLatencyMode()，**针对类似于直播娃娃机等期待超低延迟的使用场景，超低延迟播放模式下，延迟可达到 200~400ms；**

13. **【视频显示角度设置-可实时调用】**SmartPlayerSetRotation()，**针对类似于安防摄像头或其他设备出来的图像倒置现象，支持视频播放 view 顺时针旋转，当前支持 0 度，90 度，180 度，270 度 旋转，注意除了 0 度之外， 其他角度都会额外消耗性能；**

14. 【下载速度回调设置】SmartPlayerSetReportDownloadSpeed()，设置下载速度上报，默认不上报下载速度；

15. 【设置播放或录像 URL】SmartPlayerSetPlayURL，设置播放/录像 url；

16. 【快照设置】SmartPlayerSaveImageFlag()，设置是否启用快照；

17. SmartPlayerStart

18. **【快照-录像或播放后，可随时调用】**SmartPlayerSaveCurImage()，播放过程中，根据设置路径和文件名，实时快照；

19. **【播放】**SmartPlayerStart()，开始播放；

20. **【快速切换 url-可实时调用】**SmartPlayerSwitchPlaybackUrl，快速切换播放 url，快速切换时，只换播放 source 部分，适用于不同数据流之间，快速切换（如娃娃机双摄像头切换或高低分辨率流切换）；

21. **【实时静音-可实时调用】**SmartPlayerSetMute()，设置播放过程中，实时静音/取消静音；

22. **【播放】**SmartPlayerStop()，停止播放；

23. 【录像设置】SmartPlayerSetRecorderDirectory()，设置文件路径；

24. 【录像设置】SmartPlayerSetRecorderFileMaxSize()，设置每个录像文件最大 size，以兆（M）为单位，范围(5M~500M)；

25. **【录像】**SmartPlayerStartRecorder()，开始录像；

26. **【录像】**SmartPlayerStopRecorder()，停止录像；

27. **【转发】**SmartPlayerStartPullStream()，开始拉流，用于数据转发；

28. **【转发】**SmartPlayerStopPullStream()，停止拉流；

29. **【最后调用】**SmartPlayerUnInitPlayer，销毁播放实例；

30. SmartPlayerGetSDKVersionID，获取当前 SDK 版本；

31. 【YUV 回调】PlayerYuvDataBlock，YUV 数据回调（可用于客户自己绘制）；

32. **【视频数据实时回调-转发】**PullStreamVideoDataBlock，Video 数据回调，用于数据转发或客户二次处理；

33. **【音频数据实时回调-转发】**PullStreamAudioDataBlock，audio 数据回调，用于数据转发或客户二次处理；

34. 【Event 处理】handleSmartPlayerEvent，Event callback 处理。

4.5 Event 回调

```
- (NSInteger) handleSmartPlayerEvent:(NSInteger)nID param1:(unsigned long long)param1 param2:(unsigned long long)param2
param3:(NSString*)param3 param4:(NSString*)param4 pObj:(void *)pObj;
{
    if (nID == EVENT_DANIULIVE_ERC_PLAYER_STARTED) {
        NSLog(@"[event]开始播放..");
    }
    else if (nID == EVENT_DANIULIVE_ERC_PLAYER_CONNECTING)
    {
        NSLog(@"[event]连接中..");
    }
    else if (nID == EVENT_DANIULIVE_ERC_PLAYER_CONNECTION_FAILED)
    {
        NSLog(@"[event]连接失败..");
    }
    else if (nID == EVENT_DANIULIVE_ERC_PLAYER_CONNECTED)
    {
        NSLog(@"[event]已连接..");
    }
    else if (nID == EVENT_DANIULIVE_ERC_PLAYER_DISCONNECTED)
    {
        NSLog(@"[event]断开连接..");
    }
    else if (nID == EVENT_DANIULIVE_ERC_PLAYER_STOP)
    {
        NSLog(@"[event]停止播放..");
    }
    else if (nID == EVENT_DANIULIVE_ERC_PLAYER_RESOLUTION_INFO)
    {
```

```
        stream_width_ = (NSInteger)param1;
        stream_height_ = (NSInteger)param2;
        NSLog(@"[event]视频解码分辨率信息..width:%ld, height:%ld", (long)stream_width_, (long)stream_height_);
    }
    else if (nID == EVENT_DANIULIVE_ERC_PLAYER_NO_MEDIADATA_RECEIVED)
    {
        NSLog(@"[event]收不到 RTMP 数据..");
    }
    else if (nID == EVENT_DANIULIVE_ERC_PLAYER_SWITCH_URL)
    {
        NSLog(@"[event]快速切换 url..");
    }
    else if (nID == EVENT_DANIULIVE_ERC_PLAYER_CAPTURE_IMAGE)
    {
        if ((int)param1 == 0)
        {
            NSLog(@"[event]快照成功: %@", param3);

            //tmp_path = param3;

            //image_path = [ UIImage imageNamed:param3];

            //UIImageWriteToSavedPhotosAlbum(image_path, self, @selector(image:didFinishSavingWithError:contextInfo:),
NULL);
        }
        else
        {
            NSLog(@"[event]快照失败: %@", param3);
        }
    }
    else if (nID == EVENT_DANIULIVE_ERC_PLAYER_RECORDER_START_NEW_FILE)
    {
        NSLog(@"[event]录像写入新文件..文件名: %@", param3);
    }
    else if (nID == EVENT_DANIULIVE_ERC_PLAYER_ONE_RECORDER_FILE_FINISHED)
    {
        NSLog(@"[event]一个录像文件完成..文件名: %@", param3);
    }
    else if (nID == EVENT_DANIULIVE_ERC_PLAYER_START_BUFFERING)
    {
        NSLog(@"[event]开始 buffer..");
```

```
    }
    else if (nID == EVENT_DANIULIVE_ERC_PLAYER_BUFFERING)
    {
        NSLog(@"[event]buffer百分比: %lld", param1);
    }
    else if (nID == EVENT_DANIULIVE_ERC_PLAYER_STOP_BUFFERING)
    {
        NSLog(@"[event]停止buffer..");
    }
    else if (nID == EVENT_DANIULIVE_ERC_PLAYER_DOWNLOAD_SPEED)
    {
        NSInteger speed_kbps = (NSInteger)param1*8/1000;
        NSInteger speed_KBs = (NSInteger)param1/1024;

        NSLog(@"[event]download speed :%ld kbps - %ld KB/s", (long)speed_kbps, (long)speed_KBs);
    }
    else
        NSLog(@"[event]nID:%lx", (long)nID);

    return 0;
}
```