**RTP网络SDK功能设计及接口**

1. 初始化SDK:

// 返回值 0 成功 其他值失败

Int rtpInit();

1. 创建一个RTP实例:

// 返回值 0 成功，其他值失败

Int rtpCreateRtpInstance(long \*handler);

1. 设置Rtp实例为接收流模式：

// instanceHandler RTP实例句柄

// lPort 本地端口(接收端口 udp 则为接收端口 tcp则为监听端口)

// transType 传输类型(0-udp 1-tcp)

// streamType 流类型(0-h264 1-G711 2-ps)

// rtcpTimeout rtcp 超时时间(rtcp作为连接心跳使用(udp)，当超过指定时间没有收到rtcp包时，默认对端已离开)

// rtpRecvRtpData RTP数据接收回调函数

// arg1 rtpRecvRtpData 私有参数

// rtpRecvRTCPData RTCP数据接收回调函数

// arg2 rtpRecvRTCPData 私有参数

// rtpRecvMediaData 音视频数据接收回调函数(该回调函数返回音视频原始数据)(该回调函数不能为空)

// arg3 rtpRecvMediaData 私有参数

// rtpException SDK异常回调

// arg4 rtpException 私有参数

// 返回值 0成功 其他值失败

Int rtpSetRtpInstanceRecvModel(long instanceHandler,unsigned short lPort,int transType,int streamType,rtcpTimeout rt,rtpRecvRtpData recvRtp,void \*arg1,rtpRecvRTCPData recvRtcp,void \*arg2,rtpRecvMediaData recvMediaData,void \*arg3,rtpException re,void \*arg4);

注：一个RtpInstance 只能为接收流模式或推送流模式

1. 设置Rtp实例为推送流模式：

// instanceHandler RTP实例句柄

// rIp 远端ip

// rPort 远端端口(udp 为发送端口 tcp为连接端口)

// lPort 本地绑定端口

// transType 传输类型(0-udp 1-tcp)

// streamType 流类型(0-h264 1-G711 2-PS)

// rtcpTimeout rtcp 超时时间 (rtcp作为连接心跳使用，当超过指定时间没有收到rtcp包时，默认对端已离开)

// rtpException SDK异常回调(该回调返回异常事件)

// arg1 rtpException 私有参数

// rtpRecvRTCPData RTCP数据接收回调函数

// arg2 rtpRecvRTCPData 私有参数

// 返回值 0成功 其他值失败

Int rtpSetRtpInstancePushModel(long instanceHandler,char \*rIp,unsigned short rPort,unsigned short lPort,int transType,int streamType,int rtcpTimeout rt,rtpException re,void \*arg1,rtpRecvRTCPData recvRtcp,void \*arg2);

注：一个RtpInstance 只能为接收流模式或推送流模式

1. 设置SSRC字段

// instanceHandler RTP实例句柄

// ssrc RTP SSRC值

// 返回值 0成功 其他值失败

Int rtpSetRtpSSRC(long instanceHandler,int ssrc);

注：该接口可重复调用

1. 同时发送多路音频视频:

// instanceHandler RTP实例句柄

// inBuf 输入缓冲区

// intLen 输入数据真实长度

// 返回值 0 成功 其他值失败

Int rtpSendData(long instanceHandler,char \*inBuf,int inLen);

1. 从指定端口接受数据(多路)(注册回调函数方式,为空则SDK内部默认处理):

// RTP数据回调函数

// 该回调函数不允许阻塞

// outBuf 数据缓冲区(该缓冲区由SDK传给调用者,不推荐调用者直接操作该缓冲区)

// outLen 数据长度

// ssrc 该RTP所属的SSRC

// arg 用户自定义数据指针，该指针SDK不作处理

// 返回值 0成功(该回调函数成功时，调用者务必返回0，否则SDK会根据返回值的不同,对数据进行二次处理)

typedef int (\*rtpRecvRTPData)(char \*outBuf,int outLen,int ssrc,void \*arg);

1. 调用者处理RTCP数据(多路)(注册回调函数方式,为空则SDK内部默认处理):

// RTCP回调函数

// 该回调函数不允许阻塞

// outBuf 数据缓冲区(该缓冲区由SDK传给调用者)

// outLen 数据长度

// ssrc 该RTCP所属的SSRC

// arg 用户自定义数据指针，该指针SDK不会处理

// 返回值 0成功(该回调函数成功时，调用者务必返回0，否则SDK会根据返回值的不同,对数据进行二次处理)

typedef int (\*rtpRecvRTCPData)(char \*outBuf,int outLen,int ssrc,void \*arg);

1. 调用者获取音视频数据(多路)(该回调函数不能为空)

// 媒体数据回调函数

// 该回调函数不允许阻塞

// outBuf 数据缓冲区(该缓冲区由SDK传给调用者,不推荐调用者直接操作该缓冲区)

// outLen 数据长度

// ssrc 该媒体流所属的SSRC

// mediaType 媒体格式(0-h264 1-G711)

// arg 用户自定义数据指针，该指针SDK不会处理

// 返回值 0成功(该回调函数成功时，调用者务必返回0，否则SDK会根据返回值的不同,对数据进行二次处理)

typedef int (\*rtpRecvMediaData)(char \*outBuf,int outLen,int ssrc,int mediaType,void \*arg);

1. 异常回调:

// SDK内部异常回调(主要为网络断开事件)

// exceptionType 异常类型

// arg 用户自定义数据指针，该指针SDK不会处理

typedef int (\*rtpException)(int exceptionType,void \*arg);

1. 调用者在合适的时机释放实例资源:

// instanceHandler RTP实例句柄

// 返回值0 成功 其他值失败

Int rtpClearRTPInstance(long instanceHandler);

1. 获取SDK出错原因:

// 根据错误码获取出错原因

// errorCode 接口返回的错误码

char\* rtpGetError(int errorCode);

注：该接口初期可能不会实现

1. 设置SDK日志信息：

// fileName 日志文件名

// level 日志等级

Int rtpSetLog(char \*fileName,char \*level);

1. 清理SDK:

// 清理SDK

// 返回值 0 成功 其他值失败

Int rtpCleanup();

注：该文档不完善，后续会有变动更新