连麦接入

简要说明

- 在使用直播JS-SDK后如果需要接入像POYLV直播观看页的连麦服务可在直播JS-SDK中开启连麦功能即可快速实现web端接入Polyv多人视频通话和音频通话，在**无延迟**以及常规直播下通用
- 如果已经使用polyv连麦sdk：https://player.polyv.net/resp/rtc-sdk/latest/polyv-rtc.min.js的用户也可参考当前文档，更新到当前用法使用，使用上与连麦sdk基本一直，可快速替换

使用前准备

- 确认已经开通连麦功能
- 浏览器需为chrome 58及以上或firefox、safari等主流浏览器
- 页面必须在 HTTPS 协议或者 localhost 中访问

连麦流程简介

1、讲师在客户端点击开启学员视频连线/开启学员语音连线
2、`OPEN_MICROPHONE`事件被触发
3、学员可以通过sdk实例的`joinChannel`方法向讲师发出连麦请求
4、讲师看到申请点击允许连麦，这时候页面sdk监听的`ALLOW_MICROPHONE`事件被触发
5、学员自动加入连麦，开始发起推流，事件`INIT_LOCAL_STREAM_READY`被触发，设置相应参数，进行推流
6、推流成功后如果收到讲师挂断或者学员主动点击挂断则退出连麦

参考代码

示例代码

订阅频道中其他流简介

1、学员成功加入连麦后会通过`USER_STREAM_ADDED`事件接收到频道中已存在的用户(包括讲师)、在该学员后加入学员的流 2、`USER_STREAM_ADDED`事件触发调用回调中的`evt.subscribe`方法对流进行订阅 3、若有用户期间退出频道则`USER_PEER_LEAVE`事件被触发

快速开始

```
// 初始化SDK
var liveSdk = new PolyvLiveSdk({
  channelId: channelId,
  sign: sign, // 频道验证签名
  timestamp: timestamp, // 毫秒级时间戳
  appId: appId, // polyv 后台的appId
  user: {
    userId: userId,
    userName: 'polyv-test',
    pic: 'https://livestatic.videocc.net/assets/wimages/missing_face.png'
  }
});

// 监听频道信息并初始化播放器，并加入rtc参数开启连麦，如果开启了无延迟播放则默认开启连麦
liveSdk.on(PolyvLiveSdk.EVENTS.CHANNEL_DATA_INIT, (event, data) => {
  liveSdk.setupPlayer({
    pptEl: '#ppt',
    el: '#player',
    type: 'live',
    rtc: true
  });
});

// 连麦实例初始化完成，可以进行连麦相关代码调用，无延迟，普通直播并支持连麦的情况下会回调
liveSdk.player.on('rtcInitialized', function(rtc){
  // 也可以这样获取实例 liveSdk.player.rtcInstance
  console.log('连麦sdk实例', rtc);
  // 连麦sdk加载后调用相关代码
  initRTC(rtc);
});

// 监听相关事件，进行申请连麦、推流、订阅频道其他用户等操作
function initRTC() {
  rtc.on('OPEN_MICROPHONE', (evt) => {
    console.log('讲师开启连麦，可发起申请加入连麦');
  });

  rtc.on('CLOSE_MICROPHONE', (evt) => {
    console.log('讲师关闭连麦，可禁止发起连麦申请');
  });

  rtc.on('ALLOW_MICROPHONE', (evt) => {
    console.log('连麦申请通过， 开始连麦');
  });

  rtc.on('INIT_LOCAL_STREAM_READY', (evt) => {
    console.log('准备开始推流，设置推流参数');
    evt.init({
      element: document.getElementById('local') // 选择需要显示本地流的节点
    });
  });
}
```

连麦实例

实例方法

`rtc.on(event: string, eventHandler: Function): void`

用于监听开启关闭连麦、订阅、推流状态等事件

`joinChannel(callback?: Function): void`

客户端开启连麦后可调用该方法申请加入 示例代码

```
rtc.on('OPEN_MICROPHONE', (evt) => {
  // 也可以通过页面点击申请
  rtc.joinChannel(() => {
    console.log('申请成功');
  });
});
```

参数

- `callback?: Function` 申请失败的回调函数

`cancelJoinChannel(callback?: Function): void`

申请加入连麦后，在还没有收到允许连麦消息时可调用，用于取消申请

参数

- `callback?: Function` 取消申请失败的回调函数

`disableVideo(): void`

推流后用于禁用本地视频轨道

`enableVideo(): void`

禁用本地视频轨道后用于恢复视频轨道

`disableAudio(): void`

推流后用于禁用本地音频轨道

`enableAudio(): void`

禁用本地视频轨道后用于恢复音频轨道

`leaveChannel(): void`

用于挂断连麦，退出连麦频道，停止推流

`openDeviceSetting(): void`

打开设备设置面板，可用于设备调试以及更换设备

`closeDeviceSetting(): void`

关闭设备设置面板

`openInviting(): void`

打开邀请上麦面板

`closeInviting(): void`

关闭邀请上麦面板

`destroy(): void`

销毁实例，如果正在连麦会退出连麦

事件

`rtc.on('OPEN_MICROPHONE', callback: Function): void`

客户端开启连麦, 此时可以申请加入连麦 示例代码：

```
rtc.on('OPEN_MICROPHONE', (evt) => {
  console.log(evt.type); // video/audio (视频/音频通话)
});
```

`rtc.on('CLOSE_MICROPHONE', callback: Function): void`

客户端关闭连麦，此时不可以申请加入连麦 示例代码：

```
rtc.on('CLOSE_MICROPHONE', (evt) => {
  // 若页面有申请连麦的按钮可以 disable
});
```

`rtc.on('JOIN_CHANNEL_TIMEOUT', callback: Function): void`

申请连麦超时 示例代码：

```
rtc.on('JOIN_CHANNEL_TIMEOUT', (evt) => {
  alert('申请连麦超时，请重新申请');
});
```

`rtc.on('ALLOW_MICROPHONE', callback: Function): void`

客户端通过连麦申请，这时会自动初始化本地流并加入连麦频道 示例代码：

```
rtc.on('ALLOW_MICROPHONE', (evt) => {
  console.log(`开始加入连麦，频道为 ${evt.roomId}`);
});
```

`rtc.on('INIT_LOCAL_STREAM_READY', callback: Function): void`

已经加入RTC频道，准备初始化本地流，此时可对本地流进行部分配置 示例代码：

```
rtc.on('INIT_LOCAL_STREAM_READY', (evt) => {
  console.log(`准备推流，通话类型为${evt.type}`);
  evt.init({
    element: document.getElementById('local'),
    control: true,
    profile: '480p'
  });
});
```

`evt.init(config: Object)`

开始初始化本地流并推流，必须调用此方法，否则无法加入连麦
参数说明
`element: HTMLElement`： Node节点，用于显示本地流
`control?: boolean`： 是否显示控制栏，默认值为 `true`
`profile?: string`： 设置视频属性, 默认值为 `'240P'`

profile 设置

视频属性	分辨率	码率（Kbps）
240p	320 × 240	200
240p_1	320 × 240	200
240p_4	424 × 240	220
480p	640 × 480	500

视频属性	分辨率	码率（Kbps）
480p_1	640 × 480	500
480p_8	848 × 480	610
720p	1280 × 720	1130
720p_1	1280 × 720	1130
720p_2	1280 × 720	2080
720p_5	960 × 720	910

`rtc.on('INIT_LOCALSTREAM_SUCCESS', callback: Function): void`

初始化本地流成功
示例代码：

```
rtc.on('INIT_LOCALSTREAM_SUCCESS', (evt) => {
  console.log('初始化本地流成功');
});
```

`rtc.on('INIT_LOCALSTREAM_ERROR', callback: Function): void`

初始化本地流失败
示例代码：

```
rtc.on('INIT_LOCALSTREAM_ERROR', (evt, error) => {
  console.log(evt.message);
  console.log(error);
});
```

`rtc.on('PUBLIC_STREAM_SUCCESS', callback: Function): void`

推流成功
示例代码：

```
rtc.on('PUBLIC_STREAM_SUCCESS', (evt) => {
 console.log('推流成功');
});
```

`rtc.on('PUBLIC_STREAM_ERROR', callback: Function): void`

推流失败
示例代码：

```
rtc.on('PUBLIC_STREAM_ERROR', (evt,  error) => {
  console.log(error);
});
```

`rtc.on('CLIENT_BANNED', callback: Function): void`

被讲师挂断连麦，连麦会退出，无需做其他操作
示例代码：

```
rtc.on('CLIENT_BANNED', (evt) => {
  console.log('被讲师挂断连麦');
});
```

`rtc.on('USER_STREAM_ADDED', callback: Function): void`

加入连麦频道后后收到老师以及其他连麦者的流，此时需要进行订阅
无延迟模式下不广播
示例代码：

```
rtc.on('USER_STREAM_ADDED', (evt) => {
  console.log(`是否为讲师：${evt.teacher}`);
  console.log(`流Id：${evt.streamId}`);
  console.log(`用户昵称：${evt.nick}`);
  console.log(`用户头像：${evt.pic}`);
  if (evt.teacher) {
    evt.subscribe({
      element: document.getElementById('teacher')
    }, (err) => {
      console.log('订阅失败');
    });
  } else {
    evt.subscribe({
      element: document.getElementById('student'),
      control: true
    });
  }
});
```

`evt.subscribe(config: Object, failCallback: Function)`

subscribe 方法若不调用则不会显示频道内其他连麦者
参数说明
`element: HTMLElement`：显示流的Node节点
`control?: boolean`：是否显示控制栏，默认值为 `true`

`rtc.on('USER_STREAM_SUBSCRIBED', callback: Function): void`

订阅成功
示例代码：

```
rtc.on('USER_STREAM_SUBSCRIBED', (evt) => {
  cons.log('订阅成功');
  console.log(`是否为讲师：${evt.teacher}`);
  console.log(`流Id：${evt.streamId}`);
  console.log(`用户昵称：${evt.nick}`);
  console.log(`用户头像：${evt.pic}`);
});
```

`rtc.on('USER_MUTE_VIDEO', callback: Function): void`

频道内其他连麦者关闭视频推流
示例代码：

```
rtc.on('USER_MUTE_VIDEO', (evt) => {
  console.log(evt.streamId);
});
```

`rtc.on('USER_UNMUTE_VIDEO', callback: Function): void`

频道内其他连麦者重新开启视频推流
示例代码：

```
rtc.on('USER_UNMUTE_VIDEO', (evt) => {
  console.log(evt.streamId);
});
```

`rtc.on('USER_MUTE_AUDIO', callback: Function): void`

频道内其他连麦者关闭音频推流
示例代码：

```
rtc.on('USER_MUTE_AUDIO', (evt) => {
  console.log(evt.streamId);
});
```

`rtc.on('USER_UNMUTE_AUDIO', callback: Function): void`

频道内其他连麦者重新开启音频推流
示例代码：

```
rtc.on('USER_UNMUTE_AUDIO', (evt) => {
  console.log(evt.streamId);
});
```

`rtc.on('LOCAL_MUTE_VIDEO', callback: Function): void`

本地视频推流关闭，被客户端禁止或者用户主动关闭
示例代码：

```
rtc.on('LOCAL_MUTE_VIDEO', (evt) => {
  console.log('关闭视频轨道');
});
```

`rtc.on('LOCAL_UNMUTE_VIDEO', callback: Function): void`

本地视频推流重新开启，被客户端开启或者用户主动开启
示例代码：

```
rtc.on('LOCAL_UNMUTE_VIDEO', (evt) => {
  console.log('关闭视频轨道');
});
```

`rtc.on('LOCAL_MUTE_AUDIO', callback: Function): void`

本地音频推流关闭，被客户端禁止或者用户主动关闭
示例代码：

```
rtc.on('LOCAL_MUTE_AUDIO', (evt) => {
  console.log('关闭视频轨道');
});
```

`rtc.on('LOCAL_UNMUTE_AUDIO', callback: Function): void`

本地音频推流重新开启，被客户端开启或者用户主动开启
示例代码：

```
rtc.on('LOCAL_UNMUTE_AUDIO', (evt) => {
  console.log('开启音频轨道');
});
```

`rtc.on('USER_PEER_LEAVE', callback: Function): void`

订阅的流期间挂断连麦退出频道触发
无延迟模式下不广播
示例代码：

```
rtc.on('USER_PEER_LEAVE', (evt) => {
  console.log(`${evt.streamId}退出连麦`);
});
```

`rtc.on('LEAVE_CHANNEL_SUCCESS', callback: Function): void`

离开视频通话成功
示例代码：

```
rtc.on('LEAVE_CHANNEL_SUCCESS', (evt) => {
  console.log('离开视频通话成功');
});
```

`rtc.on('STOP', callback: Function): void`

主动挂断视频通话(点击默认控制栏的挂断按钮触发)，这是可以恢复到未申请连麦的抓台
示例代码：

```
rtc.on('STOP', (evt) => {
  console.log('主动挂断通话');
});
```

`rtc.on('SWITCH_MASTER', callback: Function): void`

切换主讲人事件，这时可将页面对应连麦者更换位置，建议通过css切换位置 无延迟模式下不广播
示例代码：

```
rtc.on('SWITCH_MASTER', function(evt) {
    var currentUser = evt.currentUser;
    var previousUser = evt.previousUser;

    // 上一个主讲，只在离开的情况为空
    var previousElement = previousUser.element;
    // 当前需要设置的主讲
    var currentElement = currentUser.element;

    rtc.remove(currentUser.streamId);

    // 上一个是因为离开才切换的
    if (previousUser.leave) {
      // 设置到主讲位置
      rtc.play(currentUser.streamId, {
        element: document.getElementById('playerRTC')
      });

      // 上一个主讲已经离开，不需要原来的位置了，删掉元素
      var pn = currentElement.parentNode;
      pn && pn.removeChild(currentElement);
    } else {
      // 设置到上一个主讲位置
      rtc.remove(previousUser.streamId);
      rtc.play(currentUser.streamId, {
        element: previousElement
      });

      // 不是因为离开才切换的需要重新播放上一个主讲的流到新的位置
      rtc.play(previousUser.streamId, {
        element:  currentElement
      });
    }
});
```

`rtc.on('NETWORK_QUALITY', callback: Function): void`

当前用户的上下行网络质量，大约一到两秒触发一次 uplinkNetworkQuality 为上行质量，downlinkNetworkQuality为下行质量，取值`0|1|2|3|4|5|6`

- 0 网络质量未知
- 1 网络质量优秀
- 2 网络质量良好
- 3 网络质量一般
- 4 网络质量较差
- 5 网络质量糟糕
- 6 网络质量断开
  示例代码：

```
rtc.on('NETWORK_QUALITY', function(evt) {
      console.log('上行质量为:', evt.uplinkNetworkQuality);
      console.log('下行质量为:', evt.downlinkNetworkQuality);
});
```

`rtc.on('INVITE_TO_MICROPHONE', callback: Function): void`

收到连麦邀请事件，此时可显示连麦邀请弹窗
示例代码：

```
rtc.on('INVITE_TO_MICROPHONE', function () {
    rtc.openInviting();
});
```

注意

- **在无延迟模式下sdk会自行订阅用户流所以不会广播'USER_STREAM_ADDED', 'SWITCH_MASTER', 'USER_PEER_LEAVE'事件**
- 在普通场景下连麦注意连麦成功后隐藏cdn播放器显示rtc，离开连麦后做恢复处理