# Talk SDK(JS)接口文档

# 1 接口说明

## 1.1 Room 类——SDK（JS）的主入口

Room：  
 *// 房间类变量*

DISCONNECTED = 0, //断开连接

CONNECTING = 1, //正在连接

CONNECTED = 2; //连接成功

\_room\_host, //房间所在服务器IP地址

\_room\_port, //房间所在服务器端口号

\_room\_uri, //房间连接地址

\_room\_id, //房间id

\_room\_type=0, //房间类型

\_room\_name, //房间名称

\_room\_properties, //房间属性

\_room\_video\_width = 320, //房间视频窗口的宽度

\_room\_video\_height = 240, //房间视频窗口的高度

//和管理系统有关的接口

var HTTP = "https://"; //网址一律使用https://这种认证方式访问

var WEBFUNC\_CHECKroom = "/ClientAPI/checkroom"; //房间检查

var WEBFUNC\_GETCONFIG = "/ClientAPI/getconfig"; //获取房间配置信息

var WEBFUNC\_GETFILELIST = "/ClientAPI/getroomfile"; //获取房间文件列表

var WEBFUNC\_UPLOADDOCUMENT = "/ClientAPI/uploaddocument"; //房间上传文件

var WEBFUNC\_DELROOMFILE = "/ClientAPI/delroomfile"; //房间删除文件

var ERR\_OK = 0; *// 成功*

var ERR\_INTERNAL\_EXCEPTION = -1; *// 内部异常*

var ERR\_NOT\_INITIALIZED = 1; *// 尚未初始化*

var ERR\_INVALID\_STATUS = 2; *// 当前状态不适合此操作*

var ERR\_HTTP\_REQUEST\_FAILED = 3; *// 尚未初始化*

var ERR\_BAD\_PARAMETERS = 4; *// 参数错误*

var ERR\_NO\_THIS\_USER = 5; *// 无此用户*

var ERR\_USER\_NOT\_PUBLISHING = 6; *// 此用户尚未发布视频*

var ERR\_USER\_NOT\_PLAYING = 7; *// 尚未播放此用户的视频*

var STATUS\_IDLE = 0; // 空闲

var STATUS\_CHECKING = 1; // 检查中

var STATUS\_GETTINGCFG = 2; // 检查中

var STATUS\_CONNECTING = 3; // 正在连接

var STATUS\_CONNECTED = 4; // 连接完成

var STATUS\_JOINING = 5; // 正在加入

var STATUS\_ALLREADY = 6; // 已经准备好

var STATUS\_DISCONNECTING = 7; // 正在断开连接

var STATUS\_DISCONNECTED = 8; // 断开连接

localStreams //本地音视频流（未发布）

remoteStreams //远程音视频流（已发布）

*// 进入房间*

*// 您需要提前预约好房间，详见Talksdk(JS)集成文档*

*// nickname：本地用户的昵称*

*//**defaultstream: 默认视频流*

**joinroom (defaultstream)**;

**joinroom 需要四步**

*//step1 checkroom 检查房间是否存在*

*//step2 getfilelist & getconfig 获取文件列表 和房间配置信息*

*//step3 connect 连接*

*//step4 join 加入*

*// 离开房间*

*// force*

**leaveroom (force)** ;

*// 获得房间类型*

*// 请在进入房间成功后调用*

*// 返回值：房间类型。0：一对一；1：小班，10：大讲堂（直播）*

**getRoomType();**

*// 获得房间名称*

*// 请在进入房间成功后调用*

*// 返回值：房间名称*

**getRoomName();**

*// 获得房间的详细属性*

*// 请在进入房间成功后调用*

*// 返回值：房间属性*

**getRoomProperties();**

*// 获得本地用户*

*// 请在进入房间成功后调用*

*// 返回值：本地用户对象*

**getMySelf();**

*// 获得用户*

*// 请在进入房间成功后调用*

*// peerID：用户ID*

*// 返回值：本地用户对象*  
 **getUser(String peerID);**

*// 获得当前房间内的所有用户*

*// 请在进入房间成功后调用*

*// 返回值：所有用户的Map，Key是peerID，value是用户*

**getUsers();**

*// 获得当前房间内的所有用户数量*

*// 请在进入房间成功后调用*

*// 返回值：用户数量*

**getUsersNumber ();**

*// 获得当前房间内的指定角色列表*

*// 请在进入房间成功后调用*

*//specifyKey:指定关键字*

*// 返回值：指定角色列表*

**getSpecifyRoleList(specifyKey);**

*// 获得当前房间内的所有角色列表*

*// 请在进入房间成功后调用*

*// 返回值：所有角色列表*

**getAllRoleList();**

*// 获得当前房间的配置信息*

*// 请在进入房间成功后调用*

*// 返回值：房间的配置信息*

**getConfigInfo();**

*// 获得当前房间内的上台人数的数量*

*// 请在进入房间成功后调用*

*// 返回值：上台人数的数量*

**getPublishStreamNumber();**

*// 检查我的音频和视频是否可用*

*// 请在进入房间成功后调用*

*// 返回值：音频和视频的使用状态*

**checkMyAudioAndVideoEnable(publishstate , videoInfo , audioInfo );**

*// 修改某个用户的一个属性*

*// id：要修改的用户ID*

*// tellWhom：要将此修改通知给谁。“\_\_All”：所有人；“\_\_allExceptSender”：除自己以外的所有人；“\_\_allExceptAuditor”：除旁听用户以外的所有人；“\_\_None”：不通知任何人；某用户的id：只发给该用户*

*// properties：要修改的用户属性，可以是Number、String、Map或List*

*// 返回值：请见错误码定义*

**changeUserProperty(id, tellWhom, properties);**

*// 修改某个用户的音视频发布状态*

*// id：该用户的id，可以是自己的，也可以是其他人的*

*// publish：0：不发布；1：只发布音频；2：只发布视频；3：发布音视频*

*// 返回值：请见错误码定义*

**changeUserPublish (id,publish);**

*// 发送聊天消息*

*// message：聊天内容*

*//toId: 要将此消息发送给谁。“\_\_All”：所有人；“\_\_allExceptSender”：除自己以外的所有人；“\_\_allExceptAuditor”：除旁听用户以外的所有人；“\_\_None”：不通知任何人；某用户的id：只发给该用户*

*// 返回值：请见错误码定义*

**sendMessage(message, toId);**

*// 发布自定义消息*

*// msgName：消息名字*

*// msgId：消息id*

*// toId：要通知给哪些用户。“\_\_All”：所有人；“\_\_allExceptSender”：除自己以外的所有人；“\_\_allExceptAuditor”：除旁听用户以外的所有人；“\_\_None”：不通知任何人；某用户的peerID：只发给该用户*

*// data：消息数据，可以是Number、String、Map或List*

*// save：是否保存，详见3.5：自定义信令*

*// expiresabs：到时间后自动删除，*

*//* *associatedMsgID: 父子消息绑定信息*

*// 返回值：请见错误码定义*

**pubMsg(msgName, msgId, toId, data, save, expiresabs，associatedMsgID);**

*// 删除自定义消息*

*// msgName：消息名字*

*// msgId：消息id*

*// toId：要通知给哪些用户；“\_\_All”：所有人；“\_\_allExceptSender”：除自己以外的所有人；“\_\_allExceptAuditor”：除旁听用户以外的所有人；“\_\_None”：不通知任何人；某用户的peerID：只发给该用户；“\_\_allSuperUsers”： 只发给老师和助教*

*// data：消息数据，可以是Number、String、Map或List*

*// 返回值：请见错误码定义*

**delMsg(msgName, msgId, toId, data);**

*// 将一个用户踢出房间*

*// id：该用户的id*

*// 返回值：请见错误码定义*  
 **evictUser(id);**

*//数据流发布*

*// stream：具体数据流发布对象*

*// options：*

*// callback：回调函数*

**publish(stream, options, callback);**

*//媒体文件数据流发布*

*// stream：具体媒体文件数据流发布对象*

*// options：*

*// callback：回调函数*

**publishMedia(stream, options, callback);**

*//屏幕共享数据流发布*

*// stream：具体屏幕共享数据流发布对象*

*// options：*

*// callback：回调函数*

**publishScreen(stream, options, callback);**

*//取消数据流发布*

*// stream：具体取消数据流发布对象*

*// callback：回调函数*

**unpublish(stream, callback);**

*//取消媒体文件数据流发布*

*// stream：具体取消媒体文件数据流发布对象*

*// callback：回调函数*

**unpublishMedia (stream, callback);**

*//取消屏幕共享数据流发布*

*// stream：具体取消屏幕共享数据流发布对象*

*// callback：回调函数*

**unpublishScreen (stream, callback);**

*//数据流订阅*

*// stream：具体订阅的数据流对象*

*// options：*

*// callback：回调函数*

**subscribe(stream, options, callback) ;**

*//取消数据流订阅*

*// stream：取消具体订阅的数据流对象*

*// options：*

*// callback：回调函数*

**unsubscribe(stream, options, callback) ;**

*//数据流添加*

*// stream：具体数据流对象*

**addStream(stream);**

*//数据流删除*

*// stream：具体数据流对象*

**deleteStream(stream);**

*//发送回放的seek消息给服务器*

*// positionTime：时间位置*

**seekPlayback(positionTime);**

*//发送回放的暂停消息给服务器*

**pausePlayback();**

*//发送回放的播放消息给服务器*

**pausePlayback();**

*//初始化回放信息*

*// host：服务器地址*

*// port：服务器端口号*

*// params: 参数*

*// callback：回调函数*

**initPlaybackInfo(host, port, params, callback )**

*// 检查房间*

*// host：服务器地址*

*// port：服务器端口号*

*// params: 参数*

*// callback：回调函数*

*// userid：用户ID*

**checkroom(host, port, params, callback , userid);**

*// 获取上传文件参数*

*// filename：文件名*

*// filetype：文件类型*

**getUploadFileParams(filename ,filetype);**

*// 上传文件*

*// formData：文件信息*

*// callback：回调函数*

*// progressListenCallback:进度条回调函数*

**uploadFile(formData , callback , progressListenCallback );**

*// 删除文件*

*// fileid：文件id*

*// callback：回调函数*

**deleteFile(fileid , callback );**

*// 开始录制视频 必须是发布流*

*// stream：视频流*

*// options：0,1,2 //0 不转换格式 类型mkv ；1转换为Webm；2转换为MP4*

*// callback：回调函数,两个参数，返回值和录制文件保存路径*

*Ret:0成功，-1没有开启录制，path返回的路径需要在前面加上*

*公网*[*http://global.talk-cloud.net:8081/*](http://global.talk-cloud.net:8081/)

*内网http://192.168.1.253:8000/*

**startRecordStream(stream, options, callback);**

*// 停止录制视频*

*// stream：视频流*

*// callback：回调函数,两个参数，返回值和录制文件保存路径*

**stopRecordStream(stream, callback);**

## 1.2 RoomUser类

RoomUser:  
  
 var *PUBLISH\_STATE\_NONE* = 0; //下台  
 var *PUBLISH\_STATE\_AUDIOONLY* = 1; //只发布音频  
 var *PUBLISH\_STATE\_VIDEOONLY* = 2; //只发布视频  
 var *PUBLISH\_STATE\_BOTH* = 3; //音视频都发布

var *PUBLISH\_STATE\_MUTEALL =* 4*;* //音视频都关闭  
  
 var **nickName** = **""**;*// 用户昵称*  
 var **id** = **""**;*// 用户Id*  
 var **role** = -1;*// 用户身份，0：老师；1：助教；2：学生；3：旁听；4：隐身用户*

var **hasAudio** = **true**;*// 该用户是否有麦克风*  
 var **hasVideo** = **true**;*// 该用户是否有摄像头*  
 var **canDraw** = **true**;*// 该用户是否有权在白板和文档上进行绘制*  
 var **publishState** = *PUBLISH\_STATE\_NONE*;*// 发布状态，0：未发布，1：发布音频；2：发布视频；3：发布音视频*  
var **properties**;*// 用户属性。Sdk调用者设置的自定义属性会在这里被找到*

## 1.3 Stream 类

Stream:

*// 获取流ID*

**getID ();**

*// 获取流属性*

**getAttributes ();**

*// 是否有音频设备*

*// 返回值：*

**hasAudio();**

*// 是否有音频设备*

*// 返回值：*

**hasVideo();**

*// 初始化流，并尝试从本地视频和音频检索流*

*// dispatchGetUserMediaResult\_Event：*

**init(dispatchGetUserMediaResult\_Event);**

*// 播放某个音视频*

*// 请在该用户发布视频后再调用*

*// 对同一个用户，可以调用多次此函数。当传入的view和上次传入的一致时，函数不执行任何操作，直接返回成功；当传入的view和上次传入的不一致时，换用新的view播放该用户的视频*

*//* elementID*： Id 页面div id只是播放视频用*

*//* options*：*

**play(elementID, options);**

*// 停止播放音视频*

**stop();**

*// 关闭视频*

**close();**

*// 显示视频*

**show();**

*// 隐藏视频*

**hide();**

1.4 EventDispatcher 类

EventDispatcher：

*// 添加侦听事件*

*// eventType： 事件类型*

*// listener：*

*// backupid：*

**addEventListener (eventType, listener , backupid );**

*// 移除侦听事件*

*// eventType： 事件类型*

*// listener：*

**removeEventListener(eventType, listener);**

*// 移除所有侦听事件*

*// eventTypeArr： 所有事件类型*

**removeAllEventListener(eventTypeArr);**

*// 分发侦听事件*

*// event： 事件*

*// log:日志*

**dispatchEvent(event , log);**

*// 分发侦听事件*

*// event： 事件*

*// log:日志*

**dispatchEvent(event , log);**

*// 房间事件*

*// spec：*

**RoomEvent (spec);**

*// 流事件*

*// spec：*

**StreamEvent (spec);**

1.5 Connection 类

Connection：

*// 获取浏览器类型*

**getBrowser();**

1.6 VideoPlayer 类 //视频播放

VideoPlayer：

1.7 AudioPlayer 类 //音频播放

AudioPlayer：

1.8 Constant //SDK常量

Constant：

1.9 L.Logger //SDK日志

L.Logger：

*// 调试*

**debug()；**

*// 信息*

**info()；**

*// 警告*

**warning ()；**

*// 错误*

**error ()；**

1.10 Resizer //调整

Resizer：

1.11 Utils //SDK工具类

Utils：

1.12本地设备管理类

AVMgr:

*//枚举设备信息*

*// callback： 回调函数*

*// paramsJson：Json对象，信息要进行本地存储*

*paramsJson.isSetlocalStorage = paramsJson.isSetlocalStorage!=undefined ? paramsJson.isSetlocalStorage : false ;*

**enumerateDevices(callback , paramsJson);**

*例：*

*Var paramsJson = {isSetlocalStorage: false}；*

*room.enumerateDevices( function (devicesInfo);*

*或TK.AVMgr.enumerateDevices(callback, paramsJson);*

*//获取设备数据流*

*// callback:回调函数 ,*

*//error:错误信息函数,*

*//config:配置信息object*

**getUserMedia(callback,error , config , specifiedConstraints);**

*例：*

*deviceId: 为界面选择的设备编码*

*var specifiedConstraints = {  
 video: {  
 sourceId:deviceId  
 },  
 audio:{} ,  
 exclude:{audio:true}  
};*

*room..getUserMedia (callback,error , undefined , specifiedConstraints);*

*或TK.AVMgr.getUserMedia(callback,error , undefined , specifiedConstraints)*

*//获取声音探测实例*

*//getsSoundMeterInstance*

*//audioContext:new window.AudioContext()*

*return 声音探测实例SoundMeter*

**getsSoundMeterInstance(audioContext);**

*例：*

*var audioContext = null;*

*try {  
 window.AudioContext = window.AudioContext || window.webkitAudioContext || window.mozAudioContext || window.msAudioContext;  
 audioContext = audioContext || new window.AudioContext();  
} catch (e) {  
 L.Logger.error('Web Audio API not supported ');  
}*

*var audioContext = audioContext || new window.AudioContext();*

*room..getsSoundMeterInstance(that.audioContext)*

*//为音视频节点设置扩音器输出*

*//elementArr:array(需要设置音频输出的节点数组)*

*// audiooutputDeviceId：音频输出设备编号*

*// callback:回调函数*

**setElementSinkIdToAudioouputDevice(elementArr , audiooutputDeviceId , callback);**

**例：**

*Media 为Html5 支持的video或audio组件；() =>为回调函数*

*room..setElementSinkIdToAudioouputDevice(media , undefined , () => {  
 try {  
 // 此处是可能产生例外的语句  
 media.play();  
 } catch(error) {  
 // 此处是负责例外处理的语句  
  
 }  
});*

*TK.AVMgr.setElementSinkIdToAudioouputDevice(media , undefined , () => {  
 try {  
 // 此处是可能产生例外的语句  
 media.play();  
  
 } catch(error) {  
 // 此处是负责例外处理的语句  
  
 }  
});*

*//更换本地设备，生成数据流改变本地媒体数据流轨道*

*//getUserMediaCallback:function(切换设备后获取的数据流回调) ，*

*//deviceIdMap:json ,*

*//callback:function ,*

*//audioouputElementIdArr:array(需要更新扩音器输出的节点元素数组)*

**changeLocalDeviceToLocalstream(getUserMediaCallback , deviceIdMap , callback , audioouputElementArr);**

**例：**

*var selectDeviceInfo = {} ;  
var audioouputElementIdArr = document.getElementById("room").querySelectorAll("video , audio") ;*

*room.changeLocalDeviceToLocalstream(selectDeviceInfo , function (stream) {  
 //处理回调得到的视频流  
},audioouputElementIdArr);*

*//添加设备改变事件*

*//**callback:回调函数*

**addOndevicechange(callback);**

**例：**

*room.addOndevicechange(callback);*

*//删除设备改变事件*

*//**callback:回调函数*

**removeOndevicechange (callback);**

**例：**

*room.removeOndevicechange (callback);*

# 2 FAQ

Q:SDK支持哪些浏览器版本？

A:TKSDK目前支持IE11、chorme、firefox、Safari；