cloud solution

WEBSOCKET+JSON

protocol

Revision history

Date	Version	Note	Author
2016-03-25	1.0	The original version	Chingzou
2016-04-20	1.1	1.Senduser，getuserinfo,setfp,setcard,setpwd :add user name item. 2、add deleteuserlock,cleanuserlock function	Chingzou
2016-04-21	1.2	1、add the function of “setuserinfo”,can set fingerprint,password,card. 2 、 delete the function of”set fp”,”setcard”,”setpwd”,for these are replace of setuserinfo 3、mofify the function “deleteuser”,when set the backupnum 0~9:delete finger. 10:delete password ,11:delete card ;12 : elete the user of all fingerprint; 13:delete the user all info:fingerprint,password,card and name. 4 、 modify the function of :getuserlist, getnewlog,getalllog. When the log if empty, it will return success and the cout is 0	chingzou
2016-05-17	1.3	1,add the function “reboot”.	Chingzou
2016-05-25	1.4	Add the function “settime”	Chingzou
2016-07-06	1.5	Add the note of the log	Chingzou

Catalog

Note：

1. Use websocket protocol to communication,the websocket version is RFC6455 13,The default listen port is 7788,no TLS encrypt.
2. The data format use Json.you can use javascript to Serializer and Deserialize very easy.
3. All the key value of json use lower-char.the name or all chinese char use UTF8 encoded.
4. About backupnum:0~9:fingerprint 10:password; 11:rfid card. One user can have 10 fingerprints and one password and one one rfid card.

1)Terminal active send data to server

1. Register

Terminal send register message:

```
{
   "cmd":"reg", //command
   "sn",”ZX0006827500", //Terminal serial number,fixed by the manufactory,unique
   "devinfo",{
                "modelname":"tfs30",
                “usersize”:3000,          //user capacity 1000/3000/5000
                "fpsize":3000,           //fingerprint capacity 1000/3000/5000
                "cardsize":3000，        //rfid card capacity 1000/3000/5000/10000
                "pwdsize":3000,          //password capacity
                "logsize":100000,      //logs capacity
                “useduser”:1000,
                "usedfp":1000,
                "usedcard":2000,
                "usedpwd":400,
                "usedlog":100000,
                “usednewlog”:5000,
                "fpalgo":"thbio3.0",    //fingerprint algorithm      thbio1.0 or thbio3.0
                "firmware":"th600w v6.1", //terminal firmware
                "time":"2016-03-25 13:49:30" //terminal datetime
                }
}
```

Server response message:

Success：

```
{
   "ret":"reg",    //command
   "result":true,
   "cloudtime":"2016-03-25 13:49:30”  //server now time
}
```

Fail：

```
{
   "ret":"reg",
```

```
   "result":false,
   "reason":1
}
```

2. Send the logs

Terminal send the message:

```
{
  "cmd":"sendlog",
  "count":2,
  "record":[
                {
                "enrollid":1,
                "time":"2016-03-25 13:49:30",
                "mode":0, //0 fp 1:card 2:pwd
                "inout":0, //0 in 1:out
                "event":0 // normal is 0 ,tfs20/tfs30 model have f1~f4 key pad, can customization
                },
                {
                "enrollid":2,
                “time ":"2016-03-25 13:49:30",
                "mode":0,
                "inout":0,
                "event":1
                }
  ]
}
```

Server response message:

Success：

```
{
   "ret":"sendlog",
   "result":true,
   "cloudtime":"2016-03-25 13:49:30"
}
```

Fail:

```
{
   "ret":"sendlog",
   "result":false,
   "reason":1
}
```

Note: about the logs

When the enrollid != 0 then :

Mode: 0 fp, 1,card 2,password : it means the user use the fingerprint/card/password to access

Inout: //0 in 1:out :it means the user use the master machine or the child machine to access(the access controler can add a child machine to work. Normal

the master machine inside the door and the child machine ouside the door)

Event: 0~16 : customization ,must work with the software, some machine have the key (F1~F4).when press F1 key and verifyed ok ,the value is 1.and the software can set this key as onduty

When the enrollid = 0 then

Mode: 0;

Inout : 1:

Enent: the status or the event of the door.

```
typedef enum
{
    UI_MGLOG_CLOSED,//door is closed
    UI_MGLOG_OPENED, //dorr is opened
    UI_MGLOG_HAND_OPEN, //use exit button to open the door
    UI_MGLOG_PROG_OPEN, //use software to open the door
    UI_MGLOG_PROG_CLOSE, //use software to close the door
    UI_MGLOG_ILLEGAL_OPEN, //the door is illegal opend
    UI_MGLOG_ILLEGAL_REMOVE, //the machine is removed
    UI_MGLOG_ALARM, //input alarm
} T_UI_MGLOG_TYPE;
```

3. Send user information

Note:When use keypad to add new user,and then send this message to server

Terminal send the message:

Fingerprint：

```
{
  "cmd":"senduser",
  “enrollid”:1,
“name”:”chingzou”,
  “backupnum”:0,    //0~9 fingerprint
  “admin”:0,
  “record”,”kajgksjgaglas” //the string length less then 1620 for THbio3.0 and less 1024 for THbio1.0
}
```

Rfid card：

```
{
  "cmd":"senduser",
  “enrollid”:1,
  “name”:”chingzou”,
  “backupnum”:11,
  “admin”:0,
  “record”,2352253
}
```

password：

```
{
```

"cmd":"senduser",
“enrollid”:1,
“name”:”chingzou”,
“backupnum”:10,
“admin”:0,
“record”,12345678 //max 8 digit
}
Server response message:
Success：
{
"ret":”senduser ",
"result":true,
"cloudtime":"2016-03-25 13:49:30"
}
Fail：
{
"ret":”senduser ",
"result":false,
"reason":1
}

2)Server active push message to terminal

1. Get user list

Server send the message:
{
"cmd":"getuserlist",
″stn″:true //stn:if this is the first package,set true;or response package set false
}
Terminal response message:
Success：
{
"ret":"getuserlist",
"result":true,
“count ":40, //1~40 must less then 40 records per one package
“from”,0,
“to”:39,
“record ":[
{
“enrollid ":1,
"admin ":0, // 0: normal user ; 1:adminstrator 2:super user(just only can add user and use u-disk download the log)
"backupnum ":0 //0~9 fingerprint 10:password 11:rfid card

```
            },
            {
               “enrollid ":2,
               "admin ":1,
               "backupnum ":0
            },
            {
               “enrollid ":3,
               "admin ":0,
               "backupnum ":10 //this is Rfid card
            },
            ......
   ]
}
```

Server response message:

```
{
   "cmd":"getuserlist",
   “stn”:false//response package,should set to false
}
```

Terminal send the second package again:

```
{
   "ret":"getuserlist",
   "result":true,
   "count ":40, //1~40
   “from”：40,
   “to”:79,
   “record ":[
            {
               “enrollid ":1234,
               "admin ":0,
               "backupnum ":0
            },
            {
              “enrollid ":2345,
               "admin ":1,
               "backupnum ":0
            },
            {
              “enrollid ":5677,
               "admin ":0,
               "backupnum ":10
            },
            ......
   ]
```

}

.....

Whe users is empty :the machine return:

{

"ret":"getuserlist",

"result":true,

"count ":0,

“from”:0,

“to”:0,

“record ":[]

}

Fail：

{

"ret":"getuserlist",

"result":false,

"reason":1

}

2. Get user information

Fingerprint：

{

"cmd":" getuserinfo "

"enrollid ":1,

"backupnum ":0

}

Terminal response message:

success：

{

"ret":" getuserinfo ",

"result":true,

“enrollid":1,

“name”:”chingzou”,

“backupnum":0,

“admin":0,

"record":"aabbccddeeffggddssiifdjdkjfkjdsjlkjal",

}

Fail：

{

"ret":”getuserinfo ",

"result":false,

"reason":1

}

Rfid card：

{

```
   "cmd":”getuserinfo "
   "enrollid ":1,
   "backupnum ":11
}
```

Terminal response message：

Success：

```
{
   "ret":”getuserinfo ",
   "result":true,
   “enrollid":1,
   “name”:”chingzou”,
   “backupnum":11,
   “admin":0,
   "record":23532253
}
```

Fail：

```
{
   "ret":”getuserinfo ",
   "result":false,
   "reason":1
}
```

Password：

```
{
   "cmd":”getuserinfo "
   "enrollid ":1,
   "backupnum ":10
}
```

Terminal response message：

Success：

```
{
   "ret":”getuserinfo ",
   "result":true,
   “enrollid":1,
   “name”:”chingzou”,
   “backupnum":10,
   “admin":0,
   "record":23532253
}
```

Fail：

```
{
   "ret":”getuserinfo ",
   "result":false,
   "reason":1
}
```

3. Download user information

Fingerprint：

Server send message：

```
{
   "cmd":"setuserinfo",
   "enrollid":1,
   “name”:”chingzou”,
   "backupnum",0,
   "admin":0,
   "record":"aabbccddeeffggddssiifdjdkjfkjdsjlkjalflsgsadg"
}
```

Password：

```
{
   "cmd":"setuserinfo",
   "enrollid":1,
   “name”:”chingzou”,
   "backupnum",10,
   "admin":0,
   "record":12345678
}
```

Rfid card：

```
{
   "cmd":"setuserinfo",
   "enrollid":1,
   “name”:”chingzou”,
   "backupnum",11,
   "admin":0,
   "record":2352253
}
```

2、Terminal response message：

Success：

```
{
   "ret":"setuserinfo ",
   "result":true
}
```

Fail：

```
{
   "ret":"setuserinfo ",
   "result":false,
   "reason":1
}
```

4. Delete user information

Server send message：

```
{
   "cmd":"deleteuser"
   "enrollid",1,
   "backupnum":0    //0~9 fp; 10: password 11:card
}
```

Terminal response message：

Success：

```
{
   "ret":"deleteuser",
   "result":true
}
```

Fail：

```
{
   "ret":"deleteuser",
   "result":false,
   "reason":1
}
```

5. Get user name

Server send message：

```
{
   "cmd":"getusername",
   "enrollid":1
}
```

Terminal response message：

Success：

```
{
   "ret":"getusername",
   "result":true,
   "record":"chingzou”//utf8 or ascii
}
```

Fail：

```
{
   "ret":"getusername",
   "result":false,
   "reason":1
}
```

6. Set user name

Server send message：

```
{
   "cmd":"setusername",
   “count”:50, // must less then 50 record per package
    “record”,[
```

```
                {
                  “enrollid”:1,
                  “name”:”chingzou”
                },
                {
                  “enrollid”:2,
                  “name”:”chingzou2”
                },
                ……..
               ]
}
```

Terminal response message：

Success：

```
{
  "ret":"setusername",
  "result":true
}
```

Fail：

```
{
  "ret":"setusername",
  "result":false,
  "reason":1
}
```

7. Clean all users

Server send message：

```
{
  "cmd":"cleanuser"
}
```

Terminal response message：

Success：

```
{
  "ret":"cleanuser",
  "result":true
}
```

Fail：

```
{
  "ret":"cleanuser",
  "result":false
  "reason":1
}
```

8. Get new logs

Server send message：

```
{
  "cmd":"getnewlog",
  “stn”:true
}
```

Terminal response message：

Success：

```
{
   "ret":"getnewlog ",
   "result":true,
   "count":1000,
   “from”:0,
   “to”:49,
   "record":[
                        {
                          "enrollid":1,
                          "time":"2016-03-25 13:49:30",
                          "mode":0, //0 fp 1:card 2:pwd
                          "inout":0,    //0 in 1:out
                          "event":0
                        }
                        {
                          "enrollid":2,
                          “time ":"2016-03-25 13:49:30",
                          "mode":0, //0 fp 1:card 2:pwd
                          "inout":0, //0 in 1:out
                          "event":1
                       }
                         ......
                ]
}
```

Server response message：

```
{
  "cmd":"getnewlog",
  “stn”:false
}
```

Terminal send the second package:

```
{
   "ret":"getnewlog ",
   "result":true,
   "count":1000,
   “from”:50,
   “to”:99,
   "record":[
                        {
```

```
                    "enrollid":111,
                    "time":"2016-03-25 13:49:30",
                    "mode":0, //0 fp 1:card 2:pwd
                    "inout":0,    //0 in 1:out
                    "event":0
                   }
                   {
                    "enrollid":112,
                    “time ":"2016-03-25 13:49:30",
                    "mode":0, //0 fp 1:card 2:pwd
                    "inout":0, //0 in 1:out
                    "event":1
                  }
                   ......
              ]
}
```

Whe newlog is empty :the machine return:

```
{
   "ret":"getnewlog ",
   "result":true,
   "count ":0,
   “from”:0,
   “to”:0,
   “record ":[]
}
```

Fail：

```
{
   "ret":"getnewlog ",
   "result":false
   "reason":1
}
```

9. Get all logs

Server send message：

```
{
  "cmd":"getalllog",
  “stn”:true
}
```

Terminal response message：

Success：

```
{
   "ret":"getalllog",
   "result":true,
   "count":1000,
   “from”:0,
```

```
   “to”:49,
   "record":[
                        {
                         "enrollid":1,
                         "time":"2016-03-25 13:49:30",
                         "mode":0, //0 fp 1:card 2:pwd
                         "inout":0,    //0 in 1:out
                         "event":0
                        }
                        {
                         "enrollid":2,
                         “time ":"2016-03-25 13:49:30",
                         "mode":0, //0 fp 1:card 2:pwd
                         "inout":0, //0 in 1:out
                         "event":1
                       }
                        ......
                ]
}
```

Server response message：

```
{
  "cmd":"getalllog",
  “stn”:false
}
```

Terminal send the second package:

```
{
   "ret":"getalllog",
   "result":true,
   "count":1000,
   “from”:50,
   “to”:99,
   "record":[
                        {
                         "enrollid":111,
                         "time":"2016-03-25 13:49:30",
                         "mode":0, //0 fp 1:card 2:pwd
                         "inout":0,    //0 in 1:out
                         "event":0
                        }
                        {
                         "enrollid":112,
                         “time ":"2016-03-25 13:49:30",
                         "mode":0, //0 fp 1:card 2:pwd
                         "inout":0, //0 in 1:out
```

```
                    "event":1
                }
                  ......
            ]
}
```

Whe newlog is empty :the machine return:

```
{
   "ret":"getalllog ",
   "result":true,
   "count ":0,
   “from”:0,
   “to”:0,
   “record ":[]
}
```

Fail：

```
{
   "ret":"getalllog",
   "result":false
   "reason":1
}
```

10. Clean all logs

Server send message：

```
{
   "cmd":"cleanlog"
}
```

Terminal response message：

Success：

```
{
   "ret":"cleanlog",
   "result":true
}
```

Fail：

```
{
   "ret":"cleanlog",
   "result":false
   "reason":1
}
```

11. Initialize system

Note: intialize system will delete all users and all logs,and the setting still not change.

Server send message：

```
{
   "cmd":"initsys"
```

```
}
```

Terminal response message：

Success：

```
{
   "ret":" initsys",
   "result":true
}
```

Fail：

```
{
   "ret":" initsys ",
   "result":false
   "reason":1
}
```

12. Reboot

Note:when terminal receive this message,will reboot immediately,so no repense message.

Server send message：

```
{
   "cmd":"reboot"
}
```

13. Clean all administrators

Note: this command will change all admin to normal user.

Server send message：

```
{
   "cmd":"cleanadmin"
}
```

Terminal response message：

Success：

```
{
   "ret":" cleanadmin",
   "result":true
}
```

Fail：

```
{
   "ret":" cleanadmin",
   "result":false
   "reason":1
}
```

14. settime

Note: set the terminal date and time

Server send message：

```
{
   "cmd":"settime",
```

```
   "cloudtime":"2016-03-25 13:49:30"
}
```

Terminal response message：

Success：

```
{
   "ret":" settime ",
   "result":true
}
```

Fail：

```
{
   "ret":" settime",
   "result":false
   "reason":1
}
```

15. Set terminal parameter

Server send message：

```
{
   "cmd":"setdevinfo",
   “deviceid”:1, //Termial id
   “language”:0, //as the tips option below
   “volume”:0, //0~10    default:6
   “screensaver”:0 // 0:no screensaver 1~255 :when inopration wait for 1~255 seconds and then
                        go to screensaver.
   “verifymode”:0, //opotion as show on tips below.
```

“sleep”: 0 , //0: no sleep; 1: sleep,and the fingerprint sensor will allway on
“userfpnum”:3,//how many fingerprints per user 1~10， default:3
“loghint”:1000, //when the logs will full less then 1000 ,and the terminal will hint;0:no hint
“reverifytime”:0 //reverify time 0~255 minute
}

Terminal response message：

Success：

```
{
   "ret":" setdevinfo ",
   "result":true
}
```

Fail：

```
{
   "ret":" setdevinfo ",
   "result":false
   "reason":1
}
```

Tips:1.setting the terminal common parameter, the child item is option.if don’t want to set that item,you can ignore it

For example:

If you just want to modify the item of “volume” and “sleep”

You can send the message like this:

```
{
   “cmd”:”setdevinfo”,
   “volume”:8, //volume as the first item
   “sleep”:1
}
```

Or like this:

```
{
   “cmd”:”setdevinfo”,
   “sleep”:true, //change sleep as first item;true=1 false=0,you can set whatever you want
   “volume”:8
}
```

Is it so easy?

2.the languae option:

```
enum
{
      UILANG_EN, //==0 english
      UILANG_SC, //==1 simplified Chinese
      UILANG_TC, //==2 taiwan chinese
      UILANG_JAPAN, //==3 japanese
      UILANG_NKR, //==4 north korean
      UILANG_SKR, //==5 south korean
```

```
        UILANG_THAI, //==6 thai
        UILANG_INDONESIA, //==7 Indonesian
        UILANG_VIETNAM, //==8 Vietnamese
        UILANG_SPA, //==9 Spanish
        UILANG_FAN, //==10 French
        UILANG_POR, //==11 Portuguese
        UILANG_GEN, //==12 German
        UILANG_RUSSIA, //==13 Russian
        UILANG_TUR, //==14 Turkish
        UILANG_ITALIAN, //==15 Italian
        UILANG_CZECH, //==16 Czech
        UILANG_ALB, //==17 Arabic
        UILANG_PARSI//==18 Farsi
    }
    3.Verify mode：
    enum
    {
        VERIFY_KIND_FP_CARD_PWD, //==0 Rfid card or Fingerprint or Password
        VERIFY_KIND_CARD_ADD_FP, //==1 Rfid card and Fingerprint
        VERIFY_KIND_PWD_ADD_FP, //==2 Password and Fingerprint
        VERIFY_KIND_CARD_ADD_FP_ADD_PWD, //==3 Rfid card and Fingerprint and Password
          VERIFY_KIND_CARD_ADD_PWD, //==4 Rfid card and Password
    };
```

16. Get terminal parameter

Note:1.Getting the terminal common parameter Server send message.

```
{
   “cmd”:”getdevinfo”
}
```

Terminal response message：

Success

```
{
   "ret":"getdevinfo",
   “result”:true,
   “deviceid”:1,
   “language”:0,
   “volume”:0,
   “screensaver”:0
   “verifymode”:0,
   “sleep”: 0 ,
   “userfpnum”:3,
   “loghint”:1000,
   “reverifytime”:0
}
```

Fail：

```
{
   "ret":" getdevinfo ",
   "result":false
   "reason":1
}
```

17. Open door

Note:Open the door

Server send message：

```
{
   "cmd":"opendoor"
}
```

Terminal response message：

Success：

```
{
   "ret":" opendoor",
   "result":true
}
```

Fail：

```
{
   "ret":" opendoor",
   "result":false
   "reason":1
}
```

18. Set the access paramete

Note:setting the access all common paramete,the items are option, if your don't want to set this item you can ignore them.the command so complex,ha ha ha!!!

Server send message：

```
{
   "cmd":"setdevlock",
   “opendelay”:5, //open door delay
   “doorsensor”:0, //the door sensor type: 0:disable 1:NC(normal close) 2:NO(normal open)
   “alarmdelay”:0, //door sensor alarm:when open the door and not close,the time more then 1~255 minute the access will alarm. 0:disable.
  “threat”:0, //theat alarm: 0:disable 1.open the door and alarm 2.just alarm 3.just open the door
  “InputAlarm”:0, //0.disable 1,alarm1 output 2.alarm2 output
  “antpass”: 0 ,//0.disable 1,host machine is inside.2.host machine is outside
  “interlock”:0 ,//0:disable.1.enable
  “mutiopen”:0, //0:disable;1~4:must 1~4 people press finger at the same time to open the door
  “tryalarm”:0, //0:disable 1~10:when try press the wrong finger 1~10 times ,the access will alarm
  “tamper”:0, //0:disable 1.enable
  “wgformat”:0, //weigand format 0 :26 1:34
  “wgoutput”: 0, //weigand output data：0,：enroll id ;1:1+enroll id ;2:device id+enroll id
```

```
"cardoutput":0, //if this user register a rfid card,they press finger ,weigand directly output rfid card
                  number;0:disable;1:enable;
"dayzone":[      //8 groups at most. one group means one day,and have 5 sections per day at
                  most.you can change one or more sections or groups as you want,and ignore the
                  remain
            {
              "day": [
                        {"section":"06:00~07:00"},
                        {"section":"08:30~12:00"},
                        {"section":"13:00~17:00"},
                        {"section":"18:00~21:00"},
                        {"section":"22:00~23:30"},
                      ]
              },
              {
               "day": [
                        {"section":"00:01~23:59"},
                      ]
              },
              ……..
            ],
"weekzone":[   //8 groups at most,one group means one week,you can change one or more
                  groups what you want and ignore the remain
              {"week":[
                        {"day":1}, //monday
                        {"day":1}, //tuesday
                        {"day":1}, //wednesday
                        {"day":1}, //thursday
                        {"day":1}, //friday
                        {"day":2}, //saturday
                        {"day":2}, //sunday
                       ]
              },
              {"week":[   //the second weed zone
                        {"day":1},
                        {"day":1},
                        {"day":1},
                        {"day":1},
                        {"day":1},
                        {"day":2},
                        {"day":2},
                       ]
              },
              ……..
```

```
                ],
“lockgroup”:[
                    {“group”:1234},
                    {“group”:126},
                    {“group”:348},
                    {“group”:139},
                    {“group”:15}
                ]
}
```

Terminal response message：

Success：

```
{
   "ret":" setdevlock ",
   "result":true
}
```

Fail：

```
{
   "ret":" setdevlock ",
   "result":false
   "reason":1
}
```

Tips:1, if you just have one dayzone and one weekzone,you can send the short message like this:

```
{“cmd”:”setdevlock”,
   “dayzone”:[ {“day”,[ {“section”:”07:00~18:00”}]}],
   “weekzone”:[ {“week”,[ {“day”:1}]}]
}
```

2,the relationship dayzone and weekzone like this:

(user access parameter of weekzone =3)-> weekzone[3]-> Monday[1]->dayzone[1]->sections
->Tuesday[2]->dayzone[2]->sections

If(day zone [1] section is ”00:01~23:59”) and (day zone [2] section is ”00:00~00:00”)

If this user press finger, first check his access parameter of weekzone is 3.

And then find the weekzone 3 , and find today is Monday, and then find Monday setting is dayzone 1 .

Continue find the dayzone 1, finding this dayzone just have one section:00:01~23:59

It means all day can access,then the the last action: open the door.

If today is Tuesday: oh!! this dayzone section is”00:00~00:00” allday can not access.

So the last action: refuse this user access the door.

Then this user Monday can open the door ,but Tuesday can not open the door allday.

Hope you can understand.or you can take a machine byhand,try and try.

3.About the “lockgroup”

For example: there have 3 departments in one company:

Financial department->users : TOM ,Obama ,Lily

Sale department->users: Clinton ,Bush ….

Warehouse department->users: Cruz ,Hilari

You can set the" user access paramete ->group" to financial department users as 1
Sale department users as 2
Warehouse department user as 9

And then the "lockgroup" have the section :129:

So : Obama(group 1) and Bush(group 2) and Crus(group 9) press the finger at the same time, just can open the door.

Or:Tom(group 1) and Bush(group 2) and Hilari(group 9) press the finger at the same time, can open the door.

Or: "lockgroup" have the section : 119:

TOM (group1),Obama(group 1), Cruz(group 9) press the finger at the same time can open the door.

19. Get the access parameter

Server send message：

```
{
   "cmd":" getdevlock "
}
```

Access response message:

Success

```
{
   "ret":"getdevlock",
   “result”:true
   “opendelay”:5,
   “doorsensor”:0,
   “alarmdelay”:0,
   “threat”:0,
   “InputAlarm”:0,
   “antpass”: 0 ,
   “interlock”:0 ,
“mutiopen”:0,
“tryalarm”:0,
“tamper”:0,
“wgformat”:0,
“wgoutput”: 0,
“cardoutput”:0,
“dayzone”:[
                {
                   “day”: [
                              {“section”:”06:00~07:00”},
                              {“section”:”08:30~12:00”},
                              {“section”:”13:00~17:00”},
                              {“section”:”18:00~21:00”},
                              {“section”:”22:00~23:30”},
                           ]
                 },
```

```
                {
                  “day”: [
                            {“section”:”00:01~23:59”},
                          ]
                },
                ……..
              ],
“weekzone”:[
                {“week”:[
                            {“day”:1},
                            {“day”:1},
                            {“day”:1},
                            {“day”:1},
                            {“day”:1},
                            {“day”:2},
                            {“day”:2},
                          ]
                },
                {“week”:[
                            {“day”:1},
                            {“day”:1},
                            {“day”:1},
                            {“day”:1},
                            {“day”:1},
                            {“day”:2},
                            {“day”:2},
                          ]
                },
                ……..
              ],
“lockgroup”:[
                  {“group”:1234},
                  {“group”:126},
                  {“group”:348},
                  {“group”:139},
                  {“group”:15}
              ]
}
Fail：
{
   "ret":" setdevlock ",
   "result":false
   "reason":1
}
```

20. Get the user access parameter

Server send message：

```
{
   "cmd":"getuserlock",
   “enrollid”:1
}
```

Terminal response message：

Success：

```
{
   "ret":" getuserlock ",
   "result":true,
   “enrollid”:1,
   “weekzone”:1, //the weekzone as set above
   “group”:1,    //for the 0:no group,1~9:belong the group of 1~9
     “starttime”:”2016-03-25 01:00:00”, //valid datetime
     “endtime”: ”2099-03-25 23:59:00”,
}
```

Fail：

```
{
   "ret":" getuserlock ",
   "result":false
   "reason":1
}
```

21. Set the users access parameter

Server send message：

```
{
   "cmd":"setuserlock",
   “count”:40,
   “record”:[
                {
                “enrollid”:1，
                “weekzone”:1,
                “group”:1,
                “starttime”:”2016-03-25 01:00:00”,
                “endtime”: ”2099-03-25 23:59:00”
                },
                {
                “enrollid”:2，
                “weekzone”:1,
                “group”:1,
                “starttime”:”2016-03-25 01:00:00”,
                “endtime”: ”2099-03-25 23:59:00”
                },
```

```
                    ……..
                ]
}
```

Terminal response message：

Success：

```
{
   "ret":" setuserlock ",
   "result":true
}
```

Fail：

```
{
   "ret":" setuserlock ",
   "result":false
   "reason":1
}
```

22. Delete the user access parameter

Server send message：

```
{
   "cmd":"deleteuserlock",
   “enrollid”:1
}
```

Terminal response message：

Success：

```
{
   "ret":" deleteuserlock ",
   "result":true
}
```

Fail：

```
{
   "ret":" deleteuserlock ",
   "result":false
   "reason":1
}
```

23. Clean all user access parameter

Server send message：

```
{
   "cmd":"cleanuserlock"
}
```

Terminal response message：

Success：

```
{
   "ret":" cleanuserlock ",
   "result":true
```

}

Fail：

{

"ret":" cleanuserlock ",

"result":false

"reason":1

}