屏幕共享 SDK(Java 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: 同一个房间内，允许存在多个音视频发布者发布音视频到不同的“位置”上，接收者可选择接收某个或多个位置的音视频。目前单个房间理论最多允许 32 个“位置”，实际数目由具体项目需求而定。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 时间等，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 的使用说明

屏幕录制 SDK 以 SDInterfaceScreenPublishService 这个 Service 服务形式对外提供屏幕录制、麦克风采集服务。外层需要负责：服务到 AndroidManifest 的注册、相关权限的申请、

MediaProjection 对 VirtualDisplay 的获取、省电白名单的加入，这部分可以参考 DEMO 源码实现。

三、SDK 调用步骤

1、 准备启动 service 所需的 Intent

```
mIntentForService.putExtra("SERVER_IP", serverIp);
mIntentForService.putExtra("ROOM_ID", roomId);
mIntentForService.putExtra("UP_POSITION", upPosition);
mIntentForService.putExtra("ENABLE_AUDIO", ENABLE_AUDIO_CAP);

mIntentForService.putExtra("ENCODE_VIDEO_WIDTH", width);
mIntentForService.putExtra("ENCODE_VIDEO_HEIGHT", height);
mIntentForService.putExtra("ENCODE_VIDEO_FRAMERATE",framerate);
mIntentForService.putExtra("ENCODE_VIDEO_BITRATE", bitrate);
mIntentForService.putExtra("ENCODE_VIDEO_USEHW", useHardwareEnc);

mIntentForService.putExtra("ENCODE_AUDIO_SAMPLERATE", sampleRate);
mIntentForService.putExtra("ENCODE_AUDIO_CHANNELS", channelCount);
```

参数:

@ SERVER_IP，指定服务器 IP 地址

@ROOM_ID，指定发布的房间号（非 0 值，与播放端约定）。

@UP_POSITION，指定发布到房间中的哪一个“位置”。

@ENABLE_AUDIO，是否发布音频流

@ENCODE_VIDEO_WIDTH，发布视频流的宽度，比如 1920

@ENCODE_VIDEO_HEIGHT，发布视频流的高度，比如 1080

@ENCODE_VIDEO_FRAMERATE，发布视频流的帧率，当设置的帧率超过 30fps 时，内部将使用高帧率模式，码率波动加剧。

@ENCODE_VIDEO_BITRATE，发布视频流的码率，注意其单位为 bps

@ENCODE_VIDEO_USEHW，是否使用视频硬编码，因为屏幕共享分辨率往往较高，建议使用硬编码以保证性能。

@ENCODE_AUDIO_SAMPLERATE，发布音频流的采样率，比如 44100

@ENCODE_AUDIO_CHANNELS，发布音频流的声道数，比如 2 即双声道立体声。

2、 准备绑定 service 所需的 ServiceConnection

我们通过 bind 服务，实现对服务 API 的直接调用。目前服务对外提供 2 个 API，分别为：

（1） `void setScreenCapStatusCallback(ScreenCapStatusCallback callback)`

设置屏幕录制服务的状态回调，状态回调包括三种状态:OnStart、OnRecording、OnStop，其中前两者可用于外层的界面展示更新，比如展示已共享的时长。OnStop 则用于通知外层需结束服务，此时一般发生了内部错误。

（2） `void setScreenCapSource(VirtualDisplay virtualDisplay)`

设置屏幕录制源 VirtualDisplay，该录制源由外层通过 MediaProjection 系统 API 获得。

3、绑定并启动服务

调用 Android 系统 API 绑定并启动服务，注意二者顺序不能错误，即先绑定后启动。

```
bindService(mIntentForService, mServiceConnection, Context.BIND_AUTO_CREATE);
startService(mIntentForService);
```

4、结束并解除绑定服务

调用 Android 系统 API 结束服务并解除绑定。

```
stopService(mIntentForService);
unbindService(mServiceConnection);
```

结束服务有两种情况下触发，第一种是用户 UI 操作结束分享；第二种是分享失败，服务内部申请节省，此时将通过回调 OnStop 接口触发。