客户端 SDK(Java 版)说明(V1.0.0)

www.mediapro.cc

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客户端，一个接收者客户端。

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 JitterBuff 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**未找到目录项。**接收者同样也可以设置 FEC 上行冗余度、上行 FEC Group 组大小，只是没有实际意义。

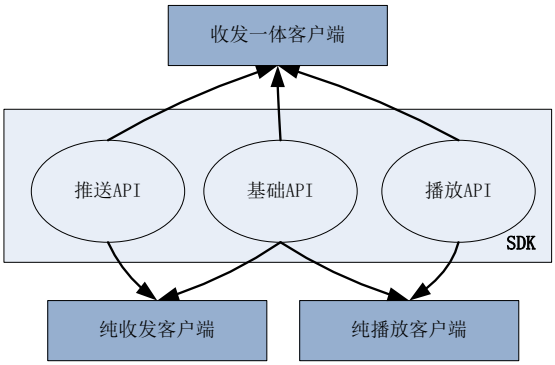
二、SDK 的组成

SDK 由如下三类 API 组成：

基础 API 接口，负责基础的 TCP 连接管理、音视频码流收发。如登录、下线、发送音视频等。

推送 API 接口，负责摄像头、麦克风的采集、滤镜、压缩编码等处理。

播放 API 接口，负责播放远端音视频。



基础 SDK 包括所有类别客户端均需调用的基础功能,纯发送型客户端无需调用播放 API，纯播放型客户端无需调用推送 API。

三、基础 API

以下接口均为线程安全，可以在多线程环境中调用，定义位于 CSDInterface.java 中。

1、系统初始化

系统初始化主要完成日志模块初始化、服务器 IP 地址配置，该 API 在整个系统中只需要调用一次即可。

int SDsysinit(String strServerIp, String strLogFileDir, byte byLogFileLevel)

参数:

@strServerIp，指定服务器 IP 地址

@strLogFileDir，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。

@byLogFileLevel，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。级别定义位于 Constant.java:

```
final byte LOG_LEVEL_DEBUG = 1;
final byte LOG_LEVEL_INFO = 2;
final byte LOG_LEVEL_WARN = 3;
final byte LOG_LEVEL_ERROR = 4;
final byte LOG_LEVEL_ALARM = 5;
final byte LOG_LEVEL_FATAL = 6;
final byte LOG_LEVEL_NONE = 7;
```

当指定为 SD_LOG_LEVEL_NONE 时，将不会生成日志文件。

返回值：返回 0 表示初始化成功，返回负数则为失败，负数值为其错误码。

2、系统反初始化

void SDsysexit()

与 SDsysinit 对应，系统退出前调用。

参数：无

返回值：无

3、登录服务器

int SDOnlineUser(int nRoomId, int nUserId, byte byUserType, int nDomainId);

参数：

@nRoomId，表示当前客户端进入的房间号，要求非 0

@nUserId，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

@byUserType，表示客户端的类型，定义位于 Constant.java：

```
final byte USER_TYPE_OTHER = 0;              //保留未使用的类型
final byte USER_TYPE_AV_SEND_RECV = 1;  //同时进行音视频发送和接收
final byte USER_TYPE_AV_RECV_ONLY = 2;  //仅接收音视频
final byte USER_TYPE_AV_SEND_ONLY = 3;  //仅发送音视频
```

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。

@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。

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。

4、下线服务器

void SDOfflineUser();

参数：无

返回值：无

5、请求上传音视频到指定位置

int SDOnPosition(byte byPosition);

参数：

@byPosition，大于等于 0，小于系统支持的最大位置数（宏 MAX_SUPPORT_DECODE_NUM 定义，目前配置为 6）。特殊值：当设置为 255 时表示由服务器分配当前空闲的位置。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 SDOnlineUser 成功后调用。

6、请求从位置上下来

void　SDOffPosition();

参数：无

返回值：无

说明：对于当前在位置上的音视频发布者，SDOffPosition 将从位置上下来并停止上行音视频数据，但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所在位置的音视频流（可能有其他客户端加入该位置）。

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

7、上传视频数据

void　SDSendVideoStreamData(byte[] byBuf,　int nlen,　int nPts);

向请求的位置发送视频码流，一次传入带 H264 起始码的一帧码流。本 API 配合推送 API 使用，具体见推送 API 说明。

参数：

@byBuf，码流存放区。

@nlen，码流长度。

@unDts，可以使用外层时间戳，也可以使用默认值 0，此时将由模块内部自行管理时间戳。内部将使用 1KHZ 时基（毫秒）生成时间戳。建议使用默认值 0。

返回值：无

8、上传音频数据

void SDSendAudioStreamData(byte[] byBuf, int nlen, int nPts);

向请求的位置发送音频码流，一次传一帧 ADTS 码流。本 API 配合推送 API 使用，具体见推送 API 说明。

参数：

@byBuf，码流存放区。

@nlen，码流长度。

@unDts，可以使用外层时间戳，也可以使用默认值 0，此时将由模块内部自行管理时间戳。内部将使用 1KHZ 时基（毫秒）生成时间戳。建议使用默认值 0。

返回值：无

9、设置音视频下行掩码

void SDSetAvDownMasks(int nAudioDownMask, int nVideoDownMask);

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;
```

SDSetAvDownMasks(unAudioDownChannelsMask, unVideoDownChannelsMask);

当本客户端仅发送音视频，不接收音视频时，设置接收掩码为 0 即可。

参数：

@nAudioDownMask，控制音频接收的掩码。

@nVideoDownMask，控制视频接收的掩码。

返回值：无

说明：请在 Online 成功后调用。

10、设置音视频传输参数

void　SDSetTransParams(int nRedunRatio, int nGroupSize, int nEnableNack, int nJitterBufTime);

参数：

@nRedunRatio，固定冗余度时对应的上行冗余比率，比如设置为 30，则表示使用 30%冗余。当设置为 0 时表示使用 AUTO_REDUN 自动冗余度。

@nGroupSize，　为上行 FEC 分组大小，512Kbps 以下建议设置为 8，512Kbps~1Mbps 建议设置为 16，1Mbps~2Mbps 建议设置 24，2Mbps 以上建议设置 28。

@nEnableNack，是否启用 NACK 功能，关于 NACK 请阅读相关文档，建议设置为 1 开启。

@nJitterBufTime，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。对延时无过高要求的场景下建议设置 200ms+。

返回值：无

说明：本函数需在 SDOnlineUser 之前调用，本 API 的使用若有疑问，请联系技术支持获得帮助。

11、获取当前统计数据

MediaTransStatis　SDGetMediaTransStatis(MediaTransStatis transStatis)

参数：

@transStatis，底层 C++获取传输统计数据后将写入本对象。MediaTransStatis 的定义请参考 MediaTransStatis.java

返回值：获取到的当前统计数据

说明：使用举例

```
private MediaTransStatis mTransStatis = new MediaTransStatis();
mTransStatis = mInterface.SDGetMediaTransStatis(mTransStatis);
Log.d(TAG,
"DownBitrate:" + mTransStatis.nVideoDownRate +
"DownLostratio:"+ mTransStatis.nVideoDownLostRatio);
```

四、推送 API

推送 API 定义位于 SDInterfaceCameraPublisher.java 中。SDK 通过单实例方式调用推送 API，形式如下：SDInterfaceCameraPublisher.Inst().API()。一个 APP 中只允许有一个推送实例。

1、 推送初始化

void Init(SDCameraView view, boolean sendAudioOnly)

参数：

@ view，指向用于本地摄像头视频渲染的 view，

view = (SDCameraView)findViewById(R.id.camera);

其中 R.id.camera 定义于 layout xml，例如：

```
<com.sd.SDCameraView
    android:id="@+id/camera"
    android:layout_width="0dp"
    android:layout_height="match_parent"/>
```

@ sendAudioOnly，是否仅发送音频，为 true 时表示纯音频发送模式，此时第一个参数 view 设置 null。

返回值：无

说明：需要在调用基础 API SDsysinit 之后调用本 API。

2、 推送反初始化

void Destroy()

参数：无

返回值：无

3、 设置编码码流输出回调接口

void setSendHandler(SDPublishHandler handler)

参数：

@ handler，码流输出处理类对象。

返回值：无

说明：本 API 用于指定经内部编码压缩后的码流交外层应用处理的回调接口。注意：外层应

用需要在回调接口中自行调用基础 API 中的音视频发送接口，我们没有将这一过程封装到 SDK 内部完成，也是考虑到某些特殊场景下，外层应用希望对发往服务端的码流进行某些特定处理。一种典型的调用示意：

外层实现 SDPublishHandler.SendListener 接口重载

```
public class MainActivity implements SDPublishHandler.SendListener
{
    @Override
    public void onSendVideoStreaming(byte[] buffer,  int size) {
        mInterface.SDSendVideoStreamData(buffer, size, 0);
    }
    @Override
    public void onSendAudioStreaming(byte[] buffer,  int size) {
        mInterface.SDSendAudioStreamData(buffer, size, 0);
    }
    SDInterfaceCameraPublisher.Inst().setSendHandler(new SDPublishHandler(this));
}
```

4、 设置摄像头采集宽高

```
boolean setPreviewResolution(int width,  int height)
```

参数：

@ width，请求采集的宽度。

@ height，请求采集的高度。

返回值：true 成功，false 失败

5、 设置视频编码宽高

```
boolean setOutputResolution(int width,  int height)
```

参数：

@ width，视频编码的宽度。

@ height，视频编码的高度。

返回值：true 成功，false 失败

说明：当采集宽高与编码宽高不等时，内部将自行缩放处理。当二者宽高比不一致时，将剪裁到编码宽高比。

6、 设置视频编码码率和帧率

boolean setVideoEncParams(int bitrate, int framerate)

参数：

@ bitrate，视频编码的码率，单位 bps。

@ framerate，视频编码的帧率，如 30 表示 30fps。

返回值：true 成功，false 失败

7、 设置使用 H264 硬编码

void switchToHardEncoder()

参数：无

返回值：无

说明：外层未指定编码器类型时，内部默认按硬编码创建，若硬编码创建失败则自动转为软编码。当使用硬编码时，要求编码宽高均为 32 的倍数。

8、 设置使用 H264 软编码

void switchToSoftEncoder()

参数：无

返回值：无

9、 开始推送音视频流

boolean startPublish()

参数：无

返回值：true 成功，false 失败

说明：需在已经登录的状态下调用，否则将不会生效。调用本 API 后内部将启动音视频的采集和编码。

10、 停止推送

boolean stopPublish()

参数： 无

返回值： true 成功，false 失败

11、 设置摄像头滤镜

boolean switchCameraFilter(MagicFilterType type)

参数：

@ type，目前可选的滤镜类型 MagicFilterType.BEAUTY、MagicFilterType.COOL 等。

返回值： true 成功，false 失败

说明： 未调用本 API 时，默认情况下无滤镜效果。

12、 切换前后置摄像头

boolean switchCameraFace(int id)

参数：

@ id，需要切换到的摄像头的 ID 号。

返回值： true 成功，false 失败

说明： 通常以如下方式实现摄像头的切换：

currentCameraId = SDInterfaceCameraPublisher.Inst().getCamraId()

nextCameraId = (currentCameraId + 1) % Camera.getNumberOfCameras()

SDInterfaceCameraPublisher.Inst().switchCameraFace(nextCameraId);

五、播放 API

播放 API 定义位于 SDInterfacePlayer.java 中。一个 APP 中可以创建多个播放实例实现多路播放功能。

1、播放初始化

void Init(Activity act, SurfaceViewRenderer surfaceView, boolean playAudioOnly, boolean playVideoOnly)

参数：

@ act，播放实例所在的 Activity。

@ surfaceView，画面渲染的 surface view。

@ playAudioOnly，是否仅播放音频。仅播放音频时 surfaceView 可设置为 null

@ playVideoOnly，是否仅播放视频。

返回值：无

说明：需要在调用基础 API　SDsysinit 之后调用本 API。

2、播放反初始化

void Destroy()

参数：无

返回值：无

3、开始播放

void startPlay(int index,　int renderWidth,　int renderHeight)

参数：

@ index，播放的音视频位置索引。

@ renderWidth，画面渲染窗口的宽度。

@ renderHeight，画面渲染窗口的高度。

返回值：无

说明：无需提供精确的 renderWidth 和 renderHeight，只需要二者比率与 surfaceView 宽高比一致即可。当然若能获得精确宽高最佳。

4、停止播放

void stopPlay()

参数：无

返回值：无