1 功能概述

common模块的连麦是使用mvp模式实现的，将UI无关的连麦业务逻辑封装在了这个模块。

2 MVP模式

连麦的mvp用IPLVLinkMicContract作为合约类，内部分别定义了`IPLVLinkMicView`作为View接口，`IPLVLinkMicPresenter`作为Presenter接口。

```
/**
 * date: 2020/7/16
 * author: hwj
 * description: 连麦业务MVP模式的合约类
 */
public interface IPLVLinkMicContract {

    interface IPLVLinkMicView {

        /**
         * 响应连麦错误
         */
        void onLinkMicError(int errorCode, Throwable throwable);

        /**
         * 响应讲师开启连麦
         */
        void onTeacherOpenLinkMic();

        /**
         * 响应讲师关闭连麦
         */
        void onTeacherCloseLinkMic();

        /**
         * 响应讲师允许连麦
         */
        void onTeacherAllowJoin();

        /**
         * 响应加入连麦频道超时
         */
        void onAllowButJoinTimeout();

        /**
         * 响应在加入频道之前，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 isAudio     是否是音频连麦
         * @param linkMicList 连麦列表。在实现中，要用该列表去渲染
         */
        void onBeforeJoinChannel(String linkMicUid, boolean isAudio, List<PLVLinkMicItemDataBean> linkMicList);

        /**
         * 响应加入连麦频道成功
         */
        void onJoinChannelSuccess();

        /**
         * 响应离开连麦频道
         */
        void onLeaveChannel();

        /**
         * 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);

    /**
     * 回调我当前不在连麦列表
     */
    void onNotInLinkMicList();

    /**
     * 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 pos  列表中的位置
     */
    void onUserMuteVideo(final String uid, final boolean mute, int pos);

    /**
     * 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 pos  列表中的位置
     */
    void onUserMuteAudio(final String uid, final boolean mute, int pos);

    /**
     * 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<PLVLinkMicItemDataBean> linkMicList);

    /**
     * 切换第一画面
     *
     * @param linkMicId 新的第一画面的连麦Id
     */
    void onSwitchFirstScreen(String linkMicId);

    /**
     * 设置第一画面的连麦ID
     *
     * @param linkMicId 新的第一画面的连麦Id
```

```
     */
    void setFirstScreenLinkMicId(String linkMicId);

    /**
     * 隐藏连麦列表中的讲师item，并根据isNeedSwitchToMain参数决定是否要把该item的view切换到主屏
     *
     * @param linkMicId          讲师的连麦id
     * @param teacherPos         讲师item在当前连麦列表中的索引
     * @param isNeedSwitchToMain 是否要切换到主屏
     */
    void onAdjustTeacherLocation(String linkMicId, int teacherPos, boolean isNeedSwitchToMain);

    /**
     * 切换PPT View和第一画面的位置
     *
     * @param toMainScreen true表示切换到主屏幕，false表示切回到悬浮窗
     */
    void onSwitchPPTViewLocation(boolean toMainScreen);

    /**
     * PPT是否被切换到连麦列表了
     *
     * @return true表示PPT在连麦列表，false表示PPT不在连麦列表
     */
    boolean isMediaShowInLinkMicList();

    /**
     * 获取PPTView在连麦列表中的位置index
     *
     * @return ppt在连麦列表中的位置
     */
    int getMediaViewIndexInLinkMicList();

    /**
     * 点击连麦列表[index]位置上的画面
     *
     * @param index 连麦列表中的位置
     */
    void performClickInLinkMicListItem(int index);

    /**
     * 更新所有整个连麦列表
     */
    void updateAllLinkMicList();
}

interface IPLVLinkMicPresenter {

    /**
     * 销毁
     */
    void destroy();

    /**
     * 请求上麦
     */
```

```
    void requestJoinLinkMic();

    /**
     * 取消请求上麦
     */
    void cancelRequestJoinLinkMic();

    /**
     * 下麦
     */
    void leaveLinkMic();

    /**
     * 静音音频
     *
     * @param mute true表示静音，false表示打开
     */
    void muteAudio(boolean mute);

    /**
     * 禁用视频
     *
     * @param mute true表示禁用视频，false表示打开视频
     */
    void muteVideo(boolean mute);

    /**
     * 切换前后置摄像头方向
     */
    void switchCamera();

    /**
     * 创建渲染器
     *
     * @param context 上下文
     * @return 渲染器
     */
    SurfaceView createRenderView(Context context);

    /**
     * 获取当前用户的连麦ID
     *
     * @return 连麦ID
     */
    String getLinkMicId();

    /**
     * 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);

    /**
     * 是否加入连麦
```

```
         */
        boolean isJoinLinkMic();

        /**
         * 设置当前是音频连麦还是视频连麦
         *
         * @param isAudioLinkMic true表示是音频连麦，false表示是视频连麦
         */
        void setIsAudioLinkMic(boolean isAudioLinkMic);

        /**
         * 设置讲师否是打开连麦
         *
         * @param isTeacherOpenLinkMic true表示讲师打开连麦，false表示讲师关闭连麦
         */
        void setIsTeacherOpenLinkMic(boolean isTeacherOpenLinkMic);

        /**
         * 讲师是否打开连麦
         *
         * @return true表示讲师打开连麦，false表示讲师关闭连麦
         */
        boolean isTeacherOpenLinkMic();

        /**
         * 当前连麦的频道是否是纯视频频道类型并且其支持RTC
         *
         * @return true表示是纯视频频道类型并且其支持RTC，false表示是纯视频频道类型且不支持RTC或者是其他的频道类型
         */
        boolean isAloneChannelTypeSupportRTC();
    }
}
```

2.1 Presenter实现逻辑

`IPLVLinkMicPresenter`的实现类是PLVLinkMicPresenter。

其内部使用两个类来完成核心业务：

1. IPolyvLinkMicManager。SDK的连麦管理器。
2. PLVLinkMicMsgHandler，common层定义的连麦消息处理器，用于接收server发送的连麦消息。

并保存和维护连麦所需的数据，如：

1. 连麦账号数据`IPLVLiveRoomDataManager.getConfig`
2. 连麦列表`List<PLVLinkMicItemDataBean> linkMicList`

2.1.1 初始化

Presenter的初始化中做了如下操作：

初始化了：连麦所需的数据，绑定View，Model，构造连麦消息处理器

```
public PLVLinkMicPresenter(final IPLVLiveRoomDataManager liveRoomDataManager, @Nullable final
IPLVLinkMicContract.IPLVLinkMicView view) {
    this.liveRoomDataManager = liveRoomDataManager;
        //数据
        String viewerId = liveRoomDataManager.getConfig().getUser().getViewerId();
        PolyvLinkMicConfig.getInstance().init(viewerId, false);
        //view
        this.linkMicView = view;
        //model
        linkMicManager = PolyvLinkMicManagerFactory.createNewLinkMicManager();
        myLinkMicId = linkMicManager.getLinkMicUid();
        if (TextUtils.isEmpty(myLinkMicId)) {
            if (linkMicView != null) {
                linkMicView.onLinkMicError(-1, new Throwable("获取到空的linkMicId"));
            }
            return;
        }
        //构造连麦消息处理器
        linkMicMsgHandler = new PLVLinkMicMsgHandler(myLinkMicId, onLinkMicDataListener);
}
```

2.2 Model实现逻辑

2.2.1 连麦消息处理器：PLVLinkMicMsgHandler

连麦mvp的model层主要的逻辑集中在PLVLinkMicMsgHandler中，它是连麦消息处理器，接收server发送的连麦相关消息，并转换为对应的业务相关事件并回调出去。

监听并处理连麦socket消息：

```
//在PLVLinkMicMsgHandler构造函数中监听连麦消息
public PLVLinkMicMsgHandler(String linkMicId, @NotNull OnLinkMicDataListener onLinkMicDataListener) {
    this.linkMicId = linkMicId;
    this.onLinkMicDataListener = onLinkMicDataListener;
    PolyvSocketWrapper.getInstance().getSocketObserver().addOnMessageListener(onMessageListener,
            PLVEventConstant.LinkMic.JOIN_REQUEST_EVENT,
            PLVEventConstant.LinkMic.JOIN_RESPONSE_EVENT,
            PLVEventConstant.LinkMic.JOIN_SUCCESS_EVENT,
            PLVEventConstant.LinkMic.JOIN_LEAVE_EVENT,
            PLVEventConstant.Class.SE_SWITCH_MESSAGE,
            Socket.EVENT_MESSAGE);
}

//处理连麦消息
private void processLinkMicSocketMessage(String message, String event) {
    if (TextUtils.isEmpty(event)) {
        return;
    }
    PLVCommonLog.d(TAG, message);
    switch (event) {
            //①讲师开启/关闭连麦；②讲师将某个人下麦了
        case PLVEventConstant.LinkMic.EVENT_OPEN_MICROPHONE:
            break;
            //服务端发回来的join request
        case PLVEventConstant.LinkMic.JOIN_REQUEST_EVENT:
            break;
            //收到其他用户加入rtc频道后发送的joinSuccess
        case PLVEventConstant.LinkMic.JOIN_SUCCESS_EVENT:
            break;
            //讲师允许加入连麦
        case PLVEventConstant.LinkMic.JOIN_RESPONSE_EVENT:
            break;
            //离开连麦
        case PLVEventConstant.LinkMic.JOIN_LEAVE_EVENT:
            break;
            //讲师禁用观众视频或麦克风
        case PLVEventConstant.LinkMic.EVENT_MUTE_USER_MICRO:
            break;
            //客户端讲师设置相关权限。这里用于接收参与者上麦消息
        case PLVEventConstant.LinkMic.TEACHER_SET_PERMISSION:
            break;
            //切换第一画面
        case PLVEventConstant.Class.SE_SWITCH_MESSAGE:
            break;
        case PLVEventConstant.Class.SE_SWITCH_PPT_MESSAGE:
            //PPT和主屏幕切换位置
            break
    }
}
```

在 `processLinkMicSocketMessage` 中对server发送的消息进行了解析和处理，并用OnLinkMicDataListener监听器对外进行回调，则Presenter可以得知此时收到了讲师发送的指令的类型：

```
public interface OnLinkMicDataListener {
    /**
     * 讲师收到了我发送的连麦请求
     */
    void onTeacherReceiveJoinRequest();
    /**
     * 讲师允许我上麦
     */
    void onTeacherAllowMeToJoin();
    /**
     * 讲师将我下麦
     */
    void onTeacherHangupMe();
    /**
     * 讲师开启连麦
     * 如果是普通连麦观众，则通过请求连麦，老师允许，来加入rtc频道
     * 如果是参与者，则应该直接在回调中调用rtc的加入连麦
     */
    void onTeacherOpenLinkMic();
    /**
     * 讲师关闭连麦
     * 所有观众离开rtc频道
     */
    void onTeacherCloseLinkMic();
    /**
     * 讲师禁用观众视频或麦克风
     *
     * @param isMute  是否禁用
     * @param isAudio true是音频，false是视频
     */
    void onTeacherMuteMedia(boolean isMute, boolean isAudio);
    /**
     * 用户加入连麦成功
     *
     * @param dataBean 用户列表数据
     */
    void onUserJoinSuccess(PLVLinkMicItemDataBean dataBean);
    /**
     * 讲师发送奖杯
     */
    void onTeacherSendCup(String linkMicId, int cupNum);
    /**
     * 更新连麦类型
     *
     * @param isAudio true表示是音频连麦，false表示是视频连麦
     */
    void onUpdateLinkMicType(boolean isAudio);
    /**
     * 切换第一画面
     *
     * @param linkMicId 要切换到第一画面的ID
     */
    void onSwitchFirstScreen(String linkMicId);
    /**
     * 切换PPT View的位置
```

```
     *
     * @param toMainScreen true表示切换到主屏幕，false表示切回到悬浮窗
     */
    void onSwitchPPTViewLocation(boolean toMainScreen);
    /**
     * 讲师下课
     */
    void onFinishClass();
}
```

2.2.2 连麦item数据

PLVLinkMicItemDataBean是连麦列表的item对应的实体类，内部封装了连麦列表业务所需的数据。

PLVLinkMicDataMapper是连麦数据映射器，将服务端返回的数据模型转换成我们业务上需要的数据模型。我们通过接口或者socket拿到的服务端的数据并不适合直接使用，因此需要数据映射器。

3 SDK核心类介绍

连麦的SDK核心类是：IPolyvLinkMicManager。

该类的使用如上述，在PLVLinkMicPresenter中。

3.1 创建

```
IPolyvLinkMicManager linkMicManager = PolyvLinkMicManagerFactory.createNewLinkMicManager();
```

3.2 初始化和销毁

```
/**
 * 初始化引擎
 *
 * @param channelId            频道
 * @param linkMicEventListener 监听器
 */
void initEngine(int channelId, PolyvLinkMicListener linkMicEventListener);
/**
 * 销毁
 */
void destroy();
```

3.3 设置监听器

```
/**
 * 添加连麦事件监听器
 *
 * @param eventHandler 监听器
 */
void addEventHandler(PolyvLinkMicEventListener eventHandler);
/**
 * 移除连麦事件监听器
 *
 * @param eventHandler 监听器
 */
void removeEventHandler(PolyvLinkMicEventListener eventHandler);
```

3.4 加入和离开频道

```
/**
 * 加入频道
 *
 * @param
 */
void joinChannel();
/**
 * 离开频道
 */
void leaveChannel();
```

3.5 渲染器设置

```
/**
 * 添加本地摄像头
 *
 * @param view      渲染器
 * @param linkMicId 连麦ID
 */
void setupLocalVideo(SurfaceView view, String linkMicId);
/**
 * 设置远程摄像头
 *
 * @param view      渲染器
 * @param linkMicId 连麦ID
 */
void setupRemoteVideo(SurfaceView view, String linkMicId);
/**
 * 创建渲染器
 *
 * @param context
 * @return
 */
SurfaceView createRendererView(Context context);
```

3.6 多媒体控制

```
/**
 * 禁用/启用本地视频功能。该方法用于只看不发的视频场景。该方法不需要本地有摄像头。
 *
 * @param mute
 * @return
 */
void muteLocalVideo(boolean mute);
/**
 * 禁用/启用本地摄像头。
 *
 * @param enable
 * @return
 */
void enableLocalVideo(boolean enable);
/**
 * 禁用/启用本地音频功能。该方法用于只看不发的音频场景。
 *
 * @param mute
 * @return
 */
void muteLocalAudio(boolean mute);
/**
 * 切换摄像头
 */
void switchCamera();
```

3.7 获取数据

```
/**
 * 获取连麦列表和状态
 *
 * @param sessionId 场次ID
 * @param callback  回调
 */
void getLinkStatus(String sessionId, PLVLinkMicDataRepository.IPLVLinkMicDataRepoListener<PLVLinkMicJoinStatus>
callback);
/**
 * 获取连麦ID
 *
 * @return 连麦ID
 */
String getLinkMicUid();
```

3.8 切换连麦角色

```
 /**
  * 设置连麦角色为：观众
  */
 void switchRoleToAudience();
 /**
  * 设置连麦角色为：主播
  */
 void switchRoleToBroadcaster();
```

3.9 发送连麦socket消息

```
/**
 * 发送加入连麦成功
 *
 * @param sessionId  场次ID
 * @param linkMicUid 连麦ID
 */
PLVLinkMicJoinSuccess sendJoinSuccessMsg(String sessionId, String linkMicUid);
/**
 * 发送请求连麦
 *
 * @param linkMicUid 连麦ID
 */
void sendJoinRequestMsg(String linkMicUid);
/**
 * 发送离开连麦
 *
 * @param sessionId  场次ID
 * @param linkMicUid 连麦ID
 */
void sendJoinLeaveMsg(String sessionId, String linkMicUid);
/**
 * 发送媒体开关Mute消息
 *
 * @param linkMicMedia mute消息实体
 * @return
 */
boolean sendMuteEventMsg(PLVLinkMicMedia linkMicMedia);
```