cloud solution

HTTP/HTTPS+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 ”setfp”,”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
2017-11-06	1.7	1.Add enable user/disable 2.add the sendlog reply the access open or not	chingzou
2018-6-7	1.8	1.add the sn for allcommand 2.add opendoor of doornum for access controller(access controller have 4 doors) 3.add timezone2 timezone3 of setuserlock and getuserlock for access controller((access controller have 4 doors)	chingzou
2019-3-27	1.9	1.sendlog add index	chingzou
2021-2-2	2.0	1.add AI device photo info(backupnum is 50,and log image and temp)	
2025-03-21	2.8	Remote Add User	
2025-06-11	2.9	send user information (Newly Added Interface),Added the interface of'Set door status'	
2025-12-08	3.0	Correct the format	

Catalog

Note:

1. Use http/https protocol to communication.use Post to submit the requests
2. Need AI face machine version upto 4.57 or 1.27(eco)
3. Set the machine domain name start with :http:// or https:// or set the default server port as 80 or 443(https)

   Ex: domain name:yunatt.com port :80.

   http://yunatt.com:8080/api port: any

   yunatt.com port 443

   https://yunatt.com:8080/api port: any
4. The data format use Json.you can use javascript to Serializer and Deserialize very easy.
5. All the key value of json use lower-char.the name or all chinese char use UTF8 encoded.
6. About backupnum:

   0~9: fingerprint

   10: password

   11:rfid card

   20~27: static face

   40~41: palm vein

   50 : photo(format is base64)

One user can have 10 fingerprints and one password and one one rfid card

A. Terminal active send data to server

1. Register

Terminal send register message:

```
{
   "cmd":"reg",                          //command
   "sn":"ZX12345678",                    //Terminal serial number,fixed by the manufactory,unique
   "cpusn":"123456789",                  //CPU serial number,fixed
   "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
      "mac":"00-01-A9-01-00-01"          //lan MAC address
   }
}
```

Server response message:

Success:

```
{
   "ret":"reg",                          //command
   "result":true,
   "cloudtime":"2016-03-25 13:49:30",    //server now time
   "nosenduser":true                     //tell the terminal,aoto send the new user message or not
}
```

Fail:

```
{
   "ret":"reg",
   "result":false,
   "reason":"did not reg"                //this message will display on screen
}
```

2. Send the logs

Terminal send the message:

```
{
   "cmd":"sendlog",
   "sn":"ZX12345678",
   "count":2,
   "logindex":10,    //add 2019-03-27
   "record":[
      {
         "enrollid":1,
         "time":"2016-03-25 13:49:30",
         "mode":0,    //1:fp 2:pwd 3:card 8:face
         "inout":0,    //0:in 1:out
         "event":0,    //normal is 0 ,tfs20/tfs30 model have f1~f4 key pad, can customization
         "temp":36.5,    //people temperature
         "image":"gesg524hgd"    //realtime punch image, encode by Base64
      },
      {
         "enrollid":2,
         "time":"2016-03-25 13:49:30",
         "mode":0,
         "inout":0,
         "event":1,
         "temp":36.5,    //people temperature,just temperature device support
         "image":"gesg524hgd"    //realtime punch image, encode by Base64
      }
      ......
   ]
}
```

Server response message:

Success:

```
{
   "ret":"sendlog",
   "result":true,
   "count":2,                    //add 2019-03-27
   "logindex":10,              //add 2019-03-27
   "cloudtime":"2016-03-25 13:49:30",
   "access":1,      //When AI face is set to Servermode 1 for open the door ,0 can not open the
                    door ,extern function,
   "message":"message"    //When AI face is set to Servermode, return device interface
                           information
}
```

Fail:

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

Note: about the logs

When the enrollid != 0 then :

Mode: 1:fp,2:password,3:card,8:face: 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",
   "sn":"ZX12345678",
   "enrollid":1,
```

```
   "name":"chingzou",
   "backupnum":0,    //0~9 fingerprint ,20-27 is static face,40-41 is palm vein,50 is photo
   "admin":0,
   "record":"kajgksjgaglas" //the string length less then 1620 for THbio3.0 and less 1024 for
                            THbio1.0
}
```

Rfid card:

```
{
   "cmd":"senduser",
   "sn":"ZX12345678",
   "enrollid":1,
   "name":"chingzou",
   "backupnum":11,
   "admin":0,
   "record":2352253
}
```

Password:

```
{
   "cmd":"senduser",
   "sn":"ZX12345678",
   "enrollid":1,
   "name":"chingzou",
   "backupnum":10,
   "admin":0,
   "record":12345678      //max 8 digit
}
```

Palm vein:

```
{
   "cmd":"senduser",
   "sn":"ZX12345678",
   "enrollid":1,
   "name":"chingzou",
   "backupnum":40,     // 0~9: Fingerprint 10: password 11: card 40:palm vein 41:palm vein
50:photo
   "admin":0,
   "record":"kajgksjgaglasdjgjjglksajlgjkdgjajkdjgksaj"    // feature data
}
```

Photo:

```
{
   "cmd":"senduser",
   "sn":"ZX12345678",
   "enrollid":1,
   "name":"chingzou",
   "backupnum":50,       //Base64
```

```
   "admin":0,
   "record":"aabbccddeeffgg..." // Base64
}
```

Server response message:

Success:

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

Fail:

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

4. Send checklive

Note: When use HTTP or HTTPS ,use this for heartbeat.

Terminal send the message:

```
{
   "cmd":"checklive",
   "sn":"ZX12345678",
   "time":"2024-04-03 10:14:10"
}
```

Server response message:

Success:

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

B. 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",
   "sn":"ZX12345678",
   "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 ,20-27 is static face,40-41 is palmer vein,50 is photo
      },
      {
         "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,
   "sn":"ZX12345678",
   "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
        }
        ......
    ]
}
```


.....

When users is empty :the machine return:


```
{
    "ret":"getuserlist",
    "sn":"ZX12345678",
    "result":true,
    "count":0,
    "from":0,
    "to":0,
    "record":[]
}
```


Fail:


```
{
    "ret":"getuserlist",
    "sn":"ZX12345678",
    "result":false,
    "reason":1
}
```


2. Get user information

Fingerprint:

Server send the message:


```
{
    "cmd":"getuserinfo",
    "enrollid":1,
    "backupnum":0
}
```

Terminal response message:

```
{
   "ret":"getuserinfo",
   "result":true,
   "sn":"ZX12345678",
   "enrollid":1,
   "name":"chingzou",
   "backupnum":0,
   "admin":0,
   "record":"aabbccddeeffggddssiifdjdkjfkjdsjlkjal"
}
```

Fail:

```
{
   "ret":"getuserinfo",
   "sn":"ZX12345678",
   "result":false,
   "reason":1
}
```

Photo:

Server send the message:

```
{
   "cmd":"getuserinfo",
   "enrollid":1,
   "backupnum":50
}
```

Terminal response message:

Success:

```
{
   "ret":"getuserinfo",
   "result":true,
   "sn":"ZX12345678",
   "enrollid":1,
   "name":"chingzou",
   "backupnum":50,
   "admin":0,
   "record":"aabbccddeeffggddssiifdjdkjfkjdsjlkjal"    //Base64
}
```

Fail:

```
{
   "ret":"getuserinfo",
   "sn":"ZX12345678",
   "result":false,
   "reason":1
}
```

Rfid card:

Server send the message:


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


Terminal response message:

Success:


```
{
   "ret":"getuserinfo",
   "result":true,
   "sn":"ZX12345678",
   "enrollid":1,
   "name":"chingzou",
   "backupnum":11,
   "admin":0,
   "record":23532253
}
```


Fail:


```
{
   "ret":"getuserinfo",
   "sn":"ZX12345678",
   "result":false,
   "reason":1
}
```


Password:

Server send the message:


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


Terminal response message:

Success:


```
{
   "ret":"getuserinfo",
   "result":true,
   "sn":"ZX12345678",
   "enrollid":1,
   "name":"chingzou",
   "backupnum":10,
   "admin":0,
   "record":23532253
```

```
}
Fail:
{
   "ret":"getuserinfo",
   "sn":"ZX12345678",
   "result":false,
   "reason":1
}
```

3. Send user information

```
Fingerprint:
Server send message:
{
   "cmd":"setuserinfo",
   "enrollid":1,
   "name":"chingzou",
   "backupnum":0,
   "admin":0,
   "record":"aabbccddeeffggddssiifdjdkjfkjdsjlkjalflsgsadg"    //Fingerprint feature data
}
Photo:
Server send message:
{
   "cmd":"setuserinfo",
   "enrollid":1,
   "name":"chingzou",
   "backupnum":50,
   "admin":0,
   "record":"aabbccddeeffggddssiifdjdkjfkjdsjlkjalflsgsadg"    //Base64
}
Password:
Server send the message:
{
   "cmd":"setuserinfo",
   "enrollid":1,
   "name":"chingzou",
   "backupnum":10,
   "admin":0,
   "record":12345678
}
Rfid card:
Server send the message:
{
   "cmd":"setuserinfo",
```

```
    "enrollid":1,
    "name":"chingzou",
    "backupnum":11,
    "admin":0,
    "record":2352253
}
```

Terminal response message:

Success:

```
{
    "ret":"setuserinfo",
    "enrollid":1,
    "sn":"ZX12345678",
    "result":true
}
```

Fail:

```
{
    "ret":"setuserinfo",
    "sn":"ZX12345678",
    "result":false,
    "reason":1
}
```

4. Delete user information

Server send message:

```
{
    "cmd":"deleteuser",
    "enrollid":1,
    "backupnum":0    //0~9 fp; 10: password 11:card // 12 for all fp // 13 for all (0~9 fp card pwd)
}
```

Terminal response message:

Success:

```
{
    "ret":"deleteuser",
    "sn":"ZX12345678",
    "result":true
}
```

Fail:

```
{
    "ret":"deleteuser",
    "sn":"ZX12345678",
    "result":false,
    "reason":1
}
```

5. Get user name

Server send message:

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

Terminal response message:

Success:

```
{
   "ret":"getusername",
   "sn":"ZX12345678",
   "result":true,
   "record":"chingzou"        //utf8 or ascii
}
```

Fail:

```
{
   "ret":"getusername",
   "sn":"ZX12345678",
   "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",
```

"sn":"ZX12345678",
"result":true
}

Fail:

{
"ret":"setusername",
"sn":"ZX12345678",
"result":false,
"reason":1
}

7. Enable user

Server send message:

{
"cmd":"enableuser",
"enrollid":1,
"enflag":1
}

Terminal response message:
Success:

{
"ret":"enableuser",
"sn":"ZX12345678",
"result":true
}

Fail:

{
"ret":"enableuser",
"sn":"ZX12345678",
"result":false,
"reason":1
}

8. Disable user

Server send message:

{
"cmd":"enableuser",
"enrolled":1,
"enflag":0
}

Terminal response message:
Success：

{
"ret":"enableuser",

"sn":"ZX12345678",
"result":true
}
Fail:
{
"ret":"enableuser",
"result":false,
"sn":"ZX12345678",
"reason":1
}

9. Clean all users

Server send message:

{
"cmd":"cleanuser"
}

Terminal response message:

Success:

{
"ret":"cleanuser",
"sn":"ZX12345678",
"result":true
}

Fail:

{
"ret":"cleanuser",
"sn":"ZX12345678",
"result":false,
"reason":1
}

10. Get new logs

Server send message:

{
"cmd":"getnewlog",
"stn":true
}

Terminal response message:

Success:

{
"ret":"getnewlog",
"sn":"ZX12345678",
"result":true,
"count":1000,

```
  "from":0,
  "to":49,
  "record":[
    {
      "enrollid":1,
      "time":"2016-03-25 13:49:30",
      "mode":0,       //1:fp 2:pwd 3:card
      "inout":0,      //0:in 1:out
      "event":0
    },
    {
      "enrollid":2,
      "time":"2016-03-25 13:49:30",
      "mode":0,       //1:fp 2:pwd 3:card
      "inout":0,      //0:in 1:out
      "event":1
    }
    ......
  ]
}
```

Server response message:

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

Terminal send the second package:

```
{
  "ret":"getnewlog",
  "sn":"ZX12345678",
  "result":true,
  "count":1000,
  "from":50,
  "to":99,
  "record":[
    {
      "enrollid":111,
      "time":"2016-03-25 13:49:30",
      "mode":0,       //1:fp 2:pwd 3:card
      "inout":0,      //0:in 1:out
      "event":0
    },
    {
      "enrollid":112,
      "time":"2016-03-25 13:49:30",
```

```
            "mode":0,       //1:fp 2:pwd 3:card
            "inout":0,      //0:in 1:out
            "event":1
        }
        ......
    ]
}
```

When newlog is empty: the machine return:

```
{
    "ret":"getnewlog",
    "sn":"ZX12345678",
    "result":true,
    "count":0,
    "from":0,
    "to":0,
    "record":[]
}
```

Fail:

```
{
    "ret":"getnewlog",
    "sn":"ZX12345678",
    "result":false,
    "reason":1
}
```

11. Get all logs

Server send message:

```
{
    "cmd":"getalllog",
    "stn":true,
    "from":"2018-11-1",    //option from date
    "to":"2018-12-30"       //option to date
}
```

Terminal response message:

Success:

```
{
    "ret":"getalllog",
    "sn":"ZX12345678",
    "result":true,
    "count":1000,
    "from":0,
    "to":49,
    "record":[
        {
```

```
            "enrollid":1,
            "time":"2016-03-25 13:49:30",
            "mode":0,       //1:fp 2:pwd 3:card
            "inout":0,      //0:in 1:out
            "event":0
        },
        {
            "enrollid":2,
            "time":"2016-03-25 13:49:30",
            "mode":0,     //1:fp 2:pwd 3:card
            "inout":0,    //0:in 1:out
            "event":1
        }
        ......
    ]
}
```

Server response message:

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

Terminal send the second package:

```
{
    "ret":"getalllog",
    "sn":"ZX12345678",
    "result":true,
    "count":1000,
    "from":50,
    "to":99,
    "record":[
        {
            "enrollid":111,
            "time":"2016-03-25 13:49:30",
            "mode":0,       //1:fp 2:pwd 3:card
            "inout":0,      //0:in 1:out
            "event":0
        },
        {
            "enrollid":112,
            "time":"2016-03-25 13:49:30",
            "mode":0,     //1:fp 2:pwd 3:card
            "inout":0,    //0:in 1:out
            "event":1
        }
```

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

When newlog is empty: the machine return:

```
{
    "ret":"getalllog",
    "sn":"ZX12345678",
    "result":true,
    "count":0,
    "from":0,
    "to":0,
    "record":[]
}
```

Fail:

```
{
    "ret":"getalllog",
    "sn":"ZX12345678",
    "result":false,
    "reason":1
}
```

12. Clean all logs

Server send message:

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

Terminal response message:

Success:

```
{
    "ret":"cleanlog",
    "sn":"ZX12345678",
    "result":true
}
```

Fail:

```
{
    "ret":"cleanlog",
    "sn":"ZX12345678",
    "result":false,
    "reason":1
}
```

13. 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",
   "sn":"ZX12345678",
   "result":true
}
```

Fail:

```
{
   "ret":"initsys",
   "sn":"ZX12345678",
   "result":false,
   "reason":1
}
```

14. Reboot

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

Server send message:

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

15. Clean all administrators

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

Server send message:

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

Terminal response message:

Success:

```
{
   "ret":"cleanadmin",
   "sn":"ZX12345678",
   "result":true
}
```

Fail:

```
{
   "ret":"cleanadmin",
   "sn":"ZX12345678",
   "result":false,
   "reason":1
```

```
}
```

16. Set time

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",
   "sn":"ZX12345678",
   "result":true
}
```

Fail:

```
{
   "ret":"settime",
   "sn":"ZX12345678",
   "result":false,
   "reason":1
}
```

17. 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",
   "sn":"ZX12345678",
```

```
   "result":true
}
Fail:
{
   "ret":"setdevinfo",
   "sn":"ZX12345678",
   "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.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
};
```

18. Get terminal parameter

Note:1.Getting the terminal common parameter.

Server send message:

```
{
   "cmd":"getdevinfo"
}
```

Terminal response message:

Success:

```
{
   "ret":"getdevinfo",
   "sn":"ZX12345678",
   "result":true,
   "deviceid":1,
   "language":0,
   "volume":0,
   "screensaver":0,
   "verifymode":0,
   "sleep":0,
   "userfpnum":3,
   "loghint":1000,
   "reverifytime":0
}
```

Fail:

```
{
   "ret":"getdevinfo",
   "sn":"ZX12345678",
   "result":false,
   "reason":1
}
```

19. Open door

Note: Open the door.

Server send message:

```
{
   "cmd":"opendoor",
   "doornum":1        //this just for access controller(1~4) because access controller have 4 doors. If delete the item, will open all doors. Normal access or attendance machine no need this item.
}
```

Terminal response message:

Success:

```
{
   "ret":"opendoor",
   "sn":"ZX12345678",
   "result":true
}
```

Fail:

```
{
   "ret":"opendoor",
   "sn":"ZX12345678",
   "result":false,
   "reason":1
}
```

20. 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;
   "access_times":0,    //Permitted Access Count.Value 0: Unlimited access
   "punchtimes":0,     //Puch times in timezon.hen set to 0, no limit applies
   "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 week 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",
   "sn":"ZX12345678",
   "result":true
}
Fail:
{
   "ret":"setdevlock",
   "sn":"ZX12345678",
   "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.

21. Get the access parameter

Server send message:

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

Access response message:

Success:

```
{
   "ret":"getdevlock",
   "sn":"ZX12345678",
   "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",
```

```
   "sn":"ZX12345678",
   "result":false,
   "reason":1
}
```

22. Get the user access parameter

Server send message:

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

Terminal response message:

Success:

```
{
   "ret":"getuserlock",
   "sn":"ZX12345678",
   "result":true,
   "enrollid":1,
   "weekzone":1, //the weekzone as set above access controller door1
   "weekzone2":1, //just for access controller door2
   "weekzone3":1, //just for access controller door3
   "weekzone4":1, //just for access controller door4
   "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",
   "sn":"ZX12345678",
   "result":false,
   "reason":1
}
```

23. Set the users access parameter

Server send message:

```
{
   "cmd":"setuserlock",
   "count":40,
   "record":[
      {
         "enrollid":1,
         "weekzone":1,    // just for access controller door1
         "weekzone2":1, // just for access controller door2
```

```
        "weekzone3":1, // just for access controller door3
        "weekzone4":1, // just for access controller door4
        "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",
  "sn":"ZX12345678",
  "result":true
}
```

Fail:

```
{
  "ret":"setuserlock",
  "sn":"ZX12345678",
  "result":false,
  "reason":1
}
```

24. Delete the user access parameter

Server send message:

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

Terminal response message:

Success:

```
{
  "ret":"deleteuserlock",
  "sn":"ZX12345678",
  "result":true
}
```

Fail:

```
{
   "ret":"deleteuserlock",
   "sn":"ZX12345678",
   "result":false,
   "reason":1
}
```

25. Clean all user access parameter

Server send message:

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

Terminal response message:

Success:

```
{
   "ret":"cleanuserlock",
   "sn":"ZX12345678",
   "result":true
}
```

Fail:

```
{
   "ret":"cleanuserlock",
   "sn":"ZX12345678",
   "result":false,
   "reason":1
}
```

26. Get time

Note: get the terminal date and time

Server send message:

```
{
   "cmd":"gettime"
}
```

Terminal response message:

Success:

```
{
   "ret":"gettime",
   "sn":"ZX12345678",
   "time":"2022-11-09 19:31:49"
}
```

27. QR code sending

Terminal send the message:

```
{
   "cmd":"sendqrcode",
   "sn":"ZX12345678",
   "record":"123456"
}
```

28. QR code server reply

Note: After receiving the QR code verification request sent by the terminal, the server returns the verification result and related operation instructions.

Server response message:

```
{
   "ret":"sendqrcode",
   "sn":"ZX12345678",
   "result":true,
   "access":1,                  //if need :access 1 can open the door, 0 can not open
   "enrollid":10,
   "username":"tom",
   "message":"ok",
   "voice":"ok",
   "fontsize":32,               // Only the 'Message Bar' mode is supported, The value of the font size
                                   can be changed (10 12 14 16 18 20 22 24 28 32 40)
   "text_color":[255,0,0]    // Only the 'Message Bar' mode is supported,RGB color model
}
```

29. Get questionnaire parameter

Server send message:

```
{
   "cmd":"getquestionnaire",
   "stn":true
}
```

Terminal response message:

Success:

```
{
   "ret":"getquestionnaire",
   "sn":"ZX12345678",
   "result":true,
   "title":"inout event",
   "voice":"please select",
   "errmsg":"please select",
   "radio":true,
   "optionflag":0,
   "usequestion":false,
   "useschedule":false,
   "card":0,
```

```
   "items":["in","out","onduty","offduty"],
   "schedules":["00:01-11:12*1","11:30-12:30*3","13:00-19:00*4","00:00-00:00*0","00:00-00:00
*0","00:00-00:00*0","00:00-00:00*0","00:00-00:00*0"]
}
```

30. Set questionnaire parameter

Server send message:

```
{
   "cmd":"setquestionnaire",
   "title":"inout event",    //display at top
   "voice":"please select", //if you want to say ,just english or chinese, if don't want to say ,delete
                              this item.
   "errmsg":"please select", //It is useful when multiple selection and mandatory selection are
                               selected, and it is displayed when no mandatory selection is
                               selected
   "radio":true,              //multiple choice or single choice
   "optionflag":0,         //In case of multiple selection, it is used to indicate which item is required
   "usequestion":true,    //enable
   "useschedule":true,    //enable
   "card":0,                  //Swipe card to start questionnaire
   "items":["in","out","onduty","offduty"], //Multiple choice up to 8, single choice can be 16
   "schedules":["00:01-11:12*1","11:30-12:30*3","13:00-19:00*4","00:00-00:00*0","00:00-00:00
   *0","00:00-00:00*0","00:00-00:00*0","00:00-00:00*0"]      //Event schedule .max 8 iter
}
```

Terminal response message:

success:

```
{
   "ret":"setquestionnaire",
   "sn":"ZX12345678",
   "result":true
}
```

Fail:

```
{
   "ret":"setquestionnaire",
   "sn":"ZX12345678",
   "result":false,
   "reason":1
}
```

31. Get holiday parameter

Server send message:

```
{
   "cmd":"getholiday",
   "stn":true    // First packet identifier (true: Start obtaining, false: Continue obtaining)
```

```
}
```

Terminal response message:

Success:

```
{
   "ret":"getholiday",
   "sn":"ZX12345678",
   "result":true,
   "holidays":[
      {
         "name":"holiday1",
         "startday":"01-01",
         "endday":"01-01",
         "shift":0,
         "dayzone":0
      },
      {
         "name":"holiday2",
         "startday":"02-01",
         "endday":"02-07",
         "shift":0,
         "dayzone":0
      },
      {
         "name":"holiday3",
         "startday":"05-01",
         "endday":"05-03",
         "shift":0,
         "dayzone":0
      }
   ]
}
```

32. Set holiday parameter

Server send message:

```
{
   "cmd":"setholiday",
   "holidays":[
      {
         "name":"holiday1",    //holiday name
         "startday":"01-01",   //holidaystartday
         "endday":"01-01",     //holidayendday
         "shift":0,            //Attendance Shift
         "dayzone":0           //day zone
      },
```

```
        {
            "name":"holiday2",
            "startday":"02-01",
            "endday":"02-07",
            "shift":0,
            "dayzone":0
        },
        {
            "name":"holiday3",
            "startday":"05-01",
            "endday":"05-03",
            "shift":0,
            "dayzone":0
        }
        ... Maximum 30 holidays
    ]
}
```

Terminal response message:

success:

```
{
    "ret":"setholiday",
    "sn":"ZX12345678",
    "result":true
}
```

Fail:

```
{
    "ret":"setholiday",
    "sn":"ZX12345678",
    "result":false,
    "reason":1
}
```

33. Set user information

Server send message:

```
{
    "cmd":"setuserprofile",
    "enrollid":1,
    "profile":"message" (Maximum 200 bytes)
}
// "enrollid":0 means public information
// "enrollid":1,2,3... means personal information
```

Terminal response message:

Success:

```
{
```

```
   "ret":"setuserprofile",
   "sn":"ZX12345678",
   "enrollid":1,
   "result":true
}
```

34. Get user information

Server send message:

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

Terminal response message:

Success:

```
{
   "ret":"getuserprofile",
   "sn":"ZX12345678",
   "enrollid":1,
   "record":"message",
   "result":true
}
```

35. Remote add user

Server send message:

```
{
   "cmd":"adduser",      // Command Word (Remote Add User)
   "enrollid":1001,      // User registration ID (unique identifier)
   "backupnum":50, // 0~9 fingerprint 10:password 11:rfid card ,20-27 is static face,40~41 is palm
                      vein,50 is photo
   "admin":0,            // Permissions (0: regular user, 1: administrator)
   "name":"TEST",        // Username (UTF-8)
   "flag":10             // "flag": 10    Automatic registration
}
```

Automatic registration (adding users with photos):

```
{
   "cmd":"adduser",
   "enrollid":1001,
   "backupnum":50,          // The initial information is a photo
   "name":"TEST",
   "flag":10                // Automatic registration
}
```

Terminal response message:

Success:

```
{
```

```
   "ret":"adduser",              // Response Command Word
   "sn":"ZX12345678",
   "enrollid":1001,              // User registration ID
   "result":true                 // Operation result
}
Fail:
{
   "ret":"adduser",
   "sn":"ZX12345678",
   "enrollid":1001,
   "result":false,
   "reason":2                    // Error code (2: ID already exists)
}
```

36. Get device information

```
Server send message:
{
   "cmd":"getdevcap"
}
Terminal response message:
Success:
{
   "ret":"getdevcap",
   "sn":"ZX12345678",
   "result":true,
   "usersize":15000,             //user capacity
   "facesize":5000,              //face capacity
   "fpsize":10000,               //fingerprint capacity
   "palmsize":0,                 //Palm Vein capacity
   "cardsize":15000,             //rfid card capacity
   "pwdsize":15000,              //password capacity
   "logsize":500000,             //logs capacity
   "useduser":746,               //Registered User Count
   "usedface":12,                //Registered face Count
   "usedfp":17,                  //Registered Fingerprint Count
   "usedpalm":0,                 //Registered Palm Vein Count
   "usedcard":703,               //Registered rfid card Count
   "usedpwd":9,                  //Registered password Count
   "usedlog":135,                //Stored Access Log Count
   "usednewlog":135,             //New Access Records
   "usedrtlog":0                 //Real-time Access Records
}
```

37. Send user information (Requires AI face recognition device firmware version 5.09 or v2.09 or above)

Note: The server sends complete user information to the terminal and supports multi-factor authentication configuration, that is, permission management.

Server send the message:

```
{
   "cmd":"setuserinfo",
   "enrollid":1,
   "name":"test",          //name
   "verifymode":0,         //Verification method (enumeration values are described as follows)
   "department":"HR",    //Department Name
   "admin":0,              // 0: normal user ; 1:adminstrator 2:super user
   "card":12345,           //card number
   "pwd":123,              //password
   "enable":1,             // 0 User disabled, 1 User enable
   "shiftid":1,            // self-service attendance device shifts
   "zoneid":1,             //Weekly Access Control Periods
   "groupid":1,            //Access Group
   "access_times":10,    //Permitted Access Count,    Value 0: Unlimited access
   "postid":1,             //Department ID      When there is a 'department' item, it can be ignore
   "starttime":"2025-03-01 00:00:00",
   "endtime":"2055-05-27 23:59:59",
   "backupnum":0,          //0~9 fingerprint,20-27is staticface,40-41is palm vein
   "record":"kajgksjgaglas" //the string length less then 1620 for THbio3.0 and less 1024 for
                            "kajgksjgaglas"
}
```

Terminal response message:

Success:

```
{
   "ret":"setuserinfo",
   "enrollid":1,
   "sn":"ZX12345678",
   "result":true
}
```

Description of verifymode enumeration values:

0: Device Verify Mode

1: Fingerprint verification

2: Password verification

3: Card verification

4: Fingerprint + Password verification

5: Fingerprint + Card verification

8: Face recognition verification
9: Face + Password verification
10: Face + Card verification
11: Card + Password verification
12: Fingerprint + Face verification
13: QR code verification
14: Face + (Card OR Password) verification
15: SFZ (Chinese ID card) verification
16: Palm verification
19: 1:N verification
20: any
21: Palm + Face verification
22: Fingerprint + Palm verification
23: Card + Palm verification
24: Palm + Password verification

38. Set door status

Server send the message:

```
{
   "cmd":"lockctrl",
   "fuc":1 // 1. Forced normally open 2. Forced normally closed 3. Software open the door 4. Relay returns to the initial state 5. Machine restarts 6. Cancel the alarm
}
```

Terminal response message:

Success:

```
{
   "ret":"lockctrl",
   "sn":"ZX12345678",
   "result":true
}
```