**SD-SRT封装库接口**

1. **系统初始化接口**

/\*\*\*

\* 环境初始化，系统只需调用一次，主要用于SRT环境以及日志模块的初始化

\* @param: outputPath 表示日志存放路径，支持相对路径和绝对路径，若目录不存在将自动创建

\* @param: outputLevel表示日志输出的级别，只有等于或者高于该级别的日志输出到文件，取值范围参考LOG\_OUTPUT\_LEVEL

\* @return:

\*/

void SDSrtAvCom\_Enviroment\_Init(const char\* outputPath, int outputLevel);

void SDSrtAvCom\_Enviroment\_Free();

1. **创建和删除SD-SRT对象**

/\*\*\*

\* 创建SrtAvCom

\* @param unLogId: 日志ID，仅用于日志输出时的对象标识。

\* @return: 返回模块指针，为NULL则失败

\*/

void\* SDSrtAvCom\_Create(UINT unLogId);

/\*\*\*

\* 销毁SrtAvCom，使用者应该做好与其他API之间的互斥保护

\* @param pRtp\_avcom: 模块指针

\* @return:

\*/

void SDSrtAvCom\_Delete(void\* pRtp\_avcom);

1. **启动**

/\*\*\*

\* 开始工作

\* @param strLocalIP: 本地IP地址，允许为NULL，为非NULL时将绑定到该IP（网卡）。

\* @param shLocalPort: 本地通信端口（该端口用于音频，视频端口号将在此基础上加1），对于客户端模式时，允许设置本地端口号为0，此时将由系统自动选择可用的端口。

\* @param strRemoteIP: 对方IP地址，当为服务端模式时设置为NULL

\* @param shRemotePort: 对方收发端口（该端口用于音频，视频端口号将在此基础上加1），当为服务端模式时设置为0

\* @param pfVideoRecvCallBack: 接收到视频数据后的对外输出回调函数

\* @param pfAudioRecvCallBack: 接收到音频数据后的对外输出回调函数

\* @param pObject: 调用上述两个回调函数时的附带透传形参，模块内部不会解析本参数仅做透传处理

\* @return: TRUE FALSE

\*/

BOOL SDSrtAvCom\_Start(

void\* pRtp\_avcom,

const char \*strLocalIP,

USHORT shLocalPort,

const char \*strRemoteIP,

USHORT shRemotePort,

CallBackFuncRecvVideoData pfVideoRecvCallBack,

CallBackFuncRecvAudioData pfAudioRecvCallBack,

void \*pObject);

1. **结束**

/\*\*\*

\* 停止SrtAvCom工作

\* @param pRtp\_avcom: 模块指针

\* @return:

\*/

void SDSrtAvCom\_Stop(void\* pRtp\_avcom);

1. **发送视频**

/\*\*\*

\* 发送视频数据

\* @param pRtp\_avcom: 模块指针

\* @param byBuf: 传入一帧带起始码的裸码流，内部自行拆分拼接。

\* @param nLen: 数据长度

\* @return:

\*/

BOOL SDSrtAvCom\_SendVideoData(void\* pRtp\_avcom, unsigned char \*byBuf, int nLen);

1. **发送音频**

/\*\*\*

\* 发送音频数据

\* @param pRtp\_avcom: 模块指针

\* @param byBuf: 传入一帧音频裸码流，可以是ADTS，内部无拆包透传

\* @param nLen: 数据长度

\* @return:

\*/

BOOL SDSrtAvCom\_SendAudioData(void\* pRtp\_avcom, unsigned char \*byBuf, int nLen);

1. **设置通用传输参数**

/\*\*\*

\* 设置基础传输参数，请在Start接口之前调用

\* @param pRtp\_avcom: 模块指针

\* @param nRecvDelayMs: 接收缓存时间，建议4\*RTT，单位ms。可在发送端或接收端设置，将取其中较大的值

\* @param nMaxBitrateKbps：最大传输码率，建议3\*VideoEncBitrate，单位kbps。需要在发送端设置，当设置为0时表示不受限。若码率本身比较平稳，可设置为2\*VideoEncBitrate。

\* @return:

\*/

BOOL SDSrtAvCom\_SetBaseTransParams(void\* pRtp\_avcom, int nRecvDelayMs, int nMaxBitrateKbps);

1. **设置视频通道传输参数**

/\*\*\*

\* 设置视频通道FEC传输参数，请在Start接口之前调用

\* @param pRtp\_avcom: 模块指针

\* @param bEnable: 是否启用FEC，收发双方需保持一致

\* @param nCols: FEC Group列数

\* @param nRows: FEC Group行数

\* @param eLayoutMode：2D FEC布局模式

\* @param eArqMode：FEC-ARQ配合模式

\* @return:

\*/

BOOL SDSrtAvCom\_SetVideoFecParams(void\* pRtp\_avcom, BOOL bEnable, int nCols, int nRows, E\_SRT\_FEC\_LAYOUT eLayoutMode, E\_SRT\_FEC\_ARQ eArqMode);

1. **设置音频通道传输参数**

/\*\*\*

\* 设置音频通道FEC传输参数，请在Start接口之前调用

\* @param pRtp\_avcom: 模块指针

\* @param bEnable: 是否启用FEC，收发双方需保持一致

\* @param nCols: FEC Group列数

\* @param nRows: FEC Group行数

\* @param eLayoutMode：2D FEC布局模式

\* @param eArqMode：FEC-ARQ配合模式

\* @return:

\*/

BOOL SDSrtAvCom\_SetAudioFecParams(void\* pRtp\_avcom, BOOL bEnable, int nCols, int nRows, E\_SRT\_FEC\_LAYOUT eLayoutMode, E\_SRT\_FEC\_ARQ eArqMode);

1. **获得视频通道统计数据**

/\*\*\*

\* 获取视频通道统计信息

\* @param pRtp\_avcom: 模块指针

\* @param pfRttMs: RTT，单位毫秒

\* @param pfUpLossRate: 上行丢包率.内部已经乘100

\* @param pfDownLossRate: 下行丢包率.内部已经乘100

\* @param pfEstimatedUpBitrate：上行带宽估算.Kbps

\* @param pfUpBitrate：上行码率.Kbps

\* @param pfDownBitrate：下行码率.Kbps

\* @return:

\*/

BOOL SDSrtAvCom\_GetVideoTransStatis(void\* pRtp\_avcom, double \*pfRttMs, double \*pfUpLossRate, double \*pfDownLossRate, double \*pfEstimatedUpBitrate, double \*pfUpBitrate, double \*pfDownBitrate);

1. **获得音频通道统计数据**

/\*\*\*

\* 获取音频通道统计信息

\* @param pRtp\_avcom: 模块指针

\* @param pfRttMs: RTT，单位毫秒

\* @param pfUpLossRate: 上行丢包率.内部已经乘100

\* @param pfDownLossRate: 下行丢包率.内部已经乘100

\* @param pfEstimatedUpBitrate：上行带宽估算.Kbps

\* @param pfUpBitrate：上行码率.Kbps

\* @param pfDownBitrate：下行码率.Kbps

\* @return:

\*/

BOOL SDSrtAvCom\_GetAudioTransStatis(void\* pRtp\_avcom, double \*pfRttMs, double \*pfUpLossRate, double \*pfDownLossRate, double \*pfEstimatedUpBitrate, double \*pfUpBitrate, double \*pfDownBitrate);

**附录：结构体说明**

//2D FEC布局模式

Typedef enum E\_SRT\_FEC\_LAYOUT

{

//普通连续模式

e\_SRT\_FEC\_LAYOUT\_EVEN = 0,

//阶梯模式，可以降低一定码率波动（冗余包分散发送）

e\_SRT\_FEC\_LAYOUT\_STAIR

} E\_SRT\_FEC\_LAYOUT;

//FEC-ARQ配合模式

typedef enum E\_SRT\_FEC\_ARQ

{

//只要丢包均会发起NAK，不管FEC能否恢复

e\_SRT\_FEC\_ARQ\_ALWAYS = 0,

//仅在FEC失败时发起NAK

e\_SRT\_FEC\_ARQ\_ONREQ,

//关闭NAK

e\_SRT\_FEC\_ARQ\_NEVER,

} E\_SRT\_FEC\_ARQ;

/\*输出接收到的视频数据 回调函数

\* @param bComplete用来表示当前帧数据是否完整（无局部丢包）

\* @param bPrevTotalFrameLost用来表示当前帧与上一次输出帧之间无整帧丢失的情况，即本帧序号与上一帧序号是否连续。

\* 通过以上两个标志，结合关键帧判定标志，外层可以很方便的实现丢帧冻结机制

\*/

typedef void (\*CallBackFuncRecvVideoData)(void\* pObj, int nLen, unsigned char \*byBuf, unsigned int unPTS, BOOL bComplete, BOOL bPrevTotalFrameLost);

/\*输出接收到的视频数据 回调函数\*/

typedef void (\*CallBackFuncRecvAudioData)(void\* pObj, int nLen, unsigned char \*byBuf, unsigned int unPTS);