1 功能概述

推流和连麦模块是手机开播场景中的基础功能，支持摄像头参数设置、音频参数设置、视频参数设置等。推流可以用于讲师上课，连麦支持音频连麦、视频连麦可自由选择，并支持多人连麦，丰富课堂内容。

推流流程：初始化推流引擎->设置推流配置参数->开始推流。

连麦流程：讲师端发起连麦->学员举手申请连麦->讲师端允许学员连麦->学员正式加入连麦。

2 使用演示

以下示例了如何创建推流和连麦布局，以及如何调用初始化方法，代码如下：

```
// 推流和连麦布局
private IPLVLSStreamerLayout plvlsStreamerLy;
// findView
plvlsStreamerLy = findViewById(R.id.plvls_streamer_ly);
// 初始化推流和连麦布局
plvlsStreamerLy.init(liveRoomDataManager);
```

上述方法的具体用例可以在polyv demo项目中的`PLVLSLiveStreamerActivity`找到。

3 接口介绍

3.1 IPLVLSStreamerLayout

手机开播场景下，针对推流和连麦布局进行封装的接口，定义了外部直接调用的方法。

```
public interface IPLVLSStreamerLayout {

    /**
     * 初始化
     *
     * @param liveRoomDataManager 直播间数据管理器
     */
    void init(IPLVLiveRoomDataManager liveRoomDataManager);

    /**
     * 开始上课
     */
    void startClass();

    /**
     * 停止上课
     */
    void stopClass();

    /**
     * 设置推流码率
     *
     * @param bitrate 码率
     */
    void setBitrate(@PLVSStreamerConfig.BitrateType int bitrate);

    /**
     * 获取推流的码率信息
     *
     * @return 码率信息<最大支持码率, 选择码率>
     */
    Pair<Integer, Integer> getBitrateInfo();

    /**
     * 是否允许录制声音
     *
     * @param enable true：允许，false：不允许
     */
    boolean enableRecordingAudioVolume(boolean enable);

    /**
     * 是否允许录制视频/打开摄像头
     *
     * @param enable true：允许，false：不允许
     */
    boolean enableLocalVideo(boolean enable);

    /**
     * 设置相机方向
     *
     * @param front true：前置，false：后置
     */
    boolean setCameraDirection(boolean front);

    /**
```

```
     * 控制成员列表中的用户加入或离开连麦
     *
     * @param position    列表中的位置
     * @param isAllowJoin true：加入，false：离开
     */
    void controlUserLinkMic(int position, boolean isAllowJoin);

    /**
     * 禁/启用用户媒体
     *
     * @param position    成员列表中的位置
     * @param isVideoType true：视频，false：音频
     * @param isMute      true：禁用，false：启用
     */
    void muteUserMedia(int position, boolean isVideoType, boolean isMute);

    /**
     * 下麦全体连麦用户
     */
    void closeAllUserLinkMic();

    /**
     * 全体连麦用户禁用/开启声音
     *
     * @param isMute true：禁用，false：开启
     */
    void muteAllUserAudio(boolean isMute);

    /**
     * 添加推流状态的监听器
     *
     * @param listener 监听器
     */
    void addOnStreamerStatusListener(IPLVOnDataChangedListener<Boolean> listener);

    /**
     * 添加网络状态监听器
     *
     * @param listener 监听器
     */
    void addOnNetworkQualityListener(IPLVOnDataChangedListener<Integer> listener);

    /**
     * 添加推流时间监听器
     *
     * @param listener 监听器
     */
    void addOnStreamerTimeListener(IPLVOnDataChangedListener<Integer> listener);

    /**
     * 因断网延迟20s断流的状态
     *
     * @param listener 监听器
     */
    void addOnShowNetBrokenListener(IPLVOnDataChangedListener<Boolean> listener);
```

```
    /**
     * 添加音频状态监听器
     *
     * @param listener 监听器
     */
    void addOnEnableAudioListener(IPLVOnDataChangedListener<Boolean> listener);

    /**
     * 添加视频状态监听器
     *
     * @param listener 监听器
     */
    void addOnEnableVideoListener(IPLVOnDataChangedListener<Boolean> listener);

    /**
     * 添加摄像头方向监听器
     *
     * @param listener 监听器
     */
    void addOnIsFrontCameraListener(IPLVOnDataChangedListener<Boolean> listener);

    /**
     * 获取网络质量
     *
     * @return 网络质量常量
     */
    int getNetworkQuality();

    /**
     * 是否推流开始成功
     *
     * @return true：成功，false：未成功
     */
    boolean isStreamerStartSuccess();

    /**
     * 获取推流和连麦的presenter
     *
     * @return presenter
     */
    IPLVStreamerContract.IStreamerPresenter getStreamerPresenter();

    /**
     * 销毁，释放资源
     */
    void destroy();
}
```

4 实现介绍

4.1 PLVLSStreamerLayout

该类是手机开播场景下的推流和连麦布局，实现 `IPLVLSStreamerLayout` 接口。

该布局包含讲师的推流摄像头画面，音频开关状态，讲师头衔、昵称以及其他已连麦的观众的相关信息的UI。

下面会列举介绍该布局中涉及到的主要的方法。

4.1.1 初始化view方法

`PLVLSStreamerLayout`继承于`FrameLayout`，在构造器中使用`initView`方法对`view`进行了初始化处理。

```
public PLVLSStreamerLayout(@NonNull Context context, @Nullable AttributeSet attrs, int defStyleAttr) {
    super(context, attrs, defStyleAttr);
    initView();
}

private void initView() {
    //填充推流和连麦布局到该view中
    LayoutInflater.from(getContext()).inflate(R.layout.plvls_streamer_layout, this);
    //推流和连麦的列表视图
    plvlsStreamerRv = findViewById(R.id.plvls_streamer_rv);

    //init RecyclerView
    plvlsStreamerRv.setLayoutManager(new LinearLayoutManager(getContext(), RecyclerView.VERTICAL, false));
    plvlsStreamerRv.addItemDecoration(new PLVMessageRecyclerView.SpacesItemDecoration(ConvertUtils.dp2px(8),
0));
    //禁用RecyclerView默认动画
    plvlsStreamerRv.getItemAnimator().setAddDuration(0);
    plvlsStreamerRv.getItemAnimator().setChangeDuration(0);
    plvlsStreamerRv.getItemAnimator().setMoveDuration(0);
    plvlsStreamerRv.getItemAnimator().setRemoveDuration(0);
    RecyclerView.ItemAnimator rvAnimator = plvlsStreamerRv.getItemAnimator();
    if (rvAnimator instanceof SimpleItemAnimator) {
        ((SimpleItemAnimator) rvAnimator).setSupportsChangeAnimations(false);
    }

    //init adapter
    streamerAdapter = new PLVLSStreamerAdapter(plvlsStreamerRv, new
PLVLSStreamerAdapter.OnStreamerAdapterCallback() {
        @Override
        public SurfaceView createLinkMicRenderView() {
            return streamerPresenter.createRenderView(Utils.getApp());
        }

        @Override
        public void setupRenderView(SurfaceView surfaceView, String linkMicId) {
            streamerPresenter.setupRenderView(surfaceView, linkMicId);
        }
    });

    //启动前台服务
    PLVLSForegroundService.startService();
}
```

4.1.2 初始化数据方法

`PLVLSStreamerLayout` 的 `init` 方法是对外API，需要外部传入 `IPLVLiveRoomDataManager` 后进行初始化。

```
//推流和连麦presenter
private IPLVStreamerContract.IStreamerPresenter streamerPresenter;

@Override
public void init(IPLVLiveRoomDataManager liveRoomDataManager) {
    this.liveRoomDataManager = liveRoomDataManager;
    //创建推流和连麦mvp-presenter
    streamerPresenter = new PLVStreamerPresenter(liveRoomDataManager);
    //注册推流和连麦mvp-view
    streamerPresenter.registerView(streamerView);
    //初始化推流和连麦mvp-presenter
    streamerPresenter.init();
}

//创建推流和连麦的mvp-view
private PLVAbsStreamerView streamerView = new PLVAbsStreamerView() {
    //...
}
```

这里的 `PLVStreamerPresenter` 是使用 `mvp` 模式封装的推流和连麦 `presenter`，用 `presenter` 将 `view` 和 `mode` 隔离开来，一切业务逻辑都是通过 `presenter` 来进行操作，也就是说 `presenter` 是视图的数据的桥梁，视图和数据相隔两端。

4.1.3 开始上课方法

`PLVLSStreamerLayout` 的 `startClass` 方法是对外API，该方法内部会调用 `PLVStreamerPresenter` 的 `startLiveStream` 方法来进行推流到登录的频道。

```
@Override
public void startClass() {
    streamerPresenter.startLiveStream();
}
```

`PLVLSStreamerLayout` 内部持有 `PLVStreamerPresenter`，因此除了内部可以自由调用推流和连麦相关的方法外，也能将其封装成接口提供给外层调用，`startClass` 方法就是其中一个例子。

4.1.4 停止上课方法

停止上课方法为 `stopClass`，该方法是对外API，内部会调用 `PLVStreamerPresenter` 的 `stopLiveStream` 方法停止推流。

```
@Override
public void stopClass() {
    streamerPresenter.stopLiveStream();
}
```

5 子目录介绍

这里的子目录指的是手机开播场景推流和连麦模块下的子目录，对应项目中`polyvLiveStreamerScene`模块包名为`com.easefun.polyv.livestreamer.modules.streamer`下的子包。

5.1 adapter目录

```
//推流和连麦列表适配器
PLVLSStreamerAdapter
```

该目录主要是放置推流和连麦功能的一些列表相关的适配器。

5.2 service目录

```
//前台服务。防止华为机型上锁屏超过1分钟导致摄像头预览画面卡住和断流
PLVLSForegroundService
```

该目录主要是放置推流和连麦功能使用到的一些服务。