腾讯云音视频多人会话解决方案服务端

1.项目简介

在构建直播业务，多人音视频业务等场景下，都需要后台配合完成诸如： [x] 生成直播地址，包括推流和播放地址 [x] 生成IM签名，用于IM独立模式下的用户登录 [x] 管理IM聊天室，聊天室的创建和销毁还有成员进出通知 [x] 双人/多人音视频管理视频位。 以上这些都有一定的学习成本，为了**降低学习成本**，我们将后台封装了一套接口，来解决以上问题。再配合IOS，Android，小程序和Win PC端的后台调用封装。对应用开发者提供一套友好的接口，方便您实现多人实时音视频，直播，聊天等业务场景。

特别说明： [1] 后台没有对接口的调用做安全校验，这需要您结合您自己的账号和鉴权体系，诸如在请求接口上加一个Sig参数，内容是您账号鉴权体系派发的一个字符串，用于校验请求者的身份。 [2] 房间管理采用 JS对象直接在内存中进行管理。房间信息动态和实效性，因此没有采用数据库做持久存储，而是在内存中动态管理。

2.项目结构

```
server
├── README.md
├── app.js
├── double_room
|   ├── index.js
|   ├── get_im_login_info.js
|   ├── get_push_url.js
|   ├── create_room.js
|   ├── destroy_room.js
|   ├── add_pusher.js
|   ├── delete_pusher.js
|   ├── get_pushers.js
|   ├── pusher_heartbeat.js
|   └── get_room_list.js
├── multi_room
|   ├── index.js
|   ├── get_im_login_info.js
|   ├── get_push_url.js
|   ├── create_room.js
|   ├── destroy_room.js
|   ├── add_pusher.js
|   ├── delete_pusher.js
|   ├── get_pushers.js
|   ├── pusher_heartbeat.js
|   └── get_room_list.js
├── live_room
|   ├── index.js
```

```
|   ├── get_im_login_info.js
|   ├── get_push_url.js
|   ├── create_room.js
|   ├── destroy_room.js
|   ├── add_pusher.js
|   ├── delete_pusher.js
|   ├── get_pushers.js
|   ├── pusher_heartbeat.js
|   └── get_room_list.js
├── utils
|   ├── index.js
|   ├── get_test_pushurl.js
|   ├── get_test_rtmpaccurl.js
|   ├── getlogfile.js
|   └── logfilelist.js
├── logic
|   ├── im_mgr.js
|   ├── live_util.js
|   ├── double_room_mgr.js
|   ├── multi_room_mgr.js
|   └── live_room_mgr.js
├── middlewares
|   ├── bodyparser.js
|   └── response.js
├── config.js
├── log.js
├── log_config.js
├── package.json
├── process.json
├── nodemon.json
├── qcloud.js
└── routes
    └── index.js
```

`app.js` 是 服务器端 的主入口文件，使用 Koa 框架，在 `app.js` 创建一个 Koa 实例并响应请求。

`routes/index.js` 是 服务器端 的路由定义文件。

`double_room` 存放 服务器端 双人房间业务逻辑的目录，`index.js` 不需要修改，他会动态的将 `double_room` 文件夹下的目录结构映射成 modules 的 Object，例如 double_room 目录中将会被映射成如下的结构：

```
// index.js 输出
{
  get_im_login_info: require('get_im_login_info'),
  get_push_url: require('get_push_url'),
  create_room: require('create_room'),
  destroy_room: require('destroy_room'),
  add_pusher: require('add_pusher'),
  delete_pusher: require('delete_pusher'),
  get_pushers: require('get_pushers'),
  pusher_heartbeat: require('pusher_heartbeat'),
  get_room_list: require('get_room_list')
}
```

`multi_room` 存放 服务器端 多人房间业务逻辑的目录， `index.js` 不需要修改，他会动态的将 `multi_room` 文件夹下的目录结构映射成 modules 的 Object。

`live_room` 存放 服务器端 直播房间业务逻辑的目录， `index.js` 不需要修改，他会动态的将 `live_room` 文件夹下的目录结构映射成 modules 的 Object。

`utils` 存放 服务器端 辅助接口的目录， `index.js` 不需要修改，他会动态的将 `utils` 文件夹下的目录结构映射成 modules 的 Object。

`config.js` 主要的配置如下：

```
{
  port: '5757',
  rootPathname: '',

  // 微信小程序 App ID
  appId: '',

  // 微信小程序 App Secret
  appSecret: '',

  // 是否使用腾讯云代理登录小程序
  useQcloudLogin: true,

  /**
   * MySQL 配置，用来存储 session 和用户信息
   * 若使用了腾讯云微信小程序解决方案
   * 开发环境下，MySQL 的初始密码为您的微信小程序 appid
   */
  mysql: {
    host: 'localhost',
    port: 3306,
    user: 'root',
    db: 'cAuth',
```

```
    pass: 'xxx',
    char: 'utf8mb4'
  },

  cos: {
    /**
     * 区域
     * 华北：cn-north
     * 华东：cn-east
     * 华南：cn-south
     * 西南：cn-southwest
     * 新加坡：sg
     * @see https://www.qcloud.com/document/product/436/6224
     */
    region: 'cn-south',
    // Bucket 名称
    fileBucket: 'wximg',
    // 文件夹
    uploadFolder: ''
  },

  /**
   * 需要开通云直播服务
   * 参考指引
@https://cloud.tencent.com/document/product/454/7953#1.-.E8.A7.86.E9.A2.91.
E7.9B.B4.E6.92.AD.EF.BC.88lvb.EF.BC.89
   * 有介绍bizid 和 pushSecretKey的获取方法。
   */
  live: {
    // 云直播 bizid
    bizid: 0,

    // 云直播 推流防盗链key
    pushSecretKey: '',

    // 云直播 推流有效期单位秒 默认7天
    validTime: 3600*24*7
  },

  /**
   * 需要开通云通信服务
   * 参考指引
@https://cloud.tencent.com/document/product/454/7953#3.-.E4.BA.91.E9.80.9A.
E8.AE.AF.E6.9C.8D.E5.8A.A1.EF.BC.88im.EF.BC.89
   * 有介绍appid 和 accType的获取方法。以及私钥文件的下载方法。
   */
  im: {
    // 云通信 sdkappid
    sdkAppID: 0,
```

```
  // 云通信 账号集成类型
  accountType: "",

  // 云通信 管理员账号
  administrator: "",

  // 云通信 派发usersig的RSA 私钥
  privateKey: ""
},

/**
 * 多人音视频房间相关参数
 */
multi_room: {
  // 房间容量上限
  maxMembers: 4,

  // 心跳超时 单位秒
  heartBeatTimeout: 20,

  // 空闲房间超时 房间创建后一直没有人进入，超过给定时间将会被后台回收，单位秒
  maxIdleDuration: 30
},

/**
 * 双人音视频房间相关参数
 */
double_room: {
  // 心跳超时 单位秒
  heartBeatTimeout: 20,

  // 空闲房间超时 房间创建后一直没有人进入，超过给定时间将会被后台回收，单位秒
  maxIdleDuration: 30
},

/**
 * 直播连麦房间相关参数
 */
live_room: {
  // 房间容量上限
  maxMembers: 4,

  // 心跳超时 单位秒
  heartBeatTimeout: 20,

  // 空闲房间超时 房间创建后一直没有人进入，超过给定时间将会被后台回收，单位秒
  maxIdleDuration: 30
},
```

```
  /**
   * 辅助功能 后台日志文件获取相关 当前后台服务的访问域名。
   */
  selfHost:"https://xxxxxxxx.qcloud.la",

  // 微信登录态有效期
  wxLoginExpires: 7200
}
```

`logic/im_mgr.js` 云通信相关的处理，主要功能有：

```
module.exports = {
  getSig,                       // 计算云通信 账号登录IM所需要的userSig票据
  createGroup,                  // 创建IM聊天室，通过云通信提供的服务端对接用的
RestFul API实现
  destroyGroup,                 // 销毁IM聊天室，通过云通信提供的服务端对接用的
RestFul API实现
  notifyPushersChange           // IM聊天室成员进入和退出系统消息通知，通过云通信提供
的服务端对接用的RestFul API实现
}
```

`logic/double_room_mgr.js` 实时音视频房间管理模块，负责 `双人` 视频房间的创建，销毁，增加成员，删除成员，获取房间列表，获取房间成员列表等功能函数；另外也负责房间成员的心跳检查，对超时的成员进行删除处理。

`logic/multi_room_mgr.js` 实时音视频房间管理模块，负责 `多人` 视频房间的创建，销毁，增加成员，删除成员，获取房间列表，获取房间成员列表等功能函数；另外也负责房间成员的心跳检查，对超时的成员进行删除处理。

`logic/live_room_mgr.js` 实时音视频房间管理模块，负责`直播`视频房间的创建，销毁，增加成员，删除成员，获取房间列表，获取房间成员列表等功能函数；另外也负责房间成员的心跳检查，对超时的成员进行删除处理。

`logic/live_util.js` 云直播辅助函数，负责生成推流地址以及播放地址。外加一些用户ID分配和房间ID分配的功能函数。

`log.js` 后台日志模块，主要记录请求响应和错误两大类日志。请求响应日志按小时存储在`logs/response/`目录下，错误日志按小时存储在`logs/error/`目录下。最多存储7天日志。以上默认配置可以通过修改`log_config.js`来调整。 log4js v2 日志配置(若是2.0以下版本请用v1 的配置具体见log_config.js文件)：

```
{
  appenders:
  {
    //错误日志
    errorLogger:{
      type: "dateFile",                    //日志类型
      filename: errorLogPath,              //日志输出位置
      alwaysIncludePattern: true,          //是否总是有后缀名
      pattern: "-yyyy-MM-dd-hh.log",       //后缀，每小时创建一个新的日志文件
      daysToKeep: 7                        //自定义属性，错误日志的根目录
    },
    //响应日志
    resLogger:{
      type: "dateFile",
      filename: responseLogPath,
      alwaysIncludePattern: true,
      pattern: "-yyyy-MM-dd-hh.log",
      daysToKeep: 7
    },
    //控制台输出
    consoleLogger:{
      type: "console"
    }
  },
  categories:                                  //设置logger名称对应的的日志等
级
  {
    default:{
      appenders: ['consoleLogger'],
      level:"info"
    },
    errorLogger:{
      appenders:["errorLogger"],
      level:"error"
    },

    resLogger:{
      appenders:["resLogger"],
      level:"info"
    }
  }
}
```

3.本地部署

以MAC OS系统为例。

3.1安装Node

打开终端，输入命令，安装node

```
brew install node
```

完成后，输入命令，验证node安装，输出node版本信息表示安装成功

```
node -v
```

3.2部署

下载源码并解压，cd到server/app.js所在目录，执行下面命令安装pm2（作用类似“看门狗”，在node程序挂掉后，立即重启node程序）。

```
npm install -g pm2
```

执行下面的命令安装node依赖的模块。

```
npm install
```

执行下面命令运行服务器程序。

```
node app.js
```

如果没有报错说明程序运行成功，可以Ctrl + C 停止，并执行下面的命令。

```
pm2 start app
```

3.3验证部署

在本地浏览器，输入下面的链接地址

```
http://localhost:5757/weapp/utils/logfilelist
```

返回日志文件列表即表示本地部署成功。

注意：至此说明程序可以正常运行，但由于源码中config.js里面的配置都是“伪造”的，通过接口获取的推流地址是无法正常推流的。因此需要替换成您自己的云直播，云通信相关的参数。并部署到服务器上，才可以作为小程序的后台。

4.服务器部署

以CentOS 系统为例，描述部署过程。采用CentOS + Nginx + Node 的 环境。小程序和IOS都要求服务器支持HTTPS请求。和远程服务器通讯一般走ssh连接，可以用工具Xshell，secureCRT连接服务器。对于小文件（小于100kB）可以用rz 命令从本机传送文件至服务器，以及sz命令从远程服务器下载文件。非常方便。

4.1准备源码

config.js 中`bizid、pushSecretKey、sdkAppID、accountType、administrator和privateKey`等配置项需要您替换成腾讯云账号下的值。

```
  /**
   * 需要开通云直播服务
   * 参考指引
@https://cloud.tencent.com/document/product/454/7953#1.-.E8.A7.86.E9.A2.91.E
7.9B.B4.E6.92.AD.EF.BC.88lvb.EF.BC.89
   * 有介绍bizid 和 pushSecretKey的获取方法。
   */
  live: {
    // 云直播 bizid
    bizid: 0,

    // 云直播 推流防盗链key
    pushSecretKey: '',

    // 云直播 推流有效期单位秒 默认7天
    validTime: 3600*24*7
  },

  /**
   * 需要开通云通信服务
   * 参考指引
@https://cloud.tencent.com/document/product/454/7953#3.-.E4.BA.91.E9.80.9A.E
8.AE.AF.E6.9C.8D.E5.8A.A1.EF.BC.88im.EF.BC.89
   * 有介绍appid 和 accType的获取方法。以及私钥文件的下载方法。
   */
  im: {
    // 云通信 sdkappid
    sdkAppID: 0,

    // 云通信 账号集成类型
    accountType: "",

    // 云通信 管理员账号
    administrator: "",

    // 云通信 派发usersig的RSA 私钥
    privateKey: ""
  }
```

4.2Nginx 配置

如果您已经有**域名**以及域名对应的**SSL证书**存放在`/data/release/nginx/`目录下，请将下面配置内容中的 [1] 替换成您自己的域名， [2-1]替换成SSL证书的crt文件名， [2-2]替换成SSL证书的key文件名。替换后的内容存成文件`ssl.conf`，存放在`/etc/nginx/conf.d/`目录下。

```
upstream app_weapp {
    server localhost:5757;
    keepalive 8;
}

#http请求转为 https请求
server {
    listen      80;
    server_name [1];

    rewrite ^(.*)$ https://$server_name$1 permanent;
}

#https请求
server {
    listen      443;
    server_name [1];

    ssl on;

    ssl_certificate           /data/release/nginx/[2-1];
    ssl_certificate_key       /data/release/nginx/[2-2];
    ssl_session_timeout       5m;
    ssl_protocols             TLSv1 TLSv1.1 TLSv1.2;
    ssl_ciphers               ECDHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES128-
GCM-SHA256:DHE-RSA-AES256-GCM-SHA384:ECDHE-RSA-AES256-SHA384:ECDHE-RSA-
AES128-SHA256:ECDHE-RSA-AES256-SHA:ECDHE-RSA-AES128-SHA:DHE-RSA-AES256-
SHA:DHE-RSA-AES128-SHA;
    ssl_session_cache         shared:SSL:50m;
    ssl_prefer_server_ciphers on;

    location / {
        proxy_pass http://app_weapp;
        proxy_http_version 1.1;
        proxy_set_header Upgrade $http_upgrade;
        proxy_set_header Connection 'upgrade';
        proxy_set_header Host $host;
        proxy_cache_bypass $http_upgrade;
    }
}
```

4.3运行服务

输入命令，启动Nginx服务。

```
nginx -s reload
```

通过浏览器访问接口，会返回502错误，是因为Node没有运行，无法处理请求。 根据3.1本地部署介绍，将Node 运行起来即可。再用浏览器访问，就会看到服务器返回的json串了。