1 功能概述

开播场景的推流和连麦模块位于`polyvLiveCommonModul`模块的`streamer`包下，推流和连麦的信息接收和发送依赖于socket，即要用`PLVSocketLoginManager`完成登录后，才能正常地使用推流和连麦的功能。推流和连麦模块采用mvp设计模式，将view层的和model层分离，将多个场景中对推流和连麦的demo调用的共用代码抽离封装到presenter层中，使页面的view层的逻辑变得简洁，封装的presenter层负责`streamerManager`的调用以及通知view层更新UI。

2 MVP模式

推流和连麦的mvp模式是用`IPLVStreamerContract`作为契约类，内部定义了推流和连麦的mvp-view接口`IStreamerView`，以及推流和连麦的mvp-presenter接口`IStreamerPresnter`。

```
public interface IPLVStreamerContract {

    // <editor-fold defaultstate="collapsed" desc="1、mvp-推流和连麦view层接口">

    /**
     * mvp-推流和连麦view层接口
     */
    interface IStreamerView {
        /**
         * 设置presenter后的回调
         */
        void setPresenter(@NonNull IStreamerPresenter presenter);

        /**
         * 推流引擎创建成功，View层应创建连麦适配器和初始化连麦布局
         *
         * @param linkMicUid  我的连麦id
         * @param linkMicList 连麦列表。在实现中，要用该列表去渲染
         */
        void onStreamerEngineCreatedSuccess(String linkMicUid, List<PLVLinkMicItemDataBean> linkMicList);

        /**
         * 响应用户开关视频
         *
         * @param uid             用户id
         * @param mute            true表示关闭视频，false表示开启视频
         * @param streamerListPos 推流和连麦列表中的位置
         * @param memberListPos   成员列表中的位置
         */
        void onUserMuteVideo(final String uid, final boolean mute, int streamerListPos, int memberListPos);

        /**
         * 响应用户开关音频
         *
         * @param uid             用户id
         * @param mute            true表示关闭音频，false表示开启音频
         * @param streamerListPos 列表中的位置
         * @param memberListPos   成员列表中的位置
         */
        void onUserMuteAudio(final String uid, final boolean mute, int streamerListPos, int memberListPos);

        /**
         * 响应本地用户麦克风音量变化
         */
        void onLocalUserMicVolumeChanged();

        /**
         * 响应远端用户麦克风音量变化
         */
        void onRemoteUserVolumeChanged(List<PLVMemberItemDataBean> linkMicList);

        /**
         * 响应用户加入连麦频道
         *
         * @param uids
```

```
     */
    void onUsersJoin(List<String> uids);

    /**
     * 响应用户离开连麦频道
     *
     * @param uids
     */
    void onUsersLeave(List<String> uids);

    /**
     * 推流网络变化
     *
     * @param quality 网络状态常量
     */
    void onNetworkQuality(int quality);

    /**
     * 更新推流时间
     *
     * @param secondsSinceStartTiming 推流时间，单位：秒
     */
    void onUpdateStreamerTime(int secondsSinceStartTiming);

    /**
     * 因断网延迟20s断流
     */
    void onShowNetBroken();

    /**
     * 当前为结束推流状态
     */
    void onStatesToStreamEnded();

    /**
     * 当前为成功推流状态
     */
    void onStatesToStreamStarted();

    /**
     * 响应推流和连麦错误
     */
    void onStreamerError(int errorCode, Throwable throwable);

    /**
     * 更新成员列表数据
     *
     * @param dataBeanList 成员列表。在实现中，要用该列表去渲染
     */
    void onUpdateMemberListData(List<PLVMemberItemDataBean> dataBeanList);

    /**
     * 相机方向改变
     *
     * @param front true：前置，false：后置
     * @param pos   成员列表中的位置
```

```
         */
        void onCameraDirection(boolean front, int pos);

        /**
         * 更新成员列表中的socket用户信息
         *
         * @param pos 成员列表中的位置
         */
        void onUpdateSocketUserData(int pos);

        /**
         * 添加成员列表数据
         *
         * @param pos 成员列表中的位置
         */
        void onAddMemberListData(int pos);

        /**
         * 移除成员列表数据
         *
         * @param pos 成员列表中的位置
         */
        void onRemoveMemberListData(int pos);

        /**
         * 达到最大连麦人数的回调，在达到最大连麦人数后再同意用户连麦时触发
         */
        void onReachTheInteractNumLimit();

        /**
         * 用户请求连麦触发
         *
         * @param uid 连麦的id
         */
        void onUserRequest(String uid);
    }
    // </editor-fold>

    // <editor-fold defaultstate="collapsed" desc="2、mvp-推流和连麦presenter层接口">

    /**
     * mvp-推流和连麦presenter层接口
     */
    interface IStreamerPresenter {
        /**
         * 注册view，可以注册多个
         */
        void registerView(@NonNull IStreamerView v);

        /**
         * 解除注册的view
         */
        void unregisterView(IStreamerView v);

        /**
         * 初始化推流和连麦配置
```

```
     */
    void init();

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

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

    /**
     * 获取设置的推流码率
     *
     * @return 设置的推流码率
     */
    int getBitrate();

    /**
     * 获取最大支持的推流码率
     *
     * @return 最大支持的推流码率
     */
    int getMaxBitrate();

    /**
     * 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);

    /**
     * 创建渲染器
     *
     * @param context 上下文
```

```
     * @return 渲染器
     */
    SurfaceView createRenderView(Context context);

    /**
     * 为特定的连麦ID的用户设置连麦渲染器
     *
     * @param renderView 渲染器
     * @param linkMicId  连麦ID
     */
    void setupRenderView(SurfaceView renderView, String linkMicId);

    /**
     * 开始推流
     */
    void startLiveStream();

    /**
     * 停止推流
     */
    void stopLiveStream();

    /**
     * 控制成员列表中的用户加入或离开连麦
     *
     * @param position    成员列表中的位置
     * @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);

    /**
     * 请求成员列表数据
     */
    void requestMemberList();
```

```
        /**
         * 获取推流状态
         *
         * @return 推流状态常量
         */
        int getStreamerStatus();

        /**
         * 获取推流和连麦的数据
         */
        @NonNull
        PLVStreamerData getData();

        /**
         * 销毁，包括销毁推流和连麦操作、解除view操作
         */
        void destroy();
    }
    // </editor-fold>
}
```

2.1 Presenter实现逻辑

`IStreamerPresenter` 的实现类为 `PLVStreamerPresenter` 。

其内部主要是使用以下两个类来完成核心业务：

1、`IPLVLiveRoomDataManager`，直播间数据管理器

2、`IPLVSStreamerManager`，sdk层的推流和连麦管理器 `streamerManager`

2.1.1 初始化

`PLVStreamerPresenter` 在构造器中进行了如下的初始化操作：引用 `IPLVLiveRoomDataManager`，创建 `PLVStreamerData`，初始化创建推流和连麦管理器，开启推流和连麦的相关信息监听器。

```
public PLVStreamerPresenter(IPLVLiveRoomDataManager liveRoomDataManager) {
    this.liveRoomDataManager = liveRoomDataManager;
    streamerData = new PLVStreamerData();

    String viewerId = liveRoomDataManager.getConfig().getUser().getViewerId();
    PolyvLinkMicConfig.getInstance().init(viewerId, true);//需先初始化，再创建manager
    streamerManager = PLVSStreamerManagerFactory.createNewStreamerManager();

    streamerMsgHandler = new PLVStreamerMsgHandler(this);
    streamerMsgHandler.run();
}
```

`PLVStreamerData` 是推流和连麦的业务数据，内部保存的数据都是 `MutableLiveData` 类型，主要是用于提供给其他的view获取/监听推流和连麦的数据，可以用作于不同业务模块间的数据监听及获取。

`PolyvLinkMicConfig.getInstance().init` 方法是初始化连麦的数据。

`PLVSStreamerManagerFactory.createNewStreamerManager()` 方法内部会创建一个推流和连麦的管理器实例并返回。

`PLVStreamerMsgHandler` 是推流和连麦的信息处理器，主要是负责监听socket和rtc的相关推流和连麦的事件，并做相应的逻辑处理。

2.1.2 注册mvp-view

在ui层创建好mvp-view后，通过调用mvp-presenter的 `regitsterView` 方法，即可把mvp-view传给mvp-presenter，这样mvp-presenter即可通知mvp-view进行一些ui更新等操作；同时mvp-presenter也会把自身传给mvp-view，使得mvp-view可以调用mvp-presenter提供的方法。

```
@Override
public void registerView(@NonNull IPLVStreamerContract.IStreamerView v) {
    if (iStreamerViews == null) {
        iStreamerViews = new ArrayList<>();
    }
    if (!iStreamerViews.contains(v)) {
        iStreamerViews.add(v);
    }
    v.setPresenter(this);
}
```

`PLVStreamerPresenter` 可以注册多个 `IStreamerView` 。

3 SDK核心类介绍

推流和连麦的SDK核心类是 `IPLVSStreamerManager` ，该类是推流和连麦的管理器接口。

3.1 创建

`IPLVSStreamerManager` 的实例可以通过 `PLVSStreamerManagerFactory.createNewStreamerManager` 方法创建。

```
IPLVSStreamerManager streamerManager = PLVSStreamerManagerFactory.createNewStreamerManager();
```

3.2 初始化

```
/**
 * 初始化推流sdk
 *
 * @param streamerListener 推流业务监听器
 */
void initEngine(PLVSStreamerListener streamerListener);
```

3.3 设置监听器

```
/**
 * 添加推流事件监听器
 *
 * @param eventHandler 监听器
 */
void addEventHandler(PLVSStreamerEventListener eventHandler);
```

3.4 推流控制

```
/**
 * 开始推流
 */
void startLiveStream();

/**
 * 停止推流
 */
void stopLiveStream();
```

3.5 销毁

销毁推流和连麦管理器，并释放资源。

```
/**
 * 销毁
 */
void destroy();
```