# Overview

1. 本工程已经配置好了关于运行后端和网页前端的webstorm工程设置。单击webstorm运行会自动打开后端并用chrome打开127.0.0.1:5555。或者:手动开启服务器后端:node main.js然后浏览器访问:127.0.0.1:5555
2. 服务器在darwin系统和win系统上会自动屏蔽EOS相关的功能。
3. 访问地址127.0.0.1:5555会自动跳转到127.0.0.1:5555/mainPage.html, mainPage.html即为电脑上的主要页面.
4. 源代码地址: <https://github.com/twoentartian/eos_grid_node>

# Important notice

1. 如果服务器没有域名的话,需要对服务器做IP静态保留,并且修改mainPage.html的第719行代码中的websocket的地址.
2. 关于局域网到广域网的穿透以及uPNP配置:uPNP请记住转发http端口和websocket端口并且需要转发TCP协议,穿透过后需要将websocket的端口更改为广域网的端口.

# Websocket document (前端后端交互接口)

WebSocket 地址: 'ws://127.0.0.1:5556'

Websocket指令:所有指令都按照JSON格式发送,服务器有关websocket的处理代码都在backend.js文件的CreateServer函数当中.

1. 获取目前后端仿真器的所有数据:

**let** command = {};  
command['type'] = 'require';  
command['content'] = '';  
websocket.send(JSON.stringify(command));

服务器返回的数据格式如下:

{

"type": "require",

"content": "",

"result": {

"market": {

"userPriceMap": {}, //这个不用管

"userPriceMap\_JsonUse": { //这个里面存储每个房屋主人的报价

"home11": "90.31",

"home12": "90.17",

"home13": "90.06",

"home14": "90.18",

"home15": "90.96",

"home21": "90.51",

"home22": "90.13",

"home23": "90.89",

"home24": "90.91",

"home25": "90.52"

}

},

"community": {

"idHomeMap": {}, //这个不用管

"homes": { //总共10个房屋的记录,这里只显示一个

"home11": { //用户ID

"state": true, //用户电源状态

"identification": {

"name": "User0", //身份记录,Name是用户名字,与id不同

"id": "home11"

},

"meter": { //电表记录

"id": "MeterId0",

"reading": 5,

"previousReading": 4

},

"cleanEnergy": 5, //用户剩余清洁能源

"fossilEnergy": 320, //用户剩余化石能源

"consumptionSpeed": 1, //消耗能源的速度,优先使用清洁能源

"usedEnergyType": "clean" //目前使用的能源类型

},

"home12":…….

},

"battery": [

{

"identification": {

"name": "battery",

"id": "battery0"

},

"capacity": 5000, //电池容量

"energy": 2659, //目前储能

"ratio": 0.5318, //电量百分比

"chargeLowThres": 0.3, //百分比低于这么多,从电网充电

"chargeHighThres": 0.5, //百分比高于这么多,停止从电网充电

"isInCriticalState": false //目前是否从电网充电,false为没有

}

],

"generator": [ //总共3台发电机,太阳能,风能和电网

{

"identification": { //发电机ID

"name": "solar",

"id": "solar0"

},

"meter": { //发电机的电表

"id": "solarMeter0",

"reading": 600,

"previousReading": 500

},

"generationSpeed": 100 //发电速度

},

{

"identification": {

"name": "wind",

"id": "wind0"

},

"meter": {

"id": "windMeter0",

"reading": 250,

"previousReading": 200

},

"generationSpeed": 50

}

],

"grid": [

{

"identification": {

"name": "grid",

"id": "grid0"

},

"meter": {

"id": "gridMeter0",

"reading": 2000,

"previousReading": 1600 //上一次读数

},

"generationSpeed": 400 //发电速率

}

]

},

"secondPerTick": 3 //仿真器每3秒会步进一次,即用户消耗电,发电机发电进电池,检测电池是否需要从电网充电以及市场的拍卖等.

}

}

1. 充值化石能源: type指定目前指令的作用,content为指令内容,对于充值化石能源而言,content格式为”{充值客户房屋的ID} {充值量}”

**let** command = {};

command['type'] = 'chargeFossil';  
command['content'] = 'home11 10';  
websocket.send(JSON.stringify(command));

服务器返回数据:

{

"type": "chargeFossil",

"content": "home11 10",

"result": "ok" //这个字段用来表示是否成功执行,如果为false,请检查指令格式和充值对象ID是否存在.

}

1. 更改清洁能源的报价: type指定目前指令的作用,content为指令内容,对于当前指令而言,content格式为”{客户房屋的ID} {报价}”

在每个周期当中.市场会根据当前报价,选出最高价进行交易.所以如果是测试需要,请把价格控制在91以上可以确保每次交易都会是这个用户.(用户home2x的初始报价是9~10之间的一个随机数, 用户home1x的报价是9~9.5之间的随机数,用户home1x的报价可以通过电脑前端修改,home2x不可通过前端修改)

**let** command = {};

command['type'] = 'changePrice';  
command['content'] = 'home11 91';  
dataSocket.send(JSON.stringify(command));

服务器返回数据:

{

"type": "changePrice",

"content": "home11 91",

"result": "ok"

}

# main.js document

主程序入口,用于初始化仿真器.

代码中初始化了10户用电户,并且进行了相应的更名(由于EOS不支持数字6~9的账户名)

# Backend.js document

主要功能:

1. 模拟小区电网和市场
2. websocket服务器

注意事项:

* Simulator采用单例模式,故simulator每次获取到的实例都是同一实例.
* Websocket中的require指令所返回的数据即是simulator的结构.
* 如果需要扩充websocket指令集,请在341行wsCommandMap中加入新的指令:

Sample:

wsCommandMap['require'] = **function**(conn, commandReceived){  
 // Send JSON  
 commandReceived['result'] = Simulator.getInstance();  
 conn.sendText(JSON.stringify(commandReceived));  
};

其中,’require’为http请求中的type, conn为websocket链接, commandReceived为接收到的整个指令object(包括content和type,已经反JSON)

# Server.js

该文件的主要职责是以下几点:

1. 路由规划:访问localhost:5555过后,判断设备类型,电脑设备导引至mainPage.html,手机设备导引至tunnel.html (也可以导引至专用页面phone.html, 代码位于38行)\
2. 发送所有需要的文件,如SVG,PNG,JS等,http请求的路径和工程共用同一路径.如果有新加的类型文件如mp3等,需要更新http响应头(目前只做了SVG的响应头)(代码51行)
3. 该文件中原生的nodejs服务器已经被express server取代,请注意函数前的注释.

# eosOperation.js document

该文件主要用于和eos交互,注意交互之前需要使用eosio-wallet文件夹中的createHome.py来为nodeos做初始化工作.(创建账户,创建代币等工作)

该文件实现了3种交互方法:

1. 纯RPC(由于EOS的插件更新导致wallet plugin不可用,所以无法签署交易).
2. RPC+eosjs, 一种用RPC获取区块信息然后使用eosjs做签署的API,该API可以使用,但是不建议使用,原因:由于仿真器中很多交互操作是同时发生的,而这些操作无法被记录在同一区块上(eos出块时间0.5s),导致很容易丢失记录.
3. Cleos, 通过wireshark抓包发现,cleos与方案2基本相同,但是cleos可以通过流量控制来防止2中的问题.

目前代码使用方案3.

请注意,如果更新了区块链账户的私钥和钱包配置,需要重新修改createHome.py中的私钥和eosOperation中的私钥.

# EOS配置指南:

1. Eos wallet请直接套用github上的eosio-wallet文件夹中的内容
2. Nodeos务必启用所有插件
3. 常用指令:

(1). 开启nodeos nodeos -e -p eosio --delete-all-blocks (-p 生产者账户 –delete-all-blocks 删除所有区块)

(2). 转账(需要先部署eos.token) cleos push action operator transfer '[ "home11", "home12", "1.0000 CLEAN", "m" ]' -p home11

operator:执行人,home11:转出的人,home12:转入的人, -p home11 请求home11的许可

(3). cleos get currency balance eosio.token home11 查询home11拥有多少eosio.token发行的货币

更多指令请查询createHome.py以及官网手册