

【视频客服-访客】Juphoon Plugin for WeChat 开发集成指南



开发指导手册（WeChat）

宁波菊风系统软件有限公司

2025年1月

版权所有©宁波菊风系统软件有限公司 2025。保留一切权利。

版本	作者	日期	说明
V2501.0	章克飞	2025.01	• 初版

一、概述

非常感谢您使用菊风系统软件的产品，我们将为您提供最好的服务。本手册可能包含技术上不准确的地方或排版错误。本手册的内容将做定期的更新，恕不另行通知；更新的内容将会在本手册的新版本中加入。我们随时会改进或更新本手册中描述的产品或程序。

1.1 系统介绍

菊风视频能力平台在实际的项目中定位为音视频能力的提供方，除此之外还包装了一些和音视频通讯强相关的业务。以银行项目为例可分为视频客服业务、视频会议业务、视频双录业务、AI 双录业务、一对一通话及消息业务等。上述业务需要客户渠道类系统或者客户业务类系统集成我们 JRTC SDK 或者插件才能形成完整的业务，在整个完整的业务中我们提供基础的音视频通讯能力和一些对应业务上所需的特色能力。如视频客服业务的智能排队服务，视频会议业务的增强会控服务等。

菊风视频能力平台提供标准 JRTC SDK 用于给客户渠道类系统和客户业务类系统集成并通过 JRTC SDK 接入到视频能力平台进行音视频通讯。

菊风为开发者提供 JRTC SDK 功能开发包，涵盖了音视频引擎终端、服务器和业务模块，支持实现智能排队、全景录像、多人音视频等业务功能。

JRTC SDK 支持 iOS、Android、鸿蒙Next、Windows、UOS、Kylin、微信小程序、H5 等操作系统平台。对于银行的其他公共平台或其他第三方平台，视频能力平台可提供标准第三方接口和其他平台进行对接。实现和银行环境的整体融入。

1.2 系统特性

菊风视频能力平台（Juphoon Video Capability Platform）提供高可用、高品质、超低延时的实时音视频通信服务，为远程银行、视频双录、视频会议、AI 双录、VoLTE 视频通话等泛金融场景化方案提供平台支撑。具有业界领先的实时音视频编码技术，以及抗啸叫降噪、ARS 码率自适应、SPo 视频甜点、智能路由等技术，应对网络质量非均衡性、网络异构性、多类型终端的接入的挑战，保证高音质、高画质。Juphoon Plugin for WeChat 插件专为微信小程序平台设计。整个平台由宁波菊风系统软件有限公司独立研发，具有自主知识产权。

二、关于菊风软件

宁波菊风系统软件有限公司（简称“菊风”，英文简称“Juphoon”）成立于2005年，现有员工200余人，注册资金2050万元，总部位于宁波，在北京、广州、长沙设有区域中心（研发、销售和交付），在郑州和杭州设有交付中心，是一家提供实时音视频通信和 RCS 融合通信解决方案的供应商。宁波总部研

发中心主要负责客户端 SDK 和 APP、音视频引擎、服务器等产品的研发；云平台和服务器的运维、网管等支撑系统研发，现中心成员有180名。

菊风经过15年+音视频底层技术积累，为众多行业合作伙伴提供了超优音视频通信服务。凭借卓越的产品以及优质的服务，迄今为止，已有数十亿终端用户以及众多企业用户通过菊风云实现了音视频场景化沟通，涉及社交、教育、医疗、智能硬件、金融、电商等多个行业领域，为其提供了有针对性的行业化解决方案。

菊风为开发者提供的优而小的 SDK 极简接入，快速助其实现实时音视频通信能力。基于客户不同需求，菊风云提供灵活的部署模式——公有云，专有云，私有云，海外云以及混合云。对主流系统平台全覆盖，支持 iOS、Android、鸿蒙Next、Windows、UOS、Kylin、微信小程序、H5 等。支持各移动设备（电脑、手机、平板）、VTM 机等多终端设备的适配。

2.1 技术支持

在您使用 Juphoon RTC SDK 的过程中，遇到任何困难，请与我们联系，我们将热忱为您提供帮助。

您可以通过如下方式与我们取得联系：

公司官网：<https://rtc.juphoon.com>

产品咨询：sales@juphoon.com

加急热线：13056832331

咨询电话：400-800-8708 / 0574-87901227

售前工程师微信二维码：



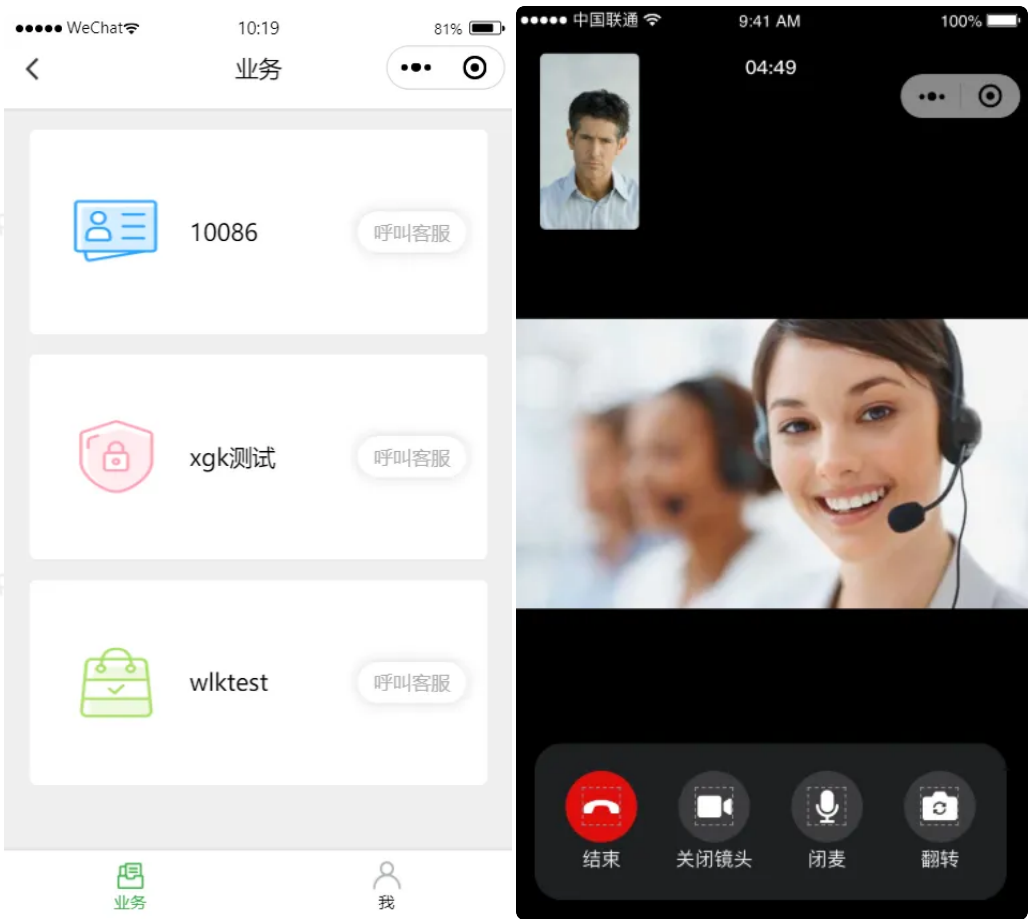
2.2 版权申明

“Juphoon RTC for WeChat Plugin”是由宁波菊风系统软件有限公司开发，拥有自主知识产权（软著正式编号 2020SR0369466号）的系统平台，宁波菊风系统软件有限公司拥有与本产品所用技术相关

的知识产权。这些知识产权包括但不限于 一项或多项发明专利或者正在进行申请的专利 (ZL202010288867.7 、 ZL201911393580.4) 。

本产品发行所依照的许可协议限制其使用、复制分发和反编译。未经宁波菊风系统软件有限公司事先书面授权，不得以任何形式或借助任何手段复制本产品的任何部分。Juphoon 是宁波菊风系统软件有限公司的商标。

三、效果图



四、快速搭建Plugin

4.1 插件概述

该插件是为小程序设计的视频客服访客插件，支持点对点呼叫和业务号呼叫，支持发送在线消息。Plugin 文件夹中包含样式文件和多个组件，style 文件夹、tools.wxs 文件、talk-view 组件、local-stream 组件、remote-stream 组件和 JRTCSDK-Wechat.js 库是必需的，视频客服访客插件还需要集成 guest 组件，将它们一起放入项目 components 文件夹下即可。

4.2 引入组件

在项目的 WXML 文件中和 JSON 文件中按以下方式引入即可，引入组件时，可以将部分登录参数通过以下方式传给组件，通过该方法传递过后，登录时则无需传递已传递的登录参数。还需要传递回调函数给组件。注：引入绑定时，需全部小写，不能使用驼峰命名。

```
XML |
1 <guest
2   id="guest"
3   signal-server="http://192.168.4.171:19993"
4   room-server="http://192.168.4.171:9000"
5   app-key="05c97ce0d0a84d2592334093"
6   app-name="视频客服访客"
7   display-name=""
8   token=""
9   token-type="jrtc_access"
10  token-extra-param=""
11  bindonlogin="onLogin"
12  bindonlogout="onLogout"
13  ...>
14 </guest>
```

```
JSON |
1 {
2   "usingComponents": {
3     "guest": "/components/guest/guest"
4   }
5 }
```

回调函数按需传递，没有使用的无需传递。

4.3 插件通用属性

属性	类型	参数/属性值	必填	说明
signal-server	String		否	signal环境
room-server	String		否	room环境
app-key	String		否	appKey
app-name	String		否	app名称

display-name	String		否	昵称
token	String		否	token
token-type	String		否	token类型
token-extra-param	String		否	token校验拓展参数
subscribeWidth	Number	1920	否	订阅视频清晰度-宽度
subscribeHeight	Number	1080	否	订阅视频清晰度-高度
closeCameraBackgroundColor	String	#1C1C1E	否	关闭摄像头后背景颜色
videoHoldLastFrame	Boolean	false	否	保留最后一帧画面
closeCameraIconSrc	String		否	关闭摄像头图标样式
newMessageToast	String	您收到了一条消息， 点击消息查看	否	收到新消息的提示语
toastColor	String	#FFFFFF	否	消息提示文字颜色
toastBackgroundColor	String	#252525cc	否	消息提示遮罩背景色
bindonlogin	Function	result, reason	否	登录的回调
bindonlogout	Function	reason	否	登出的回调
bindonclientstatechanged	Function	state, oldState	否	登录状态的回调
bindononlinemessagesreceived	Function	message, userId	否	收到在线消息的回调
bindononlinemessagesendresult	Function	result, operatorId	否	发送在线消息的回调

bindongetallgroups	Function	result, groups	否	查询所有业务组的回调
bindoncallstatechange	Function	type, inviter, reason	否	通话状态改变的回调
bindonmemberjoin	Function	participant	否	成员加入的回调
bindonmemberleave	Function	participant	否	成员离开的回调
bindonmemberupdate	Function	participant, changeParam	否	成员属性更新的回调
bindonurgentresult	Function	agree	否	加急申请的回调
bindonholdstatechange	Function	hold	否	通话保持的回调
bindoncalltypechange	Function	callType	否	通话类型改变的回调
bindonerror	Function	event	否	错误事件的回调

4.4 登录

在登录时需要设置几个参数，如下表。

参数名	参数含义	是否必填
signalServer	signal环境	是
roomServer	room环境	是
appKey	appkey	是
userId	账号	是
displayName	昵称	否
token	token	否

tokenType	token类型	否
tokenExtraParam	token校验拓展参数	否

设置好参数后，即可调用 `login` 方法进行登录。调用时需要把这些参数作为一个对象传给 `login` 方法，引入时已传递的参数则可以不用再传递。下面是一个简单的示例。

```

1  const guestComponent = this.selectComponent('#guest');
2  const clientEnv = {
3    signalServer: 'http://192.168.4.171:19993',
4    roomServer: 'http://192.168.4.171:9000',
5    appKey: '05c97ce0d0a84d2592334093',
6    userId: '',
7    displayName: '',
8    token: '',
9    tokenType: 'jrtc_access',
10   tokenExtraParam: '',
11  };
12  guestComponent.login(clientEnv);

```

4.5 登出

登出无需设置参数，只需调用 `logout` 方法即可，下面是一个简单的示例。

```

1  const guestComponent = this.selectComponent('#guest');
2  guestComponent.logout();

```

4.6 获取登录状态

可以使用登录状态的回调获取当前登录状态，它可以获取两个参数 `state`、`oldState`，`state` 表示当前的登录状态，`oldState` 表示之前的状态。下面是一个示例。


```

1  /**
2   * 登录状态变化通知
3   *
4   * @param state    当前状态值
5   * @param oldState 之前状态值
6   *    state = 1 未登录
7   *    state = 2 登录中
8   *    state = 3 已登录
9   *    state = 4 登出中
10  */
11  onClientStateChanged(e){
12      const { state, oldState } = e.detail;
13      console.log('onClientStateChanged oldState: ', oldState, 'state: ', state);
14  },

```

4.7 查询业务组

登录成功之后即可调用 `queryGroup` 方法后，可在 `onGetAllGroups` 回调中获取所有业务号信息，获取之后即可通过业务号进行呼叫。

```

1  const guestComponent = this.selectComponent('#guest');
2  guestComponent.queryGroup();

```

4.8 呼叫

在呼叫之前，需要填写一些参数。

参数名	参数含义	是否必填
telNumber	业务号	业务号呼叫时必填
agentUserId	座席ID	点对点呼叫时必填
callType	呼叫类型	是

4.8.1 点对点呼叫

在呼叫之前，需要填写坐席ID —— agentUserId。

在填好坐席ID之后，便可调用 `call` 方法进行呼叫。调用时需要传递三个对象参数 —— `callParams`、`callAgentSettings`、`mediaSettings` 给该方法，当为点对点呼叫时，`callType` 便为 `oneToOneCall`，下面是一个示例。

```
JavaScript |
1  const guestComponent = this.selectComponent('#guest');
2  const callParams = {
3    telNumber: '',
4    agentUserId: 'xxxxxx',
5    callType: 'oneToOneCall',
6  };
7  const callAgentSettings = {
8    extraInfo: '',
9    heartbeatTime: 20,
10   heartbeatTimeout: 60,
11   securityType: 0,
12   autoRecord: false,
13   vipPower: 0,
14 };
15 const mediaSettings = {
16   isSelfDefined: false,
17   videoDefinition: 0,
18   captureResolution: 1,
19   maxFrameRate: 24,
20   svcResolution: '1 90 70 180 180 360 480 720 1350',
21   maxResolution: 0,
22   renderType: 0,
23   videoEncodeType: 0,
24   audioEncodeType: 0,
25   traceId: '',
26 };
27 guestComponent.call(callParams, callAgentSettings, mediaSettings);
```

4.8.2 业务号呼叫

在查询到业务组后，便可以输入获取到的业务号进行业务号呼叫，参数名为 `telNumber`。

在填好业务号后，便可调用 `call` 方法进行呼叫。调用时需要传递三个对象参数 —— `callParams`、`callAgentSettings`、`mediaSettings` 给该方法，当为业务号呼叫时，`callType` 便为 `serviceCall`，下面是一个示例。

```

1  const guestComponent = this.selectComponent('#guest');
2  const callParams = {
3    telNumber: 'xxxxxx',
4    agentUserId: '',
5    callType: 'serviceCall',
6  };
7  ...
8  guestComponent.call(callParams, callAgentSettings, mediaSettings);

```

4.9 来电

4.9.1 接听来电

当坐席或其他用户来电时，可以调用 `answer` 方法接听来电，调用成功后会进入到通话页面，下面是一个示例。

```

1  const guestComponent = this.selectComponent('#guest');
2  guestComponent.answer();

```

4.9.2 拒接来电

当坐席或其他用户来电时，也可以调用 `term` 方法拒绝接听来电，下面是一个示例。

```

1  const guestComponent = this.selectComponent('#guest');
2  guestComponent.term();

```

4.10 获取通话状态

可以使用通话状态的回调获取当前通话状态，它可以获取四个参数 `type`、`incomingType`、`inviter`、`reason`，`type` 表示当前的通话状态，`incomingType` 表示来电类型，`inviter` 表示邀请成员对象，`reason` 表示挂断原因。下面是一个示例。

```

1  /**
2   * 通话状态改变回调
3   *
4   * @param type 访客通话状态改变类型，即以下情况会收到此回调：
5   * - {@link GuestCallStateChangeType.CALLING CALLING} 访客呼叫成功
6   * - {@link GuestCallStateChangeType.WAITING WAITING} 访客呼叫成功后等待座席接
   听
7   * - {@link GuestCallStateChangeType.INCOMING INCOMING} 作为第三方访客收到通
   话邀请
8   * - {@link GuestCallStateChangeType.TALKING TALKING} 通话接通（第三方访客）或
   被接通（主访客）
9   * - {@link GuestCallStateChangeType.TERMED TERMED} 通话挂断或被挂断
10  * @param incomingType 来电类型，当 type == {@link GuestCallStateChangeType.I
   NCOMING INCOMING} 时有效
11  * @param inviter 邀请成员对象，当 type == {@link GuestCallStateChangeType.I
   NCOMING INCOMING} 时有效
12  * @param reason 挂断原因，只在 type 为 {@link GuestCallStateChangeType.TERM
   ED TERMED} 时需要关注
13  */
14  onCallStateChange(e){
15      const { type, incomingType, inviter, reason } = e.detail;
16      console.log('onCallStateChange type:', type, 'inviter:', inviter);
17  }

```

4.11 通话参数

在呼叫之前，可以设置通话参数，如下表。

参数名	参数含义	是否必填
<code>extraInfo</code>	随路参数	否
<code>heartbeatTime</code>	心跳间隔	是
<code>heartbeatTimeout</code>	心跳超时	是
<code>securityType</code>	加密方式(Number)	是
<code>autoRecord</code>	是否开启服务器录制	是
<code>vipPower</code>	呼叫权重	是

下面是一个示例：

```

1 callAgentSettings: {
2   extraInfo: '', // 随路参数
3   heartbeatTime: 20, // 通话心跳间隔
4   heartbeatTimeout: 60, // 通话心跳超时时间
5   securityType: 0, // 媒体加密方式
6   autoRecord: false, // 服务器录制
7   vipPower: 0, // 呼叫权重
8 },

```

4.12 媒体参数

在呼叫之前，还可以设置媒体参数，如下表。

参数名	参数含义	是否必填
isSelfDefined	自定义配置	是
videoDefinition	视频清晰度	否
captureResolution	采集分辨率(Number)	自定义配置开启时必须填
frameRate	帧率(0-30)	
svcResolution	SVC(说明)	
maxResolution	最大分辨率(Number)	自定义配置关闭时必须填
renderType	渲染模式(Number)	否
videoEncodeType	视频编解码(Number)	是
audioEncodeType	音频编解码(Number)	是
traceld	日志跟踪ID	否

下面是一个示例：

```

1  mediaSettings: {
2    isSelfDefined: false, // 自定义配置
3    videoDefinition: 0, // 房间视频清晰度
4    captureResolution: 1, // 采集分辨率
5    maxFrameRate: 24, // 最大帧率
6    svcResolution: [ // svc 分辨率
7      { resolution: 90, bitRate: 70 },
8      { resolution: 180, bitRate: 180 },
9      { resolution: 360, bitRate: 480 },
10     { resolution: 720, bitRate: 1350 },
11   ],
12   maxResolution: 0, // 最大分辨率
13   renderType: 0, // 渲染模式
14   videoEncodeType: 0, // 通话视频编码
15   audioEncodeType: 0, // 通话视音频编码
16   traceId: '', // 日志跟踪ID
17 },

```

4.13 发送在线消息

在发送在线消息时，需要填写两个参数，如下表。

参数名	参数含义	是否必填
userId	接收者账号	是
message	消息内容	是

在填好账号消息内容后，即可调用 [sendOnlineMessage](#) 方法发送在线消息，调用时需要把这些参数作为一个对象传给 [sendOnlineMessage](#) 方法。下面是一个示例。

```

1  const guestComponent = this.selectComponent('#guest');
2  const sendOnlineMessageInfo = {
3    userId: '',
4    message: '',
5  };
6  guestComponent.sendOnlineMessage(sendOnlineMessageInfo);

```

4.14 预检测功能

使用预检测功能需要引入 checked 组件，该组件可用于检测设备扬声器、麦克风和摄像头是否有用。

```
XML |
1 <view wx:if="{{isChecked}}">
2   <checked
3     id="checked"
4     bindischeckingchanged="isCheckingChanged">
5   </checked>
6 </view>
```

当需要控制 checked 组件的显示隐藏时，可以传入 ischeckingchanged 方法，并定义一个参数——isChecked。下面是一个示例。

```
JavaScript |
1 data: {
2   isChecked: false,
3 },
4 isCheckingChanged() {
5   this.setData({
6     isChecked: !this.data.isChecked,
7   });
8 },
```

4.15 获取版本号信息

引入组件后，可调用 getVersion 方法获取当前组件版本信息，下面是一个示例。

```
JavaScript |
1 const guestComponent = this.selectComponent('#guest')
2 const version = guestComponent.getVersion()
3 console.log('version:', version);
```

4.16 回调函数

配置如下：

```
1 <quest
2   id="guest"
3   bindonlogin="onLogin"
4   bindonlogout="onLogout"
5   bindonclientstatechanged="onClientStateChanged"
6   bindononlinemessagereceived="onOnlineMessageReceived"
7   bindononlinemessagesendresult="onOnlineMessageSendResult"
8   bindongetallgroups="onGetAllGroups"
9   bindoncallstatechange="onCallStateChange"
10  bindonmemberjoin="onMemberJoin"
11  bindonmemberleave="onMemberLeave"
12  bindonmemberupdate="onMemberUpdate"
13  bindonurgentresult="onUrgentResult"
14  bindonholdstatechange="onHoldStateChange"
15  bindoncalltypechange="onCallTypeChange"
16  bindonerror="onError">
17 </quest>
```

4.16.1 登录的回调

示例代码:

```
1 /**
2  * 登录结果回调
3  *
4  * @param result true 表示登录成功, false 表示登录失败
5  * @param reason 登录失败原因, 当 result 为 false 时该值有效
6  */
7 onLogin(e){
8   const { result, reason } = e.detail;
9   console.log('onLogin result: ', result, ' reason: ', reason);
10 },
```

4.16.2 登出的回调

示例代码:


```
1  /**
2   * 登出回调
3   *
4   * @param reason 登出原因
5   */
6  onLogout(e){
7      const { reason } = e.detail;
8      console.log('onLogout reason: ', reason);
9  },
```

4.16.3 登录状态的回调

示例代码：

```
1  /**
2   * 登录状态变化通知
3   *
4   * @param state    当前状态值
5   * @param oldState 之前状态值
6   */
7  onClientStateChanged(e){
8      const { state, oldState } = e.detail;
9      console.log('onClientStateChanged oldState: ', oldState, 'state: ', state);
10 },
```

4.16.4 发送在线消息的回调

示例代码：

```

1  /**
2   * 在线消息发送结果回调
3   *
4   * @param result      发送结果是否成功
5   *                    - true: 发送成功
6   *                    - false: 发送失败
7   * @param operatorId 操作id, 对应 {@link JRTCClient.sendOnlineMessage sendOnlineMessage} 的返回值
8   */
9  onSendMessageResult(e){
10     const { result, operatorId } = e.detail;
11 },

```

4.16.5 收到在线消息的回调

示例代码:

```

1  /**
2   * 收到在线消息回调
3   *
4   * @param message 消息内容
5   * @param userId 对方用户ID
6   */
7  onOnlineMessageReceived(e){
8     const { userId, message } = e.detail;
9 },

```

4.16.6 获取业务组的回调

示例代码:

```

1  /**
2   * 获取业务号列表结果回调
3   *
4   * 访客调用 {@link JRTCGuest.queryAllGroups queryAllGroups} 接口获取业务号列表，会收到此回调。
5   * @param groups 座席业务实体对象列表，获取失败时为 undefined
6   * @param result 获取结果，true 表示获取成功，false 表示获取失败
7   */
8  onGetAllGroups(e){
9      const { result, groups } = e.detail;
10     console.log('onGetAllGroups result, groups', result, groups);
11 }

```

4.16.7 通话状态的回调

示例代码：

```

1  /**
2   * 通话状态改变回调
3   *
4   * @param type 访客通话状态改变类型，即以下情况会收到此回调：
5   * - {@link GuestCallStateChangeType.CALLING CALLING} 访客呼叫成功
6   * - {@link GuestCallStateChangeType.WAITING WAITING} 访客呼叫成功后等待座席接听
7   * - {@link GuestCallStateChangeType.INCOMING INCOMING} 作为第三方访客收到通话邀请
8   * - {@link GuestCallStateChangeType.TALKING TALKING} 通话接通（第三方访客）或被接通（主访客）
9   * - {@link GuestCallStateChangeType.TERMED TERMED} 通话挂断或被挂断
10  * @param incomingType 来电类型，当 type == {@link GuestCallStateChangeType.INCOMING INCOMING} 时有效
11  * @param inviter 邀请成员对象，当 type == {@link GuestCallStateChangeType.INCOMING INCOMING} 时有效
12  * @param reason 挂断原因，只在 type 为 {@link GuestCallStateChangeType.TERMED TERMED} 时需要关注
13  */
14  onCallStateChange(e){
15      const { type, incomingType, inviter, reason } = e.detail;
16      console.log('onCallStateChange type:', type, 'inviter:', inviter);
17  }

```

4.16.8 成员加入的回调

示例代码：

```
1  /**
2   * 通话中有新成员加入回调
3   *
4   * 当第三方成员加入时，已在通话中的所有成员会收到此回调，而新加入的成员不会收到此回调。
5   * @param participant 新加入的成员对象
6   */
7  onMemberJoin(e){
8      const { participant } = e.detail;
9      console.log('onMemberJoin participant: ', participant);
10 }
```

4.16.9 成员离开的回调

示例代码：

```
1  /**
2   * 通话中有成员离开回调
3   *
4   * 通话中有成员离开通话时，剩余的成员会收到此回调，而离开的成员不会收到此回调。
5   * @param participant 离开的成员对象
6   */
7  onMemberLeave(e){
8      const { participant } = e.detail;
9      console.log('onMemberLeave participant: ', participant);
10 }
```

4.16.10 成员属性更新的回调

示例代码：

```

1  /**
2   * 通话中成员属性更新回调
3   *
4   * 常用的有 {@link JRTCRoomParticipantChangeParam.audio audio}、{@link JRTC
   RoomParticipantChangeParam.video video}等。<br>
5   * 例如当通话中有成员关闭视频传输，通话中所有成员都会收到此回调。
6   * @param participant 属性更新的成员对象
7   * @param changeParam 更新的属性对象
8   */
9  onMemberUpdate(e) {
10     const { participant, changeParam } = e.detail;
11     console.log('onMemberUpdate participant, changeParam:', participant, cha
        ngeParam);
12 },

```

4.16.11 加急申请的回调

示例代码：

```

1  /**
2   * 座席处理加急的结果回调
3   *
4   * 访客调用 {@link JRTCGuest.requestUrgent requestUrgent} 接口请求加急后，座席
   同意或拒绝加急请求，访客会收到此回调获得加急请求应答结果。
5   * @param agree 加急是否通过，true 表示座席同意了访客的加急请求，false 表示不同意
6   */
7  onUrgentResult(e){
8     const { agree } = e.detail
9     console.log('onUrgentResult agree: ', agree);
10 },

```

4.16.12 通话保持的回调

示例代码：

```
1  /**
2   * 收到通话保持或取回的回调
3   *
4   * 通话中座席通过保持通话或取回通话，通话中所有成员都会收到此回调。
5   * @param hold true 表示通话被保持，false 表示通话取回
6   */
7  onHoldStateChange(e){
8      const { hold } = e.detail;
9      console.log('onHoldStateChange hold:', hold);
10 },
```

4.16.13 通话类型的回调

示例代码：

```
1  /**
2   * 音视频通话切换回调
3   *
4   * 通话中的访客和座席切换音视频通话模式，通话中所有成员都会收到此回调。
5   * @param callType 通话模式
6   * - {@link CallType.AUDIO AUDIO} 语音通话模式
7   * - {@link CallType.VIDEO VIDEO} 视频通话模式
8   */
9  onCallTypeChange(e){
10     const { callType } = e.detail;
11     console.log('onCallTypeChange callType:', callType);
12 },
```

4.16.14 错误事件的回调

示例代码：

```

1  /**
2   * 上报事件回调
3   *
4   * @param event 事件对象
5   */
6  onError(e) {
7      const { event } = e.detail;
8      console.log('onError:', event);
9  },

```

4.17 参数

4.17.1 securityType（加密方式）

securityType	说明
0	不加密
1	SRTP
2	SM4

4.17.2 captureResolution（采集分辨率）

captureResolution	说明
0	360p
1	720p
2	1080p

4.17.3 maxResolution（最大分辨率）

maxResolution	说明
0	360p
1	720p

2	1080p
---	-------

4.17.4 renderType（渲染模式）

renderType	说明
0	FullContent
1	FullScreen

4.17.5 videoEncodeType（视频编解码方式）

videoEncodeType	说明
0	H264
1	AV1

4.17.6 audioEncodeType（音频编解码方式）

audioEncodeType	说明
0	OPUS
1	PCMA
2	PCMU

4.17.7 reason

参数值	含义
NONE = 0	正常
OTHER	其他原因
INVALID_PARAM	参数错误
CLI_STATE_CANNOT_LOGIN	当前状态无法再次登录
CLI_TIMEOUT	登录超时

CLI_NETWORK	登录网络异常
ROOM_TIMEROUT	接入房间网络超时
ROOM_NERWORK	接入房间网络异常
ROOM_KICKED	被踢出房间
ROOM_OFFLINE	接入房间掉线
ROOM_QUIT	主动离开房间
ROOM_OVER	房间关闭
ROOM_FULL	房间满员
ROOM_INVALID_PASSWORD	房间密码无效
REASON_CONF_NUMBER_NOT_FOUND	该房间号的房间不存在
ROOM_APP_CONCURRENCY_FUL	服务器房间成员总数上限（移动端房间人数）
ROOM_ALL_CONCURRENCY_FUL	服务器房间成员总数上限（总房间人数）
ROOM_INTERNAL_ERROR	房间异常（服务端下发）
ROOM_JOIN_LICENCE_LIMIT	会场人员上限，licence限制
ROOM_AGENT_RELEASE	录制cd被释放

4.17.8 clientState（登录参数）

参数值	含义
1	未登录
2	登录中
3	已登录
4	登出中

4.17.9 participant（成员信息）

参数名	含义
userId	成员的账号
userUri	成员的地址
virtual	是否为虚拟用户

4.17.10 callState（通话状态）

参数值	含义
CALLING (0)	呼叫中
WAITING (1)	呼叫等待中
INCOMING (2)	收到来电
TALKING (3)	通话中
TERMED (4)	通话挂断/结束/空闲

4.17.11 callType（通话类型）

参数值	含义
AUDIO (0)	音频通话
VIDEO (1)	视频通话