

小程序·云直播

操作指南

产品文档



腾讯云

【版权声明】

©2013-2020 腾讯云版权所有

本文档（含所有文字、数据、图片等内容）完整的著作权归腾讯云计算（北京）有限责任公司单独所有，未经腾讯云事先明确书面许可，任何主体不得以任何形式复制、修改、使用、抄袭、传播本文档全部或部分内容。前述行为构成对腾讯云著作权的侵犯，腾讯云将依法采取措施追究法律责任。

【商标声明】



及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。未经腾讯云及有关权利人书面许可，任何主体不得以任何方式对前述商标进行使用、复制、修改、传播、抄录等行为，否则将构成对腾讯云及有关权利人商标权的侵犯，腾讯云将依法采取措施追究法律责任。

【服务声明】

本文档意在向您介绍腾讯云全部或部分产品、服务的当时的相关概况，部分产品、服务的内容可能不时有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系95716。

文档目录

操作指南

功能集成

直播带货组件接入说明

操作指南

功能集成

最近更新时间：2020-06-05 17:16:04

在正式接入前，请先阅读微信小程序提供的 [插件文档](#)，了解插件的使用范围和限制。

准备工作

1. 在 [微信公众平台](#) 注册并登录小程序。
2. 符合接入要求，申请插件并购买小程序·云直播服务，详见 [小程序·云直播插件](#)。
3. 开通小程序·云直播服务后，登录 [云直播控制台](#)，在【域名管理】中添加小程序直播域名，然后 [自助拼接直播地址](#)。
4. 下载并安装最新版本的 [微信开发者工具](#)，使用小程序绑定的微信号扫码登录开发者工具。



在小程序中引入插件代码

在小程序中引入插件代码，可参考 [demo 源码](#)。使用插件前需在小程序工程的 `app.json` 中声明要使用的插件，例如：

```
{
  .....
  "plugins": {
    "liveRoomPlugin": {
      "version": "1.2.1",
      "provider": "wx95a7d2b78cf30f98"
    }
  }
}
```

小程序使用插件中的播放组件

播放组件

在 page 的 json 文件中定义需要引入的 live-room-play 组件，使用 plugin:// 协议。

```
{
  "usingComponents": {
    "live-room-play": "plugin://liveRoomPlugin/live-room-play" //播放组件
  }
}
```

在 page 的 .wxml 文件加载上一步引入的 live-room-play 组件。

```
<view class="container-box">
  <view class="player-view">
    <live-room-play version="{{version}}" liveAppID="{{liveAppID}}" playUrl="{{playUrl}}" orientation="{{orientation}}" objectFit="{{objectFit}}"
    minCache="{{minCache}}" maxCache="{{maxCache}}" mode="{{mode}}" muted="{{muted}}" debug="{{debug}}" bindPlayEvent="onPlayEvent" >
    </live-room-play>
  </view>
</view>
```

推流组件

在 page 的 json 文件中定义需要引入的 live-room-push 组件，使用 plugin:// 协议。

```
{
  "usingComponents": {
    "live-room-push": "plugin://liveRoomPlugin/live-room-push" //推流组件
  }
}
```

在 page 的 .wxml 文件加载上一步引入的 live-room-push 组件。

```
<live-room-push version="{{version}}" liveAppID="{{liveAppID}}" pushUrl="{{pushUrl}}" orientation="{{orientation}}" muted="{{muted}}" mode="{{mode}}" waitingImage="{{waitingImage}}" enableCamera="{{enableCamera}}" beauty="{{beauty}}" whiteness="{{whiteness}}" backgroundMute="{{backgroundMute}}"
debug="{{debug}}" autoFocus="{{autoFocus}}" aspect="{{aspect}}" minBitrate="{{minBitrate}}" maxBitrate="{{maxBitrate}}" zoom="{{zoom}}" devicePosition="{{devicePosition}}" sdkAppID="{{sdkAppID}}" accountType="{{accountType}}" userID="{{userID}}" userSig="{{userSig}}"
roomId="{{roomId}}" nickName="{{nickName}}" avatar="{{avatar}}" bindPushEvent="onPushEvent"
bindIMEvent="onIMEvent">
```

直播播放组件相关的属性说明

直播插件的使用方法和微信原生标签的方法一致，可参考微信小程序标签 [live-player](#) 的文档说明。

属性	类型	默认值	必填	说明
liveAppID	Number	0	是	用户的腾讯云 AppID
playUrl	String	""	是	需用小程序直播播放域名生成的播放地址，详见 自助拼接直播地址
version	Number	1	否	直播控制台购买插件即为新版本，此处必填值为：2 旧方案接入：1 或不填
mode	String	"live"	否	live（直播），RTC（实时通话，该模式延时更低）
orientation	String	"vertical"	否	画面方向，可选值有 vertical、horizontal
objectFit	String	"contain"	否	填充模式，可选值有 contain、fillCrop
minCache	Number	1	否	最小缓冲区，单位 s
maxCache	Number	3	否	最大缓冲区，单位 s
muted	Boolean	false	否	是否静音
debug	Boolean	false	否	是否显示日志
soundMode	String	'speaker'	否	声音输出设备，'speaker'或者'ear'，代表扬声器或听筒

属性	类型	默认值	必填	说明
autoplay	Boolean	false	否	是否自动播放
autopause	Boolean	true	否	页面跳转时是否自动暂停
sdkAppId	String	""	否	IM 应用的 AppID，如果不需要启用内置 IM，可以不填
userId	String	"not set"	否	用户在 IM 内的唯一 ID
userSig	String	"not set"	否	用户的 IM 登录签名，签名一般由服务端根据 IM 应用的公私钥生成
roomId	String	"not set"	否	房间 ID
nickName	String	"not set"	否	用户在 IM 里的昵称
avatar	String	"not set"	否	用户在 IM 里的头像 url
bindPlayEvent	EventHandle	null	否	播放状态变化事件回调
bindfullscreenChangeEvent	EventHandle	null	否	全屏状态变化事件回调
bindNetStatus	EventHandle	null	否	网络状态变化回调
bindAttachedEvent	EventHandle	null	否	插件加载完成回调
bindIMEvent	EventHandle	null	否	播放组件内置的 IM 事件回调

直播推流组件相关属性说明

直播插件的使用方法和微信原生标签的方法一致，可参考微信小程序标签 [live-pusher](#) 的文档说明。

属性	类型	默认值	必填	说明
liveAppID	Number	0	是	用户的腾讯云 AppID
pushUrl	String	""	是	需用小程序直播播放域名生成的播放地址，详见 自助拼接直播地址
version	Number	1	否	直播控制台购买插件即为新版本，此处必填值为：2 旧方案接入：1 或不填
orientation	String	"vertical"	否	推流画面方向，可选值有 vertical、horizontal

属性	类型	默认值	必填	说明
muted	Boolean	false	否	是否静音
mode	String	"SD"	否	清晰度，可选值：SD（标清）、HD（高清）、FHD（超清）、RTC（实时通话）
waitingImage	String	""	否	进入后台推流时候的垫片图片
enableCamera	Boolean	true	否	是否开启摄像头
beauty	Number	0	否	美颜指数，取值0 - 9，数值越大效果越明显
whiteness	Number	0	否	美白指数，取值0 - 9，数值越大效果越明显
backgroundMute	Boolean	false	否	进入后台时是否静音，初始化设置生效
debug	Boolean	false	否	是否显示日志
autoFocus	Boolean	true	否	自动聚焦，初始化设置生效
aspect	String	"9:16"	否	宽高比，可选值有3:4，9:16
minBitrate	Number	200	否	最小码率，仅在 mode 为"RTC"的时候生效
maxBitrate	Number	1000	否	最大码率，仅在 mode 为"RTC"的时候生效
zoom	Boolean	false	否	自动调整焦距，初始化设置生效
devicePosition	String	"front"	否	摄像头前置或后置，值为 front，back。注意这个值只是在组件初始化时生效，不能动态修改。动态切换摄像头用接口 switchCamera
audioQuality	String	'high'	否	录音质量，'high'或者'low'
mirror	Boolean	false	否	是否镜像反转
autopush	Boolean	true	否	是否自动推流
sdkAppId	String	""	否	IM 应用的 AppID，如果不需要启用内置 IM，可以不填
userId	String	"not set"	否	用户在 IM 内的唯一 ID
userSig	String	"not set"	否	用户的 IM 登录签名，签名一般由服务端根据 IM 应用的公私钥生成
roomId	String	"not set"	否	房间 ID

属性	类型	默认值	必填	说明
nickName	String	"not set"	否	用户在 IM 里的昵称
avatar	String	"not set"	否	用户在 IM 里的头像 url
bindPushEvent	EventHandle	null	否	推流状态变化事件回调
bindNetStatus	EventHandle	null	否	网络状态变化事件回调
bindError	EventHandle	null	否	推流错误回调
bindBgmStart	EventHandle	null	否	背景音乐开始播放回调
bindBgmProgress	EventHandle	null	否	背景音乐进度回调
bindBgmComplete	EventHandle	null	否	背景音乐播放完成回调
bindAttachedEvent	EventHandle	null	否	插件加载完成回调

在播放区域叠加额外展示信息

组件提供了一个 `<slot>` 节点，用于承载组件引用时提供的子节点。本功能受限于微信，只能在组件上叠加 `cover-image`、`cover-view` 和 `canvas`。

组件实例化

获取 live-room-play 组件实例

为什么要获取 live-room-play 组件实例？

live-room 组件支持播放、停止播放、全屏等接口，要调用这些接口需要先获取到 live-room-play 的实例。

怎么获取 live-room-play 组件实例？

live-room-play 是腾讯视频云直播插件中的一个组件，在腾讯视频云直播插件中暴露了获取 live-room-play 组件实例的接口，您只需要先在 page 的 js 文件中，将插件加载进来，即可获取到 live-room-play 组件实例。

```
// 加载插件
var plugin = requirePlugin("liveRoomPlugin")
// 获取live-room组件实例
var liveRoomComponent = plugin.instance.getLiveRoomInstance();
```

获取 live-room-push 组件实例

具体步骤同上 [获取 live-room-play 组件实例](#)。

组件接口

live-room-play 组件提供如下接口

- 可参考微信小程序组件 [LivePlayerContext 方法](#)。

- start

开始播放。调用之后会启动播放。在开始播放之前，playUrl 也要保证已经设置到组件属性中。

```
// 获取live-room-play组件实例
var liveRoomComponent = plugin.instance.getLiveRoomInstance();
liveRoomComponent.start();
```

- stop

结束播放。

```
// 获取live-room-play组件实例
var liveRoomComponent = plugin.instance.getLiveRoomInstance();
liveRoomComponent.stop();
```

- requestFullScreen

全屏播放。

```
// 获取live-room-play组件实例
var liveRoomComponent = plugin.instance.getLiveRoomInstance();
liveRoomComponent.requestFullScreen(true); //全屏播放
//liveRoomComponent.requestFullScreen(false); //退出全屏
```

- pause

暂停播放。

```
// 获取live-room-play组件实例
var liveRoomComponent = plugin.instance.getLiveRoomInstance();
liveRoomComponent.pause(); //暂停播放
```

- resume

恢复播放。

```
// 获取live-room-play组件实例
var liveRoomComponent = plugin.instance.getLiveRoomInstance();
liveRoomComponent.resume(); //恢复播放
```

- mute

静音。

```
// 获取live-room-play组件实例
var liveRoomComponent = plugin.instance.getLiveRoomInstance();
liveRoomComponent.mute(); //静音
```

- 如果小程序需要后台播放纯音频流，可以使用纯音频的转码流进行播放，示例代码如下：

```
this.audio = wx.getBackgroundAudioManager();
this.audio.protocol = 'hls';// 这个属性设置为hls才支持m3u8类型的直播流
this.audio.title = '后台音乐';// 显示在状态栏的标题
this.audio.src = "http://domain/live/streamName_pureAudio.m3u8?txSecret=xxxx&txTime=xxxxx";
// 注：domain是PaaS插件申请时我方提供的，streamName是指定的流名称，
// 后面加_pureAudio就是纯音频转码流，后台播放时可以节省流量。
```

live-room-push 组件提供如下接口

- 可参考微信小程序组件 [LivePusherContext](#) 方法。

- start

开始推流。调用之后会启动推流。在开始推流之前，pushUrl 也要保证已经设置到组件属性中。

```
// 获取live-room-push组件实例
var liveRoomPushComponent = plugin.instance.getLiveRoomPushInstance();
liveRoomPushComponent.start();
```

- stop

结束推流。

```
// 获取live-room-push组件实例
var liveRoomPushComponent = plugin.instance.getLiveRoomPushInstance();
liveRoomPushComponent.stop();
```

- snapshot

截图。

```
liveRoomPushComponent.snapshot({
  success: function (res){
    wx.saveImageToPhotosAlbum({
      filePath: {{imagepath}}
    })
  },
  fail:function(res) {
  }
});
```

- switchCamera

切换前后摄像头。

```
liveRoomComponent.switchCamera();
```

- toggleTorch

打开/关闭手电筒。

```
liveRoomComponent.toggleTorch();
```

- playBGM

播放背景音乐。

```
liveRoomComponent.playBGM({  
  url: media_url,  
  success: function(res){},  
  fail: function(err){},  
  complete: function(res){}  
});
```

- stopBGM

停止背景音乐。

```
liveRoomComponent.stopBGM();
```

- pauseBGM

暂停背景音乐。

```
liveRoomComponent.pauseBGM();
```

- resumeBGM

恢复背景音乐。

```
liveRoomComponent.resumeBGM();
```

- setBGMVolume

设置背景音量。

```
liveRoomComponent.setBGMVolume({  
  volume: "0.5", // 0-1之间的浮点数字符串  
  success: function(ers){},  
  fail: function(err){},  
  complete: function(res){}  
});
```

- startPreview

开启摄像头预览。

```
liveRoomComponent.startPreview();
```

- stopPreview

关闭摄像头预览。

```
liveRoomComponent.stopPreview();
```

组件事件

播放事件

播放事件分为：

1. 普通的播放事件，tag 是 playEvent，code 含义见 [状态码](#)。
2. 错误事件，tag 是 error。现在只有白名单校验出错时会抛出，code 是-1，具体的错误原因在 detail 中给出。

code	含义	具体错误信息
-1	白名单校验出错	具体的错误原因在 detail 中给出

推流事件

推流事件分为：

1. 普通的推流事件，tag 是 pushEvent，code 含义见 [状态码](#)。
2. 错误事件，tag 是 error。现在只有白名单校验出错时会抛出，code 是-1，具体的错误原因在 detail 中给出。

直播带货组件接入说明

最近更新时间：2020-05-29 14:55:53

直播带货组件是对直播带货场景中使用即时通信 IM 能力的二次封装，在封装基本的收发消息能力的同时，针对直播带货场景封装了点赞、送礼、商品推送和优惠券领取等相关能力。

直播带货体验 Demo

提供弹幕、点赞、送礼、加购等能力。



前置条件

1. 创建即时通信 IM 应用
2. 创建群自定义字段
 - attent：记录自己关注了哪些主播
3. 创建群成员自定义字段
 - add_goods：后台上架新的直播间商品列表
 - room_tatus：控制直播间状态

组件参数

- SDKAppID
即时通信应用 ID，SDKAppID 可在 [即时通信 IM 控制台](#) 上创建应用，SDKAppID 可在应用基本信息模块获取，开始创建的应用为体验版，可永久免费试用，但配置有限，不适合正式环境试用。正式环境可购买旗舰版或专业版。
- userSig
即时通信 IM 用户签名，UserSig 为登录腾讯云即时通信 IM SDK 的必要参数，可在业务后台生成后返回给前端

使用，生成文档与代码可参考 [生成 UserSig](#)。

- roomId

直播间群聊房间 ID，此 ID 为即时通信 IM 所创建的实时音视频（AVCHATROOM）房间 ID，该房间无加入人数限制可通过 [即时通信 IM](#) 的【[群组管理](#)】创建，或通过 [restapi](#) 创建。

- TIM

即时通信 IM SDK，如是在小程序环境，请使用 **tim-wx-sdk**；web 环境则使用 **tim-js-sdk**。

- userName

与生成 userSig 的 userName 一致。

初始化示例

```
import TSL from 'tim-live-sell'
import TIM from 'tim-js-sdk' //web环境
// import TIM from 'tim-wx-sdk' 小程序环境
const tls = new TIMLiveSell({
  SDKAppID: 1400***803,
  roomId: '@TGS#E***NVLGE',
  userSig: 'eJwtzM9***-reWMQw_',
  userName: 'Ho***st',
  TIM: TIM
})
```

组件回调

TSL.EVENT.SDK_READY

组件已初始化，对应于 im sdk 的 TIM.EVENT.SDK_READY，此时才可以调用 sdk 的方法。

```
tls.on(TSL.EVENT.SDK_READY, async() => {

})
```

TSL.EVENT.ROOM_STATUS_CHANGE

房间状态改变，如主播上/下线、暂停等

```
tls.on(TSL.EVENT.ROOM_STATUS_CHANGE, async(data) => {

})
```

TSL.EVENT.JOIN_GROUP

有人加入群聊时触发

```
tls.on(TSL.EVENT.JOIN_GROUP, async(data) => {  
  const {nick,avatar,userID} = data  
})
```

TSL.EVENT.EXIT_GROUP

有人退出群聊时触发

```
tls.on(TSL.EVENT.EXIT_GROUP, async(data) => {  
  const {nick,avatar,userID} = data  
})
```

TSL.EVENT.NOTIFACATION

公告发生修改时触发

```
tls.on(TSL.EVENT.NOTIFACATION, async(data) => {  
  const { notification } = data  
})
```

TSL.EVENT.MESSAGE

有人发送群消息时触发

说明：

自己发送的消息不会在这个回调中，自己发送的消息上屏请使用 `sendMessage` 方法返回的数据，这里与 `im sdk` 保持一致。

```
tls.on(TSL.EVENT.MESSAGE, async(data) => {  
  const { nick,avatar,message,userID } = data  
})
```

TSL.EVENT.LIKE

有人给主播点赞时触发

```
tls.on(TSL.EVENT.LIKE, async(data) => {  
  const { nick,avatar,value,userID } = data  
})
```

TSL.EVENT.SEND_GIFT

有人给主播送礼时触发

```
tls.on(TSL.EVENT.SEND_GIFT, async(data) => {  
  const { nick,avatar,value,userID } = data  
})
```

TSL.EVENT.ATTENT

有人关注主播时触发

```
tls.on(TSL.EVENT.ATTENT, async(data) => {  
  const { nick,avatar,value,userID } = data  
})
```

TSL.EVENT.BUY_GOODS

有人购买商品时触发

```
tls.on(TSL.EVENT.BUY_GOODS, async(data) => {  
  const { nick,avatar,value,userID } = data  
})
```

onUseCoupon

有人领取优惠券时触发

```
tls.on(TSL.EVENT.USE_COUPON, async(data) => {  
  const { nick,avatar,value,userID } = data  
})
```

TSL.EVENT.ADD_GOODS

该直播中所推荐的商品发生改变时触发

```
tls.on(TSL.EVENT.ADD_GOODS, async(data) => {  
  const { nick,avatar,value } = data  
})
```

TSL.EVENT.KICKED

账号其他地方登录时触发

```
tls.on(TSL.EVENT.KICKED, async()) => {  
  
})
```

TSL.EVENT.NETWORK_CHANGE

网络发生改变时触发

```
tls.on(TSL.EVENT.NETWORK_CHANGE, async() => {  
  
})
```

TSL.EVENT.SDK_NOT_READY

sdk 不可用时触发

```
tls.on(TSL.EVENT.SDK_NOT_READY, async() => {  
  
})
```

TSL.EVENT.PROFILE_UPDATE

个人资料发生修改时触发

```
tls.on(TSL.EVENT.PROFILE_UPDATE, async() => {  
  
})
```

TSL.EVENT.ERROR

sdk 发生错误时触发

```
tls.on(TSL.EVENT.ERROR, async(error) => {  
  
})
```

\${type}

在调用自定义消息时触发的同名事件

```
tls.on(`${type}`, async(error) => {  
  
})
```

组件方法

sendMessage(message:string)

方法说明：发送弹幕消息

```
/**
 * 发送弹幕消息，调用此方法时，群类所有人都可收到该文本消息
 * @method sendMessage
 * @for TSL
 * @param msg 必填 弹幕消息
 * @returns Promise
 */
const {nick,avatar,message} = await tls.sendMessage(msg);
```

like(extension?:string)

- 方法说明：给主播点赞
- extension：点赞时的附加信息，如用户等级

```
/**
 * 发送点赞消息，调用此方法时，群内所有人都可收到该点赞消息
 * @method like
 * @for TSL
 * @param extension 非必填 点赞时附加信息
 * @returns Promise
 */
const {nick,avatar} = await tls.like(extension);
```

gift(extension?:string)

- 方法说明：给主播送礼
- extension：送礼时的附加信息，如礼物信息等

```
/**
 * 发送送礼消息，调用此方法时，群内所有人都可收到该送礼消息
 * @method gift
 * @for TSL
 * @param msg 非必填 送礼时附加信息
 * @returns Promise
 */
const {nick,avatar} = await tls.gift(extension);
```

exitRoom()

方法说明：退出直播间

```
/**
 * 退出房间，主播（群主）不可退出房间
 * @method exitRoom
```

```
* @for TSL
* @param
* @returns Promise
*/
const {status} = await tls.exitRoom();
```

joinRoom()

方法说明：加入直播间

```
/**
 * 加入房间
 * @method joinRoom
 * @for TSL
 * @param
 * @returns Promise
 */
const {userInfo,groupInfo} = await tls.joinRoom();
const { ownerInfo } = groupInfo;//获取主播信息
const { userID,nick,avatar } = ownerInfo
```

getRoomInfo()

方法说明：获取当前房间信息

```
/**
 * 获取该聊天室的基本信息
 * @method getRoomInfo
 * @for TSL
 * @param
 * @returns Promise
 */
const {...ownerInfo} = await tls.getRoomInfo();
//ownerInfo 为主播相关信息
const { userID,nick,avatar } = ownerInfo
```

attention()

方法说明：关注主播

```
/**
 * 关注主播
 * @method attention
 * @for TSL
 * @param
 * @returns Promise
```

```
*/  
const {nick,avatar} = await tls.attention();
```

cancelAttention()

方法说明：取消关注主播

```
/**  
 * 取消关注  
 * @method cancelAttention  
 * @for TSL  
 * @param  
 * @returns Promise  
 */  
const {nick,avatar} = await tls.cancelAttention();
```

destroy()

方法说明：销毁组件

```
/**  
 * 销毁组件  
 * @method destroy  
 * @for TSL  
 * @param  
 * @returns Promise  
 */  
tls.destroy();
```

sendCustomMsgAndEmitEvent(eventName: string, extension?: string)

- 方法说明：发送自定义消息，并触发和 type 类型的回调事件
- eventName：事件名
- extension：自定义消息发送者附带信息

```
/**  
 * 发送自定义消息，并触发同名事件  
 * @method destroy  
 * @for TSL  
 * @param eventName:事件名, someExtension : 附加信息  
 * @returns Promise  
 */  
await tsl.sendCustomMsgAndEmitEvent('eventName','someExtension')
```

内置对象

TIM 通过 TIM.create 创建的对象，可使用 TIM 所有的方法。

事件

EVENT

- TLS.EVENT.SDK_READY
- TLS.EVENT.JOIN_GROUP
- TLS.EVENT.EXIT_GROUP
- TLS.EVENT.NOTIFACATION
- TLS.EVENT.MESSAGE
- TLS.EVENT.PROFILE_UPDATE
- TLS.EVENT.ERROR
- TLS.EVENT.KICKED
- TLS.EVENT.NETWORK_CHANGE
- TLS.EVENT.SDK_NOT_READY
- TLS.EVENT.LIKE
- TLS.EVENT.BUY_GOODS
- TLS.EVENT.USE_COUPON
- TLS.EVENT.SEND_GIFT
- TLS.EVENT.ATTENT
- TLS.EVENT.ADD_GOODS