527meeting

RTCPeer接口设计说明书

文档作者：527meeting Team

创建日期：2016-06-06

当前版本：3.0.8

文档控制

修改记录

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **日期** | **作者** | **版本** | **参考版本** | **备注** |
| 2016-06-06 | 江水 | 1.0.0 |  | 支持音视频功能 |
| 2017-04-10 | 江水 | 1.0.1 |  | 增加服务器录制 |
| 2018-05-02 | 江水 | 1.0.2 |  | 增加本地录制，屏幕分享 |
| 2018-06-16 | 江水 | 2.0.0 |  | 增加IM功能 |
| 2018-09-19 | 江水 | 2.0.1 |  | 增加Java IM接口，平台对接方案 |
| 2020-09-16 | 江水 | 2.0.2 |  | 增加chat |
| 2020-09-18 | 江水 | 2.0.3 |  | 增加会议更新、删除、查询接口 |
| 2020-09-27 | 江水 | 3.0.0 |  | 替换低层，更新到remax流媒体版本 |
| 2020-10-27 | 江水 | 3.0.1 |  | 增加翻译功能 |
| 2020-12-2 | 江水 | 3.0.2 |  | 增加获取设备接口，优化public方法 |
| 2021-6-25 | 江水 | 3.0.3 |  | 增加服务器录制接口 |
| 2022-02-14 | 江水 | 3.0.4 |  | 增加修改密码和重置密码 |
| 2022-03-07 | 江水 | 3.0.5 |  | 支持视频多码流 |
| 2022-08-19 | 江水 | 3.0.6 |  | 增加本地麦克风监听，会议室人员声音事件 |
| 2022-9-29 | 江水 | 3.0.7 |  | 增加指定使用摄像头和麦克风设备 |
| 2022-12-1 | 江水 | 3.0.8 |  | 增加签名接口，修改认证接口 |

目录

[简介 8](#_Toc1052673077)

[RTCPeer接口 9](#_Toc1306121153)

[1.创建RTCPeer实例 9](#_Toc400378837)

[2. 登录（rtcPeer.login） 10](#_Toc1100847408)

[3. 登出（rtcPeer.logout） 10](#_Toc1370767351)

[4.当前用户（rtcPeer. getCurrentUser） 11](#_Toc282303241)

[5.当前公司（rtcPeer. getCurrentCompany） 11](#_Toc879195264)

[6. 查询房间（rtcPeer. getRoomById） 12](#_Toc1947310688)

[7. 创建房间（rtcPeer. createRoom） 13](#_Toc799952936)

[8. 修改房间（rtcPeer. upadteRoom） 14](#_Toc1561365132)

[9. 删除房间（rtcPeer. deleteRoom） 15](#_Toc1761090831)

[10. 创建临时房间（rtcPeer.newTempRoom） 16](#_Toc2033973663)

[11. 修改房间密码（rtcPeer. modifyRoomPwd） 18](#_Toc1350661095)

[12. 检查房间密码（rtcPeer. checkRoomPwd） 18](#_Toc1658874875)

[13. 检查房间是否有密码（rtcPeer. checkRoomHasPwd） 19](#_Toc2077318771)

[14. 添加房间成员（rtcPeer. addRoomMan） 20](#_Toc1854934918)

[15. 删除房间成员（rtcPeer. deleteRoomMen） 20](#_Toc871063327)

[16. 查询房间成员列表（rtcPeer. queryRoomMen） 21](#_Toc565315290)

[17. 查询房间列表（rtcPeer. queryMyRooms） 21](#_Toc786424702)

[18. 检查房间是否正在使用（rtcPeer. checkActiveRoom） 22](#_Toc1825602876)

[19. 创建预约会议（rtcPeer. createAppointment） 23](#_Toc1808672243)

[20. 查询会议（rtcPeer. queryAppointmentById） 24](#_Toc723364816)

[21. 修改会议（rtcPeer. updateAppointment） 26](#_Toc687536845)

[22. 删除会议（rtcPeer. deleteAppointment） 27](#_Toc1969733055)

[23. 查询会议列表（rtcPeer. queryAppointments） 27](#_Toc1843036880)

[24. 查询参会信息（rtcPeer. getRoomByInviteCode） 28](#_Toc616717832)

[25. 查询房间会议记录列表（rtcPeer. queryAppointmentsByRoomId） 29](#_Toc1420522002)

[26. 查询用户列表（rtcPeer. searchUserByOrgId） 30](#_Toc1137583915)

[27. 添加用户（rtcPeer. addUser） 31](#_Toc325950164)

[28. 修改用户（rtcPeer.modifyUser） 32](#_Toc13622851)

[29. 删除用户（rtcPeer.deleteUser） 33](#_Toc1325990175)

[30. 查询流媒体当前状态（rtcPeer. getMediaState） 34](#_Toc1479066306)

[31. 录制查询（rtcPeer.getRecordByRange） 35](#_Toc1544190917)

[32. 删除录制（rtcPeer.deleteRecords） 36](#_Toc876868024)

[33. 创建即时会议（rtcPeer.instantAppointment） 37](#_Toc1488093654)

[34. 修改密码（rtcPeer.modifyPasswrod） 38](#_Toc795489816)

[35. 重置密码（rtcPeer.resetPasswrod） 39](#_Toc1711634937)

[Webrtc音视交互功能 40](#_Toc1904934594)

[1. 进入会议（rtcPeer.meeting. enter） 41](#_Toc1549511882)

[2. 退出会议（rtcPeer.meeting. exit） 42](#_Toc112013605)

[3. 结束会议（rtcPeer.meeting. finish） 42](#_Toc1416984463)

[4. 推流到流媒体（rtcPeer.meeting. publish） 43](#_Toc1811708058)

[5. 播放远端视频（rtcPeer.meeting. play） 45](#_Toc206699993)

[6. 关闭播放远端视频（rtcPeer.meeting. closePlay） 46](#_Toc1525725152)

[7. 打开或关闭本地视频（rtcPeer.meeting. updateVideo） 47](#_Toc1907884484)

[8. 打开或关闭本地麦克风（meeting. updateMute） 47](#_Toc1736189231)

[9. 指定用户静音消息（rtcPeer.meeting. muteMsg） 48](#_Toc124609981)

[10.全体静音消息（rtcPeer.meeting. muteAll） 49](#_Toc523394842)

[11.踢人消息（rtcPeer.meeting. kickOut） 50](#_Toc604091382)

[12.举手消息（rtcPeer.meeting. raiseHand） 51](#_Toc1808657905)

[13.开始本地录制（rtcPeer.meeting.startLocalRecord） 52](#_Toc482386050)

[14.停止本地录制（rtcPeer.meeting.stopLoaclRecord） 53](#_Toc711574925)

[15.暂停本地录制（rtcPeer.meeting. pauseLocalRecord） 53](#_Toc103334332)

[16.恢复本地录制（rtcPeer.meeting.resumeLocalRecord） 53](#_Toc1573331148)

[17.设置主持人（rtcPeer.meeting.setCompere） 54](#_Toc1010458925)

[18.屏幕共享（rtcPeer.meeting. shareScreen） 55](#_Toc482471999)

[19.取消屏幕共享（rtcPeer.meeting.stopShareScreen） 56](#_Toc8636121)

[20.播放屏幕共享（rtcPeer.meeting. playShareScreen） 56](#_Toc1265881298)

[21.挂断屏幕共享（rtcPeer.meeting. hangupShareScreen） 57](#_Toc546484657)

[22.切换摄像头（rtcPeer.meeting.flipsCamera） 57](#_Toc2127555627)

[23.视频切语音（rtcPeer.meeting. audioOnlyMeeting） 58](#_Toc77216792)

[24.发送静音消息（rtcPeer.meeting.muteMsg） 60](#_Toc702500356)

[25.事件监听（rtcPeer. on） 62](#_Toc58392086)

[26. 会议聊天（rtcPeer.meeting.chat） 63](#_Toc2143246370)

[27. 会议私聊天（rtcPeer.meeting.privateChat） 64](#_Toc1798529459)

[28. 会议翻译（rtcPeer.meeting.transcribe） 65](#_Toc2052285888)

[29. 会议翻译是否打开（rtcPeer.meeting.isTranscribe） 67](#_Toc2034065149)

[30.获取当前会议时长（rtcPeer.meeting. getMeetingTimeLength） 67](#_Toc740782650)

[31.开始屏幕录制（rtcPeer.meeting.startRecordScreen） 68](#_Toc1371296891)

[32.停止屏幕录制（rtcPeer.meeting.stopRecordScreen） 68](#_Toc592347433)

[33.获取本地音视频设备（rtcPeer.meeting.getMediaDevices） 68](#_Toc1996602586)

[34.获取本地视频设备（rtcPeer.meeting.getMediaVideoDevices） 69](#_Toc320194880)

[35.开始服务端录制（rtcPeer.meeting.startRecord） 69](#_Toc2068812425)

[36.停止服务端录制（rtcPeer.meeting.stopRecord） 70](#_Toc622698398)

[37.检查地音视频设备（rtcPeer.meeting.checkDevice） 70](#_Toc1004163355)

[38.检查地视频设备（rtcPeer.meeting.checkVideo） 71](#_Toc2047009359)

[39.检查地音频设备（rtcPeer.meeting.checkAudio） 72](#_Toc1398271773)

[40.切换视频播放源（rtcPeer.meeting.switchVideoBySim） 73](#_Toc840139690)

[41.单个视频播放源切换（rtcPeer.meeting.switchVideoBySim） 73](#_Toc522790805)

[42.视频播放最大数量（rtcPeer.meeting.setVideoMax） 74](#_Toc1189459758)

[43.声控切换流是否激活（rtcPeer.meeting.publishStreamActive） 74](#_Toc324882783)

[44.修改会议名称（rtcPeer.meeting.updateMeetingName） 75](#_Toc1401503207)

[45.修改参会成员名称（rtcPeer.meeting.updateClientName） 76](#_Toc1463759753)

[46.切换音视频设备（rtcPeer.meeting.switchDevice） 78](#_Toc1984992286)

[47.监听本地麦克风音量（rtcPeer.meeting.getMicVolume） 78](#_Toc606894657)

[IM音视交互功能 79](#_Toc1678660596)

[1.发起呼叫（rtcPeer. call） 80](#_Toc1755966333)

[2.同意呼叫请求（rtcPeer. agree） 81](#_Toc1805881657)

[3.拒绝呼叫请求（rtcPeer. reject） 82](#_Toc1066626148)

[4.取消呼叫请求（rtcPeer. cancel） 84](#_Toc1739667927)

[5.响铃消息（rtcPeer. ring） 85](#_Toc608995184)

[6.忙消息（rtcPeer. busy） 86](#_Toc474995886)

[7.连接成功消息（rtcPeer. connected） 88](#_Toc1059140103)

[8.自定义事件消息 （rtcPeer. doEvent） 90](#_Toc475761138)

[9. 呼叫注册 （rtcPeer. doRegister） 90](#_Toc1035828585)

[IM音视交互功能Java接口 91](#_Toc1668585513)

[1.呼叫接口 92](#_Toc2075254465)

[2.同意呼叫请求接口 93](#_Toc1519882328)

[3.拒绝呼叫请求接口 93](#_Toc344305631)

[4.取消呼叫请求接口 94](#_Toc1423795199)

[5.事件消息通用接口 95](#_Toc315631072)

[6.呼叫注册接口 96](#_Toc526819014)

[7.响铃消息 97](#_Toc172091717)

[8.忙消息 97](#_Toc1832498757)

[9.连接成功消息 97](#_Toc1743627272)

[10.创建临时房间接口 97](#_Toc581713542)

[11.登出接口 98](#_Toc1513939250)

[12. 创建预约会议 98](#_Toc1390725094)

[轻会议平台对接方案 100](#_Toc704640910)

[1. 生成签名接口 101](#_Toc1674944812)

[2. 用户信息认证接口 101](#_Toc1581810408)

[Web Url接口 104](#_Toc1787461043)

[1.通过房间ID进入指定房间 104](#_Toc709011818)

[2.通过会议ID进入指定房间 104](#_Toc2122351570)

[Bean类说明 105](#_Toc659460320)

[1.User 106](#_Toc386496073)

[2. Company 106](#_Toc1848950383)

[3. Room 106](#_Toc1220714991)

[4.Client 107](#_Toc1645573946)

[5. Appointment 107](#_Toc1866904356)

简介

527meeting包括一套技术平台和多种终端SDK，用于简化WebRTC视频通讯开发，并同时提高了WebRTC的稳定性，也增加更多易用的功能。

使用527meeting，开发者可以在Web端和移动端，通过较简单的直接调用，将视频通讯功能嵌入任何软件中。还可以进一步对通讯的连接、效果、录制等进行编程控制。527meeting控制台的图形界面，即可容易直观地调试通讯状况，查看统计信息，分配管理权限。

总之，527meeting将音视频实时通讯中复杂的底层连接和优化工作封装到云端和SDK中。这样一来，开发者既能享受WebRTC本身高效、低价、无插件、点对点等优秀特性，又可以大大简化开发，避免将珍贵的时间和人力投入到无休止的与最新WebRTC标准对接、适配和质量调试的过程中。未来还能逐步享受到527meeting 平台更加全面易用的功能。为了更好的对接，我们编写RTCPeer.js,以Javascript类包的形式提供对接接口，并提供了原型Demo,希望能够更好的提供我们的服务。

527meeting Team

2016-06-06

RTCPeer接口

RTCPeer.js 是以jquery为基础包的类包，提供基于WebRTC技术的音视频相关功能接口，通过RTCPeer.js您可以轻松构建一个音视频通读。在使用RTCPeer时，我们首先要新建RTCPeer实例，然后通过RTCPeer实例来调用音视频相关功能接口。

1.创建RTCPeer实例

* 1. 概述

构造函数

* 1. 参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| host | String | M | 服务器地址 |
| relayServer | String | M | Relay 服务器地址，域名+端口号（域名不要加https） |
| socket | Boolean | O | true 支持呼叫功能（和控制服务器保持socket连接），false不支持,  默认不保持连接，使用时才连接 |
| nginx | Boolean | O | 是否使用nginx和流媒体连接 |
| transport | String | O | 协议：udp或tcp,默认传tcp |
| relay | String | O | relay或all, 默认传relay |
| **sign** | **String** | **M** | **api签名，有签名才能调用，通过java生成签名接口可获取，文档中有** |
| logout | Boolean | O | 支持非登录用户呼叫 |

* 1. 示例

1. window.rtcPeer = new RTCPeer({ mode: 'pc', host: 'https://shipinhuiyi.hedait.cn/app', relayServer: "shipinhuiyi.hedait.cn:30001",relay: 'relay',transport: 'tcp'
2. });

2. 登录（rtcPeer.login）

2.1概述

用户登录接口

2.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| name | String | M | 账号 |
| password | String | M | 密码 |

2.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| user | Object | 用户信息 |
| company | Object | 公司信息 |
| result | number | 0：成功，1：用户名密码不正确 |

2.4示例

1. rtcPeer.login({
2. name: username,
3. password: password
4. }).then(function(d) {
5. $("#account").text(d.user.name);
6. });

3. 登出（rtcPeer.logout）

3.1概述

用户登录接口

3.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| userId | number | M | 用户唯一标识 |

3.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| result | number | 0：退出成功，1：退出失败 |

3.4示例

1. rtcPeer.logout({ userId: rtcPeer.getCurrentUser().id }).then(function(d) {
2. showLogin(true);
3. });

4.当前用户（rtcPeer. getCurrentUser）

4.1概述

登录后，获取当前用户信息

4.2参数

无

4.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| user | Object | 当前用户对象 |

4.4示例

1. var currentUser = rtcPeer.getCurrentUser();

5.当前公司（rtcPeer. getCurrentCompany）

5.1概述

登录后，获取当前用户所在公司信息

5.2参数

无

5.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| company | Object | 当前用户所在公司信息 |

5.4示例

1. var currentOrg = rtcPeer. getCurrentCompany();

6. 查询房间（rtcPeer. getRoomById）

6.1概述

根据房间唯一标识查询房间信息

6.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| roomId | number | M | 房间唯一标识码 |

6.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| room | Object | 虚拟房间信息 |

6.4示例

1. var room=null;
2. rtcPeer. getRoomById ({
3. roomId:1012
4. }).then(function(d){
5. room=d.room
6. });

7. 创建房间（rtcPeer. createRoom）

7.1概述

创建虚拟房间

7.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| roomtype | number | M | 房间类型 1音频会议 2视频会议 3视频回传 |
| name | String | M | 房间名称 |
| desc | String | M | 房间信息描述 |
| pictureuri | String | M | 房间Logo |
| definition | String | M | 清晰度 1080p 720p 480p 240p |
| numberLimit | number | M | 房间人数上限 |
| isPublic | Boolean | M | ture公共会议室 false 私有会议室 |

7.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| roomId | number | 虚拟房间唯一标识 |
| result | number | 0 成功 1失败 |

7.4示例

1. var req = {
2. isPublic: false,
3. roomtype: 2,
4. name: 'jiangshui',
5. desc: "test",
6. definition: '720p',
7. numberLimit: 100
8. }
9. rtcPeer.createRoom(req).then(function(data) {
10. Load.hide();
11. if (data.result == 0) {
12. $location.url("room");
13. } else {
14. $scope.message = $translate.instant("message\_037");
15. }
16. apply($scope);
17. });

8. 修改房间（rtcPeer. upadteRoom）

8.1概述

修改虚拟房间

8.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| roomType | number | M | 房间类型 1音频会议 2视频会议 3视频回传 |
| roomId | number | M | 虚拟房间唯一标识 |
| name | String | M | 房间名称 |
| desc | String | M | 房间信息描述 |
| pictureuri | String | M | 房间Logo |
| definition | String | M | 清晰度 1080p 720p 480p 240p |
| numberLimit | number | M | 房间人数上限 |
| isPublic | Boolean | M | ture公共会议室 false 私有会议室 |

8.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| roomId | number | 虚拟房间唯一标识 |
| result | number | 0 成功 1失败 |

8.4示例

1. var req = {
2. roomId: room.id,
3. isPublic: $scope.rightIndex == 1,
4. roomtype: $scope.typeObj.type,
5. name: $scope.nameTxt,
6. desc: $scope.descTxt,
7. pictureuri: $scope.logo,
8. definition: $scope.definition.value,
9. numberLimit: $scope.numberTxt \* 1
10. }
11. rtcPeer.updateRoom(req).then(function(data) {
12. Load.hide();
13. if (data.result == 0) {
14. Session.setItem('roomdetailid', data.roomId);
15. $location.url("roomdetail");
16. } else {
17. $scope.message = $translate.instant("message\_037");
18. }
19. apply($scope);
20. });

9. 删除房间（rtcPeer. deleteRoom）

9.1概述

删除虚拟房间

9.1概述

根据房间唯一标识查询房间信息

9.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| roomId | number | M | 房间唯一标识码 |

9.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| result | number | 0 成功 1失败 |

9.4示例

1. rtcPeer.deleteRoom({
2. roomId: $scope.room.id
3. }).then(function(data) {
4. Load.hide();
5. if (data.result == 0) {
6. $location.url("room");
7. } else {
8. note($translate.instant("message\_024"), false);
9. }
10. apply($scope);
11. });

10. 创建临时房间（rtcPeer.newTempRoom）

10.1概述

创建临时房间

10.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| name | String | M | 房间名称 |
| orgId | number | M | 公司唯一标识，私有部署可传2 |
| userId | number | M | 用户唯一标识，私有部署可传2 |
| roomtype | number | M | 房间类型 1音频会议 2视频会议 3 远程培训 |
| name | String | M | 房间名称 |
| desc | String | O | 房间信息描述 |
| pictureuri | String | O | 房间Logo |
| definition | String | O | 清晰度 1080p 720p 480p 240p,不传默认480p |
| numberLimit | number | M | 房间人数上限 |
| isPublic | Boolean | M | ture公共会议室 false 私有会议室 |

10.3返回

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| result | Long | 0成功 |
| roomId | Long | 会议室房间唯一标识 |

10.4示例

1. rtcPeer.newTempRoom({
2. "roomtype": 2,
3. "numberLimit": 200,
4. "name": "Test Room",
5. "userId":2,
6. "orgId":2,
7. "isPublic":false
8. }).then(function(d){
9. console.log(d)
10. })

11. 修改房间密码（rtcPeer. modifyRoomPwd）

11.1概述

修改房间密码

11.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| roomId | number | M | 会议室房间唯一标识 |
| password | String | M | 密码 |

11.3返回

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| result | Long | 0成功 |

11.4示例

1. rtcPeer.modifyRoomPwd({
2. roomId:599,
3. password:"1234"}).then(function(d){
4. console.log(d
5. )})

12. 检查房间密码（rtcPeer. checkRoomPwd）

12.1概述

检查房间密码是否正确

12.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| roomId | number | M | 会议室房间唯一标识 |
| password | String | M | 密码 |

12.3返回

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| result | Long | 1成功0 失败 |

12.4示例

1. rtcPeer.checkRoomPwd({
2. roomId:599,
3. password:"1234"}).then(function(d){
4. console.log(d)
5. })

13. 检查房间是否有密码（rtcPeer. checkRoomHasPwd）

13.1概述

检查房间密码是否有密码

13.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| roomId | number | M | 会议室房间唯一标识 |

13.3返回

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| isHasPwd | Boolean | true: 有密码; flase：没有密码 |

13.4示例

1. rtcPeer.checkRoomHasPwd({
2. roomId:599
3. }).then(function(d){
4. console.log(d)
5. })

14. 添加房间成员（rtcPeer. addRoomMan）

14.1概述

添加私有房间成员，添加到房间里的成员可以通过房间列表查询到此房间。

14.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| roomId | number | M | 会议室房间唯一标识 |
| userId | number | M | 用户唯一标识 |

14.3返回

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| result | Long | 0成功 |

14.4示例

1. rtcPeer.addRoomMan({roomId:9,userId:8})

15. 删除房间成员（rtcPeer. deleteRoomMen）

15.1概述

将指定成员从私有房间中移除。

15.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| roomId | number | M | 会议室房间唯一标识 |
| moderatorId | number | M | 用户唯一标识 |

15.3返回

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| result | Long | 0成功 |

15.4示例

1. rtcPeer.deleteRoomMen({roomId:81,moderatorId:138})

16. 查询房间成员列表（rtcPeer. queryRoomMen）

16.1概述

根据房间唯本标识码查询房间内成员。

16.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| roomId | number | M | 会议室房间唯一标识 |

16.3返回

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| result | Long | 0成功 |
| moderators | Array | moderator对象 |

16.4示例

1. rtcPeer.queryRoomMen({roomId:8}).then(function(d){console.log(d)})

17. 查询房间列表（rtcPeer. queryMyRooms）

17.1概述

查询当前用户有权限进入的房间列表

17.2参数

无

17.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| rooms | Array | 虚拟房间列表 |

17.4示例

1. var rooms=[];
2. rtcPeer. queryRoomsUniq().then(function(d){
3. rooms=d.rooms
4. });

18. 检查房间是否正在使用（rtcPeer. checkActiveRoom）

18.1概述

检查房间是正在使用

18.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| roomId | number | M | 房间唯一标识码 |

18.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| result | number | 0 无使用 1正在使用中 |

18.4示例

1. rtcPeer.checkActiveRoom({
2. roomId: $scope.room.id \* 1
3. }).then(function(data) {
4. Load.hide();
5. if (data.result == 0) {
6. Session.setItem('room', $scope.room);
7. $location.url("editroom");
8. } else {
9. note($translate.instant("message\_107"), false);
10. }
11. apply($scope);
12. });

19. 创建预约会议（rtcPeer. createAppointment）

19.1概述

创建指定时间的会议记录，可以根据会议相关信息进入虚拟房间视频通信。

19.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| name | String | O | 会议名称，默认‘即时会议’ |
| description | String | O | 描述 |
| definition | String | O | 清晰度 1080p 720p 480p 240p,不传默认480p |
| type | Number | O | 1:音频会议 2网络会议 3在线培训 |
| startTime | Date | O | 会议开始时间，默认接口调用时间 |
| endTime | Date | O | 会议结束时间，默认开始时间+1小时 |
| members | Array | O | 邀请成员，数组内成员格式为字符串,  格式如下：  item.memberId + "," + item.name + ", ," + item.email + "," + item.id + ",Etc/GMT+8," + item.role  注：memberId 默认为0，  id为视频系统的用户id, 可以不传  name:用户昵称  email:邮件，可以为null  角色role:  会议场模式：1是主持人 2普通参会人员；  培训模式：1是主持人 2是讲师，3普通参会人员，建议只邀请主持人和讲师 |
| topics | Array | O | 网络会议 子议题  数组内的Objet格式：  { "title":“议题”, "content": "具体内容" } |

19.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| roomId | number | 会议对应房间的唯一标识码 |
| meetingId | number | 会议唯一标识码 |

19.4示例

1. rtcPeer.createAppointment({name:"test",description:'abc',type:2})
2. .then(function(d){console.log(d)})
3. var date = new Date();
4. rtcPeer.createAppointment({
5. name:"shui test abc",
6. description:'abc',
7. type:2,
8. members:["0,ceshi,,,80,Etc/GMT+8,2"],
9. startTime:new Date(),
10. endTime:new Date(date.setHours(date.getHours() + 1))})
11. .then(function(d){
12. console.log(d)
13. })

20. 查询会议（rtcPeer. queryAppointmentById）

20.1概述

根据会议唯一标识码查询会议信息。

20.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| meetingId | Number | M | 会议唯一标识码 |

20.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| meeting | Object | 会议信息 |
| members | Array | 邀请成员，数组内成员格式为对象,  格式如下：  {“codeId”: “test” , “name”：“章三”，“email”：test@qq.com”,”role”:”2”,id:1**}**  注：  codeId 为用户对接的唯一标示  id为视频系统的用户id,  name:用户昵称 email:邮件，  角色role: 可以不传 默认普通参会人员  会议场模式：1是主持人 2普通参会人员；  培训模式：1是主持人 2是讲师，3普通参会人员，建议只邀请主持人和讲师 |
| topics | Array | 网络会议 子议题  数组内的Objet格式：  { "title":“议题”, "content": "具体内容" } |

20.4示例

1. var meeting=null;
2. rtcPeer. queryAppointmentById ({meetingId:1012})
3. .then(function(d){
4. meeting =d.meeting;
5. });

21. 修改会议（rtcPeer. updateAppointment）

21.1概述

更新会议相关信息。

21.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| name | String | M | 会议名称，默认‘即时会议’ |
| description | String | O | 描述 |
| type | Number | M | 1:音频会议 2网络会议 3在线培训 |
| startTime | Date | M | 会议开始时间，默认接口调用时间 |
| endTime | Date | M | 会议结束时间，默认开始时间+1小时 |
| meetingId | number | M | 会议唯一标识码 |
| members | Array | O | 邀请成员，数组内成员格式为String,  格式如下：  “**memberId,name,email,id, Etc/GMT+8,role**”  注：memberId 默认为0，  id为用户的唯一标识码id,  name:用户昵称  email:邮件，可以为null  角色role:  会议场模式：1是主持人 2普通参会人员；  培训模式：1是主持人 2是讲师，3普通参付人员，建议只邀请主持人和讲师 |
| topics | Array | O | 网络会议 子议题  数组内的Objet格式：  { "title":“议题”, "content": "具体内容" } |

21.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| roomId | number | 会议对应房间的唯一标识码 |
| meetingId | number | 会议唯一标识码 |

21.4示例

1. rtcPeer.updateAppointment({
2. name:"testabc",
3. description:'abceee',
4. type:1,
5. meetingId:255270}).then(function(d){
6. console.log(d)
7. })

22. 删除会议（rtcPeer. deleteAppointment）

22.1概述

根据会议唯一标识码删除指定会议。

22.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| type | Number | O | 1:音频会议 2网络会议 3在线培训 |
| meetingId | number | M | 会议唯一标识码 |

22.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| result | number | 0 删除成功 2正在使用中，不能删除 |

23. 查询会议列表（rtcPeer. queryAppointments）

23.1概述

根据时间段查询会议列表。

23.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| startTime | Date | O | 查询会议开始时间 |
| endTime | Date | O | 查询会议结束时间，默认开始时间+24\*3小时 |
| pageNo | number | O | 页码 从0开始 |
| pageSize | number | O | 一页数据量 |

23.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| meetings | Array | 会议列表 |

23.4示例

1. rtcPeer.queryAppointments({}).then(function(d){console.log(d)})

24. 查询参会信息（rtcPeer. getRoomByInviteCode）

24.1概述

根据会议唯一标识码或随机码查询会议信息。

24.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| code | Number | M | 会议唯一标识码 |
| inviteCode | Number | O | 随机码,随码入会，有值用随机码，没值用code |

24.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| result | number | 0成功，3未开始，5没有查询到，其它无效会议号 |
| room | Object | 当前会议所在虚拟房间信息 |
| meeting | Object | 当前会议信息 |

24.4示例

1. rtcPeer.getRoomByInviteCode({
2. code: +code,
3. *//inviteCode: +code*
4. }).then(function(d) {
5. if (d.result == 0) {
6. \_enterRoom(d.room);
7. } else if (d.result == 3) {
8. *//当前会议还未开始，请稍后再打开链接！*
9. *//var minutes = Math.floor(d.milliseconds / (1000 \* 60));*
10. $("#message").text("当前会议还未开始，请稍后再打开链接！");
11. } else {
12. *//"当前会议链接已失效!",*
13. $("#message").text("当前会议链接已失效!");
14. }
15. });

25. 查询房间会议记录列表（rtcPeer. queryAppointmentsByRoomId）

25.1概述

根据房间唯一标识码查询对应的会议列表。

25.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| roomId | number | M | 房间唯一标识码 |
| pageNo | number | M | 页码 从0开始 |
| pageSize | number | M | 一页数据量 |

24.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| resultList | Array | 会议列表 |

25.4示例

1. rtcPeer.queryAppointmentsByRoomId({
2. roomId: roomId \* 1,
3. pageNo: (pageNo - 1),
4. pageSize: 10
5. }).then(function(data) {
6. $scope.roomLogs = data.resultList;
7. showTotalCount(data.totalCount || 0)
8. if (data.totalPage > 1) {
9. $scope.totalPage = data.totalPage;
10. var pageList = [];
11. for (var i = 1; i <= data.totalPage; i++) {
12. pageList.push(i);
13. };
14. $scope.pageList = pageList;
15. } else {
16. $scope.totalPage = 1;
17. $scope.pageList = [1];
18. }
19. apply($scope);
20. })

26. 查询用户列表（rtcPeer. searchUserByOrgId）

26.1概述

根据公司唯一标识码查询用户列表。

26.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| orgId | number | M | 公司唯一标识码 |
| userId | number | M | 用户唯一标识 |
| pageNo | number | M | 页码 从0开始 |
| pageSize | number | M | 一页数据量 |

26.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| users | Array | 用户列表 |
| totalCount | number | 数据总数 |
| totalPage | number | 数据总页数 |

26.4示例

1. rtcPeer.searchUserByOrgId({
2. orgId:2,
3. userId:6,
4. pageNo: 0,
5. pageSize: 12,}).then(function(d){
6. console.log(d)
7. })

27. 添加用户（rtcPeer. addUser）

27.1概述

将用户添加到指定公司。

27.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| orgId | number | M | 公司唯一标识码 |
| account | number | M | 用户帐号 |
| password | String | M | 默认密码 |
| level | number | M | 用户级别，4:普通用户,5:参会用户 |
| name | String | M | 用户名称 |

27.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| result | number | 0 成功1 账号重复 |

27.4示例

1. rtcPeer.addUser({
2. account:'jiangshui123456',
3. password:'123456',
4. email:'12345@qq.com',
5. orgId:2,level:2,
6. name:'jiangabc'}).then(function(d){
7. console.log(d)
8. })

28. 修改用户（rtcPeer.modifyUser）

28.1概述

将用户添加到指定公司。

28.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| userId | number | M | 用户唯一标识码 |
| phone | String | O | 手机号 |
| address | String | O | 地址 |
| job | String | O | 用户级别，4:普通用户,5:参会用户 |
| name | String | O | 用户名称 |
| companyname | String | O | 对应的所在公司 |

28.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| result | number | 0 成功1 账号重复 |
| user | Object | 更新后的用户对象 |

28.4示例

1. rtcPeer.modifyUser({
2. userId:2
3. name:'jiangshui123456',
4. job:'123456',
5. address:'12345',
6. phone:’15012345678’,
7. name:'jiangabc'}).then(function(d){
8. console.log(d)
9. })

29. 删除用户（rtcPeer.deleteUser）

29.1概述

根据用户唯一标识码删除用户。

29.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| orgId | number | M | 公司唯一标识码 |
| userId | number | M | 用户唯一标识 |

29.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| result | number | 0 成功1 账号重复 |

29.4示例

1. rtcPeer.deleteUser({
2. orgId:2,
3. userId:84
4. }).then(function(d){
5. console.log(d)
6. })

30. 查询流媒体当前状态（rtcPeer. getMediaState）

30.1概述

查询流媒体当前状态。

30.2参数

无

30.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| object | Array | 多台流媒体数组 |

数组中的media参数说明

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| mediaAddress | String | 流媒体地址 |
| mediaId | number | 流媒体对应的唯一标识 |
| mediaName | String | 流媒体名称 |
| participants | number | 参会总人员 |
| roomSize | String | 会议室数量 |
| status | number | 0正常，其它:不正常 |

30.4示例

1. rtcPeer.getMediaState().then(function(d){
2. })

Received data::

*[{…}]*

0:

* + 1. mediaAddress: "huawei.527meeting.com"
    2. mediaId: 1
    3. mediaName: "huawei"
    4. participants: 9
    5. roomSize: "2"
    6. status: 0

31. 录制查询（rtcPeer.getRecordByRange）

31.1概述

网络录制查询接口

31.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| orgId | number | M | 公司唯一标识码 |
| userId | number | O | 用户唯一标识，传值查询当前用户的录制，不传查询公司下所有录制 |
| name | String | O | 模糊查询 |
| startDate | String | O | 查询会议开始时间，传入UTC时间格式 “yyyyMMddHHmmss” |
| endDate | String | O | 查询会议结束时间，传入UTC时间格式 “yyyyMMddHHmmss” |
| pageNo | number | M | 页码 从0开始 |
| pageSize | number | M | 一页数据量 |

31.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| recordList | Array | 录制数据 |
| totalPage | number | 总页数 |

31.4示例

1. rtcPeer.getRecordByRange({
2. userId:2,
3. name:"",
4. pageNo:0，
5. pageSize: 10}).then(function(d){
6. console.log(d)
7. })

**注：域名+’app/upload/videos/’+record.comment**

32. 删除录制（rtcPeer.deleteRecords）

32.1概述

删除录制接口

32.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| recordIds | Array | M | 录制Id数组 |

32.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| result | number | 0成功 1失败 |

32.4示例

1. rtcPeer.deleteRecords({
2. recordIds: [1,2,3]
3. }).then(function(data) {
4. });

33. 创建即时会议（rtcPeer.instantAppointment）

33.1概述

创建即时会议

33.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| name | String | M | 会议名称 |
| orgId | number | M | 公司唯一标识，私有部署可传2 |
| startTime | String | M | 会议开始时间 ，传入UTC时间格式 “yyyyMMddHHmmss” |
| endTime | String | M | 会议结束时间 ，传入UTC时间格式“yyyyMMddHHmmss” |
| type | number | M | 房间类型 1音频会议 2视频会议 3 远程培训 |
| isPublic | Boolean | M | ture公共会议室 false 私有会议室 |

33.3返回

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| meetingId | number | 会议唯一标识 | |
| roomId | number | 对应的房间id | |
| result | number | 0 成功 1失败 | |

33.4示例

1. rtcPeer.instantAppointment({
2. "type": 2,
3. "name": "Test Room",
4. "userId":2,
5. "orgId":2,
6. "isPublic":false
7. }).then(function(d){
8. console.log(d)
9. })

34. 修改密码（rtcPeer.modifyPasswrod）

34.1概述

修改用户密码接口

34.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| oldpassword | String | M | 旧密码 |
| password | String | M | 新密码 |
| userId | Number | M | 用户唯一标识 |

34.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| result | number | 0成功 1旧密码不正确，2修改失败 |

34.4示例

1. rtcPeer.modifyPasswrod({
2. userId:2,
3. oldpassword:“222222”，
4. password:”123456”
5. }).then(function(data) {
6. });

35. 重置密码（rtcPeer.resetPasswrod）

35.1概述

重置密码接口

35.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| password | String | M | 新密码 |
| userId | Number | M | 用户唯一标识 |

35.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| result | number | 0成功 1失败 |

35.4示例

1. rtcPeer.resetPasswrod({
2. userId:2,
3. password:”123456”
4. }).then(function(data) {
5. });

Webrtc音视交互功能

针对音视频互通功能需求，提示围绕音视频相关的功能接口，使用开发用户能够方便快捷的搭建音视频会议。

注：监听onPublished事件，vCode == 'rePlay'时，一定要修改成员列表中的client的peerId 值，不然播放远端流播放不成功

1. rtcPeer.off("onPublished").on("onPublished", function (data) {
2. if (data.roomId == room.id && data.client.id != myClient.id) {
3. if (data.vCode == 'rePlay') {
4. *//同步流媒体通道成员Id \*\*\*\*\*\*\*\*\*重要，不同步会影响play\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\**
5. clientList.forEach(function (item) {
6. if (item.id == data.client.id) {
7. item.peerId = data.client.peerId;
8. }
9. });
10. }
11. });

1. 进入会议（rtcPeer.meeting. enter）

1.1概述

进入会议房间

1.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| userId | Object | M | 用户唯一标识 |
| roomId | number | M | 会议虚拟房间唯一标识 |
| roomType | number | M | 1.音频会议 2视频会议 3在线培训 |
| username | String | O | 用户名称，游客时使用 |

1.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| client | Object | 虚拟参会人员信息,当前用户client |
| members | Array | 虚拟参会人员列表，不包含当前用户client |
| room | Object | 当前会议所在虚拟房间信息 |

1.4示例

1. rtcPeer.meeting.enter({
2. roomId: tempRoom.id,
3. roomType: tempRoom.roomType,
4. maxNumber: 4,
5. codec: currentCodec
6. }).then(function(d) {
7. room = d.room;
8. myClient = d.client;
9. clientList.push(d.client);
10. clientList = clientList.concat(d.members);
11. *//播放自己的视频*
12. rtcPeer.meeting.publish({
13. audio: true,
14. video: true,
15. domId: 'local'
16. });
17. ……
18. }, function(d) {
19. console.error("createMeeting:::  " + d);
20. });
21. });

2. 退出会议（rtcPeer.meeting. exit）

2.1概述

退出会议

2.2参数

无

2.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| result | number | 0：成功，1：失败 |

2.4示例

1. meeting.exit().then(function() {
2. showVideoArea(false);
3. $('#local').empty();
4. $('#network').empty();
5. });

3. 结束会议（rtcPeer.meeting. finish）

3.1概述

所有参会人员全部退出会议。

3.2参数

无

3.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| result | number | 0：成功，1：失败 |

3.4监听事件

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| onFinish | String | M | 会议结束消息事件 |

事件获取数据

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| userId | number | 调用finish操作的用户唯一标识 |

3.5示例

1. rtcPeer.meeting. finish ().then(function() {
2. showVideoArea(false);
3. $('#local').empty();
4. $('#network').empty();
5. });

4. 推流到流媒体（rtcPeer.meeting. publish）

4.1概述

创建通道和流媒体建立连接，使用本地的摄像头和麦克风获取音视频流，并将本地流推送到流媒体服务器上。Html video 标签要加autoplay muted，html要增加audio标签作为声音外放，如： <audio id="mixedaudio" autoplay playsinline></audio>，不加自动创建

4.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| audio | Boolean | M | 本地麦克风是否使用获取声音 |
| video | Boolean | M | 本地视频是否使用获取视频 |
| useVideo | Boolean | O | 是否使用摄像头,如果不使用摄像头，video属性设置为false,默认true |
| localTag | String | O | 本地视频挂载的Dom(Video tag id) |
| ~~remoteTag~~ | ~~String~~ | ~~O~~ | ~~远端视频挂载的Dom(Video tag id)，展示时是声控切换流（默认流）~~ |
| definition | String | O | 清晰度 1080p 720p 480p 240p |
| maxFrameRate | number | O | 最大帧率 ，范围1~30 |
| minFrameRate | number | O | 最小帧率,范围1~30,**注：**minFrameRate<=maxFrameRate |
| callback | Function | O | 成功后回调 |
| useCamera | String | O | 传摄像头deviceId时，使用指定摄像头，传any随机选择一个;不传默认any  **注**：可以使用rtcPeer.meeting.getMediaVideoDevices拿本地摄像头设备或rtcPeer.meeting.getMediaDevices拿所有设备，在结果中选择其中一个摄像头设备的deviceId |
| useMic | String | O | 传麦克风deviceId时，使用指定麦克风，传any随机选择一个;不传默认any  注：可以使用rtcPeer.meeting.getMediaDevices拿所有设备，在结果中选择其中一个麦克风设备的deviceId |
| device | Object | O | 通过rtcPeer.meeting.checkDevice检查本地设备结果，处理没有摄像头、麦克风设备等情况下，仍然可以进入；不传结果默认有摄像头、麦克风设备。 |

4.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| getBitrate | Function | 获取当前通道码率和帧率 |
| client | object | 虚拟参会方 |

4.4示例

Html:

1. <audio id="mixedaudio" autoplay playsinline></audio>
2. <video id="video\_0" autoplay muted></video>
3. <video id="remote" autoplay muted></video>

Js code:

1. rtcPeer.meeting.publish({
2. audio: true,
3. video: true,
4. localTag: 'video\_0',
5. remoteTag: 'remote',
6. });

5. 播放远端视频（rtcPeer.meeting. play）

5.1概述

播放其它参会议成员的视频。如：播放A的视频，必须A已经打开自己的本地视频

5.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| client | Object | M | 用户参会的虚拟身份 |
| domId | String | M | 视频挂载的Dom(Video tag id) |
| callback | Function | O | 播放成功回调 |

5.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| getBitrate | Function | 获取当前通道码率和帧率 |
| client | object | 虚拟参会方 |

5.4示例

1. meeting.play({
2. client: data.client,
3. domId: 'network',
4. callback: function(option) {
5. $('#network').append('<div class="name-tag" id="client' + option.client.id + '"><span>' + option.client.name + '</span></div>');
6. }
7. })

6. 关闭播放远端视频（rtcPeer.meeting. closePlay）

6.1概述

停止播放其它参会议成员的视频。

6.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| object | Object | M | 用户参会的虚拟身份 |
| callback | Function | O | 停止播放成功回调 |

6.3示例

1. rtcPeer.meeting.stopPlay({
2. client: data.client,
3. callback: function(option) {
4. $('#client' + option.client.id).remove();
5. }
6. });

7. 打开或关闭本地视频（rtcPeer.meeting. updateVideo）

7.1概述

打开或关闭本地视频。

7.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| client | Object | M | 用户参会的虚拟身份 |
| video | Boolean | M | true :打开视频 false:关闭视频 |
| callback | Function | O | 回调函数 |

7.3示例

1. meeting.updateVideoMode({
2. video: isOpenVideo,
3. client: myClient,
4. callback: function(myClient) {}
5. });

8. 打开或关闭本地麦克风（meeting. updateMute）

8.1概述

打开或关闭本地麦克风。

8.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| isMute | Boolean | M | false:打开麦克风 true:关闭麦克风 |

8.3示例

1. rtcPeer.meeting. updateAudioMode (isAudio);

9. 指定用户静音消息（rtcPeer.meeting. muteMsg）

9.1概述

发送关闭或打开指定用户的麦克风消息,指定用户接收到消息后，将自己静音或取消静音。

9.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| userId | number | M | Client中的id值 |
| roomId | number | M | 当前会议房间的唯一标识（room id） |
| isMute | Boolean | M | false :打开麦克风 true:关闭麦克风 |
| super | String | O | ‘fromSuper’ 主持人发送 |

9.3监听事件

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| onMute | String | M | 静音消息事件 |

事件获取数据

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| userId | number | M | Client中的id值 |
| roomId | number | M | 当前会议房间的唯一标识（room id） |
| isMute | Boolean | M | false :打开麦克风 true:关闭麦克风 |
| super | String | O | ‘fromSuper’ 主持人发送 |

9.4示例

1. rtcPeer.meeting. muteMsg ({
2. userId:1234,
3. roomId:1008,
4. isMute:true,
5. super: "fromSuper"
6. });
7. *//事件监听*
8. *//接收到主持人的是否禁音指令，执行对应操作*
9. rtcPeer.off("onMute ").on("onMute ", function(data) {
10. if (room.id == data.roomId) {
11. if (data.userId == myClient.id && data.fromSuper) {
12. rtcPeer.meeting. updateAudioMode (!data.isMute;);
13. }
14. }
15. });

10.全体静音消息（rtcPeer.meeting. muteAll）

10.1概述

主持人发送全体静音消息,用户接收到消息后，将自己静音。

10.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| userId | number | M | Client中的id值 |
| roomId | number | M | 当前会议房间的唯一标识（room id） |
| isMute | Boolean | M | false:打开麦克风 true:关闭麦克风 |

10.3监听事件

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| onMuteAll | String | M | 全体静音消息事件 |

事件获取数据

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| userId | number | M | Client中的id值 |
| roomId | number | M | 当前会议房间的唯一标识（room id） |
| isMute | Boolean | M | false:打开麦克风 true:关闭麦克风 |

10.4示例

1. meeting. muteAll ({
2. userId:1234,
3. roomId:1008,
4. isMute:true
5. });
6. *//事件监听*
7. *//接收到全体静音事件，不是主持人的用户将执行关闭麦克风操作*
8. rtcPeer.off("onMuteAll").on("onMuteAll", function(data) {
9. if (room.id == data.roomId) {
10. meeting.updateMute (data.isMute);
11. }
12. }）

11.踢人消息（rtcPeer.meeting. kickOut）

11.1概述

发送指定用户退出会议室消息，指定用户收到消息后，调用退出会议室方法，退出会议室。

11.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| userId | number | M | Client中的id值 |
| roomId | number | M | 当前会议房间的唯一标识（room id） |

11.3监听事件

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| tiren | String | M | 踢人消息事件 |

事件获取数据

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| userId | number | M | Client中的id值 |
| roomId | number | M | 当前会议房间的唯一标识（room id） |

11.4示例

1. rtcPeer.meeting. kickOut ({
2. userId:1234,
3. roomId:1008
4. });
5. *//事件监听*
6. rtcPeer.off("tiren ").on("tiren ", function(data) {
7. if (room.id == data.roomId) {
8. *//如果得到的userid和自己的id一样，调用退出会议方法，退出会议室*
9. if (data.userId == currentId) {
10. *//退出会议室并提示用户*
11. }
12. }
13. });

12.举手消息（rtcPeer.meeting. raiseHand）

12.1概述

发送举手消息给当前会议中的所有成员。

12.2参数

无

12.3监听事件

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| onRaiseHand | String | M | 踢人消息事件 |

事件获取数据

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| client | Object | M | Client |
| roomId | number | M | 当前会议房间的唯一标识（room id） |

12.4示例

1. rtcPeer.meeting. raiseHand()
2. *//事件监听*
3. rtcPeer.off("onRaiseHand").on("onRaiseHand", function(d) {
4. if (d.client.id != myClient.id) {
5. console.log('举手----', d.client.name);
6. }
7. });

13.开始本地录制（rtcPeer.meeting.startLocalRecord）

13.1概述

通过屏幕共享和录制插件抓屏，将屏幕和声音录制下来。**只支持PC端。**

13.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| fileName | String | M | 保存文件名 |
| callback | Function | M | 回调函数 |

13.3示例

1. rtcPeer.meeting.startLocalRecord ({
2. fileName：”abcd”
3. });

14.停止本地录制（rtcPeer.meeting.stopLoaclRecord）

14.1概述

停止本地录制。**只支持PC端，浏览器需要安装插件。**

14.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| callback | Function | M | 回调函数 |

14.3示例

1. rtcPeer.meeting.stopLocalRecord();

15.暂停本地录制（rtcPeer.meeting. pauseLocalRecord）

15.1概述

暂停本地录制。**只支持PC端。**

15.2参数

无

15.3示例

1. rtcPeer.meeting.pauseLocalRecord ();

16.恢复本地录制（rtcPeer.meeting.resumeLocalRecord）

16.1概述

恢复本地录制。**只支持PC端。**

16.2参数

无

16.3示例

1. rtcPeer.meeting.resumeLocalRecord ();

17.设置主持人（rtcPeer.meeting.setCompere）

17.1概述

指定用户设置成主持人,全体会议成员各终端会收到设置主持人消息，各终端收到事件可将在成员列表中找到对应人标记在主持人。

17.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| userId | number | M | Client中的id值 |
| roomId | number | M | 当前会议房间的唯一标识（room id） |

17.3监听事件

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| onCompere | String | M | 设置主持人消息事件 |

事件获取数据

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| userId | number | M | Client中的id值 |
| roomId | number | M | 当前会议房间的唯一标识（room id） |

17.4示例

1. rtcPeer.meeting. setCompere({
2. userId:1234,
3. roomId:1008
4. });
5. *//事件监听*
6. rtcPeer.off("onCompere ").on("onCompere ", function(data) {
7. if (room.id == data.roomId) {
8. }
9. });

18.屏幕共享（rtcPeer.meeting. shareScreen）

18.1概述

将自己的屏幕共享到其它客户端。**只支持PC端。一会议中，同一时间只能有一个人分享屏幕，html video标签一定要加autoplay muted。**

18.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| shareScreenId | number | M | 临时创建的分享唯一标识 |
| domId | String | M | video标签唯一dom id |

18.3监听事件

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| onShareScreen | String | M | 分享控制事件 |

事件获取数据

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| shareScreenId | number | shareScreenId 有值表是分享，为空表册关闭 |
| roomId | number | 分享所在的房间唯一标识 |
| client | Object | 分享的发起者 |

18.4监听本地事件

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| shareScreen | String |  | 分享屏幕者的本地事件 |

事件获取数据

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| type | String | type值： play, stop,事件类型  play:屏幕分享  stop:停止分享 |
| shareScreenId | number | shareScreenId 分享id |

18.5示例

Html:

1. <video id='screenvideo' autoplay muted></video>

Js code:

1. rtcPeer.meeting.shareScreen({
2. domId:’screenvideo’
3. });

19.取消屏幕共享（rtcPeer.meeting.stopShareScreen）

19.1概述

取消将自己的屏幕共享到其它客户端。**只支持PC端。**

19.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| callback | Function | O | 回调函数 |

19.3示例

1. rtcPeer.meeting. stopShareScreen ();

20.播放屏幕共享（rtcPeer.meeting. playShareScreen）

20.1概述

播放远端屏幕共享。**只支持PC端。html video标签一定要加autoplay muted。**

20.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| shareScreenId | number | M | 临时创建的分享唯一标识 |
| domId | String | M | video标签唯一dom id |

20.3示例

Html:

1. <video id='screenvideo' autoplay muted></video>

Js code:

1. rtcPeer.meeting.playShareScreen({
2. shareScreenId: data.shareScreenId \* 1,
3. domId:’’screenvideo’’
4. });

21.挂断屏幕共享（rtcPeer.meeting. hangupShareScreen）

21.1概述

非主持人停止播放主持人共享时，调用些接口**。**

21.2参数

无

21.3示例

1. rtcPeer.meeting.hangupScreenshare();

22.切换摄像头（rtcPeer.meeting.flipsCamera）

22.1概述

移动端切换摄像头。

22.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| callback | Function | O | 回调函数 |

22.4示例

1. rtcPeer.meeting.flipsCamera(function(){});

23.视频切语音（rtcPeer.meeting. audioOnlyMeeting）

23.1概述

将视频会议切换成语音会议模式。

23.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| client | Object | M | 用户参会的虚拟身份 |
| callback | Function | O | 回调函数 |

23.3监听事件

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| onPublished | String | M | 视频消息事件 |

事件获取数据

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| event | String | event 值： play, close, audioMode事件类型  play:播放视频  close:关闭视频  audioMode:视频电话切语音电话 |
| client | Object | 发起动作的用户信息 |

23.5示例

1. rtcPeer.meeting. audioOnlyMeeting ({
2. client: myClient,
3. callback: function(myClient) {
4. $("#local").css('display', 'block');
6. clientList.forEach(function(item) {
7. if (item.id == myClient.id) {
8. item.isOpenVideo = false;
9. }
10. });
11. changeVideoSize();
12. }
13. });
14. *//接收其他客户端播放或停止事件*
15. rtcPeer.off("onPublished").on("onPublished", function(data) {
16. if (data.roomId == room.id && data.client.id != myClient.id) {
17. if (data.event == 'close' || data.event == 'audioMode') {
18. console.info("关闭播放client  " + data.client.name);
19. rtcPeer.meeting.stopPlay({
20. client: data.client,
21. callback: function(client) {
22. $('#network' + client.id).remove();
23. clientList.forEach(function(item) {
24. if (item.id == data.client.id) {
25. item.isOpenVideo = false;
26. }
27. });
28. changeVideoSize();
29. }
30. });
31. if (isOpenVideo && data.event == 'audioMode') {
32. isOpenVideo = false;
33. $("#toggerVideo").removeClass('camera\_focus');
34. rtcPeer.meeting.updateVideoMode({
35. video: isOpenVideo,
36. client: myClient,
37. callback: function(myClient) {
38. $("#local").css('display', 'none');
39. clientList.forEach(function(item) {
40. if (item.id == myClient.id) {
41. item.isOpenVideo = isOpenVideo;
42. }
43. });
44. changeVideoSize();
45. }
46. });
47. }
48. }
49. }
50. });

24.发送静音消息（rtcPeer.meeting.muteMsg）

24.1概述

通过发送指定某用户的麦克风打开关闭。

24.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| userId | number | M | 用户唯一标识 |
| roomId | number | M | 房间唯一标识 |
| isMute | Boolean | M | 是否静音，true静音，false取消静音 |

24.3监听事件

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| onMute | String | M | 静音消息事件 |

事件获取数据

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| userId | number | 用户唯一标识 |
| roomId | number | 房间唯一标识 |
| isMute | Boolean | 是否静音，true静音，false取消静音 |

24.4示例

1. $scope.muteOther = function(user) {
2. rtcPeer.meeting.muteMsg({
3. roomId: room.id,
4. userId: user.id,
5. isMute: !user.muted
6. })
7. };
8. *//接收是否静音*
9. $scope.$on('onMute', function(e, data) {
10. if (room.id == data.roomId) {
11. angular.forEach($scope.userList, function(member) {
12. *//其它人看到自己的状态*
13. if (member.id == data.userId) {
14. member.muted = data.isMute;
15. return;
16. }
17. });
18. if (data.userId == client.id) {
19. $scope.isMute = data.isMute;
20. rtcPeer.meeting.setAudio(!$scope.isMute);
21. }
22. }
23. });

25.事件监听（rtcPeer. on）

25.1概述

监听事件方法。

25.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| Key | String | M | 消息事件名：换成对应事件名 |

主要事件

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| EnterRoom | String | 进入房间事件 |
| ExitRoom | String | 退出房间事件 |
| onLocalPublished | String | 本地流连接流媒体服务器成功 本地播放成功事件 |
| onPublished | String | 视频播放或停止事件 |
| onLinkSlow | String | code: 1001本地视频网络上传较差 code: 1002远端视频网络下载较差 |
| onLocalRecord | String | 录制本地流事件，type为  startLocalRecord 开始录制，  stopLocalRecord 停止录制，  pauseLocalRecord 暂停录制，  resumeLocalRecord 恢复录制 |

25.3示例

1. rtcPeer. on("EnterRoom", function(data) {
2. if (room.id == data.roomId) {
3. if (data.result == 9) {
4. *//当前账号在其他客户端上进入当前会议*
5. meeting.exitRoom().then(function() {
6. showVideoArea(false);
7. });
8. } else if (data.client.id != myClient.id) {
9. console.info("进入房间client  " + data.client.name);
10. clientList.push(data.client);
11. }
12. }
13. })

26. 会议聊天（rtcPeer.meeting.chat）

26.1概述

会议聊天。

26.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| userId | number | M | Client中的id值 ，当前用户 |
| roomId | number | M | 当前会议房间的唯一标识（room id） |
| content | String | M | 聊天内容 |

26.3监听事件

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| onChat | String | M | 聊天消息事件 |

事件获取数据

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| userId | number | Client中的id值 |
| roomId | number | 当前会议房间的唯一标识（room id） |
| content | String | 聊天内容 |
| client | Client | 发送信息的人信息 |

26.4示例

1. rtcPeer.meeting. chat({content:"1233eeeeeee"})
2. *//事件监听*
3. *//接收到主持人的是否禁音指令，执行对应操作*
4. rtcPeer.off("onChat").on("onChat", function(data) {
5. if (room.id == data.roomId) {
6. console.log("chat message", data.content);
7. }
8. });

27. 会议私聊天（rtcPeer.meeting.privateChat）

27.1概述

会议指定和某一个人聊天。

27.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| userId | number | M | Client中的id值 ，当前用户 |
| otherUserId | number | M | 指定聊天的的client id，当前会议已存的用户 |
| roomId | number | M | 当前会议房间的唯一标识（room id） |
| content | String | M | 聊天内容 |

27.3监听事件

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| onChat | String | M | 聊天消息事件 |

事件获取数据

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| userId | number | Client中的id值 |
| roomId | number | 当前会议房间的唯一标识（room id） |
| content | String | 聊天内容 |
| client | Client | 发送信息的人信息 |

27.4示例

1. rtcPeer.meeting.privateChat({content:"1233eeeeeee",otherUserId})
2. *//事件监听*
3. *//接收到主持人的是否禁音指令，执行对应操作*
4. rtcPeer.off("onPrivateChat").on("onPrivateChat", function(data) {
5. if (room.id == data.roomId) {
6. console.log("chat message", data.content);
7. }
8. });

28. 会议翻译（rtcPeer.meeting.transcribe）

28.1概述

会议中参会人员的声音翻译。

28.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| action | String | M | start|stop|pause|resume]启动翻译，停止翻译，暂停和恢复 |

28.3监听事件

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| onTranscription | String | M | 翻译消息事件 |

事件获取数据

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| var | String | M | 翻译文字 |
| onebest | String | M | 最终调整后的翻译文字（不是每次翻译后都 有onebest） |
| bg | String | M | 同一句翻译，唯一标识 |

28.4示例

1. *//开始翻译*
2. rtcPeer.meeting.transcribe({ action: start});
3. *//停止翻译*
4. rtcPeer.meeting.transcribe({ action: 'stop' });
5. *//接收翻译事件*
6. var transcriptionTxt = "";
7. var transcriptionNumber = ""; *//同样的编号属于一句话*
8. rtcPeer.off("onTranscription").on("onTranscription", function(d) {
9. console.log(d);
10. if (d.data) {
11. transcriptionTxt = d.data.var || d.data.onebest;
12. $("#transcribeTxt").text(transcriptionTxt);
13. }
14. });

29. 会议翻译是否打开（rtcPeer.meeting.isTranscribe）

29.1概述

会议中参会人员的声音翻译。

29.2参数

无

29.3返回数据

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| message | String | 描述 |
| result | Boolean | ture翻译打开状态，false翻译 关闭状态 |

29.4示例

1. Mixer.instance.api.isTranscribe().then(function(d){console.error(d)})
2. *//结果：*
3. {message: "+OK conference 100000508 transcription is on", sessid: "ba63d78c-5a4b-c541-dff4-ce13b337720b", result: true}
4. message:"+OK conference 100000508 transcription is on"
5. result:true
6. sessid:"ba63d78c-5a4b-c541-dff4-ce13b337720b"

30.获取当前会议时长（rtcPeer.meeting. getMeetingTimeLength）

30.1概述

获取当前正在进行的会议，已经进行多长时间，返回毫秒。

30.2示例

1. rtcPeer.utils.dateTimeFormate(rtcPeer.meeting.getMeetingTimeLength());
2. ……
3. "00:00:48"

31.开始屏幕录制（rtcPeer.meeting.startRecordScreen）

31.1概述

通过屏幕共享和录制插件抓屏，将屏幕和声音录制到服务端。**只支持PC端。**

30.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| fileName | String | O | 保存文件名 |

30.3示例

rtcPeer.meeting.startRecordScreen();

32.停止屏幕录制（rtcPeer.meeting.stopRecordScreen）

32.1概述

停止录制。**只支持PC端，浏览器需要安装插件。**

32.2示例

1. rtcPeer.meeting.stopRecordScreen();

33.获取本地音视频设备（rtcPeer.meeting.getMediaDevices）

33.1概述

获取本地麦克风、摄像头、外放列表。

33.2示例

1. rtcPeer.meeting.getMediaDevices(function(list){
2. console.log(“list”,list);
3. });

34.获取本地视频设备（rtcPeer.meeting.getMediaVideoDevices）

34.1概述

获取本地摄像头设备列表。如果没有摄像头设备，返回为空。

34.2示例

1. rtcPeer.meeting.getMediaVideoDevices(function(list){
2. console.log(“list”,list);
3. });

35.开始服务端录制（rtcPeer.meeting.startRecord）

35.1概述

发启服务端录制，将会议中用户声控切换的视频画面和声音录制下来**。**

35.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| fileName | String | O | 保存文件名，不传已当前时间为名称 |
| startTime | Date | O | 录制开始时间，不传默认当前时间 |

35.3返回

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| result | number | 0成功 |
| recordId | number | 录制唯一标识 |

35.4示例

1. rtcPeer.meeting.startLocalRecord ({
2. fileName：”abcd”
3. });

36.停止服务端录制（rtcPeer.meeting.stopRecord）

36.1概述

停止服务端录制。

36.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| recordId | number | M | 录制唯一标识 |

35.3返回

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| result | number | 0成功 |

36.4示例

1. rtcPeer.meeting.stopRecord({recordId:123});

37.检查地音视频设备（rtcPeer.meeting.checkDevice）

37.1概述

检查当前机器设备情况。

37.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| definition | String | M | 分辨率：1080p或720p |
| callback | function | M | 回调函数，结果返回 |

37.3返回

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| video | Object | code:1000有摄像头，1001:没有权限，1002：没有发现、不能读取或不支持，1003其它错误 |
| audio | Object | code:1000有麦克风，1001:没有权限，1002：没有发现、不能读取或不支持，1003其它错误 |

37.2示例

1. rtcPeer.meeting.checkDevice({
2. definition: tmpRoom.definition,
3. callback: function (data) {
4. if (data.video.code == 1001) {
5. *//摄像头或麦克风不存在*
6. showMessage("请允许使用当前摄像头");
7. } else if (data.audio.code == 1001) {
8. showMessage("请允许使用当前麦克风");
9. } else {
10. checkDeviceData = data;
11. }
12. }
13. }）

38.检查地视频设备（rtcPeer.meeting.checkVideo）

38.1概述

检查当前机器设备情况。

38.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| definition | String | M | 分辨率：1080p或720p |
| callback | function | M | 回调函数，结果返回 |

38.3返回

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| video | Object | code:1000有摄像头，1001:没有权限，1002：没有发现、不能读取或不支持，1003其它错误 |

38.2示例

1. rtcPeer.meeting.checkVideo({
2. definition: tmpRoom.definition,
3. callback: function (data) {}
4. })

39.检查地音频设备（rtcPeer.meeting.checkAudio）

39.1概述

检查当前机器设备情况。

39.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| callback | function | M | 回调函数，结果返回 |

39.3返回

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| audio | Object | code:1000有摄像头，1001:没有权限，1002：没有发现、不能读取或不支持，1003其它错误 |

39.4示例

1. rtcPeer.meeting.checkAudio({
2. callback: function (data) {}
3. })

40.切换视频播放源（rtcPeer.meeting.switchVideoBySim）

40.1概述

根据传参切换视频流大小，当前所有播放的视频流都将切换到同一大小。

40.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| sim | number | M | 0小流，1中流，2大流，默认大流 |

40.3示例

1. rtcPeer.meeting.switchVideoBySim(2);

41.单个视频播放源切换（rtcPeer.meeting.switchVideoBySim）

41.1概述

根据视频的domid和sim碜数，指定切换视频流大小。

41.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| sim | number | M | 0小流，1中流，2大流，默认大流 |
| domId | String | M | 对应的video标签Id |

41.3示例

1. rtcPeer.meeting.switchVideoSimByDom(‘video\_1’,2);

42.视频播放最大数量（rtcPeer.meeting.setVideoMax）

42.1概述

当前会议最大展示视频个数，包括自己本地视频。如设置4，播放第5个会自动将最先播放的视频流替换。

42.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| number | number | M | 最大播放视频个数 |

42.3示例

1. rtcPeer.meeting.setVideoMax(25);

43.声控切换流是否激活（rtcPeer.meeting.publishStreamActive）

43.1概述

声控切换流是否激活,true会从流媒体将声控切换流拉下来播放，false将将声控切换流关闭。

43.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| isPlay | Boolean | M | 是否打开 |

43.3示例

1. rtcPeer.meeting.publishStreamActive(false);

44.修改会议名称（rtcPeer.meeting.updateMeetingName）

44.1概述

在会议中，修改会议名称，并发送消息给所有参会方。

44.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| userId | number | M | 用户唯一标识 |
| roomId | number | M | 房间唯一标识 |
| appointmentName | String | M | 会议名称 |
| appointmentId | String | M | 会议唯一标识 room对象中appointmentId，进入会议时返回 |

24.3监听事件

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| onUpdateMeeting | String | M | 静音消息事件 |

事件获取数据

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| userId | number | 用户唯一标识 |
| roomId | number | 房间唯一标识 |
| appointmentName | String | 修改后的会议名称 |

44.4示例

1. rtcPeer.meeting.updateMeetingName({
2. roomId: room.id,
3. userId: user.id,
4. appointmentName: "abc",
5. appointmentId: room.appointmentId
6. })
8. *//接收是否静音*
9. $scope.$on('onUpdateMeeting', function(e, data) {
10. if (room.id == data.roomId) {
11. room.appointmentName = data.appointmentName;
12. }
13. });

45.修改参会成员名称（rtcPeer.meeting.updateClientName）

45.1概述

在会议中，修改参会成员名称，并发送消息给所有参会方。

45.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| userId | number | M | 用户唯一标识 |
| roomId | number | M | 房间唯一标识 |
| name | String | M | 成中名称 |

45.3监听事件

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| onUpdateClient | String | M | 修改成员名称消息事件 |

事件获取数据

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| userId | number | 用户唯一标识 |
| roomId | number | 房间唯一标识 |
| client | Object | 成员对象 |

45.4示例

1. rtcPeer.meeting.muteMsg({
2. roomId: room.id,
3. userId: user.id,
4. name: 'abc'
5. })
7. *//接收是否静音*
8. $scope.$on('onUpdateClient', function(e, data) {
9. if (room.id == data.roomId) {
10. angular.forEach($scope.userList, function(member) {
11. *//其它人看到自己的状态*
12. if (member.id == data.client.id) {
13. member.name= data.client.name;
14. return;
15. }
16. });
17. if (data.userId == user.id) {
18. user.name = data.client.name;
19. }
20. }

46.切换音视频设备（rtcPeer.meeting.switchDevice）

46.1概述

在会议中，通过rtcPeer.meeting.getMediaDevices接口获取音视频设备，将需要的音视频对象传切换到对应的设备上。

46.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| video | Object | O | 摄像头对象 |
| audio | Object | O | 麦克风对象 |
| callback | function | M | 成功后回调 |

**注：video、audio对象必须有一个，否者报错**

47.监听本地麦克风音量（rtcPeer.meeting.getMicVolume）

47.1概述

在会议中，获取当前使用的麦克风音量值。

47.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| callback | function | M | 返回声音大小 |

47.3示例

1. rtcPeer.meeting.getMicVolume(function (volume) {
2. console.log("volume",(volume / 30000).toFixed(2) \* 100 + '%');
3. })

48.监听会议中当前说话（rtcPeer.on("memberTalking")）

48.1概述

在会议中，监听会议中当前说话人

48.2参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| member | Object | 当前说话人 |

48.3示例

1. rtcPeer.off("memberTalking").on("memberTalking", function (data) {
2. console.log("memberTalking", data.member);
3. });

IM音视交互功能

提供在线用户之间的呼叫过程中的交互功能接口，通过接口可以实现发起呼叫、接收或拒绝呼叫，进入指定会议房间等需求。

1.发起呼叫（rtcPeer. call）

1.1概述

指定用户发起呼叫。

1.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| targetUserId | number | M | 用户User中的id值 |
| ownDeviceType | String | O | 设备类型 pc 、Android、iOS、webphone、tv |
| roomId | number | O | 呼叫后进入指定会议室 |

1.3监听事件

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| doInvite | String | M | 被呼叫者收到的邀请消息事件 |

事件获取数据

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| ownUserId | number | 发起呼叫者的用户唯一标识Id |
| ownDeviceType | String | 发起呼叫者的用户设备类型 |

1.4示例

1. rtcPeer.call({
2. targetUserId: callUser.id
3. }).then(function(d) {
4. if (d.result == 1) {
5. clearTimeout(ringTimeoutIndex);
6. clearTimeout(ringCallbackIndex);
7. showMessage("您呼叫的用户当前不在线。");
8. } else if (d.result == 0) {
9. room = { id: d.roomId, roomType: 2, name: '即时会议', isCall: true };
10. callRoom = room;
11. if (!isOtherBusy) {
12. callMessage(callUser);
13. }
14. clearTimeout(ringCallbackIndex);
15. }
16. });
17. *//事件监听*
18. rtcPeer.off("doInvite").on("doInvite", function(data) {
19. console.info("doInvite", data);
20. if (isAtRoom) {
21. rtcPeer.tellBusy({
22. 'targetUserId': data.ownUserId,
23. 'targetDeviceType': data.ownDeviceType
24. });
25. } else {
26. rtcPeer.ring({
27. 'targetUserId': data.ownUserId,
28. 'targetDeviceType': data.ownDeviceType
29. });
30. }
31. });

2.同意呼叫请求（rtcPeer. agree）

2.1概述

同意呼叫请求，并发送信息告诉呼叫。

2.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| targetUserId | number | M | 呼叫者用户唯一标识码 |
| targetDeviceType | String | O | 设备类型 pc 、Android、iOS、webphone、tv |
| ownDeviceType | String | O | 设备类型 pc 、Android、iOS、webphone、tv |

2.3监听事件

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| doAgree | String | M | 被呼叫者发送同意请求消息事件 |

事件获取数据

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| ownUserId | number | 被呼叫者的用户唯一标识Id |
| ownDeviceType | String | 被呼叫者的用户设备类型 |

2.4示例

1. rtcPeer.agree({
2. 'targetUserId': receiveUser.id,
3. 'targetDeviceType': receiveUser.deviceType
4. });
6. rtcPeer.off("doAgree").on("doAgree", function(data) {
7. });

3.拒绝呼叫请求（rtcPeer. reject）

3.1概述

拒绝呼叫请求，并发送信息告诉呼叫。

3.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| targetUserId | number | M | 呼叫者用户唯一标识码 |
| targetDeviceType | String | O | 设备类型 pc 、Android、iOS、webphone、tv |
| ownDeviceType | String | O | 设备类型 pc 、Android、iOS、webphone、tv |

3.3监听事件

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| reject | String | M | 被呼叫者发送拒绝请求消息事件 |

事件获取数据

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| ownUserId | number | 被呼叫者的用户唯一标识Id |
| ownDeviceType | String | 被呼叫者的用户设备类型 |

3.4示例

1. rtcPeer.reject({
2. 'targetUserId': receiveUser.id,
3. 'targetDeviceType': receiveUser.deviceType
4. });
6. rtcPeer.off("reject").on("reject", function(data) {
7. console.info("reject", data);
8. if (callUser.id == data.ownUserId) {
9. $("#callMessage").hide();
10. $("#ringAudio")[0].src = "";
11. clearTimeout(ringCallbackIndex);
12. clearTimeout(ringTimeoutIndex);
13. }
14. });

4.取消呼叫请求（rtcPeer. cancel）

4.1概述

取消呼叫请求，并发送信息告诉被呼叫者。

4.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| targetUserId | number | M | 呼叫者用户唯一标识码 |
| targetDeviceType | String | O | 设备类型 pc 、Android、iOS、webphone、tv |
| ownDeviceType | String | O | 设备类型 pc 、Android、iOS、webphone、tv |

4.3监听事件

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| cancel | String | M | 取消呼叫请求消息事件 |

事件获取数据

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| ownUserId | number | 被呼叫者的用户唯一标识Id |
| ownDeviceType | String | 被呼叫者的用户设备类型 |

4.4示例

1. rtcPeer.cancel({
2. 'targetUserId': callUser.id
3. });
4. rtcPeer.off("cancel").on("cancel", function(data) {
5. console.info("cancel", data);
6. if (receiveUser.id == data.ownUserId) {
7. $("#receiveMessage").hide();
8. $("#ringAudio")[0].src = "";
9. clearTimeout(ringCallbackIndex);
10. clearTimeout(ringTimeoutIndex);
11. }
12. });

5.响铃消息（rtcPeer. ring）

5.1概述

被呼叫方接收到邀请事件（doInvite）后，被呼叫方当前处于空闲状态时，被呼叫方自动发送响铃消息，告诉呼叫方被呼叫方已响铃。

5.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| targetUserId | number | M | 呼叫者用户唯一标识码 |
| targetDeviceType | String | O | 呼叫者设备类型 pc 、Android、iOS、webphone、tv |
| ownDeviceType | String | O | 设备类型 pc 、Android、iOS、webphone、tv |

5.3监听事件

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| doEvent | String | M | 消息事件 event:  onRinging响铃事件  onBusy 正在忙事件  onConnected 连接成功事件，开始音视频连接 |

事件获取数据

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| ownUserId | number | 被呼叫者的用户唯一标识Id |
| ownDeviceType | String | 被呼叫者的用户设备类型 |

5.4示例

1. rtcPeer.ring({
2. 'targetUserId': data.ownUserId,
3. 'targetDeviceType': data.ownDeviceType
4. });
5. rtcPeer.off("doEvent").on("doEvent", function(data) {
6. console.info("doEvent", data);
7. switch (data.event) {
8. case "onRinging":
9. *//clearTimeout(ringTimeoutIndex);*
10. break;
11. }
12. });

6.忙消息（rtcPeer. busy）

6.1概述

被呼叫方接收到邀请事件（doInvite）后，被呼叫方当前处于忙状态（正在开会中）时，被呼叫方自动发送正在忙消息，告诉呼叫方被呼叫方无法接听。

6.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| targetUserId | number | M | 呼叫者用户唯一标识码 |
| targetDeviceType | String | O | 呼叫者设备类型 pc 、Android、iOS、webphone、tv |
| ownDeviceType | String | O | 设备类型 pc 、Android、iOS、webphone、tv |

6.3监听事件

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| doEvent | String | M | 消息事件 event:  onRinging响铃事件  onBusy 正在忙事件  onConnected 连接成功事件，开始音视频连接 |

事件获取数据

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| ownUserId | number | 被呼叫者的用户唯一标识Id |
| ownDeviceType | String | 被呼叫者的用户设备类型 |

6.4示例

1. rtcPeer.busy({
2. 'targetUserId': data.ownUserId,
3. 'targetDeviceType': data.ownDeviceType
4. });
5. rtcPeer.off("doEvent").on("doEvent", function(data) {
6. console.info("doEvent", data);
7. switch (data.event) {
8. case 'onBusy':
9. isOtherBusy = true;
10. $("#callMessage").hide();
11. $("#ringAudio")[0].src = "";
12. clearTimeout(ringTimeoutIndex);
13. clearTimeout(ringCallbackIndex);
14. showMessage("您呼叫的用户当前正在忙。");
15. break;
16. }
17. });

7.连接成功消息（rtcPeer. connected）

7.1概述

接收到被呼叫方接同意事件（doAgree）后，呼叫方进入会议并发送连接成功消息给被呼叫方，被呼叫方接收到connected事件后，进入指定会议。

7.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| targetUserId | number | M | 呼叫者用户唯一标识码 |
| targetDeviceType | String | O | 呼叫者设备类型 pc 、Android、iOS、webphone、tv |
| roomId | Number | M | 虚拟房间唯 一标识码 |
| roomType | Number | M | 房间类型：1：音频会议 2 ：视频会议  3： 在线培训 |

7.3监听事件

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| doEvent | String | M | 消息事件 event:  onRinging响铃事件  onBusy 正在忙事件  onConnected 连接成功事件，开始音视频连接 |

事件获取数据

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| ownUserId | number | 被呼叫者的用户唯一标识Id |
| ownDeviceType | String | 被呼叫者的用户设备类型 |

7.4示例

1. rtcPeer.connected({
2. 'targetUserId': data.ownUserId,
3. 'targetDeviceType': data.ownDeviceType,
4. 'roomId': d.room.id,
5. 'roomType': d.room.roomType
6. }).then(function(data) {
7. clearTimeout(ringCallbackIndex);
8. clearTimeout(ringTimeoutIndex);
9. $("#callMessage").hide();
10. $("#ringAudio")[0].src = "";
11. });
12. rtcPeer.off("doEvent").on("doEvent", function(data) {
13. console.info("doEvent", data);
14. switch (data.event) {
15. case 'onConnected':
16. clearTimeout(ringCallbackIndex);
17. clearTimeout(ringTimeoutIndex);
18. $("#ringAudio")[0].src = "";
19. room = { id: data.roomId, name: '即时会议', isCall: true };
20. callRoom = room;
21. \_enterRoom({
22. id: data.roomId,
23. roomType: data.roomType,
24. name: '即时会议'
25. });
26. break;
27. }
28. });

8.自定义事件消息 （rtcPeer. doEvent）

8.1概述

响铃消息、忙消息、连接成功消息都是通过自定doEvent 接口中event 的值来实现

8.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| targetUserId | number | M | 呼叫者用户唯一标识码 |
| targetDeviceType | String | M | 呼叫者设备类型 pc 、Android、iOS、webphone、tv |
| roomId | Number | O | 虚拟房间唯 一标识码 |
| roomType | Number | O | 房间类型：1：音频会议 2 ：视频会议  3： 在线培训 |
| event | String | M | 自定义消息事件名：如onRinging |

8.3监听事件

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| doEvent | String | M | 消息事件 event:  onRinging响铃事件  onBusy 正在忙事件  onConnected 连接成功事件，开始音视频连接 |

事件获取数据

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| ownUserId | number | 被呼叫者的用户唯一标识Id |
| ownDeviceType | String | 被呼叫者的用户设备类型 |

9. 呼叫注册 （rtcPeer. doRegister）

9.1概述

登录后，调用呼叫注册接口，注册当前用户已在线，可以被呼叫,非SID对接方案不使用，类库内会自动在登录后注册

9.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| userId | number | M | 当前用户唯一标识码 |

9.3返回

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| result | number | 0呼叫注册成功 |

IM音视交互功能Java接口

提供在线用户之间的呼叫过程中的交互功能接口，通过接口可以实现发起呼叫、接收或拒绝呼叫，进入指定会议房间等需求。在客户端通过建立socket连接控制服务器，实现消息的实时传输。

1.呼叫接口

1.1概述

指定用户发起呼叫。

1.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| module | String | M | 值：call |
| method | String | M | 方法名：doInvite |
| ownUserId | Long | M | 当前用户唯一标识 |
| targetUserId | Long | M | 被呼叫用户User中的id（用户唯一标识） |
| ownDeviceType | String | O | 设备类型 pc 、Android、iOS、webphone、tv |
| targetDeviceType | String | O | 被呼叫设备类型 pc 、Android、iOS、webphone、tv |
| type | Long | O | 1音频会议 2视频会议 3视频回传 |
| roomId | Long | M | 会议室房间唯一标识，通过创建房间接口获取(newTempRoom接口) |

1.3监听事件

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| doInvite | String | M | 被呼叫者收到的邀请消息事件 |

事件获取数据

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| ownUserId | number | 发起呼叫者的用户唯一标识Id |
| ownDeviceType | String | 发起呼叫者的用户设备类型 |
| type | number | 1音频会议 2视频会议 3视频回传 |

2.同意呼叫请求接口

2.1概述

同意呼叫请求，并发送信息告诉呼叫。

2.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| module | String | M | 值：call |
| method | String | M | 方法名：doAgree |
| ownUserId | Long | M | 当前用户唯一标识 |
| targetUserId | Long | M | 呼叫用户User中的id（用户唯一标识） |
| ownDeviceType | String | O | 设备类型 pc 、Android、iOS、webphone、tv |
| targetDeviceType | String | O | 呼叫设备类型 pc 、Android、iOS、webphone、tv |
| type | Long | O | 1音频会议 2视频会议 3视频回传 |
| roomId | Long | M | 通过呼叫接口得到 |

2.3监听事件

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| doAgree | String | M | 被呼叫者发送同意请求消息事件 |

事件获取数据

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| ownUserId | number | 被呼叫者的用户唯一标识Id |
| ownDeviceType | String | 被呼叫者的用户设备类型 |

3.拒绝呼叫请求接口

3.1概述

拒绝呼叫请求，并发送信息告诉呼叫。

3.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| module | String | M | 值：call |
| method | String | M | 方法名：reject |
| ownUserId | Long | M | 当前用户唯一标识 |
| targetUserId | Long | M | 呼叫用户User中的id（用户唯一标识） |
| ownDeviceType | String | O | 设备类型 pc 、Android、iOS、webphone、tv |
| targetDeviceType | String | O | 呼叫设备类型 pc 、Android、iOS、webphone、tv |

3.3监听事件

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| reject | String | M | 被呼叫者发送拒绝请求消息事件 |

事件获取数据

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| ownUserId | number | 被呼叫者的用户唯一标识Id |
| ownDeviceType | String | 被呼叫者的用户设备类型 |

4.取消呼叫请求接口

4.1概述

取消呼叫请求，并发送信息告诉被呼叫者。

4.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| module | String | M | 值：call |
| method | String | M | 方法名：cancel |
| ownUserId | Long | M | 当前用户唯一标识 |
| targetUserId | Long | M | 呼叫用户User中的id（用户唯一标识） |
| ownDeviceType | String | O | 设备类型 pc 、Android、iOS、webphone、tv |
| targetDeviceType | String | O | 呼叫设备类型 pc 、Android、iOS、webphone、tv |

4.3监听事件

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| cancel | String | M | 取消呼叫请求消息事件 |

事件获取数据

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| ownUserId | number | 被呼叫者的用户唯一标识Id |
| ownDeviceType | String | 被呼叫者的用户设备类型 |

5.事件消息通用接口

5.1概述

响铃消息、忙消息、连接成功消息都是通过自定doEvent 接口中event 的值来实现

5.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| module | String | M | 值：call |
| method | String | M | 方法名：doEvent |
| ownUserId | Long | M | 当前用户唯一标识 |
| targetUserId | Long | M | 呼叫用户User中的id（用户唯一标识） |
| ownDeviceType | String | O | 设备类型 pc 、Android、iOS、webphone、tv |
| targetDeviceType | String | O | 呼叫设备类型 pc 、Android、iOS、webphone、tv |
| event | String | M | 自定议事件名：onRinging响铃事件  onBusy 正在忙事件  onConnected 连接成功事件，开始音视频连接 |

5.3监听事件

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| doEvent | String | M | 消息事件 event:  onRinging响铃事件  onBusy 正在忙事件  onConnected 连接成功事件，开始音视频连接 |

事件获取数据

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| ownUserId | number | 被呼叫者的用户唯一标识Id |
| ownDeviceType | String | 被呼叫者的用户设备类型 |

.

6.呼叫注册接口

6.1概述

登录后，调用呼叫注册接口，注册当前用户已在线，可以被呼叫。

6.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| module | String | M | 值：call |
| method | String | M | 方法名：doRegister |
| SID | String | M | Session ID 由认证接口成功后返回 |
| userId | Long | M | 当前用户唯一标识 |
| deviceType | String | O | 设备类型 pc 、Android、iOS、webphone、tv |

6.3返回

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| result | Long | 呼叫注册成功 |

7.响铃消息

7.1概述

被呼叫方接收到邀请事件（doInvite）后，被呼叫方当前处于空闲状态时，被呼叫方自动发送响铃消息，告诉呼叫方被呼叫方已响铃。

8.忙消息

8.1概述

被呼叫方接收到邀请事件（doInvite）后，被呼叫方当前处于忙状态（正在开会中）时，被呼叫方自动发送正在忙消息，告诉呼叫方被呼叫方无法接听。

9.连接成功消息

9.1概述

接收到被呼叫方接同意事件（doAgree）后，呼叫方进入会议并发送连接成功消息给被呼叫方，被呼叫方接收到connected事件后，进入指定会议。

10.创建临时房间接口

10.1概述

创建临时房间。

10.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| module | String | M | 值：room |
| method | String | M | 方法名：newAppRoom |
| name | String | M | 房间名称 |
| numberLimit | Long | M | 会议室人员上限，最大200人 |
| roomtype | Long | M | 1音频会议 2视频会议 3在线培训 |
| userId | Long | M | 当前用户唯一标识 |

10.3返回

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| result | Long | 0成功 |
| roomId | Long | 会议室房间唯一标识 |

11.登出接口

11.1概述

登出。

11.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| module | String | M | 值：account |
| method | String | M | 方法名：logout |
| SID | String | M | Session ID 由认证接口成功后返回 |
| userId | Long | M | 当前用户唯一标识 |

11.3返回

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| result | Long | 0成功 |

12. 创建预约会议

12.1概述

创建预约会议接口, 创建即时会议,将开始时间设置为当前时间。

12.2参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| method | String | M | 方法名，值：new |
| module | String | M | 值：meeting |
| SID | String | M | Session ID 用户登录后产生的身份标识码 |
| name | String | M | 会议名称 |
| startTime | String | M | 会议开始时间 ，传入UTC时间格式 “yyyyMMddHHmmss” |
| endTime | String | M | 会议结束时间 ，传入UTC时间格式“yyyyMMddHHmmss” |
| hour | String | O | 会议时长（小时） |
| leverage | String | O | 会议容量人数（自然数，200方）；主要看流媒体服务器的CPU |
| type | Long | M | 会议类型 ：  1音频会议 2视频会议 3视频回传 |
| isPublic | Boolean | M | 是否公开，true公共会议，false:私人会议。建议传false, |
| pin | Long | O | 会议进入校验密码 |
| members | Array | O | 邀请成员，数组内成员格式为String,  格式如下：  “**memberId,name,email,id, Etc/GMT+8,role**”  注：memberId 默认为0，  id为用户的唯一标识码id,  name:用户昵称  email:邮件，可以为null  角色role:  会议场模式：1是主持人 2普通参会人员；  培训模式：1是主持人 2是讲师，3普通参付人员，建议只邀请主持人和讲师 |
| template | Long | O | 模板 |
| postUrl | String | O | 海报 |
| orgId | Long | M | 公司唯一标识码Id，可以提供，一般情况传2 |
| roomId | Long | O | 房间唯 一标识id  不传id会重新创建一个，传存在的room id，用此room |
| baseURL | String | O | 邮件链接地址 |
| topics | Array | O | 网络会议 子议题  数组内的Objet格式：  { "title":“议题”, "content": "具体内容" } |

12.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| id | Int | 会议唯一标识 |
| roomId | Int | 房间的唯一标识，新建会议，如果不传roomId,会议和房间的关系是一对一。 |
| result | Int | 0：成功，1：失败 |

轻会议平台对接方案

第三方平台和私有部署轻会议平台对接方案。

1. 生成签名接口

1）轻会议给第三方平台配置appid值

2）第三方平台通过 appid和secret 到轻会议平台拿签名（轻会议平台通过appid和secret加密保存返回 ），第三方平台调用轻会议平台api,必需要有签名才能调用

1.1生成签名接口

1.2.1请求格式

|  |  |
| --- | --- |
| 请求地址 | https:// xxx.xxx.com/app/getSign |
| 请求模式 | GET |
| Content-Type | multipart/form-data |

1.2.2 参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| appId | String | M | 公司接入api认证key（由轻会议提供） |
| secret | String | M | 钥匙 |

1.2.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| code | int | 200 |
| data | String | api签名 |
| message | String | 描述 |

 1.2.4 入参格式

https://xxx.xxx.com/app/getSign?appId=56789537388&secret=kksakjkjkk11

2. 用户信息认证接口

1.1概述

用户信息对接，具体：

1）调用轻会议平台getSign接口生成签名；

2）调用轻会议平台dotokenLogin接口进行认证，认证成功返回用户信息

3）第三方平台可以通过 SID和用户唯一标识调用轻会议平台相关接口；

注：第三方提供认证接口，认证成功返回第三方用户信息。

1.2 dotokenLogin接口

1.2.1请求格式

|  |  |
| --- | --- |
| 请求地址 | https:// gj.527meeting.com/app/account |
| 请求模式 | POST |
| Content-Type | application/x-www-form-urlencoded |

1.2.2 参数

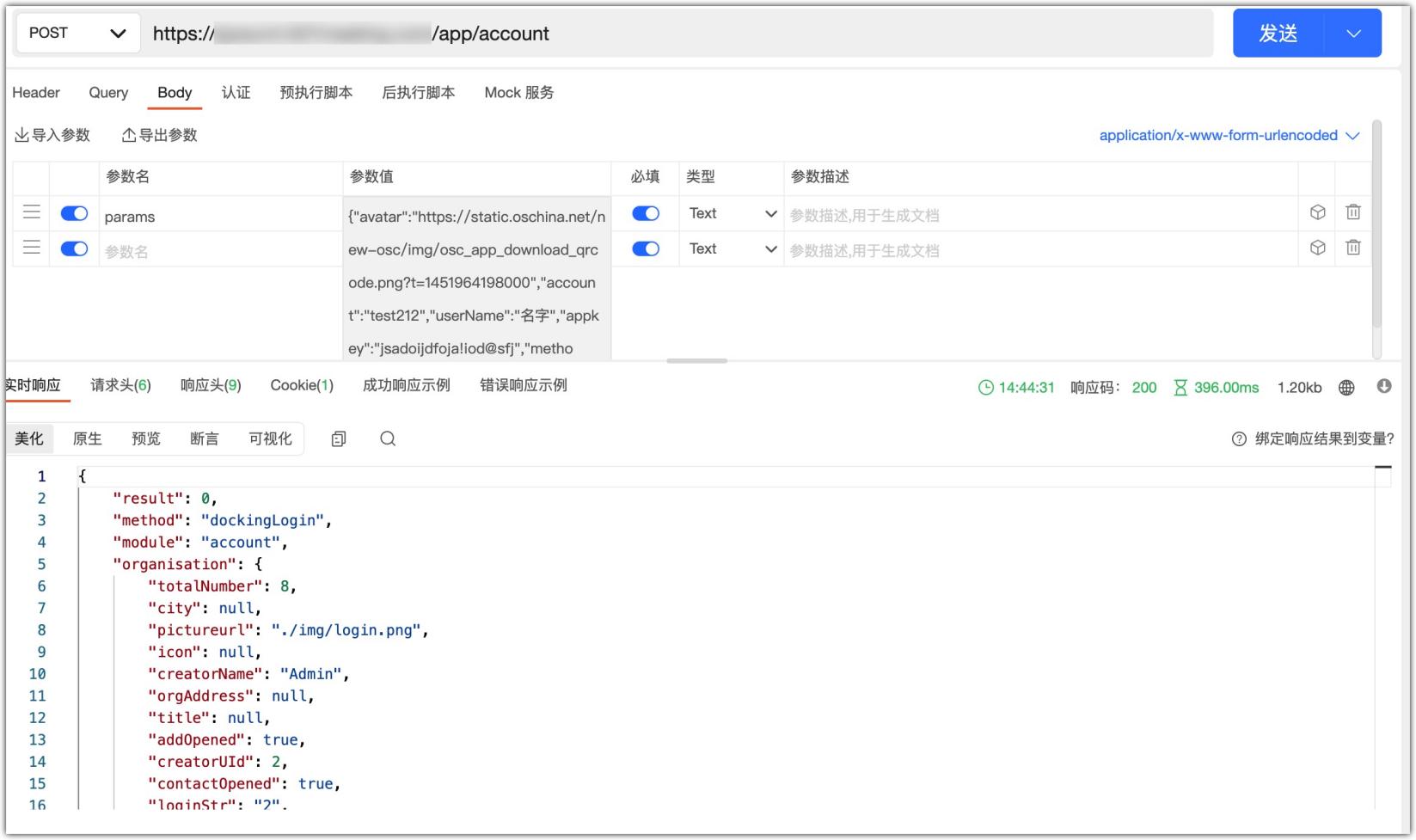
|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| method | String | M | 方法名，值：dockingLogin |
| avatar | String | O | 第三方用户头像 |
| account | String | M | 第三方用户唯一标识 |
| userName | String | O | 第三方用户名 |
| appkey | String | O | 公司接入api认证key（由轻会议提供） |
| sign | String | M | api签名，有签名才能调用，通过java接口可获取 |
| job | String | O | 职位 |
| level | Long | O | 用户级别 |

1.2.3返回参数

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| SID | String | Session ID 用户登录后产生的身份标识码 |
| organisation | Company | 公司信息，参考been类中的Company。 |
| user | User | 用户信息，参考been类中的User. |
| result | Long | 0 成功；1 公司名不正确 ；2 公司appkey验证失败；3 回调请求失败；4 注册失败 |

1.2.4 入参格式

“params”:{"avatar":"https://static.oschina.net/new-osc/img/osc\_app\_download\_qrcode.png?t=1451964198000","account":"test212","userName":"名字","appkey":"jsadoijdfoja!iod@sfj","method":"dockingLogin"}



Web Url接口

登录用户或非登录用户可以通过url链接跳转到会议室中。

1.通过房间ID进入指定房间

1.1概述

通过房间ID,进入指定房间,登录用户传有效SID,非登录用房不传。

1.2请求格式

https:// 61.182.226.42:10443/app/#/?role=1&rd= 6656&name=小明

1.3参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| SID | String | M | Session ID 用户登录后产生的身份标识码 |
| uid | String | O | 用户唯一标识，登录用户传 |
| role | number | O | 角色role:  会议场模式：1是主持人 2普通参会人员；  培训模式：1是主持人 2是讲师，3普通参付人员 |
| rd | number | M | 房间唯一标识码 |
| v | number | O | 进入会议室默认：1:摄像头打开，0：摄像头不打开 |
| name | String | O | 参会人员昵称 |

2.通过会议ID进入指定房间

1.1概述

通过会议ID,进入指定房间,登录用户传有效SID,非登录用房不传。

1.2请求格式

https:// 61.182.226.42:10443/app/#/?role=1&code= 6656&name=小明

1.3参数

|  |  |  |  |
| --- | --- | --- | --- |
| 参数 | 类型 | 是否必选 | 参数说明 |
| SID | String | M | Session ID 用户登录后产生的身份标识码 |
| uid | String | O | 用户唯一标识，登录用户传 |
| role | number | O | 角色role:  会议场模式：1是主持人 2普通参会人员；  培训模式：1是主持人 2是讲师，3普通参付人员 |
| code | number | M | 会议唯一标识码 |
| v | number | O | 进入会议室默认：1:摄像头打开，0：摄像头不打开 |
| name | String | O | 参会人员昵称 |

Bean类说明

对文中的Object的对象补充说明。

1.User

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| id | Number | 用户唯一标识码 |
| account | String | 账号 |
| name | String | 昵称 |
| email | String | 邮件地址 |
| phone | String | 手机号 |
| photo | String | 头像 |
| company | String | 公司名称 |

2. Company

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| id | Number | 公司唯一标识码 |
| name | String | 公司名 |
| email | String | 公司邮件地址 |
| phone | String | 公司手机号 |
| logo | String | 公司Logo |
| address | String | 公司地址 |

3. Room

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| id | Number | 唯一标识码 |
| name | String | 房间名 |
| ownerId | Number | 房间拥有者用户唯一标识码 |
| roomType | Number | 房间类型 1音频 2视频 3培训 |
| definition | String | 清晰度 1080p 720p 480p 240p |
| numberLimit | Number | 房间人数上限 |
| isPublic | Boolean | true 公共会议室 false 私有会议室 |
| isHasPwd | Boolean | 是否有密码 |
| appointmentId | Number | 房间对应的预约会议唯一标识，仅在会议中使用 |
| appointmentName | String | 对应会议的名称，仅在会议中使用 |
| superModeId | Number | 仅在会议中使用,会议中的管理员的唯一标识 |
| logo | String | Logo |
| comment | String | 描述 |

4.Client

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| id | Number | 用户唯一标识码 |
| name | String | 昵称 |
| email | String | 邮件地址 |
| phone | String | 手机号 |
| pictureurl | String | 头像 |
| company | String | 公司名称 |
| mode | String | 终端类型 |
| streamId | Number | 用户唯一标识码 |
| broadCastId | number | 用户唯一标识码 流媒体使用 |
| isMuted | Boolean | 是否是静音状态 |
| isSuperModerator | Boolean | 是否是主持人 |

5. Appointment

|  |  |  |
| --- | --- | --- |
| 参数 | 类型 | 参数说明 |
| id | Number | 唯一标识码 |
| name | String | 会议名称 |
| startTime | Date | 开始时间 |
| endTime | Date | 结束时间 |
| from | User | 创建者 |
| timeLength | Number | 分钟 |
| description | String | 描述 |
| streamId | Number | 用户唯一标识码 |
| broadCastId | number | 用户唯一标识码 流媒体使用 |
| actuallyStartTime | Date | 真实开始时间 |
| actuallyEndTime | Date | 真实结束时间 |