客户端无内置编解码 SDK(C 版)说明(V1.0.0)

一、基本概念

房间：服务器上一个虚拟的概念，进入同一个房间的客户端才允许数据交互，不同房间之间的客户端并无交集。一个房间内客户端的最大数目理论不限，具体根据服务器端设置。服务器允许最大的房间数目同样依据服务端设置而定。最简单的模型是一个房间内一个发布者，一个接收者。

客户端 UID: 用于标识每个客户端的唯一 ID,在系统中不允许有相同 UID 的客户端同时在线，否则新登录的客户端会将之前的客户端“顶下去”。

音视频发布者: 具备上行音视频到服务器（房间）能力的客户端类型，同时也能接收来自服务器（房间）的音视频数据。

音视频接收者: 只从服务器（房间）接收音视频的客户端类型，不具备上行能力，比如直播业务中的普通观众。

音视频位置: 一个房间内允许同时多个音视频发布者，它们发布音视频到不同的“位置”上，接收者可选择接收某个或某些位置的音视频。目前单个房间允许 32 个“位置”。发布者登录后可以选择由服务器自动分配空闲的“位置”供其发布，也可以指定“位置”发布，若当前该“位置”上已有其他发布者，后者将顶替前者以保证同一位置上同时只有一个发布者。

音视频接收掩码: 客户端可以根据业务需要选择接收房间内某个或某几个位置的音频或视频数据，可以通过设置自己期望接收的掩码到服务器。音频和视频使用各自独立的掩码，每个掩码 32bit 对应房间内的 32 个位置。

传输参数: 本文传输参数是指视频通道的 FEC 上行冗余度、上行 FEC Group 组大小、接收 Qos 丢包等待时间，音频通道在内部已经根据经验数据配置好了合适的值且不对外开放。对于纯接收端（纯下行）同样也可以设置 FEC 上行冗余度、上行 FEC Group 组大小，内部将忽略该参数。

二、API 接口

所有 API 接口定义均位于 SDTerminalSdk.h 文件中。

1、系统环境初始化，仅需调用一次

void SDTerminal_Enviroment_Init(const char * outputPath, int outputLevel)

参数：

@param: outputPath: 日志文件输出的目录，若目录不存在，SDK 将自动创建，支持相对路径或绝对路径。日志对于问题定位非常重要，建议开启。

@param: outputLevel: 日志输出的级别，只有等于或者高于该级别的日志会输出到文件，日志级别有 ： LOG_OUTPUT_LEVEL_DEBUG 、 LOG_OUTPUT_LEVEL_INFO 、 LOG_OUTPUT_LEVEL_WARNING 、 LOG_OUTPUT_LEVEL_ERROR 、 LOG_OUTPUT_LEVEL_ALARM 、 LOG_OUTPUT_LEVEL_FATAL 、 LOG_OUTPUT_LEVEL_NONE，当指定为 LOG_OUTPUT_LEVEL_NONE 时，将不会生成日志文件。

2、系统退出时调用一次反初始化

void SDTerminal_Enviroment_Free ()

3、创建客户端 SDK 对象

```
void*  SDTerminal_Create();
```

4、销毁客户端 SDK 对象

```
void  SDTerminal_Delete(void** ppTerminal);
```

参数：

@ ppTerminal，模块指针的指针

说明：使用者应该做好与其他 API 之间的互斥保护

5、登录服务器

```
int  SDTerminal_Online(
    void* pTerminal,
    char* strMediaServerIp,
    unsigned int unDomainId,
    unsigned int unUid,
```

```
    unsigned int unRoomId,
    USER_ONLINE_TYPE eUserType
    );
```

参数：

@pTerminal，模块指针

@strMediaServerIp，服务器 IP 地址

@unDomainId，域 ID，默认为 1 即可，具体含义请参考 SRTP-Server 服务器设计相关文档。

@unUid，表示当前客户端使用的唯一用户 ID，需要由用户自行保障唯一性。要求非 0。

@unRoomId，表示当前客户端进入的房间号，要求非 0。

@eUserType，表示客户端的类型，定义如下

```
typedef enum USER_ONLINE_TYPE
{
    //其他类型，保留
    USER_ONLINE_TYPE_OTHER = 0,
    //音视频收发类型
    USER_ONLINE_TYPE_AV_SEND_RECV,
    //仅接收音视频类型
    USER_ONLINE_TYPE_AV_RECV_ONLY,
    //仅发送音视频类型
    USER_ONLINE_TYPE_AV_SEND_ONLY
} USER_ONLINE_TYPE;
```

用户根据自身业务选择合适的客户端类型，以获得资源的最低占用。

返回值：

返回 0 表示登录成功，返回负数则为失败，负数值为其错误码。

说明：本 API 为阻塞式同步登录接口，登录失败的超时时间较长，请避免在 UI 线程中调用。若需要异步登录接口，请参考 SDTerminal_OnlineAsync，后者即刻返回，并通过回调接口通知外层登录结果。

6、下线服务器

```
void  SDTerminal_Offline(void* pTerminal);
```

参数:

@pTerminal，模块指针

返回值: 无

说明: 下线服务器，若此时用户在音视频位置上，将同时从位置上下来（自动调用 OffPosition）。

7、请求上传音视频到指定位置

```
int  SDTerminal_OnPosition(void* pTerminal, unsigned char *pucPosition);
```

参数:

@pTerminal，模块指针

@pucPosition，大于等于 0，小于系统支持的最大位置数（宏 MAX_SUPPORT_POSITION_NUM 定义）。特殊值：当设置为 255 时表示由服务器分配当前空闲的位置，分配结果将写回本参数。本函数调用成功后，客户端将自动停止接收自己位置的音视频流（不看不听自己）。

返回值:

返回 0 表示请求发布成功，返回负数则为失败，负数值为其错误码。

说明: 请在 Online 成功后调用

8、请求从位置上下来

```
void  SDTerminal_OffPosition(void* pTerminal);
```

参数:

@pTerminal，模块指针

返回值: 无

调用 OffPosition 后，客户端将从服务器所在位置上下来并停止发送音视频数据，此时它仍然保持在线状态并能接收其他位置的音视频流。本函数调用成功后，将自动恢复接收自己之前位置的音视频流（可能有其他客户端加入该位置）。

说明: 本接口只能在 Online 成功后调用。SDTerminal_Offline 内部自带本 API 功能，客户端下线可以不用单独调用本 API，直接调用 SDTerminal_Offline 即可。

9、上传视频数据

```
void  SDTerminal_SendVideoData(void* pTerminal, unsigned char *byBuf, unsigned int unLen,
unsigned int unDts);
```

向请求的位置发送视频码流，一次传入**一帧带起始码**的码流。

参数：

@pTerminal，模块指针

@byBuf，码流存放区。

@unLen，码流长度。

@unDts，SDK 默认使用内部时间戳，此时本参数将被忽略。当用户期望使用外部时间戳时，需调用 SDTerminal_SetUseInternalTimeStamp API来指定时间戳工作模式。

返回值：无

10、上传音频数据

```
void SDTerminal_SendAudioData(void* pTerminal, unsigned char *byBuf, unsigned int unLen, unsigned int unDts);
```

向请求的位置发送音频码流，一次传一帧 **ADTS** 码流。

参数：

@pTerminal，模块指针

@byBuf，码流存放区。

@unLen，码流长度。

@unDts，SDK 默认使用内部时间戳，此时本参数将被忽略。当用户期望使用外部时间戳时，需调用 SDTerminal_SetUseInternalTimeStamp API来指定时间戳工作模式。

返回值：无

11、设置音视频下行掩码

```
void SDTerminal_SetAvDownMasks (void* pTerminal, unsigned int unAudioMask, unsigned int unVideoMask);
```

通过设置音视频下行掩码，可以选择从服务器接收哪几个位置的音视频数据。每一个 bit 对应一个位置，最低位对应 0 号位置，最高位对应 31 号位置。比如希望接收某个 index 位置的音视频时，可以设置其对应 bit 设置为 1，否则设置为 0。

允许接收某个 index 位置的音视频时，可以设置为：

```
UINT unMask = 0x1 << (index);
unAudioDownChannelsMask |= unMask;
```

```
unVideoDownChannelsMask |= unMask;
```

停止接收某个 index 位置的音视频时，可以设置为：

```
UINT unMask = 0x1 << (index);
unMask = ~unMask;
unAudioDownChannelsMask &= unMask;
unVideoDownChannelsMask &= unMask;
```

```
SDTerminal_SetAvDownMasks (pTerminal, unAudioDownChannelsMask, unVideoDownChannelsMask);
```

当本客户端仅发送音视频，不接收音视频时，设置音视频接收掩码全为 0 即可。

参数：

@pTerminal，模块指针

@unAudioMask，控制音频接收的掩码。

@unVideoMask，控制视频接收的掩码。

返回值：无

说明：请在 Online 成功后调用

12、设置音视频传输参数

```
void SDTerminal_SetTransParams(void* pTerminal,
unsigned int unJitterBuffDelay, FEC_REDUN_TYPE eFecRedunMethod,
unsigned int unFecRedunRatio,
unsigned int unFecMinGroupSize, unsigned int unFecMaxGroupSize, BOOL bEnableNack);
```

参数：

@pTerminal，模块指针

@unJitterBuffDelay，本客户端接收码流时的内部缓存时间（毫秒），范围 0~600。设置为 0 时，将关闭内部接收 JitterBuff 功能。

@eFecRedunMethod，为上行 FEC 冗余度方法，包括 AUTO_REDUN 自动冗余度、FIX_REDUN 固定冗余度。

@unFecRedunRatio，上行冗余比率，比如设置为 30，则表示使用 30%冗余。自动冗余度时以该冗余度作为上限，根据网络情况向下调整。

@unFecMinGroupSize，为上行 FEC 分组的下限值， 1Mbps 以下建议设置为 16，1Mbps~2Mbps 建议设置 24，2Mbps 以上建议设置 28。

@unFecMaxGroupSize，为上行FEC分组的上限值，根据终端CPU能力而定，最大不超过72，越大FEC所

消耗的CPU越高，抗丢包能力也越强，建议性能足够的设备上设置为64。

@bEnableNack，是否启用 NACK 功能。

返回值：无

说明：本函数需在 Online 之前调用。

13、获取当前 SDK 版本信息

```
UINT SDTerminal_GetVersion (void* pTerminal);
```

参数：

@pTerminal，模块指针

返回值：获得当前 SDK 的版本信息

14、获取当前丢包率数据

```
void  SDTerminal_GetVideoAudioUpDownLostRatio(void* pTerminal, float *pfVideoUpLostRatio,
float *pfVideoDownLostRatio, float *pfAudioUpLostRatio, float *pfAudioDownLostRatio);
```

参数：

@pTerminal，模块指针

@pfVideoUpLostRatio，获取视频上行丢包率

@pfVideoDownLostRatio，获取视频下行丢包率

@pfAudioUpLostRatio，获取音频上行丢包率

@pfAudioDownLostRatio，获取音频下行丢包率

返回值：内部已经乘 100 得到百分比

15、获取当前视频通道的实时 RTT

```
unsigned int  SDTerminal_GetCurrentRtt(void* pTerminal);
```

@pTerminal，模块指针

返回值：获得当前视频通道的实时 RTT 值

16、指定时间戳工作模式

```
void  SDTerminal_SetUseInternalTimeStamp(void* pTerminal, BOOL bUseInternalTimestamp);
```

@pTerminal，模块指针

@bUseInternalTimestamp，是否采用内部时间戳模式，TRUE-内部，FALSE-外部。默认情况下未调用本 API 时，系统采用内部时间戳模式，此时将在每次调用 Send 接口时自动产生时间戳。当用户需要使用外部时间戳时，需调用本 API 指定。

返回值： 无

说明： 本函数需在 Online 之前调用。

17、指定本客户端最大接收的音视频路数

```
void  SDTerminal_SetSupportDecodeNum(void* pTerminal, unsigned int unSupportDecodeNum);
```

指定最大支持的音视频流接收路数，默认为 MAX_SUPPORT_POSITION_NUM，外层可根据实际情况在 Online 接口之前调整本值以达到节省资源的目的。仅在有把握时调用本 API。

@pTerminal，模块指针

@unSupportDecodeNum，实际需要支持的最大接收路数，比如设置为1，则仅接收0号位置音视频。

返回值： 无

说明： 本函数需在 Online 之前调用。

三、回调输出相关 API 接口

内部状态、接收的远端音视频数据均通过回调函数的方式交给外层，SDK 提供了相关的回调函数设置接口。

1、设置系统状态通知回调

```
void  SDTerminal_SetSystemStatusNotifyCallback(void* pTerminal, SystemStatusNotifyFunc pfStatusNotifyCallback, void* pObject);
```

参数：

@pTerminal，模块指针

@pfStatusNotifyCallback,回调函数指针

@ pObject，透传指针，将透传给回调函数。

返回值：无

2、设置视频数据接收回调

```
void SDTerminal_SetRecvRemoteVideoCallback(void* pTerminal, RecvRemoteVideoFunc pfRecvRemoteVideoCallback, void* pObject)
```

参数：

@pTerminal，模块指针

@pfRecvRemoteVideoCallback,回调函数指针

@pObject，透传指针，将透传给回调函数。

RecvRemoteVideoFunc 的定义如下：

```
void (*RecvRemoteVideoFunc)(void* pObject, unsigned char ucPosition, unsigned char* data, unsigned int unLen, unsigned int unPTS, VideoFrameInfor* pFrameInfo);
```

其中，VideoFrameInfor 提供了当前帧以及当前流的重要信息：

```
typedef struct VideoFrameInfor
{
    unsigned int unWidth;
    unsigned int unHeight;
    unsigned int unFps;
    BOOL bPacketLost;
    BOOL bKeyFrame;
    BOOL bInfoUpdated;
    unsigned char bySps[512];
    unsigned int unSpsSize;
    unsigned char byPps[512];
    unsigned int unPpsSize;
}VideoFrameInfor;
```

模块内部会获得当前码流的宽、高、帧率、SPS、PPS 告知外层，当其中某些参数发生变更时将置位 bInfoUpdated。另外 bPacketLost 与 bKeyFrame 变量可用于外层实现丢帧冻结机制，bPacketLost 表示当前帧是否接收完整，若网络丢包且 FEC 未能恢复时，该标志将置位。bKeyFrame 表示当前帧是否

为 IDR 关键帧。如何使用这两个标志实现丢帧冻结可以联系技术支持获得帮助。需要说明的是，当没有丢包发生时，本函数的输出与对方调用 SDTerminal_SendVideoData 函数的输入完全一致。

返回值： 无

说明： SDK 内部是在独立于网络接收线程之外的线程中调用本接口，所以外层可以将一定耗时的操作（比如解码）放置在此。

3、设置音频数据接收回调

```
void  SDTerminal_SetRecvRemoteAudioCallback(void* pTerminal, RecvRemoteAudioFunc pfRecvRemoteAudioCallback, void* pObject);
```

参数：

@pTerminal，模块指针

@pfRecvRemoteVideoCallback,回调函数指针

@pObject，透传指针，将透传给回调函数。

RecvRemoteAudioFunc 的定义如下：

```
void (*RecvRemoteAudioFunc)(void* pObject, unsigned char ucPosition, unsigned char* data, unsigned int unLen, unsigned int unPTS, AudioFrameInfor* pFrameInfo);
```

其中，AudioFrameInfor 提供了当前帧以及当前流的重要信息：

```
typedef struct AudioFrameInfor
{
    unsigned int unCodecType;
    unsigned int unSampleRate;
    unsigned int unChannelNum;
    unsigned int unFrameNo;
    BOOL bInfoUpdated;
}AudioFrameInfor;
```

音频帧为 ADTS 格式，其每个包头部均附带了采样率、通道数、编码格式等信息。

返回值： 无

说明： SDK 内部是在独立于网络接收线程之外的线程中调用本接口，所以外层可以将一定耗时的操作（比如解码）放置在此。

4、设置远端请求 IDR 通知回调

void SDTerminal_SetRemoteIdrRequestNotifyCallback(void* pTerminal,

RemoteIdrRequestNotifyFunc pfRemoteIdrRequestNotifyCallback, void* pObject);

参数:

@pTerminal, 模块指针

@pfRemoteIdrRequestNotifyCallback,回调函数指针

@pObject, 透传指针, 将透传给回调函数。

RemoteIdrRequestNotifyFunc 的定义如下:

BOOL (*RemoteIdrRequestNotifyFunc)(void* pObject);

当服务端接收失败时, 即会通过本回调通知本端, 本端可以尽快编码 IDR 帧以避免远端长时间画面冻结。

返回值: 无

说明: 注意 SDK 内部是在网络接收线程中调用本回调, 因此外层不应在回调中执行耗时操作, 应尽快返回。

5、设置码率自适应通知回调

void SDTerminal_SetAutoBitrateNotifyCallback(void* pTerminal,

AUTO_BITRATE_TYPE eAutoBitrateMethod,

AutoBitrateNotifyFunc pfAutoBitrateNotifyCallback,void* pObject);

参数:

@pTerminal, 模块指针

@eAutoBitrateMethod, 码率自适应方法: AUTO_BITRATE_TYPE_ADJUST_DISABLE (关闭码率自适应, 默认不调用本 API 时即关闭状态)、AUTO_BITRATE_TYPE_ADJUST_FRAME_FIRST (优先降低帧率, 其次降低码率)、AUTO_BITRATE_TYPE_ADJUST_BITRATE_FIRST (优先降低码率, 其次降低帧率)。

@pfAutoBitrateNotifyCallback,回调函数指针

@pObject, 透传指针, 将透传给回调函数。

AutoBitrateNotifyFunc 的定义如下:

BOOL (*AutoBitrateNotifyFunc)(void* pObject, unsigned int unFrameDropInterval, float fBitrateRatio);

当使用模块的码率自适应评估时, 评估结果由本接口送出, 外层负责具体的实施。比如

unFrameDropInterval=2 表示每 2 帧丢 1 帧。unFrameDropInterval 为 0 表示不丢帧。比如 fBitrateRatio=0.8 表示需要将码率降低为原始码率的 0.8 倍。外层需同时响应帧率调整和码率调整，当外层执行了码率自适应动作时，返回 TRUE，否则返回 FALSE。

返回值： 无

说明： 注意 SDK 内部是在网络接收线程中调用本回调，因此外层不应在回调中执行耗时操作，应尽快返回。

四、一种简单场景下的使用示意

下面演示一种纯发送端（不接收）的 API 调用情况，其他场景的 API 使用可以以此为参考或咨询技术支持。

1、调用环境初始化 SDTerminal_Enviroment_Init，整个系统只需要调用一次。

2、调用 SDTerminal_Create 创建一个客户端对象。

3、调用 SDTerminal_SetSystemStatusNotifyCallback 注册消息通知回调函数，一个消息通知函数举例：

```
static void SDTerminal_SystemStatusNotify(void* pObject, STATUS_CHANGE_NOTIFY unStatus)
    {
        switch (unStatus)
        {
            case STATUS_NOTIFY_EXIT_KICKED:
                // 本客户端账号(uid)在其他位置登录，被踢出房间
                printf("kicked by other user, exiting...\n");
                break;
            case STATUS_NOTIFY_RECON_START:
                // 客户端掉线，内部开始自动重连
                printf("starting reconnect\n");
                break;
            case STATUS_NOTIFY_RECON_SUCCESS:
```

```
                // 内部自动重连成功
                printf("reconnect success\n");
                break;
            default:
                printf("unprocessed status(%d)\n", unStatus);
        }
    }
```

消息通知函数中不要放耗时操作，一般仅用于日志或界面提示。

4、调用一次 SDTerminal_SetTransParams 设置发端上行传输 FEC 参数，作为纯发送端，推荐调用为：

```
SDTerminal_SetTransParams (pSdk, 0, FIX_REDUN, 30, 28, 64, TRUE);
```

参数说明：接收时 QOS 丢包延时为 0（发端没有接收，直接设置 0），使用固定冗余度方式 FIX_REDUN，使用 30%的上行冗余，使用 GroupMin 大小为 28，使用 GroupMax 大小为 64，使能 NACK。

5、调用一次 SDTerminal_Online 登录服务器，举例调用为：

```
SDTerminal_Online(pSdk, strServerIp, 1, unUserId, unRoomId,
USER_ONLINE_TYPE_AV_SEND_ONLY);
```

6、调用一次 SDTerminal_OnPosition 设置上传音视频到房间的哪个位置，举例调用为：

```
unsigned char byUpPosition = 0;
   SDTerminal_OnPosition(pSdk, byUpPosition);
```

7、调用一次 SDTerminal_SetAvDownMasks 设置发端从服务器接收音视频的情况，推荐调用为：

```
   SDTerminal_SetAvDownMasks(pSdk, 0, 0);
```

参数说明：第一个参数表示从服务器接收的音频掩码，第二个参数表示接收的视频掩码；二者均为 32 位整数，每个 bit 对应一个音视频位置，比如最低 bit 位对应 0 号位置，若希望接收该位置的视频或音频则设置该 bit 为 1。因为发端只发不收，所以我们直接设置音频视频接收掩码都为 0，表示什么也不接收。

8、N 次调用 SDTerminal_SendVideoData、SDTerminal_SendAudioData 向服务器发送音视频数据

9、 结束前调用一次 SDTerminal_Offline (内部将自动 OffPosition)

10、 调用 SDTerminal_Delete 回收客户端对象

11、调用 SDTerminal_Enviroment_Free 反初始化环境