

点对点 SDK(C 版)说明(V1.0.1)

www.mediapro.cc

一、重要说明

FEC: 前向纠错技术, 通过增加冗余带宽方式, 提高网络对于丢包的抵抗力。

NACK: 重传请求机制, 与 FEC 配合, 在预估 FEC 无法恢复时触发接收端到发送端的重传请求, 发送端予以重发响应。**NACK** 仅重传一次, 不保证数据一定传输成功, 优点是可以获得稳定的低延时, 具体表现为: 弱网延时无累积, 画面可能因丢包而卡顿。

ACK: 接收端收到包后给予发送端应答的传输方式。**ACK** 保证数据一定传输成功, 但延时可能出现不可控的放大。具体表现为: 非极端网络环境下, 稳定性流畅度较 **NACK** 方案好。极端环境下可出现延时放大。

Smooth 平滑: 发送端对一帧较大的码流在时间限内平滑发送, 避免一次性发出给网络带来的压力。

JitterBuff: 接收端为了抵消网络传输、丢包恢复、**NACK** 重传引入的抖动, 引入 JitterBuff 缓存。缓存时间越大, 画面流畅度越高, 但延时也同步增大。当需要极低延时时, 可设置 JitterBuff 为 0。设置为非 0 值时, 该值将作为 JitterBuff 初始值, 后续本模块将根据网络情况自行调整。

IDR 请求: 接收端若出现视频帧丢包无法恢复时, 将向发送端发起编码 IDR 帧请求, 发送端可以即刻编码 IDR 帧, 使得接收端尽快恢复。

丢包冻结机制: 接收端若出现视频帧丢包无法恢复时, 可以采取停止解码、渲染, 直至完整的 IDR 帧到来再继续解码渲染的方式, 避免用户察觉到画面花屏。本 SDK 内部实现了丢包冻结, 确保外层始终获得可解码可渲染码流。

时间戳模式: 本 SDK 提供了内置时间戳(默认)和外部时间戳两种模式, 当用户选择外部时间戳模式时, 需在发送端提供音视频时间戳(1KHZ 时基, 即毫秒单位), 该时间戳将在接收端透传露出。当用户选择内置时间戳模式时, SDK 将在用户调用 SDK 发送接口时自动生成时间戳, 以简化用户使用。

码率自适应: 本 SDK 提供了码率自适应能力, 内部将根据网络状况实时对外输出码率调整建议。支持的码率自适应模式包括: 关闭码率自适应、优先调整帧率模式、优先调整编码码率模式三种。其中优先调整帧率模式: 是在保持编码器参数不变的情况下, 通过降低实际送编码器的帧率(在采集之后、编码之前进行丢帧), 实现画面质量基本不变的同时间接降低带宽。其中优先调整编码码率模式: 该模式要求外层编码器支持动态设置编码码率, 保持画

面流畅性，降低编码码率（画质）来适应网络。目前业内以优先调整编码码率模式居多（流畅性优先），推荐使用。注意当使用 ACK 模式时，外层必须实现码率自适应策略。注意当启用码率自适应时，本 SDK 提供了两个码率自适应接口，均需实现。

内部拆包传输：外层传入一帧码流后，SDK 内部将进行拆分发送，拆分包大小与 FEC min group 取值相关，原则上 400~1000 字节之间波动，拆分后的包大小加上私有头部、UDP\IP 头部后不会超过 1500 字节。

FEC 相关参数：FEC 相关参数包括 FEC 冗余度方法、FEC 上行冗余度、FEC min group 组大小、FEC max group 组大小。FEC 冗余度方法包括：固定冗余度、自动冗余度两种。固定冗余度即全程使用用户指定的冗余度进行 FEC 编码，自动冗余度则以用户指定的 FEC 冗余度为上限，根据网络情况进行向下调整，对于网络较好的场合尽量降低带宽。FEC 是分组进行的，即多个拆分包组成一个 group，产生其冗余包。group 越大，同样冗余度的情况下，产生的冗余包越多，抵抗连续丢包能力越强，同时因丢包产生的抖动也会越大（因为 FEC 的 group 可能产生跨多帧的情况，前面包的丢失不得不等到后续包的到来才能恢复）；FEC 分组越大消耗 CPU 资源也越大。建议配置 FEC min group 大小 16；建议根据芯片处理性能设置 FEC max group，在性能足够的设备上建议设置为 64（PC、主流 Android 手机均可设置为 64，嵌入式平台则根据实际情况设置）。

二、API 接口

所有 API 接口定义均位于 SDTerminalP2PSdk.h 文件中。

1、系统环境初始化，仅需调用一次

```
void SDTerminalP2P_Environment_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，日志级别取值见 LOG_OUTPUT_LEVEL_P2P，有 DEBUG、INFO、WARNING、ERROR、ALARM、FATAL、NONE 几个级别可选，当指定为 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 SDTerminalP2P_Enviroment_Free ()
```

3、创建 SDK 对象

```
void* SDTerminalP2P_Create();
```

4、销毁 SDK 对象

```
void SDTerminalP2P_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、准备会话

```
int SDTerminalP2P_Online(  
    void* pTerminal,  
    const char* strLocalIp,  
    unsigned short shLocalPort,  
    const char* strRemoteIP,  
    unsigned short shRemotePort,  
    CLIENT_USER_TYPE_P2P eUserType  
);
```

参数：

@pTerminal, 模块指针

@strLocalIp, 绑定的本地 IP 地址，当设置为 NULL 时，内部将使用 INADDR_ANY，交由操作系统选择一个网卡 IP。建议指定 IP 以防多 IP 时系统选择错误

@shLocalPort, 绑定的本地端口号，允许设置本地端口为 0，由操作系统选择一个当前可用的端口。

作为(服务端\播放端\接收端)使用时，需要指定本地监听的端口。

@strRemoteIP, 远端 IP 地址。(服务端\播放端\接收端)上设置 NULL 即可，作为(服务端\播放端\接收端)时，一般是不知道(客户端\发送端)的 IP 和端口，内部将在收到远端数据后自动翻转 IP 和端

口，从而获得可用于向远端发送数据的 IP 和端口。注意：当用户通过本参数指定了远端 IP 时，SDK 仅会接收来自该 IP 的数据，丢弃来自其他 IP 的数据。

@shRemotePort，远端端口号。（服务端\播放端\接收端）上设置远端端口号为 0 即可。

@eUserType，本客户端类型：收发一体、纯发送、纯接收，按需设置类型可以实现最小的资源开销。

返回值：

返回大于等于 0 表示成功，返回负数则为失败，负数值为其错误码。

6、结束会话

```
void SDTerminalP2P_Offline(void* pTerminal);
```

参数：

@pTerminal，模块指针

返回值：无

7、发送视频数据

```
void SDTerminalP2P_SendVideoData(void* pTerminal, unsigned char *byBuf, unsigned int unLen, unsigned int unDts, BOOL bIsHevc);
```

发送视频码流，一次传入一帧带起始码(0x 00 00 00 01 或 0x00 00 01)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模式，内部自动产生时间戳，此时本参数传入 0 即可。当用户调用 SDTerminalP2P_SetUseInternalTimeStamp API 来指定外部时间戳模式时，在此传入外部时间戳。

@bIsHevc，当前码流是否为 HEVC (H265) 码流，是则设置为 TRUE，H264 码流设置为 FALSE，请务必按实际情况准确设置。

返回值：无

8、发送音频数据

```
void SDTerminalP2P_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。内部将校验 ADTS 头合法性。

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模式，内部自动产生时间戳，此时本参数传入 0 即可。当用户调用 SDTerminalP2P_SetUseInternalTimeStamp API 来指定外部时间戳模式时，在此传入外部时间戳。

返回值：无

9、设置音视频传输参数

```
void SDTerminalP2P_SetTransParams(void* pTerminal, unsigned int unJitterBuffDelay,
FEC_REDUN_TYPE_P2P eFecRedunMethod, unsigned int unFecRedunRatio,
unsigned int unFecMinGroupSize, unsigned int unFecMaxGroupSize, BOOL bEnableAck);
```

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，此时可以获得最低延时。

@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。自动冗余度将根据网络情况自行调整冗余。固定冗余度则全程使用固定值。

@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基础值，根据网络情况调整。

@unFecMinGroupSize，为上行 FEC 分组的下限值，建议设置为 16。

@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。

@bEnableAck，是否启用 ACK 功能，当设置为 TRUE 时，走 ACK 模式。当设置为 FALSE 时走 NACK 模式。二者的区别见文档说明。

返回值：无

说明：注意，本函数需在 Online 之前调用。

10、获取当前 SDK 版本信息

```
UINT SDTerminalP2P_GetVersion(void* pTerminal);
```

参数：

@pTerminal, 模块指针

返回值： 获得当前 SDK 的版本信息

11、获取当前丢包率数据

```
void SDTerminalP2P_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： 无

说明： 上述值内部已经乘 100.0 转换为百分比

12、获取当前视频通道的实时 RTT

```
unsigned int SDTerminalP2P_GetCurrentRtt(void* pTerminal);
```

@pTerminal, 模块指针

返回值： 无

说明： 获得当前视频通道的实时 RTT 值

13、设置时间戳工作机制

```
void SDTerminalP2P_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 FALSE。

返回值: 无

说明: 本函数需在 Online 之前调用。不调用本函数时, 默认使用内部时间戳模式。

14、设置丢包冻结机制

```
void SDTerminalP2P_EnableFreezeFrameWhenLost (void* pTerminal,  
BOOL bEnable);
```

@pTerminal, 模块指针

@bEnable, 设置 TRUE 表示启用丢包冻结机制, 外层可只管解码, 可确保不会因丢包而露出花屏。设置 FALSE 表示关闭丢包冻结机制, 此时所有接收到的数据均返回外层, 同时提供丢包标识、关键帧标识。

返回值: 无

说明: 本函数需在 Online 之前调用。不调用本函数时, 默认开启丢包冻结机制。

15、设置发送端 Smooth 机制

```
void SDTerminalP2P_SetVideoFrameRateForSmoother(void* pTerminal,  
unsigned int unFrameRate);
```

@pTerminal, 模块指针

@unFrameRate, 设置视频帧率信息, 该值将作为内部发送时 Smoother 处理的参考。注意该帧率要符合实际帧率, 可以高于实际帧率, 但不能低于实际帧率, 否则将导致发送速度不足, 触发内部自行关闭 Smooth 功能。

返回值: 无

说明: 本函数需在 Online 之前调用。不调用本函数时, 默认关闭 smooth 处理。

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通知外层，SDK 提供了相关的回调函数设置接口。

1、设置视频数据接收回调

```
void SDTerminalP2P_SetRecvRemoteVideoCallback(void* pTerminal,  
RecvP2PRemoteVideoFunc 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。

RecvP2PRemoteVideoFunc 的定义如下：

```
void (*RecvP2PRemoteVideoFunc)(void* pObject, unsigned char* data, unsigned int unLen,  
unsigned int unPTS, VideoFrameInforP2P* pFrameInfo);
```

其中，VideoFrameInforP2P 提供了当前帧以及当前流的重要信息：

```
typedef struct VideoFrameInforP2P
```

```
{  
  
    unsigned int unWidth;  
  
    unsigned int unHeight;  
  
    unsigned int unFps;  
  
    BOOL bPacketLost;  
  
    BOOL bKeyFrame;  
  
    BOOL bInfoUpdated;  
  
    BOOL bIsHvc;  
  
    unsigned char bySps[512];  
  
    unsigned int unSpsSize;  
  
    unsigned char byPps[512];  
  
    unsigned int unPpsSize;  
  
}VideoFrameInforP2P;
```


模块内部会获得当前码流的宽、高、帧率、VPS（HEVC 时）、SPS、PPS 告知外层，当其中某些参数发生变更时将置位 bInfoUpdated 以通知外层。

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。默认情况下内部已开启丢包冻结机制，外层可忽略 bPacketLost 与 bKeyFrame。

当没有丢包发生时，本函数的输出与对方调用 SDTerminalP2P_SendVideoData 函数的输入完全一致。

返回值：无

说明：SDK 内部将在独立于网络接收线程之外的线程中调用本接口，所以外层可以将相对耗时的操作（比如解码）放置在此回调中。

2、设置音频数据接收回调

```
void SDTerminalP2P_SetRecvRemoteAudioCallback(void* pTerminal, RecvP2PRemoteAudioFunc  
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。

RecvP2PRemoteAudioFunc 的定义如下：

```
void (*RecvP2PRemoteAudioFunc)(void* pObject, unsigned char* data, unsigned int unLen,  
unsigned int unPTS, AudioFrameInforP2P* pFrameInfo);
```

其中，AudioFrameInforP2P 提供了当前帧以及当前流的重要信息：

```
typedef struct AudioFrameInforP2P
```

```
{
```

```
    unsigned int unCodecType;
```

```
    unsigned int unSampleRate;
```

```
    unsigned int unChannelNum;
```

```
    unsigned int unFrameNo;
```

```
    BOOL bInfoUpdated;
```

```
}AudioFrameInforP2P;
```

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（目前固定 O-AAC）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，所以外层可以将相对耗时的操作（比如解码）放置在此回调中。

3、设置远端请求 IDR 通知回调

```
void SDTerminalP2P_SetRemoteIdrRequestNotifyCallback(void* pTerminal,  
P2PRemoteIdrRequestNotifyFunc 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。

P2PRemoteIdrRequestNotifyFunc 的定义如下：

```
BOOL (*P2PRemoteIdrRequestNotifyFunc)(void* pObject);
```

当远端接收失败时，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。

4、设置码率自适应通知回调（一）

```
void SDTerminalP2P_SetAutoBitrateNotifyCallback(void* pTerminal,  
AUTO_BITRATE_TYPE_P2P eAutoBitrateMethod,  
P2PAutoBitrateNotifyFunc pfAutoBitrateNotifyCallback, void* pObject);
```

参数：

@pTerminal, 模块指针

@eAutoBitrateMethod, 码率自适应方法：P2P_AB_TYPE_DISABLE（关闭码率自适应，默认不调用本 API 时即关闭状态）、P2P_AB_TYPE_ADJUST_FRAME_FIRST（优先降低帧率，其次降低码率）、P2P_AB_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。

回调函数 P2PAutoBitrateNotifyFunc 的说明如下：

```
BOOL (*P2PAutoBitrateNotifyFunc)(void* pObject, BOOL bSelectOrDropFrame,  
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。

假设 unFrameSelectOrDropInterval 为 N，bSelectOrDropFrame 为 TRUE 时，每 N 帧“取”一帧编码发送；为 FALSE 时，每 N 帧“丢”一帧。

比如 bSelectOrDropFrame 为 FALSE，unFrameSelectOrDropInterval=2，表示每 2 帧丢 1 帧。

比如 bSelectOrDropFrame 为 TRUE，unFrameSelectOrDropInterval=3，表示每 3 帧取 1 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，本回调函数返回 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。

5、设置码率自适应通知回调（二）

```
void SDTerminalP2P_SetDropNextFrameNotifyCallback(void* pTerminal,  
DropNextFrameNotifyFunc pfDropNextFrameNotifyCallback,  
void* pObject);
```

参数：

@pTerminal，模块指针

@pfDropNextFrameNotifyCallback，回调函数指针。当内部码率自适应请求单次丢帧时，将触发本回调函数，单次丢帧不同于周期性丢帧，它仅执行本次丢帧动作，丢完即恢复之前帧率状态。码率自适应利用了人眼对瞬间帧率降低不敏感的特点，来缓解瞬间的发送压力，降低触发周期性丢帧或降码率的概率。

@pObject，透传指针，将透传给回调函数。

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。