网宿推流 SDK-iOS 版本嵌入说明书

网宿科技股份有限公司

2018-01-24

一、嵌入说明

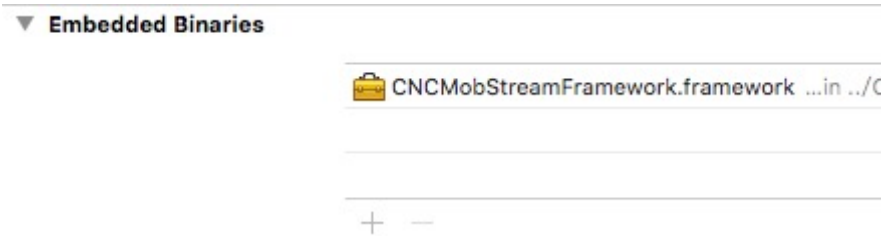
1、文件说明

移动端 SDK-iOS 版本提供的是 Framework，文件名为 CNCMobStreamFramework.framework。其中重要的头文件有：

CNCMobComDef.h	主要是错误码及一些关键参数的枚举值定义
CNCMobStreamSDK.h	主要是 SDK 提供的接口定义
CNCMobStruct.h	主要是关键参数结构定义
CNCComDelegateDef.h	包含编码、发送相应代理、接口。及参数获取、设置接口。
CNCVideoSourceInput.h	包含音频、编码、录制、推流行为控制等相关接口。
CNCCaptureVideoDataManager.h	视频采集及相关设置接口
CNCMobStreamAudioEngine.h	音频采集及相关设置接口
CNCMobStreamRtmpSender.h	发送层包括头部及音视频帧等
CNCMobStreamTimeStampGenerator.h	时间戳发生器 音视频时间戳生成器
CNCMobStreamVideoDisplayer.h	预览层 显示采集数据
CNCMobStreamVideoEncoder.h	编程层 视频帧编码 代理返回编码结果
CNCRecordFileSeesionManager.h	文件录制 flv、MP4 格式的文件录制

2、工程配置

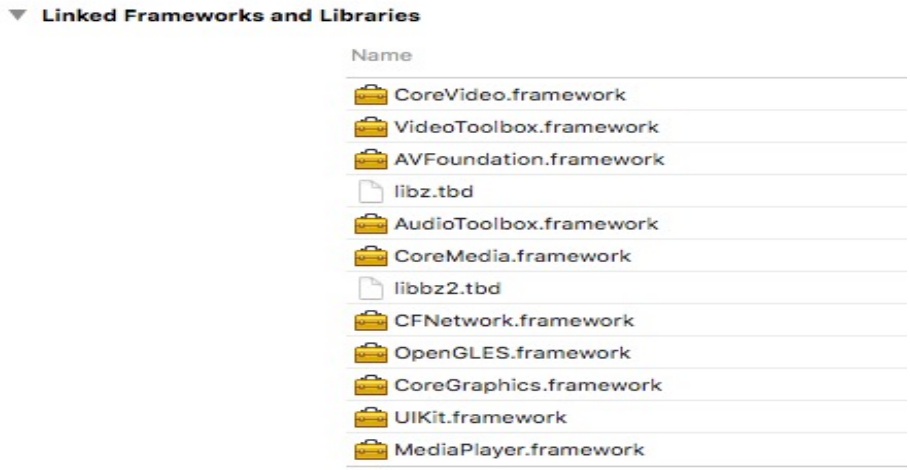
a、在工程的 General 中的 Embedded Binaries 中添加 CNCMobStreamFramework.framework；



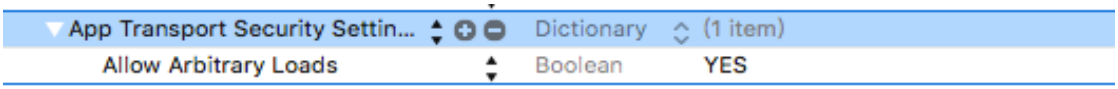
b、在工程的 Build Setting 中的 Framework Search Pahts 中添加 framework 的所

在路径，并将 Enable Bitcode 设置为 YES

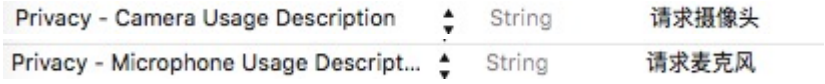
c、在工程的 General 中的 Linked Frameworks and Libraries 添加如下相应的库



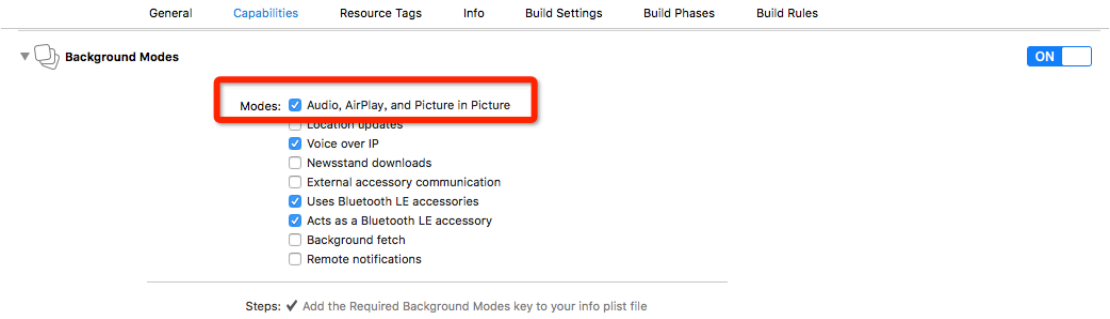
d、在工程的 Info.plist 添加如下 Http 权限



e、在工程的 Info.plist 添加如下摄像头、麦克风申请权限提示语以适配 iOS 10

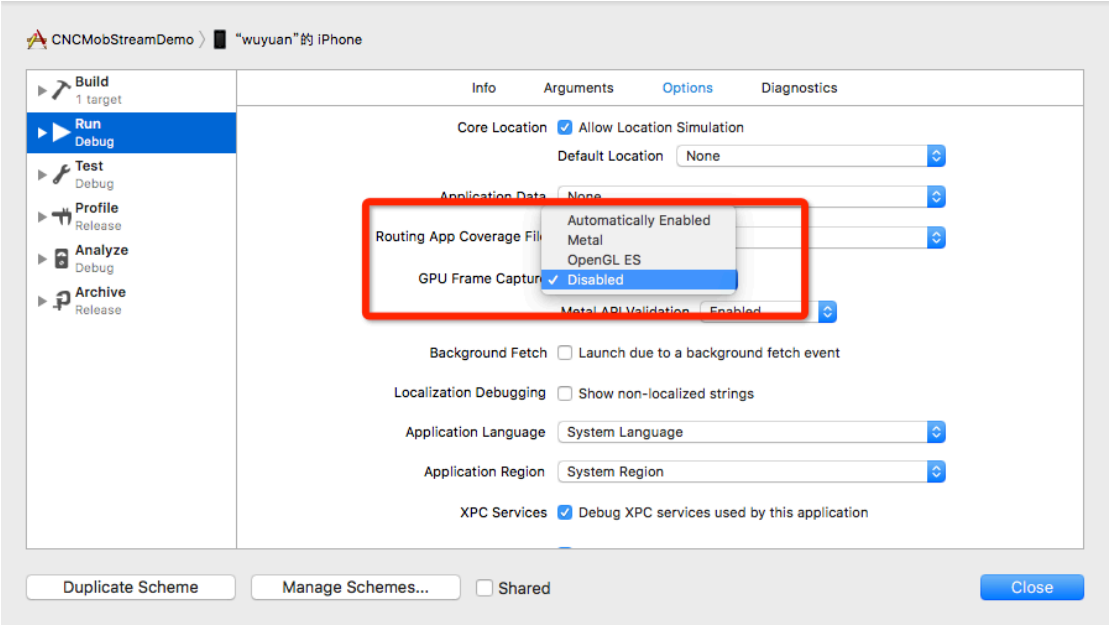


f、后台推流使用前需在工程设置中打开相应开关

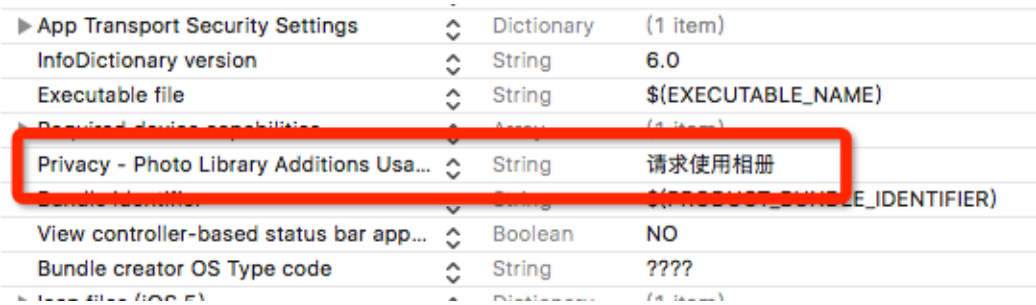


g、提交 appstore 之前请打开 framework 的 Info.plist 文件 选择最低支持版本为 8.0，然后打包提交

h、使用 Xcode 9.0 时为确保 framework 正常使用请 GPU Frame Capture 为 Disable



i、使用 Xcode 9.0 时为确保 iOS 11 设备的相册功能正常使用请添加 NSPhotoLibraryAddUsageDescription 字段



3、支持的软硬件版本

CNCMobStreamFramework.framework 支持的硬件是 iPhone 4S 及高于此硬件的设备，支持的软件版本为 iOS7.0 及高于此版本的 iOS 系统。

二、推流器功能使用说明

以下描述主要接口的使用场景及方法，接口详细说明可见 CNCMobStreamSDK.h

1 推流 SDK 初始化

调用 CNCMobStreamSDK 的

```
/*! @brief 向CNC注册第三方应用。
 *
 * 需要在每次启动第三方应用程序时调用， 此接口为同步接口 网络情况不好时 可能会耗时
较长。
 * @param   appid    CNC分配给客户的ID
 authKey  CNC分配给客户的key
 * @return 详见错误码CNC_Ret_Code定义。
 */
+ (CNC_Ret_Code)regist_app:(NSString *)app_id auth_key:(NSString
*)auth_key;
```

或者

```
/*! @brief 向CNC注册第三方应用。
 *
 * 需要在每次启动第三方应用程序时调用， 此接口为异步接口 请在调用前监听
CNC_Notification_MobStreamSDK_Init_DidFinished 消息 将返回结果的
NSNotification中的object对象从NSNumber转化为CNC_Ret_Code 错误信息详见错误
码CNC_Ret_Code定义 。
 * @param   appid    CNC分配给客户的ID
 authKey  CNC分配给客户的key
 * @return 无。
 */
+ (void)async_regist_app:(NSString *)app_id auth_key:(NSString
*)auth_key;
```

一个为同步接口，一个为异步接口，用户视情况可以自行选择调用，app_id 和 auth_key 为客户权限信息，请向相关人员垂询获取，详见 CNCMobStreamSDK.h 头文件中的说明。

示例代码：

```
  NSString *app_id = @"ws.com";
   NSString *auth_key = @"5F9D375CB57B40E6A788B9069886B1AF";

   CNC_Ret_Code ret = [CNCMobStreamSDK regist_app:app_id
auth_key:auth_key];
   if (ret == CNC_RCode_Success) {

      NSLog(@"SDK鉴权成功 业务接口可用 ret = %@", @(ret));

   } else {

      NSLog(@"SDK鉴权失败 业务接口不可用ret = %@", @(ret));

   }
```

2 启动推流

1) 设置推流参数

参数结构体为 CNCStreamCfg 具体各个参数详见 CNCMobStruct.h 定义说明，大部分参数都有默认值 用户可视具体应用场景另行赋值，定义好结构体之后，调用

```
+ (BOOL)set_stream_config:(CNCStreamCfg *)para;
```

2) 在设置好参数之后再设置预览的 view

```
+ (BOOL)set_show_video_preview:(UIView *)preview;
```

3) 调用启动推流的接口

设置好参数及预览 view 之后调用启动推流的接口

```
+ (BOOL)start_push;
```

即可开始进入录制推流的流程。

4) 暂停推流

```
+ (BOOL)pause_push;
```

采集正常，仅停止推流

5) 推流前重置推流地址 url

```
+ (BOOL)reset_url_string_before_pushing:(NSString *)urlString;
```

6) 获取当前推流使用的 url

```
+ (NSString *)get_rtmp_url_string;
```

7) 设置码率自适应

```
+ (void)set_adaptive:(BOOL)isAdaptive;
```

8) 查询当前版本号

```
+ (NSString *)get_sdk_version;
```

9) 开启日志系统

```
+ (void)set_log_system_enable:(BOOL)enable;
```

10) 预览横竖屏重置

```
 /*! @brief 推流前 预览时中重置横竖屏 且要同时重置预览页

 * 这个接口要和reset_preview_frame配合着用 因为旋转之后必然伴随着preview

的frame发生变化 如长宽对换 另 如有水印也要重新设置

 *

 * @param new_direct 新的方向

 * @return 设置结果 成功为YES 失败为NO

 */
+ (BOOL)reset_direct:(CNCENMDirectType)new_direct;
```

11) 动态设置采集帧率

```
 /*! @brief 根据手机性能设置动态设置摄像头采集帧率

 *
```

```
 * @param auto_fps 开启动态设置

 *
 */
+ (void)global_enable_capture_auto_fps:(BOOL)auto_fps;
```

3 推流过程中动态修改参数的接口

1) 修改码率

```
+ (NSInteger)reset_bit_rate:(NSInteger)new_bit_rate;
```

配套的有获取当前使用码率的接口:

```
+ (NSInteger)get_bit_rate;
```

2) 修改使用的摄像头

前后摄像头切换：

```
+ (void)swap_cameras;
```

切换到指定的摄像头：

```
+ (void)reset_came_pos:(AVCaptureDevicePosition)new_position;
```

配套的有获取当前使用的摄像头的接口：

```
+ (AVCaptureDevicePosition)get_came_pos;
```

3) 聚焦变焦

手动聚焦

```
+ (BOOL)tapFocusAtPoint:(CGPoint)point;
```

变焦

```
+ (void) videoZoomFactorWithScale:(CGFloat)effectiveScale;
```

4) 获取帧率

```
+ (NSInteger)get_video_fps;
```

5) 闪关灯

```
+ (void)set_torch_mode:(BOOL)enable;
```

6) 静音功能

```
+ (void)set_muted_statu:(BOOL)isMute;
```

7) 获取静音状态

```
+ (BOOL)isMuted;
```

8) 重设预览页 frame

```
+ (void)reset_preview_frame:(CGRect)frame;
```

9) 获取当前摄像头最大放大倍数

```
+ (CGFloat)get_current_camera_upscale;
```

10) 获取视频帧信息

```
/*! @brief 获取视频帧统计信息。

 * 默认为nil

* @return 成功返回realTimeFrame，失败返回nil。

*/

+ (realTimeFrame *)get_statistics_message;
```

4 停止推流

停止推流的接口：

```
+ (BOOL)stop_push;
```

5 推流过程中消息监听

1) 实时传输速度监听

添加监听注册消息：

```
kMobStreamSDKSendSpeedNotification
```

可以获取当前推流实时的传输速度，单位为 KB/s。

6 美颜功能

1) 磨皮

调用如下方法

```
/*! @brief  美颜磨皮
 *
 * @param beauty 磨皮等级参数 值域为 0~2.0 默认为 1.0
 * @param effect 附加效果  值域为 0~1.0 默认为 0.5          (若无附加效果 此参数设置无效)
 * @param filter_effect 滤镜效果 值域为 0~1.0 默认为 0.5 (若滤镜效果不可调节 此参数设置无效)
 *
 */
+ (void)set_beauty_with:(CGFloat)beauty effect:(CGFloat)effect filter_effect:(CGFloat)fliter_effect;
```

2) 美颜样式选择

```
/*! @brief  美颜样式选择
 *
 * @param type  CNCBEAUTY
 *
```

```
 */
+ (void)set_beauty_filter_with_type:(CNCBEAUTY)type;
```

3) 滤镜选择

```
/*! @brief  滤镜选择

 * @param type  CNCCOMBINE
 */
+ (void)set_combine_filter_with_type:(CNCCOMBINE)type;
```

7 混音混响功能

1. 混音功能（背景音功能）

```
/*!@brief  传入播放音乐的地址，并设置是否循环播放

 *

 * @param  filePath  传入音乐文件的存放路径

 * @param  enable  是否循环播放

 * 注:传入路径后会自动开始播放，切换音乐可以调用此接口

 */
+ (BOOL)setUpAUFilePlayer:(NSString *)filePath
loopEnable:(BOOL)enable;
```

2. 播放控制

```
///开始播放音乐或恢复播放音乐
+ (void)startPlayMusic;

///结束播放音乐
+ (void)stopPlayMusic;

///暂停播放
+ (void)pausePlayMusic;
```

```
/*!@brief  调节播放
 *
 * @param  position  0->1，传入进度条的位置值
 *
 */
+ (void)seekPlayheadTo:(CGFloat)position;
```

3. 混响功能

```
/*! @brief  设置混响房间大小类型
 *
 * @param type  CNCAudioRoomType
 *
 */
+ (void)setAudioRoomType:(CNCAudioRoomType)type;
```

```
/// 设置混响环境类型枚举值
typedef CF_ENUM(UInt32, CNCAudioRoomType) {
    CNCAudioRoomType_SmallRoom      = 0,
    CNCAudioRoomType_MediumRoom     = 1,
    CNCAudioRoomType_LargeRoom      = 2,
    CNCAudioRoomType_MediumHall     = 3,
    CNCAudioRoomType_LargeHall      = 4,
    CNCAudioRoomType_Plate      = 5,
    CNCAudioRoomType_MediumChamber  = 6,
    CNCAudioRoomType_LargeChamber   = 7,
    CNCAudioRoomType_Cathedral      = 8,
    CNCAudioRoomType_LargeRoom2     = 9,
    CNCAudioRoomType_MediumHall2 = 10,
    CNCAudioRoomType_MediumHall3 = 11,
    CNCAudioRoomType_LargeHall2     = 12
};
```

```
/*! @brief  设置混响影响因子
 *
 * @param value  [0->100]，默认60
```

```
 *
 */

+ (void)setAudioMixReverb:(Float32)value;
```

4. 音量调节

```
///设置人声大小 0->1,默认 1.0f
+ (void)setHumanVolume:(Float32)value;

///设置音乐大小 0->1,默认 0.3f
+ (void)setMusicVolume:(Float32)value;

///设置总输出音量大小 0->1,默认 1.0f
+ (void)setOutPutVolume:(Float32)value;
```

5. 状态值获取

```
///获取状态

///当前人声音量大小
+ (Float32)currentHumanVolume;
///当前音乐音量大小
+ (Float32)currentMusicVolume;
///当前输出音量大小
+ (Float32)currentOutPutVolume;
///当前混响因子大小
+ (Float32)currentAudioMixReverb;
///音乐总时长 格式：0:00
+ (NSString *)getDurationString;

///当前播放时间字符串 格式：0:00
+ (NSString*)getPlayTimeString;
```

```
///当前播放进度【0~1】
+ (float)getPlayProgress;
```

6. 耳返开关控制

```
/**
 @abstract  查询当前是否有耳机
 */
+ (BOOL)is_headset_plugged_in;

// 是否开启耳返
+ (BOOL)isOpenMicVoiceReturnBack;

/*! @brief 是否开启耳返
 *
 * 默认为关闭,耳返功能只适用插入耳机的情况

 * @return。
 */
+ (void)set_audio_returnback:(BOOL)enable;
// 开启新音频采集，默认为开启状态
+ (void)setUsingAudioManagerEnable:(BOOL)enable;
```

7. 后台推流的控制

```
///切换到后台时，是否继续推流 默认为NO
+ (BOOL)isContinuePushInBk;

/*! @brief 设置应用切换到后台时，推拉流是否继续

* 继续推流传YES,终止推流传NO

* 不设置时，默认为NO

* @return 成功返回yes，失败返回NO。
```

```
* 注：后台继续推流以及后台继续拉流的状态必须统一
*/
+ (BOOL)set_whether_continue_push_inBk:(BOOL)enable;
```

8. 设置音频输出

```
/*! @brief 设置全局音频默认输出到扬声器
 * 注：设置为NO时，可解决某些设备音频采集会有'哒哒哒'破音问题，
 * 同时音乐播放的声音从听筒出来会造成声音小问题
 * @param enable 扬声器传YES，听筒传NO
 * 不设置时，默认为YES，即音频输出到扬声器
 * @return 无。
 */
+ (void)globalSetAudioExportDefaultToSpeaker:(BOOL)enable;
```

8 水印功能

```
/*! @brief 设置水印 仅支持图片UIImageView及文字UILabel
 *
 * @param object 添加的水印 设置为nil为清除水印
 * @param scale 水印位置比例（x,y,width,height）范围为0~1
 * @param updateBlock 更新block 支持在block内对水印内容及图片进行修改
 * @return 添加水印结果
 */
+ (BOOL)overlayMaskWithObject:(UIView *)object rect:(CGRect)scale
block:(void (^)(void))updateBlock;
```

9 录制、截屏功能

录制文件存储位置默认：APP/Document/CNC_Save
录制文件名为 streampusher+当前时间戳.flv/mp4/jpg

1. 录制功能

```
/*! @brief 录制开始

* 录制分为长短视频、GIF录制，其中视频录制又细分为flv、MP4两种。其中短视频、GIF录制可以设置录制时长控制，长视频录制可设置大小控制。

* 当录制时长、大小达到预设值时自动结束当前录制并保存结果。

* 另有提示码提示录制结果 录制开始前请监听返回码CNC_Notification_MobStreamSDK_ReturnCode

* @param path 录制文件存储地址

* @param type 录制文件类型

* @param max_time 录制时长 短视频录制时长控制值域为 3.0s~60.0s。GIF录制时长值域为100.0ms~5.0s。

* @param long_store 录制长、短视频类型

* @param size 录制文件大小 长视频录制大小控制最小值为100kb，不足最小值以最小值为准。

* @param return_id 确认ID 随同录制结束返回

*

* @return 录制start结果

*/
+ (BOOL)start_store_video:(NSString *)path
                 fileType:(CNCRecordVideoType)type
                 max_time:(NSInteger)max_time
               long_store:(BOOL)long_store
                     size:(uint64_t)size
                return_id:(NSString *)return_id;
```

```
/*! @brief 录制停止

 * @return 录制文件存储路径

 */
+ (NSString *)stop_store_video;
```

```
/*! @brief 录制状态获取

 * @return 是否处于录制状态

 */
+ (BOOL)is_doing_store;
```

2. 截屏功能

```
 /*! @brief  截屏

 *

 * @param path 截屏文件存储地址

 * @return 截屏是否成功

 *
 */
+ (BOOL)screen_shot:(NSString *)path;
```

10 Socks5 代理

1. 是否开启 Socks 代理

```
/*! @brief  是否打开Socket5

 */
+ (BOOL)isUseSocks5Push;
```

2. 开启 Socks5 推流

```
/*! @brief 打开Socks5推流

 * @param ip    代理IP地址

 * @param port  IP端口号

 * @param userName 用户名

 * @param passw    密码

 *

 * @return 是否设置参数成功；（并不对用户名以及密码做验证）；

 */
+ (BOOL)openSocks5WithIp:(NSString *)ip withPort:(NSInteger)port
withUser:(NSString *)userName withPass:(NSString *)passw;
```

3. 关闭 Socks5 推流

```
/*! @brief  关闭Socks5

 */
+ (void)closeSocks5;
```

11 镜像功能

```
/*! @brief 设置编码层镜像及预览层镜像

 * @param source_mirror 编码层镜像

 * @param preview_mirror 预览层镜像

*/
+(void)set_source_mirror:(BOOL)source_mirror
preview_mirror_:(BOOL)preview_mirror;
```

12 SEI

```
 /*! @brief 软编编码开启SEI。
 *
 * @param json_str 发送字段
```

```
 * @return 无。
 */
+ (void)set_sw_encoder_push_time_stamp:(NSString *)json_str;
```

三、分层嵌入、接口说明

1 分层架构初始化

1. 推流器相关参数设置（CNCVideoSourceCfg）

```
/*! @brief 推流过程中参数设置
 * @param para
 * @return 设置结果
 */
- (BOOL)preset_para:(CNCVideoSourceCfg *)para;
```

2. 开启音频采集

```
/*! @brief 开启音频采集 */
- (void)start_audio_run;
```

2 推流及参数设置

1. 开始推流

```
/*! @brief 开始推流
* @param 无
* @return 开启是否成功
*/
- (BOOL)start_push;
```

2. 暂停推流

```
/*! @brief 暂停推流

 * @param 无

 * @return 暂停是否成功

 * 注：并没有关闭摄像头。

 */
- (BOOL)pause_push;
```

3. 停止推流

```
/*! @brief 停止推流

* @param 无

* @return 停止是否成功

* 注：关闭摄像头、释放内存。

*/
- (BOOL)stop_push;
```

4. 重置推流地址

```
/*! @brief 推流前重置推流地址url

* 只可在开始推流前配置，推流过程中无法重置

* @param 推流地址url

* @return 布尔值

*/
- (BOOL)reset_url_string_before_pushing:(NSString *)urlString;
```

5. 码率重置

```
/*! @brief 推流过程中重置推流使用的码率
```

```
* @param 新的码率 单位kbps

* @return 重置后推流使用的码率 单位kbps

*/
- (NSInteger)reset_bit_rate:(NSInteger)new_bit_rate;
```

6. 获取当前推流地址

```
/*! @brief 获取当前推流使用的url

 * @param

* @return 当前推流使用的url

*/
- (NSString *)get_rtmp_url_string;
```

7. 获取码率

```
/*! @brief 获取当前推流码率自适应的码率 单位kbps

* @return 当前推流码率自适应的码率 单位kbps

*/
- (NSInteger)get_bit_rate;
```

8. 获取帧率

```
/*! @brief 获取当前推流使用的帧率 单位fps

*
* @param

* @return 当前推流使用的帧率 单位fps

*/
- (NSInteger)get_video_fps;
```

9. 视频采集帧发送到编码层

```
/*! @brief 将视频帧数据buf发给SDK用于编码及RTMP推流。

* @param buf 视频帧数据块起始地址

* @param pix_width 数据块横向字节数

* @param pix_height 数据块纵向字节数 pix_width*pix_height是BUF的总大小

* @param format 数据块格式

* @param ts 视频帧的时间戳

* @return 无。

*/
- (void)send_frame_buf:(void *)buf pix_width:(int)pix_width
pix_height:(int)pix_height format:(CNCENM_Buf_Format)format
time_stamp:(long long)ts;
```

```
/*! @brief 将视频帧数据pixelBuffer发给SDK用于编码及RTMP推流。
 *
 * @param pixelBuffer 视频帧数据块起

 * @param format 数据块格式

 * @param ts 视频帧的时间戳

 * @return 无。

 */
-   (void)send_frame_pixelBufferRef:(CVPixelBufferRef)pixelBuffer
format:(CNCENM_Buf_Format)format time_stamp:(long long)ts;
```

10. 获取统计信息

```
/*! @brief 获取统计信息。

 * @return 成功返回realTimeLayeredFrame，失败返回nil。

 */
- (realTimeLayeredFrame *)get_statistics_message;
```

3 音频相关设置

1. 静音功能

```
/*! @brief 设置 开启或关闭 静音功能

* 默认为关闭静音

* @return 无。

*/
- (void)set_muted_status:(BOOL)is_mute;
```

2. 静音状态

```
/*! @brief 获取  静音状态

* @return 静音状态。

 */
- (BOOL)get_is_muted;
```

3. 后台推流

```
/*! @brief 设置应用切换到后台时，推拉流是否继续

* 继续推流传YES,终止推流传NO

* 不设置时，默认为NO

* @return 成功返回yes，失败返回NO。

* 注：后台继续推流以及后台继续拉流的状态必须统一

*/
- (BOOL)set_whether_continue_push_inBk:(BOOL)enable;
```

4. 后台推流状态获取

```
///切换到后台时，是否继续推流 默认为NO
- (BOOL)isContinuePushInBk;
```

5. 混音功能（背景音功能）

```
///开始播放音乐或恢复播放音乐
- (void)startPlayMusic;

///结束播放音乐
- (void)stopPlayMusic;

///暂停播放
- (void)pausePlayMusic;
/*!@brief  调节播放
 *
 * @param  position  0->1，传入进度条的位置值
 *
 */
- (void)seekPlayheadTo:(CGFloat)position;
```

6. 混响功能

```
/*! @brief  设置混响房间大小类型
 * @param type  CNCAudioRoomType
 */
- (void)setAudioRoomType:(CNCAudioRoomType)type;
```

```
/// 设置混响环境类型枚举值
typedef CF_ENUM(UInt32, CNCAudioRoomType) {
    CNCAudioRoomType_SmallRoom      = 0,
    CNCAudioRoomType_MediumRoom     = 1,
    CNCAudioRoomType_LargeRoom      = 2,
    CNCAudioRoomType_MediumHall     = 3,
    CNCAudioRoomType_LargeHall      = 4,
```

```
    CNCAudioRoomType_Plate          = 5,
    CNCAudioRoomType_MediumChamber  = 6,
    CNCAudioRoomType_LargeChamber   = 7,
    CNCAudioRoomType_Cathedral      = 8,
    CNCAudioRoomType_LargeRoom2     = 9,
    CNCAudioRoomType_MediumHall2= 10,
    CNCAudioRoomType_MediumHall3= 11,
    CNCAudioRoomType_LargeHall2     = 12
};
```

```
/*! @brief  设置混响影响因子

 * @param value  [0->100], 默认60

 */

- (void)setAudioMixReverb:(Float32)value;
```

7. 音量调节

```
///设置人声大小 0->1,默认 1.0f

- (void)setHumanVolume:(Float32)value;


///设置音乐大小 0->1,默认 0.3f

- (void)setMusicVolume:(Float32)value;


///设置总输出音量大小 0->1,默认 1.0f

- (void)setOutPutVolume:(Float32)value;
```

8. 状态值获取

```
///获取状态

///当前人声音量大小

- (Float32)currentHumanVolume;

///当前音乐音量大小

- (Float32)currentMusicVolume;
```

```
///当前输出音量大小
- (Float32)currentOutPutVolume;
///当前混响因子大小
- (Float32)currentAudioMixReverb;
///音乐总时长 格式：0:00
- (NSString *)getDurationString;

///当前播放时间字符串 格式：0:00
- (NSString*)getPlayTimeString;

///当前播放进度【0~1】
- (float)getPlayProgress;
```

9. 耳返开关控制

```
/**
 @abstract  查询当前是否有耳机
 */
- (BOOL)is_headset_plugged_in;

// 是否开启耳返
- (BOOL)isOpenMicVoiceReturnBack;

/*! @brief 是否开启耳返
 * 默认为关闭,耳返功能只适用插入耳机的情况
 * @return。
 */
- (void)set_audio_returnback:(BOOL)enable;
```

4 录制功能

录制文件存储位置默认：APP/Document/CNC_Save
录制文件名为 streampusher+当前时间戳.flv/mp4

1. 录制初始化

```
/*! @brief 录制开始

 *  录制分为长短视频、GIF录制，其中视频录制又细分为flv、MP4两种。其中短视频、GIF录制可以设置录制时长控制，长视频录制可设置大小控制。

 *  当录制时长、大小达到预设值时自动结束当前录制并保存结果。

 *  另有提示码提示录制结果 录制开始前请监听返回码CNC_Notification_MobStreamSDK_ReturnCode

 * @param path 录制文件存储地址

 * @param type 录制文件类型

 * @param max_time 录制时长 短视频录制时长控制值域为 3.0s~60.0s。GIF录制时长值域为100.0ms~5.0s。

 * @param long_store 录制长、短视频类型

 * @param size 录制文件大小 长视频录制大小控制最小值为100kb，不足最小值以最小值为准。

 * @param return_id 确认ID 随同录制结束返回

* @return 录制start结果

 */
+ (BOOL)start_store_video:(NSString *)path
                 fileType:(CNCRecordVideoType)type
                 max_time:(NSInteger)max_time
               long_store:(BOOL)long_store
                     size:(uint64_t)size
                return_id:(NSString *)return_id;
```

2. 停止录制

```
/*! @brief 录制停止

 * @return 录制文件存储路径

 */
+ (NSString *)stop_store_video;
```

3. 录制状态

```
/*! @brief 录制状态获取

 * @return 是否处于录制状态

 */
+ (BOOL)is_doing_store;
```

四、架构细致分层

1.视频采集及美颜类------ CNCCaptureVideoDataManager

1. CNCCaptureVideoDataManagerDelegate

```
 @protocol CNCCaptureVideoDataManagerDelegate <NSObject>
///采集sample_buf输出
-
(void)capture_sample_bufferRef_data:(CMSampleBufferRef)sam
ple_buf time_stamp:(unsigned int)time_stamp;
///采集pixelBuffer输出
-
(void)capture_pixel_bufferRef_data:(CVPixelBufferRef)pixel
Buffer time_stamp:(unsigned int)time_stamp;
///采集buf输出
- (void)video_capture_buf:(void *)buf
pix_width:(int)pix_width pix_height:(int)pix_height
format:(CNCENM_Buf_Format)format time_stamp:(long long)ts;
///添加水印buf输出
- (void)overlayMask_buf:(void *)buf pix_width:(int)pix_width
pix_height:(int)pix_height format:(CNCENM_Buf_Format)format
time_stamp:(long long)time_stamp;
```

```
@end
```

2. 初始化视图宽和高

```
/*! @brief 设置视频显示以及展示的view
 *
 * @param width 展示的view的宽
 * @param height 展示的view的高
 * @return 开启是否成功
 */
- (BOOL)init_capture_width:(int)width height:(int)height;
```

3. 预设摄像头参数

```
/*! @brief 预设摄像头参数
 *
 * @param para CNCCaptureInfo类型的摄像头参数
 * @return 设置摄像头参数是否成功
 */
- (BOOL)preset_para:(CNCCaptureInfo *)para;
```

4. 开始启动摄像头采集

```
/*! @brief 启动摄像头采集
 * @return 是否启动成功
 */
- (BOOL)start_capture;
```

5. 停止摄像头采集

```
/*! @brief 停止摄像头采集
 * @return 无
```

```
 */
- (void)stop_capture;
```

6. 暂停摄像头采集

```
/*! @brief 暂停摄像头采集

 * @return 无

 */
- (void)pause_capture;
```

7. 恢复摄像头采集

```
/*! @brief 恢复摄像头采集

 * @return 无

 */
- (void)resume_capture;
```

8. 闪光灯开启或关闭

```
/*! @brief 闪光灯开启或关闭

 *

 * @param enable YES:开启，NO：关闭

 * @return 无

 */
- (void)set_torch_mode:(BOOL)enable;
```

9. 重设 frame

```
 /*! @brief  重设采集frame大小
 *
 * @param new_frame CGRect
 *
 */
- (void)resetFilterViewFrame:(CGRect)new_frame;
```

10. 摄像头操作

```
/*! @brief 推流过程中切换摄像头
*
* @param 无
* @return
*/
- (void)swap_cameras;
```

```
/*! @brief 推流过程中切换到指定位置的摄像头
*
* @param 目标摄像头位置
* @return
*/
-
(void)reset_came_pos:(AVCaptureDevicePosition)new_position
;
```

```
/*! @brief 获取采集摄像头位置
*
* @return目标摄像头位置
*/
- (AVCaptureDevicePosition)get_came_pos;
```

11. 聚焦变焦

```
/*! @brief 手动聚焦
*
* @param 传入聚焦的坐标
* @return 是否设置成功
*/
- (BOOL)tapFocusAtPoint:(CGPoint)point;
```

```
/*! @brief 变焦-拉远拉近、放大缩小
*
* @param 改变的倍数 值域 1.0~videoZoomFactorUpscaleThreshold
* @return
*/
- (void)videoZoomFactorWithScale:(CGFloat)effectiveScale;
```

```
/*! @brief 获取当前相机的最大放大倍数
*
* @param
* @return videoZoomFactorUpscaleThreshold 上限阈值
*/
```

```
- (CGFloat)get_current_camera_upscale;
```

12. 获取采集内部 context_queue

```
/*! @brief  返回context_queue
 *
 * @return
 *
 */
- (dispatch_queue_t)get_context_queue;
```

13. 获取当前 context

```
 /*! @brief  返回当前context
 *
 * @return
 *
 */
- (EAGLContext*)get_readonly_context;
```

14. 美颜及其他设置 详见第二章第六节美颜（见上方）

15. 水印设置

分层水印支持回传预处理结果进行水印

a. 回传视频帧

b. 开启水印

```
/*! @brief  预处理后视频帧添加水印
 * @param pixelbuffer 视频帧
 * @param buffer_time 采集样本时间戳
CMSampleBufferGetPresentationTimeStamp
 * @param time_stamp  编码统一时间戳
 * @return
 *
 */
-   (void)overlay_pixelbuffer:(CVImageBufferRef)pixelbuffer
time:(CMTime)buffer_time                time_stamp:(unsigned
int)time_stamp;
```

```
 /*! @brief 设置水印 仅支持图片UIImageView及文字UILabel
```

```
 *
 * @param object 添加的水印 设置为nil为清除水印
 * @param scale 水印位置比例 (x,y,width,height) 范围为0~1
 * @param updateBlock 更新block 支持在block内对水印内容及图片进行修改
 * @return 添加水印结果
 */
-       (BOOL)overlayMaskWithObject:(UIView       *)object rect:(CGRect)scale block:(void (^)(void))updateBlock;
```

16. 镜像功能

```
 /*! @brief  采集流镜像
 * @param source_mirror 视频采集源镜像
 *
 * @return
 *
 */
- (void)set_source_mirror:(BOOL)source_mirror;
```

2.视频编码类---------- CNCMobStreamVideoEncoder

1. 唯一初始化方法

```
/*! @brief 初始化方法
 *
 * @param bounds 视频分辨率宽高

 * @return CNCMobStreamVideoEncoder句柄
 */
- (instancetype)initWithVideoSize:(CGSize)bounds;
```

2. 开始编码，软硬编选择

```
/*! @brief 开始编码
 *
 * @param type CNCENMEncoderType 软硬编选择
```

```
 * @return 无

 */
- (void)startEncodedWithType:(CNCENMEncoderType)type;
```

3. 停止编码

```
/*! @brief 停止编码

 * @return 无

 */
- (void)stopEncoded;
```

4. 传入编码前数据

```
/*! @brief 传入帧数据，开始编码

 *
 * @param buffer image buffer data
 * @param pix_width bytesPerRow
 * @param pix_height planeTotalHeight
 * @param format 图像数据的格式

 * @param ts 时间戳

 * @return 无

 */
-           (void)inputFrameBuffer:(void           *)buffer
pix_width:(int)pix_width         pix_height:(int)pix_height
format:(CNCENM_Buf_Format)format       time_stamp:(unsigned
int)ts;
```

```
/*! @brief 将视频帧数据pixelBuffer发给SDK用于编码及RTMP推流。

 * @param pixelBuffer 视频帧数据块起

 * @param format 数据块格式

 * @param ts 视频帧的时间戳

 * @return 无。

 */
```

```
-   (void)send_frame_pixelBufferRef:(CVPixelBufferRef)pixelBuffer
format:(CNCENM_Buf_Format)format time_stamp:(long long)ts;
```

5. 两个参数

```
///实时码率，如果开启码率自适应的时候，要相应更新该参数
@property (nonatomic, assign) NSUInteger real_bit_rate;

///实时帧率
@property (nonatomic, assign) NSUInteger real_fps;
```

6. CNCMobStreamVideoEncoderDelegate

```
@required
///视频帧时间戳来源
- (unsigned int)timeStampForCurrentVideoFrame;

@optional

///SPS、PPS数据回调出口，用于RTMP发送videoHeader;
- (void)outPutH264SPS:(NSData *)spsData PPS:(NSData
*)ppsData;

///请求发送meta data头信息
- (void)requireSendMetaDataWithVideoSize:(CGSize)size
bitRate:(int)rate videoFPS:(int)fps;

///H264编码数据回调
-       (void)didEncodedCallBack:(char       *)compressData
dataLength:(int)length                frameSize:(CGSize)size
isKey:(BOOL)isKey timestamp:(unsigned int)timestamp;
```

3.视频显示类---------- CNCMobStreamVideoDisplayer

1. 初始化视图

```
 /*! @brief 初始化视图
 *
```

```
 * @param preview 父视图
 * @param fboDisplayConfigs CNCDisplayConfigs配置参数
 * @return 句柄
 */
-        (instancetype)initWithView:(UIView        *)preview
displayConfigs:(CNCDisplayConfigs)fboDisplayConfigs;
```

```
typedef struct CNCDisplayConfigs {
    /// 填充模式
    CNCDisplayFillModeType fill_mode;
    /// 数据是否为YUV格式
    BOOL bUseCaptureYUV;
    //  横竖屏
    CNCENMDirectType direction;
    CGFloat capture_width;
    CGFloat capture_height;
    CNCDisplayType displayType;
} CNCDisplayConfigs;
```

2. 设置镜像

```
 /*! @brief 设置预览镜像
 *
 * @param is_mirror 镜像
 * @return 无
 */
- (void)set_display_mirror:(BOOL)is_mirror;
```

3. 显示图像数据传入接口

```
 /*! @brief 预览视频帧样本
 *
 * @param sampleBuffer 视频帧样本
 * @return 无
 */
-
(void)processVideoSampleBuffer:(CMSampleBufferRef)sampleBu
ffer;
```

```
/*! @brief 预览视频帧样本
 *
 * @param pixelBuffer 视频帧样本
 * @return 无
 */
-
(void)processVideoImageBuffer:(CVPixelBufferRef)pixelBuffe
r;
```

```
/*! @brief 数据buf预览
 * @param buf 视频帧数据
 * @param bytesPerRow  视频帧样本行字节数
 * @param pHeight 视频帧样本高度
 * @return 无
 */
-          (void)processVideoBufferData:(void          *)buf
bytesPerRow:(int)bytesPerRow planeHeight:(int)pHeight;
```

4. 纹理预览

```
/*! @brief 纹理预览
 * @param texture buffer纹理
 * @param width  宽度
 * @param height 高度
 * @return 无
 */
- (void)renderTexture:(GLuint)texture width:(int)width
height:(int)height;
```

5. 预览设置水印

```
/*! @brief 设置预览层水印
 * @param object 水印对象
 * @param scale 水印位置比例 (x,y,width,height) 范围为0~1
 * @return 设置结果
 */
- (BOOL)overlay_mask:(UIView *)object rect:(CGRect)scale;
```

4.音频采集编码类--------- CNCMobStreamAudioEngine

1. 音频初始化设置见第二章第 7 混音混响功能

2. 接口说明

```
///是否开启回音消除功能，默认开启；关闭后，可能会有环境噪声
- (BOOL)setAecEnable:(BOOL)bEnable;

///是否开启环境音自动增益减益功能，默认开启，开启时，人声大小会触发自动
调节环境音量大小
- (BOOL)setAgcEnable:(BOOL)bEnable;
```

3. CNCMobStreamAudioEngineDelegate

```
@required

///音频帧时间戳获取
- (unsigned int)timestampForCurrentAudioFrame;

@optional

///获取 静音状态、后台是否推流、是否正在推流状态 参数返回值
- (BOOL)audioEngine:(CNCMobStreamAudioEngine *)audioEng
valueForOption:(AudioEngineOption)option
withDefault:(BOOL)defaultValue;

///编码后的AAC数据回调接口
- (void)audioEngine:(CNCMobStreamAudioEngine *)audioEng
sendAudioBufferData:(char *)src_sz len:(unsigned int)len
time_stamp:(unsigned int)time_stamp;

///请求发送audio header
-   (void)audioEngine:(CNCMobStreamAudioEngine   *)audioEng
sendAudioHeaderRate:(int)rate          channel:(int)channel
time_stamp:(unsigned int)time_stamp;
```

5.时间戳生成器---------- CNCMobStreamTimeStampGenerator

1. 方法简介

```
///重置时间戳生成器
- (void)resetGeneratorZero;

///生成一个视频时间戳
- (unsigned int)generateVideoTimeStamp;
```

```
///生产一个音频时间戳
- (unsigned int)generateAudioTimeStamp;

///获取视频时间戳队列最小时间戳
- (unsigned int)popTheMinimumInVideoTimeStampArray;

///移除时间时间戳队列比当前发送时间戳小的元素
-      (void)removeSmallThanInVideoTimeStampArray:(unsigned
int)currentTimeStamp;
```

6.RTMP 发送器---------- CNCMobStreamRtmpSender

1. 初始化

```
/*! @brief 初始化方法
 *
 * @param urlString 推流地址

 * @return 句柄，失败返回nil
 */
-             (instancetype)initWithRtmpUrlString:(NSString
*)urlString;
```

```
///推流地址，开始推流前需设置
@property (nonatomic, copy) NSString *rtmp_url;

///码率自适应回调
@property (nonatomic, assign)
id<CNCMobStreamRtmpSenderDelegate> delegate;
```

2. 开始推流

```
/*! @brief 开始连接推流
 *
```

```
 * @param timeGenerator 时间戳管理器，用于控制帧的发送队列

 * @return 推流地址为空，推流失败返回NO，成功返回yes；

 */
-        (BOOL)startConnect:(CNCMobStreamTimeStampGenerator
*)timeGenerator;
```

3. 停止推流

```
/*! @brief 停止推流

 * @return 失败返回NO，成功返回yes

 */
- (BOOL)stopConnect;
```

4. 开启码率自适应

```
/*! @brief 开启码率自适应

 *

 * @param minRate 最小码率

 * @param maxRate 最大码率

 * @param currentRate 初始化码率

 * @return 参数不对，设置失败返回NO，成功返回yes；

 */
-
(BOOL)openVideoBitRateAutoFitWithMiniRate:(NSUInteger)minR
ate                                MaxiRate:(NSUInteger)maxRate
CurrentRate:(NSUInteger)currentRate;
```

5. 关闭码率自适应

```
/*! @brief 关闭码率自适应

 * @return 无
```

```
 */
- (void)closeVideoBitRateAutoFit;
```

6. 发送 meta data 头

```
/*! @brief 发送meta data 头
 *
* @param videoSize 视频分辨率 videoSize.witdh宽
videoSize.height高
* @param frame_rate 视频帧率
* @param video_bit_rate 视频码率
* @param audio_rate 音频采样率
* @param audio_channel 音频channel
* @return 无;
    */
-                    (void)send_meta_data:(CGSize)videoSize
frame_rate:(int)fps      video_bit_rate:(int)video_bit_rate
audio_rate:(int)rate audio_channel:(int)channel;
```

7. 发送视频头信息

```
/*! @brief 发送视频头信息
 *
 * @param spsData
 * @param ppsData
 * @return 无;
 */
-   (void)send_video_header:(NSData   *)spsData   PPS:(NSData
*)ppsData;
```

8. 发送音频头信息

```
/*! @brief 发送音频头信息

* @param rate 音频采样率

 * @param channel 音频声道数

 * @return 无;

 */
- (void)send_acc_header:(int)rate channel:(int)channel;
```

9. 发送视频帧数据

```
/*! @brief 发送视频帧数据

* @param (char *)sz H264数据

 * @param len H264数据长度

 * @param is_key 是否关键帧

 * @param width 视频图像的宽

 * @param height 视频图像的高

 * @param time_stamp 视频帧的时间戳

 * @return 无;

 */
- (void)send_video:(char *)sz len:(unsigned int)len
is_key:(bool)is_key width:(int)width height:(int)height
time_stamp:(unsigned int)time_stamp;
```

10. 发送音频帧数据

```
/*! @brief 发送音频帧数据

* @param (char *)src_sz AAC数据

 * @param len AAC数据长度
```

```
 * @param time_stamp 音频帧的时间戳

 * @return 无;

 */
-  (void)send_audio:(char  *)src_sz  len:(unsigned  int)len
time_stamp:(unsigned int)time_stamp;
```

11. CNCMobStreamRtmpSenderDelagate

```
///码率自适应回调，需要相应去设置视频编码器

 (CNCMobStreamVideoEncoder) 的 real_bit_rate 属性

-
(void)videoBitRateAdaptiveCorrection:(NSUInteger)bit_rate;
```

7.文件录制---------- CNCRecordFileSeesionManager

1. 录制开始

```
 /*! @brief 录制开始

 *  录制分为长短视频、GIF录制，其中视频录制又细分为flv、MP4两种。其中短视频、

GIF录制可以设置录制时长控制，长视频录制可设置大小控制。

 *  当录制时长、大小达到预设值时自动结束当前录制并保存结果。

 *  另有提示码提示录制结果 录制开始前请监听返回码

CNC_Notification_MobStreamSDK_ReturnCode

 * @param path 录制文件存储地址

 * @param record_info 录制参数设置 具体格式详见 CNCRecordFileInfo

 * @param type 录制文件类型

 * @param max_time 录制时长 短视频录制时长控制值域为 3.0s~60.0s。GIF录制

时长值域为100.0ms~5.0s。
```

```
 * @param long_store 录制长、短视频类型

 * @param size 录制文件大小 长视频录制大小控制最小值为100kb，不足最小值以最小值为准。

 * @param return_id 确认ID 随同录制结束返回

 *

 * @return 录制start结果

 */
-(BOOL)start_store_video:(NSString *)path info:(CNCRecordFileInfo *)record_info fileType:(CNCRecordVideoType)type max_time:(NSInteger)max_time long_store:(BOOL)long_store size:(uint64_t)size return_id:(NSString *)return_id
```

2. 录制停止

```
/*! @brief 录制停止

 * @return 录制文件存储路径

 */
- (NSString *)stop_store_video;
```

3. 录制状态

```
 /*! @brief 录制状态获取

 * @return 是否处于录制状态

 */
- (BOOL)is_doing_store;
```

4. GIF 录制

```
/*! @brief gif录制

 * @param buf 未编码前数据
```

```
 * @param pix_width 数据宽度

 * @param pix_height 数据高度

 * @param format 采集格式

 * @param time_stamp 时间戳

 * @return 此帧是否成功加入录制队列

 */
-  (BOOL)store_gif_record:(void  *)buf  pix_width:(int)pix_width
pix_height:(int)pix_height      format:(CNCENM_Buf_Format)format
time_stamp:(long long)ts;
```

5. 音频数据

```
/*! @brief 音频数据传入

 * @param sz 编码后音频数据

 * @param len 数据长度

 * @param time_stamp 时间戳

 * @return 此帧是否成功加入录制队列

 */
-    (BOOL)do_store_audio:(char*)sz    len:(unsigned    int)len
time_stamp:(unsigned int)time_stamp;
```

6. 视频数据

```
/*! @brief 视频数据传入

 * @param sz 编码后音频数据

 * @param len 数据长度

 * @param is_key 是否为关键帧

 * @param time_stamp 时间戳
```

```
 * @return 此帧是否成功加入录制队列

 */
-    (BOOL)do_store_video:(char*)sz    len:(unsigned    int)len
is_key:(bool)is_key     width:(int)width     height:(int)height
time_stamp:(unsigned int)time_stamp;
```

五、第三方采集预处理嵌入

为了满足用户使用更加个性化的美颜/滤镜/动态贴纸的功能，我们支持用户集成第三方的厂商来满足自身业务的发展。现在我们已经实现了几种方案的集成，以达到快速集成和使用的目的。

注：商汤的鉴权文件 SENSEME.lic、FaceUnity 的鉴权字段 g_auth_package 请自行替换后方可正常使用。使用前请确认使用第三方库的.h .a 等。说明文档中以当前使用版本为准。

1. 商汤采集预处理嵌入

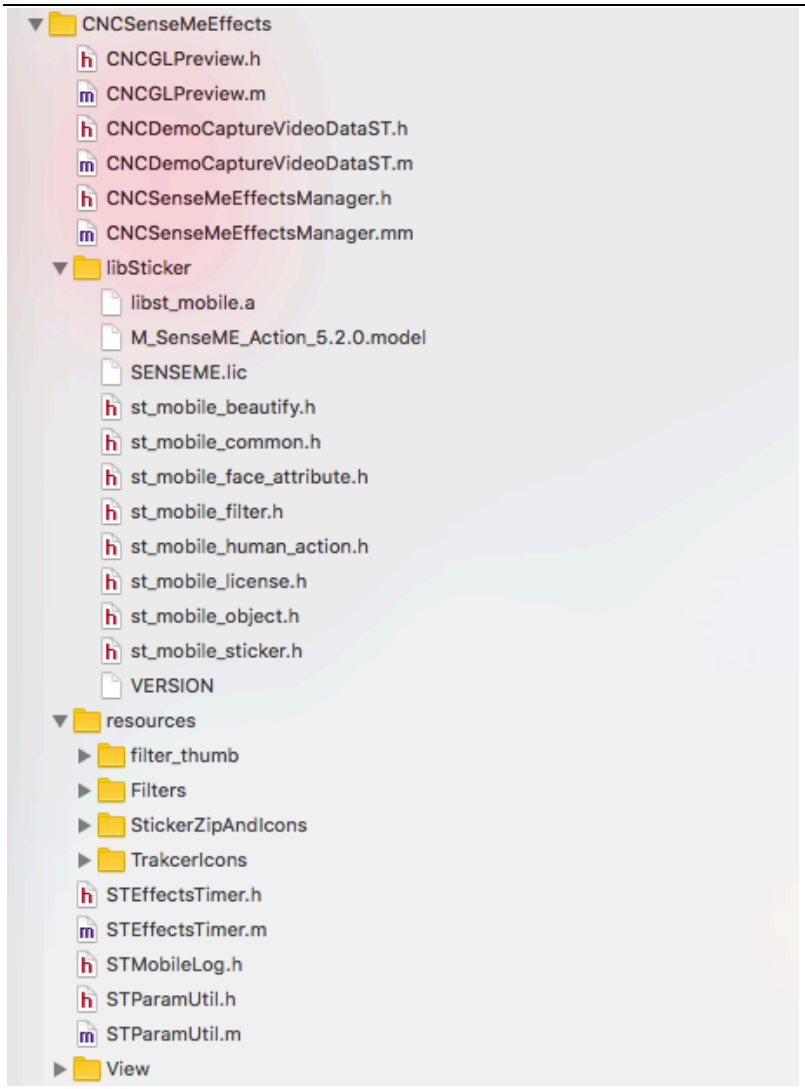
目前嵌入版本为商汤 5.2

1. 功能

商汤功能包括：

1.人脸识别及追踪	2.人脸属性检测
3.手势检测及追踪	4.前后背景分割
5.美颜	6.面部 2D、3D 特效
7.美形	8.通用物体跟踪

2. 嵌入目录结构



i. libSticker 为商汤 SDK 及算法文件（M_SenseME_Action_5.2.0.model）、鉴权文件（SENSEME.lic）等。

ii. resources 包含滤镜文件（DeliciousFoodFilters.bundle 等）、素材、贴纸 zip 包。

iii. CNCGLPreview 为商汤配合接口所使用的预览渲染类。

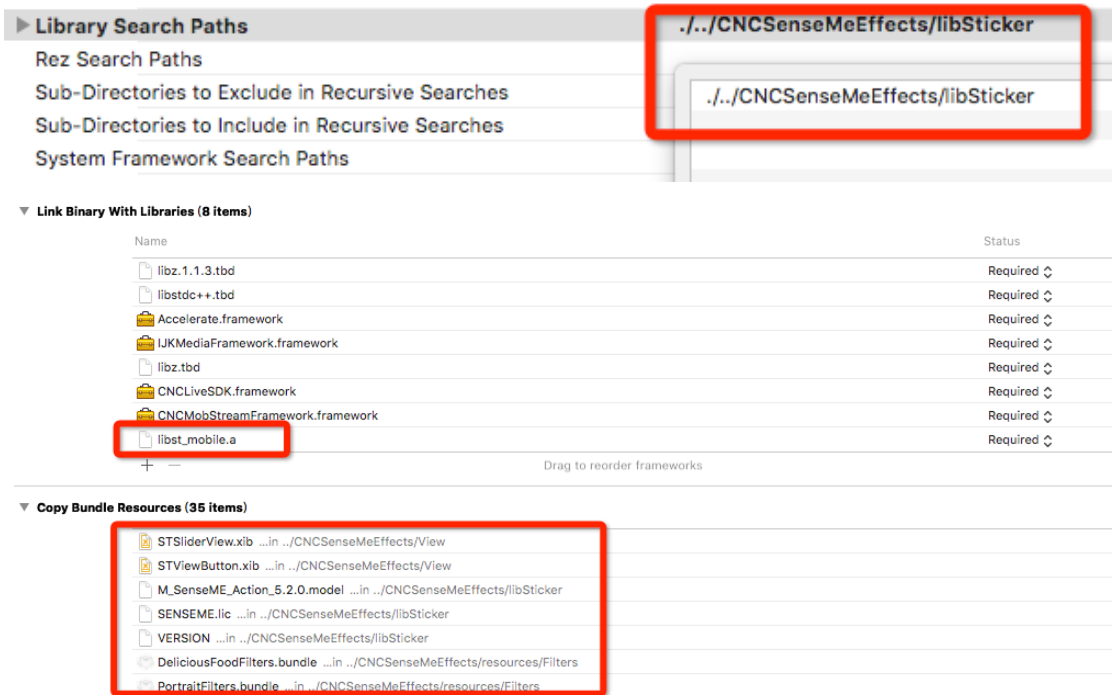
iv. CNCDemoCaptureVideoDataST 为 Demo Viewcontroller。

v. CNCSenseMeEffectsManager 为使用商汤 SDK 的管理类 内嵌有初始化 SDK,功能使用、商汤 UI 界面等各方法。

vi. View 为商汤自带控件及方法

3. 嵌入方式

将 CNCSenseMeEffects 文件下所有文件添加进工程内，并添加 lib 库依赖，其中鉴权文件以商汤官方出具的为准（需自行更换），详见工程设置。



i. 初始化管理类

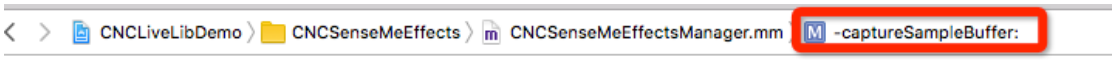
在 setup 之前要先设置好正确的鉴权、贴纸、滤镜等文件路径，否则会导致失败

```
if (self.st_manager == nil) {
    [self check_capture_width_and_height];
    self.st_manager = [[[CNCSenseMeEffectsManager alloc] initWith:self.preview direct:self.video_direct_type
        width:capture_width height:capture_height] autorelease];
    self.st_manager.delegate = self;
    self.st_manager.isFrontCamera = YES;
    self.st_manager.isVideoMirrored = YES;
}
```

ii. 预处理每帧数据，通过代理传入编码进行推流

```
#pragma mark- CNCSenseMeEffectsManagerDelegate
- (void)do_sense_data:(void *)src_buf pix_width:(int)pix_width pix_height:(int)pix_height format:(CNCRTC_Buf_Format)format
    time_stamp:(long long)ts {
    if (self.delegate && [self.delegate
        respondsToSelector:@selector(video_capture_st:buf:pix_width:pix_height:format:time_stamp:)]) {
        [self.delegate video_capture_st:self buf:src_buf pix_width:pix_width pix_height:pix_height format:format time_stamp:ts];
    }
}
```

iii. texture，实现预览




```
[self.glPreview renderTexture:textureResult];
```

4. 接口

详见各类头文件，Manager 类为适配当前 SDK 的使用情况，若 SDK 有接口及功能变动以实际为准。第三方内部头文件及接口说明不在本文档内体现，可见于该 SDK 官方说明文档。

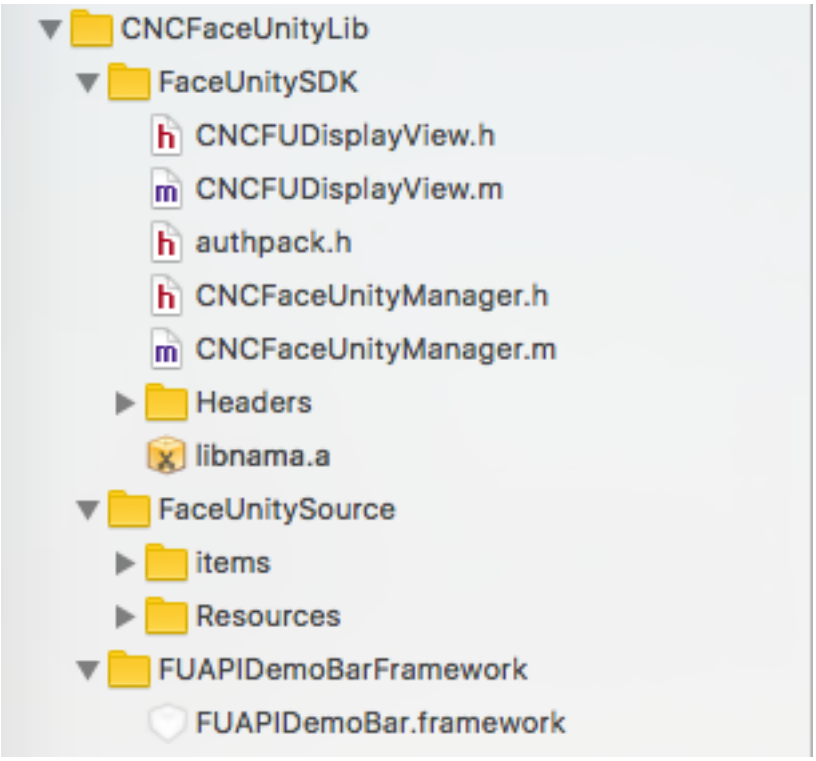
2. FaceUnity 预处理嵌入

目前嵌入版本为 FaceUnity 4.0

1. 功能

功能包括：人脸识别、贴纸、磨皮美颜、美型（大眼小眼）、滤镜

2. 嵌入目录结构



i. CNCFUDisplayView 为 FaceUnity 配合接口所使用的预览渲染文件,也可使用 CNCMobStreamVideoDisplayer 镜像预览渲染。

ii. FaceUnitySource 为配套贴纸及配套初始化二进制文件。

iii. FaceUnitySDK 中包含库及相应头文件，其中 authpack.h 中字符串为鉴权 key。

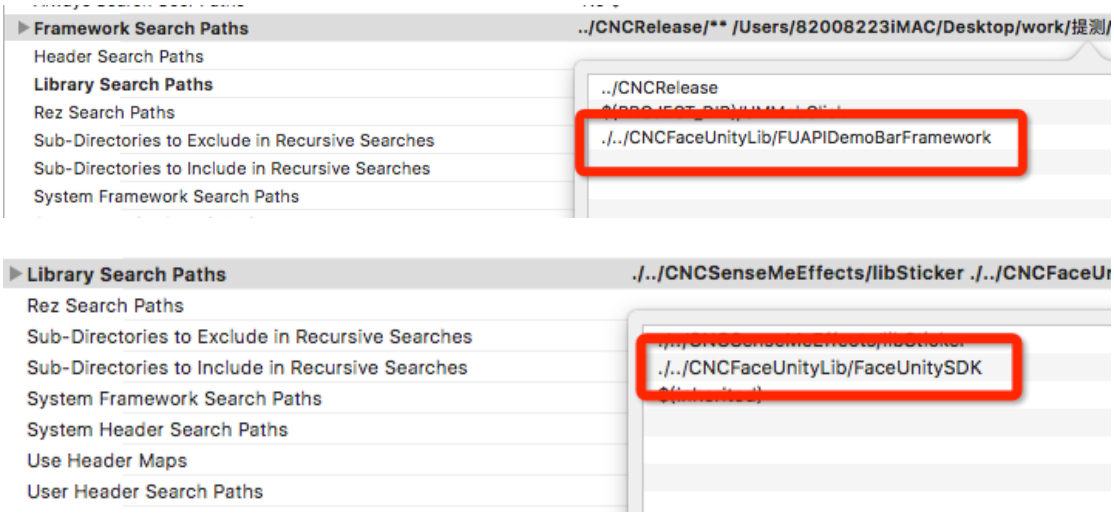
iv. CNCFaceUnityManager 为 FU SDK 的管理类 内嵌有初始化 SDK,功能使用等各方法。

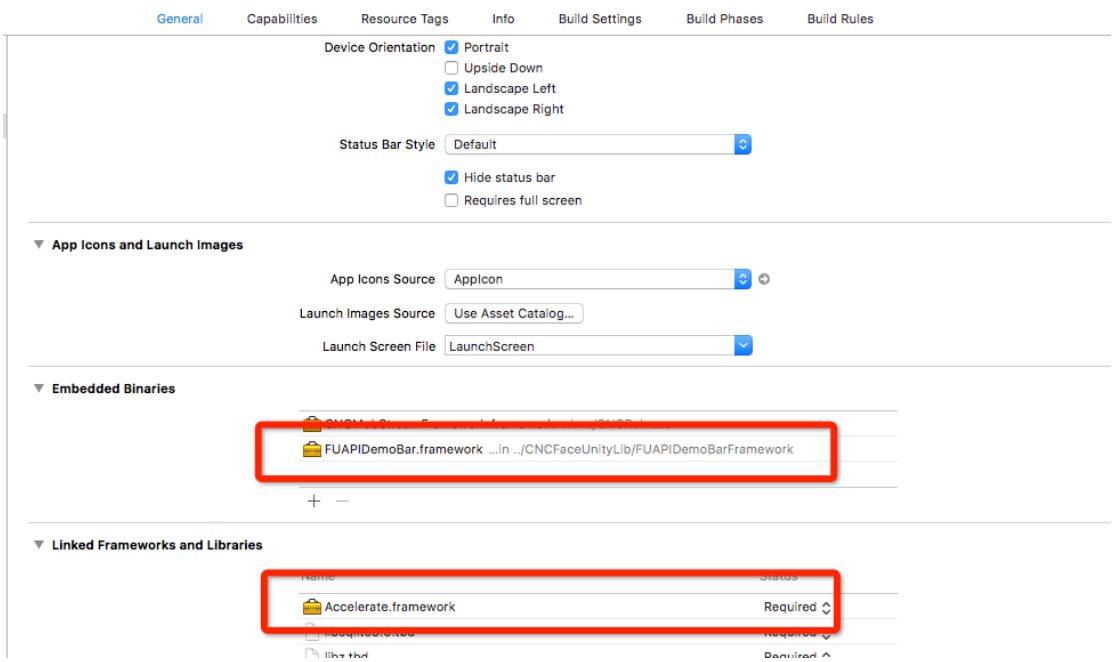
v. FUAPIDemoBarFramework 为 FU 控件

3. 嵌入方式

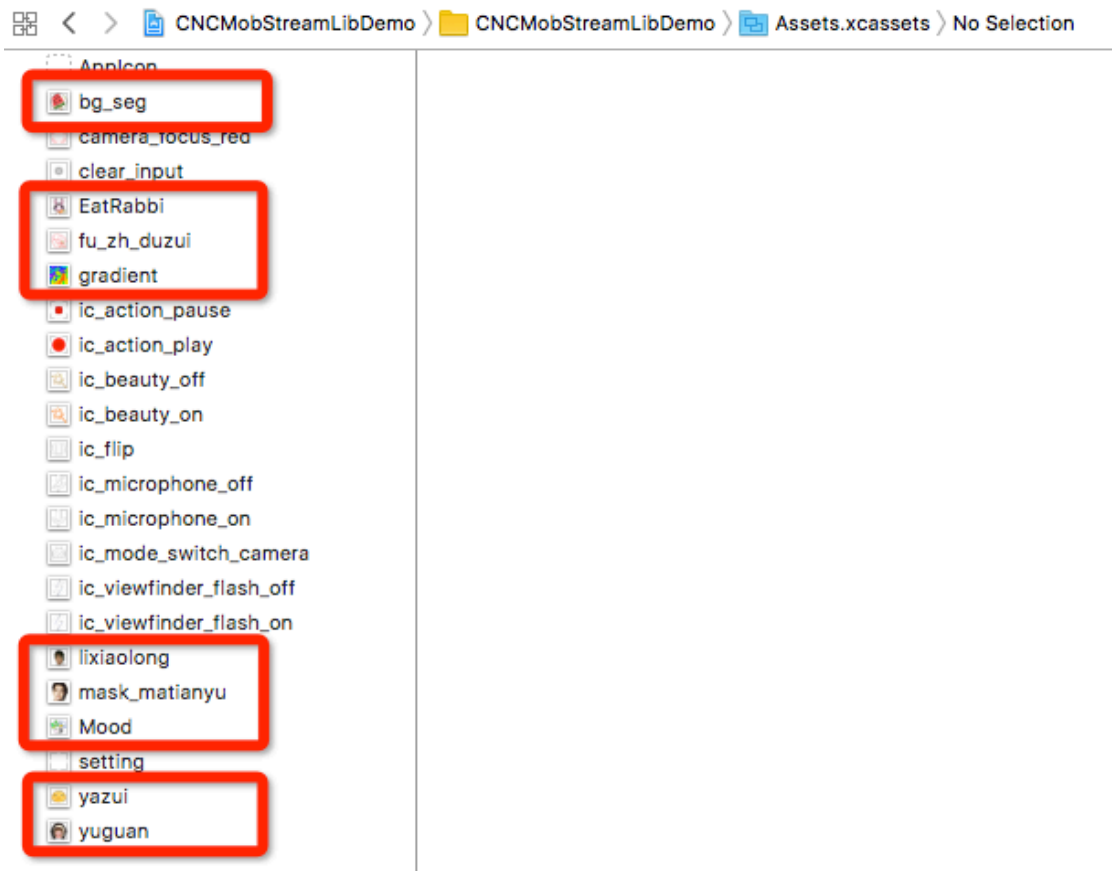
将 CNCFaceUnityLib 文件下所有文件添加进工程内，并添加 lib 库依赖，其中鉴权文件、初始化二进制文件以 FaceUnity 官方出具的为准（需自行更换）。

i. 添加依赖库及相应 framework 资源文件。





ii. 添加 FUAPIDemoBarFramework 相应的图片资源，拖拽进工程即可。

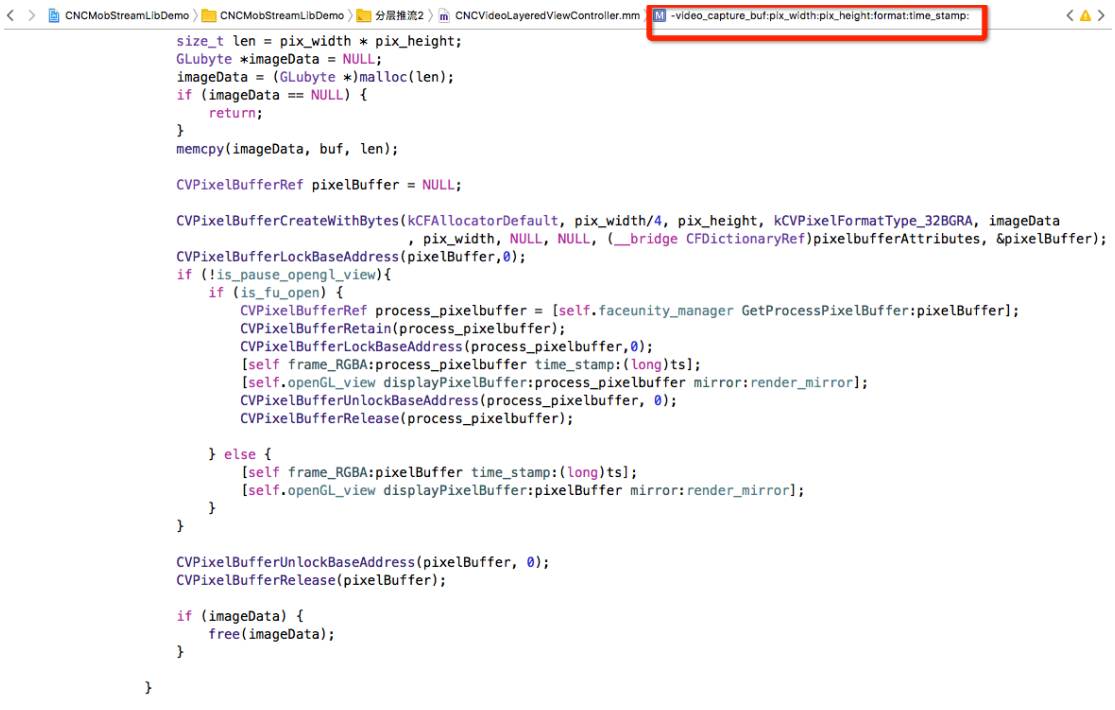


iii. 初始化管理类及渲染类。

```
#pragma mark FU 美颜
- (void)init_fu_bar {
    self.demo_bar = [[[FUAPIDemoBar alloc] initWithFrame:CGRectMake(0, screen_h_, screen_w_, 208)] autorelease];

    {
        UIButton *btn = [[[UIButton alloc] initWithFrame:CGRectMake(self.demo_bar.bounds.size.width-30, 60, 40, 40)] autorelease];
        btn.backgroundColor = [UIColor clearColor];
        [btn setImage:[UIImage imageNamed:@"clear_input"] forState:UIControlStateNormal];
        [btn addTarget:self action:@selector(actionCloseFuBar:) forControlEvents:UIControlEventTouchUpInside];
        [self.demo_bar addSubview:btn];
    }
    [self.view addSubview:self.demo_bar];

}
- (void)init_faceunity_manager {
    if ((self.has_video)){
        self.faceunity_manager = [[[CNCFaceUnityManager alloc] init] autorelease];
        [self.faceunity_manager setUpFaceunity];
        dispatch_async(dispatch_get_main_queue(), ^{
            [self init_fu_bar];
            [self setDemoBar];
        });
    }

}
```

iv. 预处理每帧数据并实现渲染

FaceUnity 内部有两种方式反馈预处理结果，一为直接修改传入的 pixelbuffer，另一种为返回新的 pixelbuffer。嵌入时可自行选用。




```
                size_t len = pix_width * pix_height;
                GLubyte *imageData = NULL;
                imageData = (GLubyte *)malloc(len);
                if (imageData == NULL) {
                    return;
                }
                memcpy(imageData, buf, len);

                CVPixelBufferRef pixelBuffer = NULL;

                CVPixelBufferCreateWithBytes(kCFAllocatorDefault, pix_width/4, pix_height, kCVPixelFormatType_32BGRA, imageData
                                             , pix_width, NULL, NULL, (__bridge CFDictionaryRef)pixelbufferAttributes, &pixelBuffer);
                CVPixelBufferLockBaseAddress(pixelBuffer,0);
                if (!is_pause_opengl_view){
                    if (is_fu_open) {
                        CVPixelBufferRef process_pixelbuffer = [self.faceunity_manager GetProcessPixelBuffer:pixelBuffer];
                        CVPixelBufferRetain(process_pixelbuffer);
                        CVPixelBufferLockBaseAddress(process_pixelbuffer,0);
                        [self frame_RGBA:process_pixelbuffer time_stamp:(long)ts];
                        [self.openGL_view displayPixelBuffer:process_pixelbuffer mirror:render_mirror];
                        CVPixelBufferUnlockBaseAddress(process_pixelbuffer, 0);
                        CVPixelBufferRelease(process_pixelbuffer);

                    } else {
                        [self frame_RGBA:pixelBuffer time_stamp:(long)ts];
                        [self.openGL_view displayPixelBuffer:pixelBuffer mirror:render_mirror];
                    }
                }

                CVPixelBufferUnlockBaseAddress(pixelBuffer, 0);
                CVPixelBufferRelease(pixelBuffer);

                if (imageData) {
                    free(imageData);
                }

            }
```

v. 预处理结果编码

```
- (void)frame_RGBA:(CVImageBufferRef)imageBuffer time_stamp:(long)time_stamp {

    uint8_t *p_src_buf = (uint8_t*)CVPixelBufferGetBaseAddress(imageBuffer);

    size_t byte_per_row = CVPixelBufferGetBytesPerRow(imageBuffer);
    //    NSLog(@"byte_per_row :  %zu",byte_per_row);
    size_t height = CVPixelBufferGetHeight(imageBuffer);
    size_t width  = CVPixelBufferGetWidth(imageBuffer);

    int real_width = int(width * 4);

    UInt8 *dst_buf = NULL;
    int len = int(real_width * height);
    dst_buf = new UInt8[len];
    do {

        if (dst_buf == NULL) {
            break;
        }
        memset(dst_buf, 0x0, len);
        UInt8 *p_tmp_dst = dst_buf;
        UInt8 *p_tmp_src = p_src_buf;

        if (byte_per_row != real_width) {
            for (unsigned int y = 0; y < height; ++y) {
                memcpy(p_tmp_dst, p_tmp_src, real_width);
                p_tmp_dst += real_width;
                p_tmp_src += byte_per_row;
            }
        } else {
            memcpy(dst_buf, p_src_buf, len);
        }

        if (self.is_pushing) {
            if (self.videoEncoder) {
                if ([self.record_manager is_doing_store] && store_type == CNCRecordVideoType_GIF){
                    [self.record_manager store_gif_record:dst_buf pix_width:int(real_width) pix_height:int(height)
                        format:CNCENM_buf_format_BGRA time_stamp:time_stamp];
                }
                [self.videoEncoder inputFrameBuffer:dst_buf pix_width:int(real_width) pix_height:int(height)
                    format:CNCENM_buf_format_BGRA time_stamp:(unsigned int)time_stamp];
            }
        }
    } while (0);

    if (dst_buf != NULL) {
        delete []dst_buf;
    }

}
```

4. 接口

详见各类头文件，Manager 类为适配当前 SDK 的使用情况，若 SDK 有接口及功能变动以实际为准。第三方内部头文件及接口说明不在本文档内体现，可见于该 SDK 官方说明文档。

六、推流器相关状态码说明

附 SDK 中相关的状态码，以备针对出现的不同情况进行 APP 层级处理。

分类	编号	提示	使用场景	分类

	1101	输入参数错误	初始化时未指定鉴权 key 或 value	错误
	1104	SDK 初始化失败	SDK 初始化失败	错误
	1002	录制结束	视频、GIF 录制结束	提示
	1003	录制成功	视频/GIF 录制成功	提示
	1110	录制失败	视频/GIF 录制失败	错误
	1113	截图失败	截图失败	错误
	1114	磁盘空间不足	录制视频前检查磁盘空间 ,磁盘空间不足时抛出此通知，并取消录制	错误
	1116	代理接口连接失败	代理连接失败	错误
	1502	行为提前结束风险	短视频在低于 100M 时开始录制时	警告
鉴权	2001	正在鉴权中...	正在鉴权中	提示
	2101	过期	鉴权时间过期	错误
	2102	SDK 版本过低	鉴权 SDK 版本过低	错误
	2103	SDK 类型不匹配	SDK 类型不匹配	错误
	2104	AppID 类型不匹配	AppID 类型不匹配	错误
	2105	authKey 不匹配	authKey 不匹配	错误
	2106	鉴权网络异常	网络鉴权失败	错误
推流器		推流		
	5301	推流成功	连接服务器成功，并开始推流	提示
	3302	推流初始化失败	未指定推流 url	错误
	3303	推流连接失败	服务器或网络原因等引起推流连接失败，并会不断重连	错误

	3304	数据发送失败	数据包发送失败	错误
	3305	推流已断开	推流从连接状态断开	提示
	4301	网络环境差	网络环境差，推流器不断丢包	警告
	4302	无法支持设定分辨率	开启摄像头与切换摄像头时 ,当前设备不支持分辨率	警告
	3375	混音加载失败	混音文件加载失败	错误
	5306	摄像头开启成功	摄像头开启成功	提示
	5307	摄像头切换成功	摄像头切换成功	提示
	5308	分辨率切换	内部根据屏幕方向自动对调视频宽高时发送此通知	提示
	3342	硬件错误	摄像头或 mic 异常	错误
		视频编码		
	3341	相机权限错误	未获取到摄像头权限	错误
	3343	硬编码器初始化失败	创建硬编码器失败	错误
	3345	视频编码失败	视频编码失败	错误
		音频		
	3371	麦克风权限错误	未获取到麦克风权限	错误