视沃科技

大牛直播 SDK

Unity3D 调用说明

官网：http://daniusdk.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://daniusdk.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

声 明.....2
1. Android 播放端 SDK 说明.....4
1.1 demo 说明.....4
1.2 集成说明.....4
1.3 调用时序.....5
1.4 相关实现.....7
2. iOS 播放端 SDK 说明.....14
2.1 demo 说明.....14
2.2 集成说明.....14
2.3 调用时序.....15
2.4 相关实现.....18
3. Windows 播放端 SDK 说明.....24
3.1 demo 说明.....24
3.2 集成说明.....24
3.3 调用时序.....25

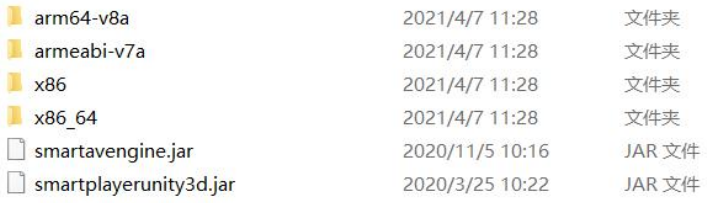
1. Android 播放端 SDK 说明

1.1 demo 说明

- SmartU3dAndroidPlayer：大牛直播 SDK Unity3D Android 平台 RTMP/RTSP 直播播放端工程。

1.2 集成说明

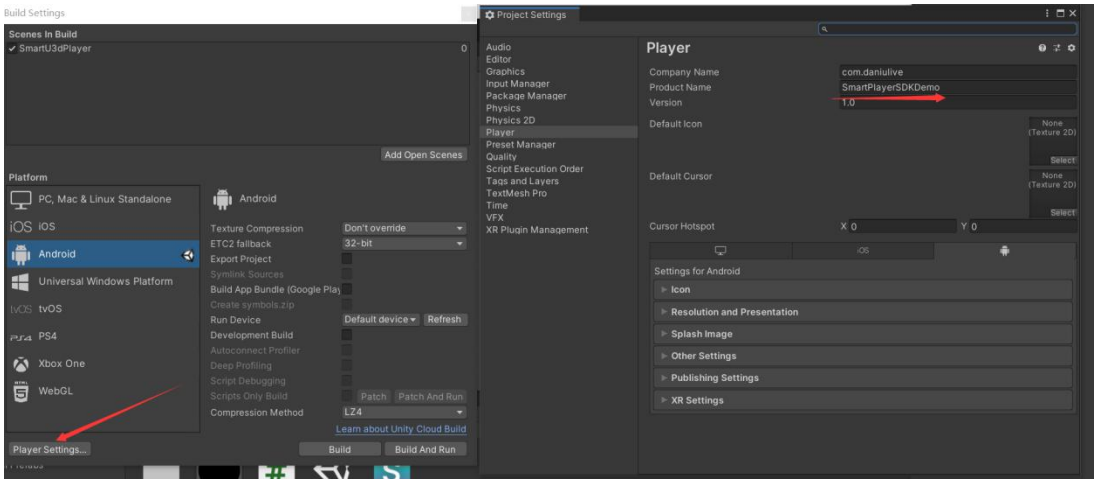
- Unity3D 接口和调用 demo，参见：SmartPlayerAndroidMono.cs
- SmartU3dAndroidPlayer\Assets\Plugins\Android\libs 下相关库到工程：



  - smartavengine.jar 加入到工程；
  - smartplayerunity3d.jar 加入工程；
  - 32/64 位 arm 和 x86 下 libSmartPlayer.so。
- 在 SmartU3dAndroidPlayer\Assets\Plugins\Android\AndroidManifest.xml 配置相关权限：

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

- 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；
- 正式授权版，需要调用 license 设置接口，确保 license key 正确设置方可正常使用；
- 如何改 app-name：
  Unity3D 模式下：File-->Build Settings-->Android-->Player Settings:



1.3 调用时序

1. **【最先调用】**NT_U3D_Init：player 初始化，目前预留；

2. **【正式授权版设置 license key】**NT_U3D_SetSDKClientKey，设置正式授权版 license，请早于 NT_U3D_Open 接口调用；

3. **【获得 player 句柄】**NT_U3D_Open，设置上下文信息，返回 player 句柄；

4. 【设置 GameObject】NT_U3D_Set_Game_Object，注册 Game Object，用于消息传递；

5. 【设置硬解码】NT_U3D_SetVideoDecoderMode，设置特定机型硬解码播放，如硬解码检测到不支持，切换到软解码；

6. 【audio 输出类型】NT_U3D_SetAudioOutputType()，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，一般建议使用 audiotrack 模式；

7. 【缓冲设置】NT_U3D_SetBuffer，设置播放端缓存数据 buffer，以毫秒(ms)为单位，如超低延迟模式下，不需 buffer 数据，设置为 0；

8. 【RTSP TCP/UDP 设置】NT_U3D_SetRTSPTcpMode，设置 TCP/UDP 播放模式，**注意：此接口仅用于 RTSP；**

9. 【RTSP 超时时间设置】NT_U3D_SetRTSPTimeout，设置 RTSP 超时时间, timeout 单位为秒，必须大于 0；

10. 【RTSP TCP/UDP 自动切换】NT_U3D_SetRTSPAutoSwitchTcpUdp，设置 RTSP TCP/UDP 自动切换，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；

11. **【实时静音-可实时调用】**NT_U3D_SetMute，设置播放过程中，实时静音/取消静音；

12. 【快速启动】NT_U3D_SetFastStartup，设置快速启动后，如果 CDN 缓存 GOP，播放端可快速出帧；

13. 【低延迟模式】NT_U3D_SetPlayerLowLatencyMode，设置低延迟模式；

14. **【视频垂直反转-可实时调用】**NT_U3D_SetFlipVertical，**视频垂直反转；**

15. **【视频水平反转-可实时调用】**NT_U3D_SetFlipHorizontal，**视频水平反转；**

16. **【视频显示角度设置-可实时调用】**NT_U3D_SetRotation，**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；**

17. 【下载速度回调设置】NT_U3D_SetReportDownloadSpeed，设置下载速度上报，默认不上报下载速度；

18. 【快照设置】NT_U3D_SetSaveImageFlag，设置是否需要在播放或录像过程中快照；

19. **【快照-录像或播放后，可随时调用】**NT_U3D_SaveCurImage，播放过程中，根据设置路径和文件名，实时快照；

20. **【快速切换 url-可实时调用】**NT_U3D_SwitchPlaybackUrl，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；

21. 【录像设置】NT_U3D_CreateFileDirectory，创建文件路径；

22. 【录像设置】NT_U3D_SetRecorderDirectory，设置文件路径；

23. 【录像设置】NT_U3D_SetRecorderFileMaxSize，设置每个录像文件最大 size，以兆（M）为单位，范围[5M~500M]；

24. 【录像设置】NT_U3D_SetRecorderAudioTranscodeAAC，支持拉取的 RTMP/RTSP 的 PCMA/PCMU/SPEEX 音频格式转 AAC 后录制；

25. 【设置播放或录像 URL】NT_U3D_SetUrl，设置播放/录像 url；

26. **【播放】**NT_U3D_StartPlay，开始播放；

27. **【播放】**NT_U3D_GetVideoFrame，获取底层回调的 YUV 数据；

28. **【播放】**NT_U3D_StopPlay，停止播放；

29. **【录像】**NT_U3D_StartRecorder，开始录像；

30. **【录像】**NT_U3D_StopRecorder，停止录像；

31. **【关闭】**NT_U3D_Close，关闭播放器实例；

32. **【最后调用】**NT_U3D_UnInit，UnInit Player，最后调用。

1.4 相关实现

OpenPlayer：

```
private void OpenPlayer()
```

```
{
    if ( java_obj_cur_activity_ == null )
    {
        Debug.LogError("getApplicationContext is null");
        return;
    }

    player_handle_ = NT_U3D_Open();
```

```
    if (player_handle_ != 0)
        Debug.Log("open success");
    else
        Debug.LogError("open fail");
}
```

开始播放

完成相关的初始化和接口参数设定后，调用 NT_U3D_StartPlay()实现音视频数据播放。

```
public void Play()
{
    if (is_running)
    {
        Debug.Log("已经在播放。。");
        return;
    }

    //获取输入框的 url
    string url = input_url_.text.Trim();
```

```
    if (!url.StartsWith("rtmp://") && !url.StartsWith("rtsp://"))
    {
        videoUrl = "rtsp://admin:12345@192.168.0.120:554/h265/ch1/main/av_stream";
    }
    else
    {
        videoUrl = url;
    }
```

```
    OpenPlayer();
```

```
    if ( player_handle_ == 0 )
        return;
```

```
        NT_U3D_Set_Game_Object(player_handle_, game_object_);

        /* ++ 播放前参数配置可加在此处 ++ */
        int is_using_tcp = 0;        //TCP/UDP 模式设置
        NT_U3D_SetRTSPTcpMode(player_handle_, is_using_tcp);

        int is_report = 0;
        int report_interval = 1;
        NT_U3D_SetReportDownloadSpeed(player_handle_, is_report, report_interval);  //下载速度回调

        NT_U3D_SetBuffer(player_handle_, play_buffer_time_);                        //设置 buffer time

        NT_U3D_SetPlayerLowLatencyMode(player_handle_, is_low_latency_ ? 1 : 0);    //设置是否启用低延迟模
式

        NT_U3D_SetMute(player_handle_, is_mute_ ? 1 : 0);                           //是否启动播放的时候静
音

        NT_U3D_SetVideoDecoderMode(player_handle_, is_hw_decode_ ? 1 : 0);          //设置 H.264 软硬解模式

        NT_U3D_SetVideoHevcDecoderMode(player_handle_, is_hw_decode_ ? 1 : 0);          //设置 H.265 软硬解
模式

        int is_fast_startup = 1;
        NT_U3D_SetFastStartup(player_handle_, is_fast_startup);                     //设置快速启动模式

        int rtsp_timeout = 10;
        NT_U3D_SetRTSPTimeout(player_handle_, rtsp_timeout);                        //设置 RTSP 超时时间

        int is_auto_switch_tcp_udp = 1;
        NT_U3D_SetRTSPAutoSwitchTcpUdp(player_handle_, is_auto_switch_tcp_udp);    //设置 TCP/UDP 模式自动
切换

        NT_U3D_SetUrl(player_handle_, videoUrl);
        /* -- 播放前参数配置可加在此处 -- */

        int flag = NT_U3D_StartPlay(player_handle_);

        if (flag  == DANIULIVE_RETURN_OK)
        {
```

```
            is_need_get_frame_ = true;
            Debug.Log("播放成功");
        }
        else
        {
            is_need_get_frame_ = false;
            Debug.LogError("播放失败");
        }
```

```
        is_running = true;
    }
```

停止播放：

调用 NT_U3D_StopPlay()后，如果没有录像，调用 NT_U3D_Close()，播放 handler 置空，然后调用 NT_U3D_UnInit()即可。

```
    private void ClosePlayer()
    {
        is_need_get_frame_ = false;
        is_need_init_texture_ = false;

        int flag = NT_U3D_StopPlay(player_handle_);
        if (flag == DANIULIVE_RETURN_OK)
        {
            Debug.Log("停止成功");
        }
        else
        {
            Debug.LogError("停止失败");
        }

        flag = NT_U3D_Close(player_handle_);
        if (flag == DANIULIVE_RETURN_OK)
        {
            Debug.Log("关闭成功");
        }
        else
        {
            Debug.LogError("关闭失败");
        }
```

```
        player_handle_ = 0;
```

```
        NT_U3D_UnInit();
```

```
        is_running = false;
    }
```

相关 Event 处理：

```
    /// <summary>
    /// android 传递过来 code
    /// </summary>
    /// <param name="code"></param>
    public void onNTSmartEvent(string param)
    {
        if (!param.Contains(","))
        {
            Debug.Log("[onNTSmartEvent] android 传递参数错误");
            return;
        }

       string[] strs = param.Split(',');
```

```
       string player_handle =strs[0];
       string code = strs[1];
       string param1 = strs[2];
       string param2 = strs[3];
       string param3 = strs[4];
       string param4 = strs[5];

       Debug.Log("[onNTSmartEvent] code: 0x" + Convert.ToString(Convert.ToInt32(code), 16));
```

```
        switch (Convert.ToInt32(code))
        {
            case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_STARTED:
                Debug.Log("开始。。");
                break;
            case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_CONNECTING:
                Debug.Log("连接中。。");
                break;
            case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_CONNECTION_FAILED:
                Debug.Log("连接失败。。");
                break;
```

```
            case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_CONNECTED:
                Debug.Log("连接成功。。");
                break;
            case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_DISCONNECTED:
                Debug.Log("连接断开。。");
                break;
            case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_STOP:
                Debug.Log("停止播放。。");
                break;
            case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_RESOLUTION_INFO:
                Debug.Log("分辨率信
息: width: " + Convert.ToInt32(param1) + ", height: " + Convert.ToInt32(param2));
                break;
            case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_NO_MEDIADATA_RECEIVED:
                Debug.Log("收不到媒体数据，可能是 url 错误。。");
                break;
            case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_SWITCH_URL:
                Debug.Log("切换播放 URL。。");
                break;
```

```
            case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_CAPTURE_IMAGE:
                Debug.Log("快照: " + param1 + " 路径: " + param3);
```

```
                if (Convert.ToInt32(param1) == 0)
                {
                    Debug.Log("截取快照成功。.");
                }
                else
                {
                    Debug.Log("截取快照失败。.");
                }
                break;
```

```
            case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_RECORDER_START_NEW_FILE:
                Debug.Log("[record]开始一个新的录像文件 : " + param3);
                break;
            case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_ONE_RECORDER_FILE_FINISHED:
                Debug.Log("[record]已生成一个录像文件 : " + param3);
                break;
```

```
            case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_START_BUFFERING:
```

```
                Debug.Log("Start_Buffering");
                break;
```

```
            case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_BUFFERING:
                Debug.Log("Buffering: " + Convert.ToInt32(param1));
                break;
```

```
            case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_STOP_BUFFERING:
                Debug.Log("Stop_Buffering");
                 break;
```

```
             case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_DOWNLOAD_SPEED:
                 Debug.Log("download_speed:" + param1 + "Byte/s" + ", "
                         + (Convert.ToInt32(param1) * 8 / 1000) + "kbps" + ", " + (Convert.ToInt32(
param1) / 1024)
                         + "KB/s");
                 break;
         }
```

```
    }
```

2. iOS 播放端 SDK 说明

2.1 demo 说明

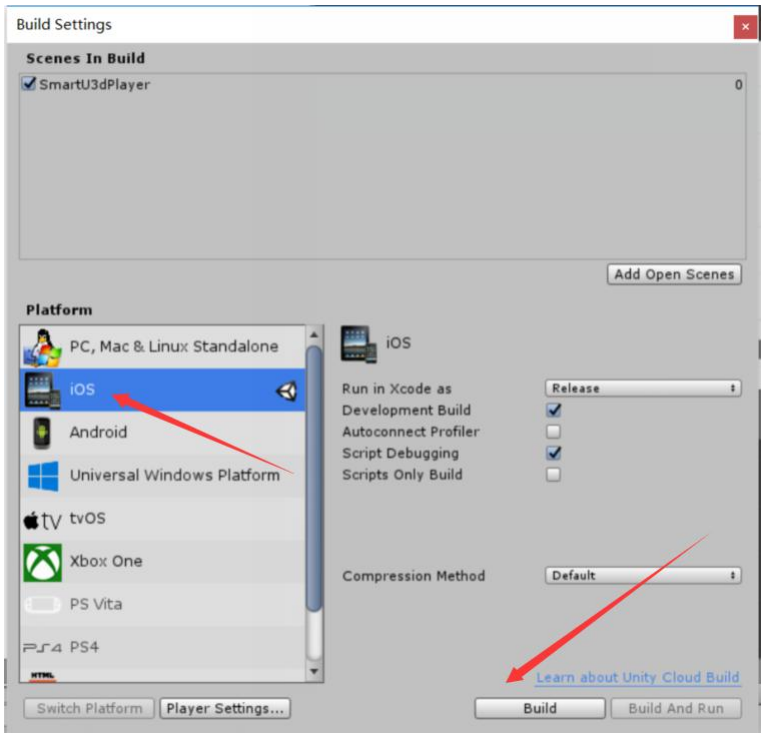
- SmartU3diOSPlayer：大牛直播 SDK Unity3D iOS 平台 RTMP/RTSP 直播播放端工程。

2.2 集成说明

拷贝以下文件，到 Assets-->Plugins-->iOS 目录：

相关头文件和调用说明，参见：SmartPlayeriOSMono.cs

Unity3D 工程下，File-->Build Settings，Platform 选择 iOS，然后点击 build，设置目录，生成 xcode 工程：



生成后的 xcode 工程，添加以下依赖库：

- 相关库：libSmartPlayerSDK.a
- 引入以下依赖 framework
  - libbz.tbd
  - Libbz2.tbd
  - libiconv.tbd
  - libstdc++.tbd
  - Libc++.tbd
  - Accelerate.framework
  - AssetsLibrary.framework
  - AudioToolBox.framework
  - AVFoundation.framework
  - CoreMedia.framework
  - Foundation.framework
  - GLKit.framework
  - OpenGLES.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>

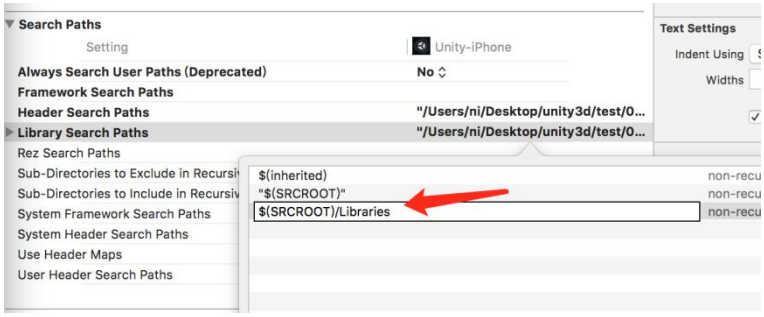
  <string>SmartiOSPlayer</string>
- 快照添加到“照片”权限：

  Info.plist-->右键 Open As-->Source Code

  添加

  <key>NSPhotoLibraryUsageDescription</key>

  <string>1</string>
- 导出后的 xcode 工程，如编译不过，参考以下设置：



2.3 调用时序

1. 【最先调用】NT_U3D_Init：player 初始化，目前预留；

2. **【正式授权版设置 license key】**NT_U3D_SetSDKClientKey，设置正式授权版 license，请早于 NT_U3D_Open 接口调用；

3. **【获得 player 句柄】**NT_U3D_Open，设置上下文信息，返回 player 句柄；

4. 【设置 GameObject】NT_U3D_Set_Game_Object，注册 Game Object，用于消息传递；

5. 【设置硬解码】NT_U3D_SetVideoDecoderMode，设置是否用硬解码播放，如硬解码不支持，切换到软解码；

6. 【缓冲设置】NT_U3D_SetBuffer，设置播放端缓存数据 buffer，以毫秒(ms)为单位，如超低延迟模式下，不需 buffer 数据，设置为 0；

7. 【RTSP TCP/UDP 设置】NT_U3D_SetRTSPTcpMode，设置 TCP/UDP 播放模式，**注意：此接口仅用于 RTSP；**

8. 【RTSP 超时时间设置】NT_U3D_SetRTSPTimeout，设置 RTSP 超时时间, timeout 单位为秒，必须大于 0；

9. 【RTSP TCP/UDP 自动切换】NT_U3D_SetRTSPAutoSwitchTcpUdp，设置 RTSP TCP/UDP 自动切换，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；

10. **【实时静音-可实时调用】**NT_U3D_SetMute，设置播放过程中，实时静音/取消静音；

11. 【快速启动】NT_U3D_SetFastStartup，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】NT_U3D_SetPlayerLowLatencyMode，设置低延迟模式；

13. **【视频垂直反转-可实时调用】**NT_U3D_SetFlipVertical，**视频垂直反转；**

14. **【视频水平反转-可实时调用】**NT_U3D_SetFlipHorizontal，**视频水平反转；**

15. **【视频显示角度设置-可实时调用】**NT_U3D_SetRotation，**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；**

16. 【下载速度回调设置】NT_U3D_SetReportDownloadSpeed，设置下载速度上报，默认不上报下载速度；

17. 【快照设置】NT_U3D_SetSaveImageFlag，设置是否需要在播放或录像过程中快照；

18. **【快照-录像或播放后，可随时调用】**NT_U3D_SaveCurImage，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】**NT_U3D_SwitchPlaybackUrl，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；

20. 【录像设置】NT_U3D_CreateFileDirectory，创建文件路径，注意：iOS 只提供接口，未提供具体实现；

21. 【录像设置】NT_U3D_SetRecorderDirectory，设置文件路径；

22. 【录像设置】NT_U3D_SetRecorderFileMaxSize，设置每个录像文件最大 size，以兆（M）为单位，范围[5M~500M]；

23. 【录像设置】NT_U3D_SetRecorderAudioTranscodeAAC，支持拉取的 RTMP/RTSP 的 PCMA/PCMU/SPEEX 音频格式转 AAC 后录制；

24. 【设置播放或录像 URL】NT_U3D_SetUrl，设置播放/录像 url；

25. **【播放】**NT_U3D_StartPlay，开始播放；

26. **【播放】**NT_U3D_GetVideoFrame，获取底层回调的 YUV 数据；

27. **【播放】**NT_U3D_StopPlay，停止播放；

28. **【录像】**NT_U3D_StartRecorder，开始录像；

29. **【录像】**NT_U3D_StopRecorder，停止录像；

30. **【关闭】**NT_U3D_Close，关闭播放器实例；

31. **【最后调用】**NT_U3D_UnInit，UnInit Player，最后调用。

2.4 相关实现

开始播放

完成相关的初始化和接口参数设定后，调用 NT_U3D_StartPlay()实现音视频数据播放。

```
public void Play(int sel)
{
    if (videoctrl[sel].is_running)
    {
        Debug.Log("已经在播放。。");
        return;
    }

    videoctrl[sel].player_handle_ = NT_U3D_Open();
```

```
    if (videoctrl[sel].player_handle_ == IntPtr.Zero)
    {
        Debug.LogError("open fail");
        return;
    }
```

```
    NT_U3D_Set_Game_Object(videoctrl[sel].player_handle_, game_object_);
```

```
    /* ++ 播放前参数配置可加在此处 ++ */
    int is_using_tcp = 1;        //TCP 模式
    NT_U3D_SetRTSPTcpMode(videoctrl[sel].player_handle_, is_using_tcp);
```

```
    int is_report = 0;
```

```
        int report_interval = 1;
        NT_U3D_SetReportDownloadSpeed(videoctrl[sel].player_handle_, is_report, report_interval);  //下载速度回调
```

```
        int play_buffer_time_ = 100;
        NT_U3D_SetBuffer(videoctrl[sel].player_handle_, play_buffer_time_);                         //设置buffer time
```

```
        int is_low_latency_ = 0;
        NT_U3D_SetPlayerLowLatencyMode(videoctrl[sel].player_handle_, is_low_latency_);    //设置是否启用低延迟模式
```

```
        int is_mute_ = 0;
        NT_U3D_SetMute(videoctrl[sel].player_handle_, is_mute_);                            //是否启动播放的时候静音
```

```
        int is_hw_decode_ = 1;
        NT_U3D_SetVideoDecoderMode(videoctrl[sel].player_handle_, is_hw_decode_);                //设置软硬解模式
```

```
        int is_fast_startup = 1;
        NT_U3D_SetFastStartup(videoctrl[sel].player_handle_, is_fast_startup);                    //设置快速启动模式
```

```
        int rtsp_timeout = 10;
        NT_U3D_SetRTSPTimeout(videoctrl[sel].player_handle_, rtsp_timeout);                       //设置RTSP 超时时间
```

```
        int is_auto_switch_tcp_udp = 1;
        NT_U3D_SetRTSPAutoSwitchTcpUdp(videoctrl[sel].player_handle_, is_auto_switch_tcp_udp);    //设置TCP/UDP 模式自动切换

        NT_U3D_SetUrl(videoctrl[sel].player_handle_, videoctrl[sel].videoUrl);
        /* -- 播放前参数配置可加在此处 -- */
```

```
        int isEnableYuvBlock = 1;  //是否回调 YUV 数据
        int flag = NT_U3D_StartPlay(videoctrl[sel].player_handle_, isEnableYuvBlock);
```

```
        if (flag == DANIULIVE_RETURN_OK)
        {
            videoctrl[sel].is_need_init_texture_ = true;
```

```
            videoctrl[sel].is_need_get_frame_ = true;
            Debug.Log("播放成功");
        }
        else
        {
            videoctrl[sel].is_need_get_frame_ = false;
            Debug.Log("播放失败");
        }
```

```
        videoctrl[sel].is_running = true;
    }
```

停止播放：

调用 NT_U3D_StopPlay()后，如果没有录像，调用 NT_U3D_Close()，播放 handler 置空，然后调用 NT_U3D_UnInit()即可。

```
    private void ClosePlayer(int sel)
    {
        videoctrl[sel].is_need_get_frame_ = false;
        videoctrl[sel].is_need_init_texture_ = false;

        int flag = NT_U3D_StopPlay(videoctrl[sel].player_handle_);
```

```
        if (flag == DANIULIVE_RETURN_OK)
        {
            Debug.Log("停止成功");
        }
        else
        {
            Debug.Log("停止失败");
        }
```

```
        flag = NT_U3D_Close(videoctrl[sel].player_handle_);
        if (flag == DANIULIVE_RETURN_OK)
        {
            Debug.Log("关闭成功");
        }
        else
        {
            Debug.Log("关闭失败");
        }
```

```
        videoctrl[sel].player_handle_ = IntPtr.Zero;
```

```
        videoctrl[sel].is_running = false;
    }
```

相关 Event 处理：

```
/// <summary>
/// iOS 传递过来 code
/// </summary>
/// <param name="code"></param>
public void onNTSmartEvent(string param)
{
    if (!param.Contains(","))
    {
        Debug.Log("[onNTSmartEvent] iOS 传递参数错误");
        return;
    }

    string[] strs = param.Split(',');

    string player_handle = strs[0];
    string code = strs[1];
    string param1 = strs[2];
    string param2 = strs[3];
    string param3 = strs[4];
    string param4 = strs[5];
```

```
    Debug.Log("[onNTSmartEvent] code: 0x" + Convert.ToString(Convert.ToInt32(code), 16));
```

```
    switch (Convert.ToInt32(code))
    {
        case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_STARTED:
            Debug.Log("开始。。");
            break;
        case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_CONNECTING:
            Debug.Log("连接中。。");
            break;
        case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_CONNECTION_FAILED:
            Debug.Log("连接失败。。");
            break;
```

```
        case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_CONNECTED:
            Debug.Log("连接成功。。");
            break;
        case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_DISCONNECTED:
            Debug.Log("连接断开。。");
            break;
        case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_STOP:
            Debug.Log("停止播放。。");
            break;
        case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_RESOLUTION_INFO:
            Debug.Log("分辨率信
息: width: " + Convert.ToInt32(param1) + ", height: " + Convert.ToInt32(param2));
            break;
        case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_NO_MEDIADATA_RECEIVED:
            Debug.Log("收不到媒体数据，可能是 url 错误。。");
            break;
        case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_SWITCH_URL:
            Debug.Log("切换播放 URL。。");
            break;
```

```
        case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_CAPTURE_IMAGE:
            Debug.Log("快照: " + param1 + " 路径: " + param3);
```

```
            if (Convert.ToInt32(param1) == 0)
            {
                Debug.Log("截取快照成功。.");
            }
            else
            {
                Debug.LogError("截取快照失败。.");
            }
            break;
```

```
        case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_RECORDER_START_NEW_FILE:
            Debug.Log("[record]开始一个新的录像文件 : " + param3);
            break;
        case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_ONE_RECORDER_FILE_FINISHED:
            Debug.Log("[record]已生成一个录像文件 : " + param3);
            break;
```

```
        case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_START_BUFFERING:
```

```
            Debug.Log("Start_Buffering");
            break;
```

```
        case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_BUFFERING:
            Debug.Log("Buffering: " + Convert.ToInt32(param1));
            break;
```

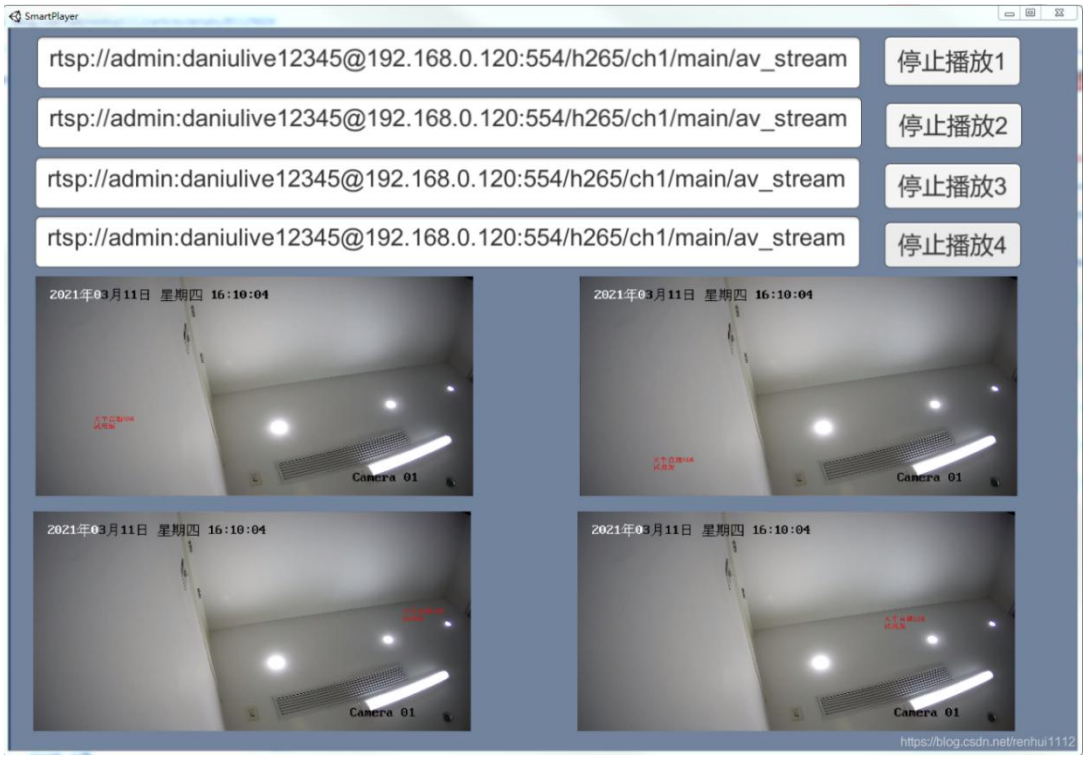
```
        case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_STOP_BUFFERING:
            Debug.Log("Stop_Buffering");
            break;
```

```
        case EVENTID.EVENT_DANIULIVE_ERC_PLAYER_DOWNLOAD_SPEED:
            Debug.Log("download_speed:" + param1 + "Byte/s" + ", "
                    + (Convert.ToInt32(param1) * 8 / 1000) + "kbps" + ", " + (Convert.ToInt32(param1) /
1024)
                    + "KB/s");
            break;
    }
```

3. Windows 播放端 SDK 说明

3.1 demo 说明

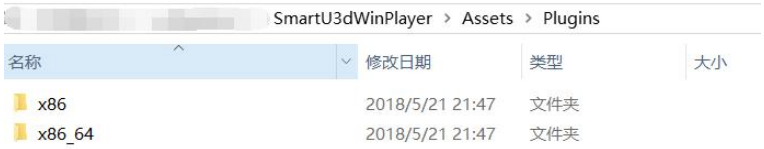
➢ SmartU3dWinPlayer：大牛直播 SDK Unity3D Windows 平台 RTMP/RTSP 直播播放端工程。Demo 界面如下：



3.2 集成说明

➢ Unity3D 接口和调用 demo，参见：SmartPlayerWindowsMono.cs

➢ SmartU3dWinPlayer\Assets\Plugins\x86 和 SmartU3dWinPlayer\Assets\Plugins\x86_64 下相关库到工程：



- nt_base_code_define.cs 加入到工程；
- smart_player_define.cs 加入工程；

- smart_player_sdk.cs 加入工程；
- smart_log.cs 加入工程；
- smart_log_define.cs 加入工程。

➢ 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，windows 测试 app name：SmartPlayer。

3.3 调用时序

Windows 平台 Unity3D 实际上调用的是大牛直播 SDK 的 C#相关接口实现，无需二次封装，相关调用时序，也可参照 Windows C#调用 SDK 调用说明(参见“视沃科技-Windows-SDK 集成说明.pdf”)。

调用流程：

1．开始播放的时候，调用 OpenPlayer()，完成 PlayerInit 和 PlayerOpen 操作，并设置 eventcallback 和 videoframecallback，并完成相关参数配置，设置 YUV 回调输出，调用 PlayerStart，完成播放端的启动。

注意：正式授权版，需在调用 NT_SP_Init()接口之前，调用 NT_SP_SetSDKClientKey()，设置相关 license key，确保验证通过。

```
public void Play(int sel)
 {
     if (videoctrl[sel].is_running)
     {
         Debug.Log("已经在播放..");
         return;
     }

     lock (videoctrl[sel].frame_lock_)
     {
         videoctrl[sel].cur_video_frame_ = null;
     }
```

```
     OpenPlayer(sel);
```

```
     if (videoctrl[sel].player_handle_ == IntPtr.Zero)
         return;
```

```
     //设置播放 URL
```

```
        NTSmartPlayerSDK.NT_SP_SetURL(videoctrl[sel].player_handle_, videoctrl[sel].videoUrl);

        /* ++ 播放前参数配置可加在此处 ++ */

        int play_buffer_time_ = 100;
        NTSmartPlayerSDK.NT_SP_SetBuffer(videoctrl[sel].player_handle_, play_buffer_time_);
    //设置 buffer time

        int is_using_tcp = 0;         //TCP 模式
        NTSmartPlayerSDK.NT_SP_SetRTSPTcpMode(videoctrl[sel].player_handle_, is_using_tcp);

        int timeout = 10;
        NTSmartPlayerSDK.NT_SP_SetRtspTimeout(videoctrl[sel].player_handle_, timeout);

        int is_auto_switch_tcp_udp = 1;
        NTSmartPlayerSDK.NT_SP_SetRtspAutoSwitchTcpUdp(videoctrl[sel].player_handle_, is_auto_switch_tcp
_udp);

        Boolean is_mute_ = false;
        NTSmartPlayerSDK.NT_SP_SetMute(videoctrl[sel].player_handle_, is_mute_ ? 1 : 0);
    //是否启动播放的时候静音

        int is_fast_startup = 1;
        NTSmartPlayerSDK.NT_SP_SetFastStartup(videoctrl[sel].player_handle_, is_fast_startup);
    //设置快速启动模式

        Boolean is_low_latency_ = false;
        NTSmartPlayerSDK.NT_SP_SetLowLatencyMode(videoctrl[sel].player_handle_, is_low_latency_ ? 1 : 0);
    //设置是否启用低延迟模式

        //设置旋转角度(设置 0， 90， 180， 270 度有效，其他值无效)
        int rotate_degrees = 0;
        NTSmartPlayerSDK.NT_SP_SetRotation(videoctrl[sel].player_handle_, rotate_degrees);


        int volume = 100;
        NTSmartPlayerSDK.NT_SP_SetAudioVolume(videoctrl[sel].player_handle_, volume);   //设置播放音量，范
围是[0, 100]，0 是静音，100 是最大音量，默认是 100


        // 设置上传下载报速度
        int is_report = 0;
```

```
        int report_interval = 1;
        NTSmartPlayerSDK.NT_SP_SetReportDownloadSpeed(videoctrl[sel].player_handle_, is_report, report_i
nterval);
        /* -- 播放前参数配置可加在此处 -- */
```

```
        //video frame callback (YUV/RGB)
        videoctrl[sel].video_frame_call_back_ = new SP_SDKVideoFrameCallBack(NT_SP_SetVideoFrameCallBack)
;
        NTSmartPlayerSDK.NT_SP_SetVideoFrameCallBack(videoctrl[sel].player_handle_, (Int32)NT.NTSmartPla
yerDefine.NT_SP_E_VIDEO_FRAME_FORMAT.NT_SP_E_VIDEO_FRAME_FROMAT_I420, window_handle_, videoctrl[sel].vid
eo_frame_call_back_);
```

```
        UInt32 flag = NTSmartPlayerSDK.NT_SP_StartPlay(videoctrl[sel].player_handle_);
```

```
        if (flag == DANIULIVE_RETURN_OK)
        {
            videoctrl[sel].is_need_get_frame_ = true;
            Debug.Log("播放成功");
        }
        else
        {
            videoctrl[sel].is_need_get_frame_ = false;
            Debug.LogError("播放失败");
        }
```

```
        videoctrl[sel].is_running = true;
    }
```

2. OpenPlayer 实现：

```
   private void OpenPlayer(int sel)
    {
        window_handle_ = IntPtr.Zero;

        if (videoctrl[sel].player_handle_ == IntPtr.Zero)
        {
            videoctrl[sel].player_handle_ = new IntPtr();
            UInt32 ret_open = NTSmartPlayerSDK.NT_SP_Open(out videoctrl[sel].player_handle_, window_hand
le_, 0, IntPtr.Zero);
            if (ret_open != 0)
            {
                videoctrl[sel].player_handle_ = IntPtr.Zero;
```

```
                Debug.LogError("调用 NT_SP_Open 失败..");
                return;
            }
        }
```

```
        videoctrl[sel].event_call_back_ = new SP_SDKEventCallBack(NT_SP_SDKEventCallBack);
        NTSmartPlayerSDK.NT_SP_SetEventCallBack(videoctrl[sel].player_handle_, window_handle_, videoctrl
[sel].event_call_back_);
```

```
        videoctrl[sel].sdk_video_frame_call_back_ = new VideoControl.SetVideoFrameCallBack(SDKVideoFrame
CallBack);
        videoctrl[sel].sdk_event_call_back_ = new VideoControl.SetEventCallBack(SDKEventCallBack);
    }
```

3. Video frame 实时处理并绘制：

开始播放后，daniulive 直播播放端 SDK 回调 yuv 数据及相关信息，unity3d 获取到数据信息后，调用 InitYUVTexture()，完成初始化工作，调用 UpdateYUVTexture()实现数据实时刷新，当数据信息发生变化时，会二次调用 InitYUVTexture()，完成初始化。

```
    private void SDKVideoFrameCallBack(UInt32 status, IntPtr frame, int sel)
    {
        //这里拿到回调 frame，进行相关操作
        NT_SP_VideoFrame video_frame = (NT_SP_VideoFrame)Marshal.PtrToStructure(frame, typeof(NT_SP_Vide
oFrame));

        VideoFrame  u3d_frame = new VideoFrame();
```

```
        u3d_frame.width_  = video_frame.width_;
        u3d_frame.height_ = video_frame.height_;
```

```
        u3d_frame.timestamp_ = (UInt64)video_frame.timestamp_;
```

```
        int d_y_stride = video_frame.width_;
        int d_u_stride = (video_frame.width_ + 1) / 2;
        int d_v_stride = d_u_stride;
```

```
        int d_y_size = d_y_stride * video_frame.height_;
        int d_u_size = d_u_stride * ((video_frame.height_ + 1) / 2);
        int d_v_size = d_u_size;
```

```
        int u_v_height = ((u3d_frame.height_ + 1) / 2);

        u3d_frame.y_stride_ = d_y_stride;
        u3d_frame.u_stride_ = d_u_stride;
        u3d_frame.v_stride_ = d_v_stride;

        u3d_frame.y_data_ = new byte[d_y_size];
        u3d_frame.u_data_ = new byte[d_u_size];
        u3d_frame.v_data_ = new byte[d_v_size];

        CopyFramePlane(u3d_frame.y_data_, d_y_stride,
            video_frame.plane0_, video_frame.stride0_, u3d_frame.height_);

        CopyFramePlane(u3d_frame.u_data_, d_u_stride,
           video_frame.plane1_, video_frame.stride1_, u_v_height);

        CopyFramePlane(u3d_frame.v_data_, d_v_stride,
           video_frame.plane2_, video_frame.stride2_, u_v_height);

        lock (videoctrl[sel].frame_lock_ )
        {
            videoctrl[sel].cur_video_frame_ = u3d_frame;
            //Debug.LogError("sel: " + sel + " w:" + u3d_frame.width_ + "h:" + u3d_frame.height_);
        }
    }
```

4．停止播放

调用 NT_U3D_StopPlay()后，调用 NT_U3D_UnInit()即可。

```
    private void ClosePlayer(int sel)
    {
        videoctrl[sel].is_need_get_frame_ = false;
        videoctrl[sel].is_need_init_texture_ = false;

        if (videoctrl[sel].player_handle_ == IntPtr.Zero)
        {
            return;
        }

        UInt32 flag = NTSmartPlayerSDK.NT_SP_StopPlay(videoctrl[sel].player_handle_);
        if (flag == DANIULIVE_RETURN_OK)
        {
```

```
            Debug.Log("停止成功");
        }
        else
        {
            Debug.LogError("停止失败");
        }
```

```
        videoctrl[sel].player_handle_ = IntPtr.Zero;
```

```
        videoctrl[sel].is_running = false;
    }
```

5. Event 回调

```
    private void SDKEventCallBack(UInt32 event_id,
        Int64 param1,
        Int64 param2,
        UInt64 param3,
        [MarshalAs(UnmanagedType.LPStr)] String param4,
        [MarshalAs(UnmanagedType.LPStr)] String param5,
        IntPtr param6,
        int sel)
    {
        if ((UInt32)NTSmartPlayerDefine.NT_SP_E_EVENT_ID.NT_SP_E_EVENT_ID_CONNECTING == event_id
            || (UInt32)NTSmartPlayerDefine.NT_SP_E_EVENT_ID.NT_SP_E_EVENT_ID_CONNECTION_FAILED == event_
id
            || (UInt32)NTSmartPlayerDefine.NT_SP_E_EVENT_ID.NT_SP_E_EVENT_ID_CONNECTED == event_id
            || (UInt32)NTSmartPlayerDefine.NT_SP_E_EVENT_ID.NT_SP_E_EVENT_ID_DISCONNECTED == event_id)
        {
            connection_status_ = event_id;
        }
        if ((UInt32)NTSmartPlayerDefine.NT_SP_E_EVENT_ID.NT_SP_E_EVENT_ID_START_BUFFERING == event_id
            || (UInt32)NTSmartPlayerDefine.NT_SP_E_EVENT_ID.NT_SP_E_EVENT_ID_BUFFERING == event_id
            || (UInt32)NTSmartPlayerDefine.NT_SP_E_EVENT_ID.NT_SP_E_EVENT_ID_STOP_BUFFERING == event_id)
        {
            buffer_status_ = event_id;
            if ((UInt32)NTSmartPlayerDefine.NT_SP_E_EVENT_ID.NT_SP_E_EVENT_ID_BUFFERING == event_id)
            {
                buffer_percent_ = (Int32)param1;
            }
        }
        if ((UInt32)NTSmartPlayerDefine.NT_SP_E_EVENT_ID.NT_SP_E_EVENT_ID_DOWNLOAD_SPEED == event_id)
```

```
        {
            download_speed_ = (Int32)param1;
        }

        String t_show_str = "";
        if (connection_status_ != 0)
        {
            t_show_str += "链接状态: ";
            if ((UInt32)NTSmartPlayerDefine.NT_SP_E_EVENT_ID.NT_SP_E_EVENT_ID_CONNECTING == connection_s
tatus_)
            {
                t_show_str += "链接中";
            }
            else if ((UInt32)NTSmartPlayerDefine.NT_SP_E_EVENT_ID.NT_SP_E_EVENT_ID_CONNECTION_FAILED ==
connection_status_)
            {
                t_show_str += "链接失败";
            }
            else if ((UInt32)NTSmartPlayerDefine.NT_SP_E_EVENT_ID.NT_SP_E_EVENT_ID_CONNECTED == connecti
on_status_)
            {
                t_show_str += "链接成功";
            }
            else if ((UInt32)NTSmartPlayerDefine.NT_SP_E_EVENT_ID.NT_SP_E_EVENT_ID_DISCONNECTED == conne
ction_status_)
            {
                t_show_str += "链接断开";
            }
        }
```

```
        if (download_speed_ != -1)
        {
            String ss = "  下载速
度: " + (download_speed_ * 8 / 1000).ToString() + "kbps " + (download_speed_ / 1024).ToString() + "KB/s"
;
            t_show_str += ss;
        }
```

```
        if (buffer_status_ != 0)
        {
             t_show_str += "  缓冲状态: ";
```

```
            if ((UInt32)NTSmartPlayerDefine.NT_SP_E_EVENT_ID.NT_SP_E_EVENT_ID_START_BUFFERING == buffer_
status_)
            {
                t_show_str += "开始缓冲";
            }
            else if ((UInt32)NTSmartPlayerDefine.NT_SP_E_EVENT_ID.NT_SP_E_EVENT_ID_BUFFERING == buffer_s
tatus_)
            {
                String ss = "缓冲中 " + buffer_percent_.ToString() + "%";
                t_show_str += ss;
            }
            else if ((UInt32)NTSmartPlayerDefine.NT_SP_E_EVENT_ID.NT_SP_E_EVENT_ID_STOP_BUFFERING == buf
fer_status_)
            {
                t_show_str += "结束缓冲";
            }
        }
        //show_message = "DaniuEvent: instance" + sel + ": " + t_show_str;
        //Debug.LogError(show_message);
    }
```