组播 SDK(windows 版-内置编解码)

(V1.0.0)

一、重要说明

音视频组播发送端：作为组播源将自身音视频组播发送。

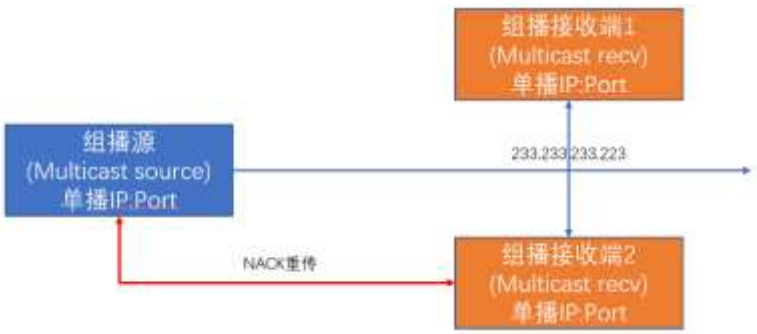
音视频组播接收端：接收音视频组播流。

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。

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（本 SDK 仅支持固定冗余度）、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 冗余度为基础，根据网络情况进行调整，对于网络较好的场合使用低冗余尽量降低带宽。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）。

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，缺点是画面仍可能因丢包而卡顿。

组播 NACK 支持：组播是一种单向的传输技术，正常情况下无法使用 NACK 重传机制，因为每一个接收端都可能有不同的丢包情况。我们使用单播作为组播的一种补充，从而实现了为不同组播接收端提供个性化的 NACK 服务。



Smooth 平滑：发送端对一帧较大的码流在时间窗口内平滑发送，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。

传输参数：本 SDK 中传输相关参数包括：视频通道的上行 FEC 冗余度、上行 FEC Group 分组大小、NACK 重传支持、接收端 JitterBuff 缓存时间。对于音视频组播接收端，同样可以设置上行 FEC 冗余度、上行 FEC Group 分组大小，只是没有实际意义(因为不会发送数据，当然也不会进行 FEC 编码)。

二、API 接口

所有 API 接口定义均位于 SDTerminalSdk.h 文件中。本 API 主要实现音视频编码、网络收发、音视频解码三大功能，同时提供了上下行码流的 TS 录制、统计信息获取等辅助功能。

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： DEBUG、INFO、WARNING、ERROR、ALARM、FATAL、NONE，当指定为 NONE 时，将不会生成日志文件。具体见 TERMINAL_LOG_OUTPUT_LEVEL 定义。

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_New(TerminalEncodeParams* ptEncodeParams, RemoteVideoYuvDataCallback
```

pfOutputVideo, RemoteAudioPcmDataCallback pfOutputAudio, void* pObject);

参数:

@ptEncodeParams，音视频编码相关参数，包括编码标准类型、是否采用硬编码、编码码率、编码帧率、编码分辨率、音频采样率、声道数等。

@ pfOutputVideo，视频接收解码输出回调函数。

@ pfOutputAudio，音频接收解码输出回调函数。

@ pObject，输出回调函数透传数据。

返回值:

对象指针，返回 NULL 表示失败。

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

返回值: 无

5、建立连接

```
BOOL  SDTerminal_Online(
    void* pTerminal,
    TerminalLogonParams* ptLogonParams,
    TerminalTransParams* ptTransParams);
```

参数:

@ pTerminal，模块指针

@ ptLogonParams，建立连接相关参数，包括客户端类型、组播 IP、组播端口、本地 IP、组播 NACK 端口。本地 IP 的指定用于避免在多网卡时系统默认 IP 不可控的问题。组播 NACK 端口用于单播方式传输 NACK 信令与媒体包，需配置为与组播源一致端口。

@ ptTransParams，音视频传输相关参数，包括 FEC 相关、接收缓存相关等。

返回值:

返回 TRUE 表示成功，返回 FALSE 则为失败。

6、断开连接

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

参数:

@ pTerminal, 模块指针

返回值: 无

7、编码并发送一帧视频数据

```
void  SDTerminal_SendVideoData(void* pTerminal, unsigned char* buf, unsigned int unWidth,
unsigned int unHeight, TERMINAL_VIDEO_INPUT_FORMAT eFormat);
```

参数:

@ pTerminal, 模块指针

@ buf, 输入图像存放区。

@ unWidth, 输入图像宽度, 当与编码宽高不一致时, 内部自行缩放。建议与编码宽高一致。

@ unHeight, 输入图像高度, 当与编码宽高不一致时, 内部自行缩放。建议与编码宽高一致。

@ eFormat, 输入图像格式, 建议与编码器当前使用的格式一致。

说明: 若采用硬编码, 外层可先通过 SDTerminal_GetHardwareEncInfo 接口获得当前硬编码支持的输入格式, 以此作为外层采集格式。这样避免内部再进行色度空间转换、缩放处理, 提高性能。

返回值: 无

8、编码并发送一帧音频数据

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

参数:

@pTerminal, 模块指针

@ buf, 输入音频 PCM 存放区。

@ unLen, 输入音频 PCM 字节数。

返回值: 无

9、获取硬编码支持情况

```
BOOL SDTerminal_GetHardwareEncInfo(void* pTerminal, BOOL* pbWillUseHwEnc,
```

TERMINAL_VIDEO_INPUT_FORMAT *peHwRecomInputFormat);

参数:

@ pTerminal, 模块指针

@ pbWillUseHwEnc, 是否将最终采用硬编码。用户在 Online 接口中启用硬编码, 当最终是否采用将取决于当前机器设备是否支持, 不支持时将自动使用软编码。

@ peHwRecomInputFormat, 若最终采用硬编码, 返回硬编码器支持的输入格式。

说明: 需在 Online 接口调用成功后, 才能有效调用本 API。

返回值: 返回 TRUE 表示成功, 返回 FALSE 则为失败。

10、设置上行码流录制

BOOL SDTerminal_StartCapUpStream(void* pTerminal, const char* strTsFileDir);

void SDTerminal_StopCapUpStream(void* pTerminal);

参数:

@ pTerminal, 模块指针

@ strTsFileDir, 上行录制 TS 文件名, 需带完整路径。

说明: SDTerminal_StartCapUpStream 需在 Online 接口成功后调用生效。

返回值: 返回 TRUE 表示成功, 返回 FALSE 则为失败。

11、设置下行码流录制

BOOL SDTerminal_StartCapDownStream(void* pTerminal, unsigned int unAvDownIndex, const char* strTsFileDir);

void SDTerminal_StopCapDownStream(void* pTerminal, unsigned int unAvDownIndex);

参数:

@ pTerminal, 模块指针

@ unAvDownIndex, 需要录制的下行位置。

@ strTsFileDir, 下行录制 TS 文件名, 需带完整路径。

说明: SDTerminal_StartCapDownStream 需在 Online 接口成功后调用生效。

返回值: 返回 TRUE 表示成功, 返回 FALSE 则为失败。

12、获取当前 SDK 版本信息

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

参数:

@ pTerminal, 模块指针

返回值: 获得当前 SDK 的版本信息

13、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