Whatsminer API
V2.0.0

MicroBT Electronics Technology Co.,Ltd

Content

Whatsminer API....1
1. Summary....3
2. Writable API....4
2.1 Update pools information....4
2.2 Restart btminer....4
2.3 Power off miner....4
2.4 Power on miner....4
2.5 Manage led....5
2.6 Switch power mode....5
2.7 Firmware upgrading....5
2.8 Reboot system....6
2.9 Restore to factory setting....6
2.10 Modify the password of admin account....6
2.11 Modify network configuration....7
2.12 Download logs....7
2.14 Enable btminer fast boot....8
2.15 Disable btminer fast boot....8
2.16 Enable web pools....8
2.17 Disable web pools....8
2.18 Set hostname....8
2.19 Set zone....9
2.20 Load log....9
2.21 Set power pct....9
2.22 Pre power on....9
3.Readable API....9
3.1 API summary....10
3.2 API pools....10
3.3 API edevs/devs....11
3.5 API get psu....11
3.6 API get version....11
3.7 Get token....12
3.8 Status....12
3.9 Get miner info....12
4. Others....14
4.1 API ciphertext....15

1. Summary

This article describes how to use the mining machine API. The intended audience is mine management software developers.

Using WhatMinerTools gain privilege to the miner. The function of remote batch management can be realized through API.
Follow these steps:
1.Change the default password(admin)
2.Turn on the API

2. Writable API

2.1 Update pools information

Json:

```
{
        "token":"str",
        "cmd":"update_pools",
        "pool1":"str",
        "worker1":"str",
        "passwd1":"str",
        "pool2":"str",
        "worker2":"str",
        "passwd2":"str",
        "pool3":"str",
        "worker3":"str",
        "passwd3":"str"
}
```

2.2 Restart btminer

Json:

```
{
        "token":"str",
        "cmd":"restart_btminer"
}
```

2.3 Power off miner

This operation simply stops mining and turns off the power output of the power board. There was no power outage on the control board

Json:

```
{
        "token":"str",
        "cmd":"power_off",
        "respbefore":"str"                    //"false"/"true"
}
```

2.4 Power on miner

This operation simply starts mining and turns on the power output of the power board.

Json:

```
{
        "token":"str",
        "cmd":"power_on",
}
```

2.5 Manage led

Recovery to automatic control:

Json:

```
{
        "token":"str",
        "cmd":"set_led",
        "param":"auto"
}

{
        "token":"str",
        "cmd":"set_led",
        "color":"str",          //red green
        "period":inter,         //flash cycle ms
        "duration":inter,       //led on time in cycle(ms)
        "start":inter           //led on time offset in cycle(ms)
}
```

2.6 Switch power mode

Json:

```
{
        "token":"str",
        "cmd":"set_low_power"
}
```

2.7 Firmware upgrading

Upgrade flow:

Client -> whatsminer(text flow): "update_firmware"

Json:

```
{
"token":"str",
"cmd":"update_firmware"
}
```

Whatsminer -> client(text flow): "ready"
Json:
{
"STATUS":"S",
"When":1594179080,
"Code":131,"Msg":"ready",
"Description":"whatsminer v1.3"
}

Client -> whatsminer(binary flow): file_size(4Byte) file_data

file_size: size of upgrade file,send integer to stream as little endian.
file_data:file binary flow

Check upgrading by the value of "Firmware Version" returned by summary.

All interactions are done in once TCP connection.

2.8 Reboot system

Json:
{
"token":"str",
"cmd":"reboot"
}

2.9 Restore to factory setting

Json:
{
"token":"str",
"cmd":"factory_reset"
}

2.10 Modify the password of admin account

The maximum password length is 8byte

Notice: you must regain token form whatsminer for encrypted transmission
Json:
{
"token":"str",
"cmd":"update_pwd",
"old":"str", //use letter,mumber,underline

```
        "new":"str"       //use letter,mumber,underline
}
```

2.11 Modify network configuration

Notice:after modify configuration whatsminer machine will be reboot.

Json:

```
{
        "token":"str",
        "cmd":"net_config",
        "param":"dhcp"
}
```

Json:

```
{
        "token":"str",
        "cmd":"net_config",
        "ip":"str",
        "mask":"str",
        "gate":"str",
        "dns":"str",       //"114.114.114.114 192.168.0.1" Divide by a space
        "host":"str"
}
```

2.12 Download logs

Download flow:

Client -> whatsminer(text flow):

Json:

```
{
        "token":"str",
        "cmd":"download_logs"
}
```

Whatsminer -> client(text flow):

Json:

```
{
        "STATUS":"S",
        "When":1603280777,
        "Code":131,
        "Msg":{"logfilelen":"str"},
        "Description":"whatsminer v1.3"
}
```

Whatsminer -> client(binariy flow):

The whatsminer sends the file contents after 10ms delay.

2.13 Set target freq

```
{
        "cmd": "set_target_freq",
        "percent": "str",            //range: -10 ~ 100
        "token": "str"
}
```

2.14 Enable btminer fast boot

```
{
        "cmd": "enable_btminer_fast_boot",
        "token": "str"
}
```

2.15 Disable btminer fast boot

```
{
        "cmd": "disable_btminer_fast_boot",
        "token": "str"
}
```

2.16 Enable web pools

```
{
        "cmd": "enable_web_pools",
        "token": "str"
}
```

2.17 Disable web pools

```
{
        "cmd": "disable_web_pools",
        "token": "str"
}
```

2.18 Set hostname

```
{
        "cmd": "set_hostname",
        "hostname": "str",
        "token": "str"
```

}

2.19 Set zone

```
{
        "cmd": "set_zone",
        "timezone": "CST-8",
        "zonename": "Asia/Shanghai",
        "token": "str"
}
```

2.20 Load log

```
{
        "cmd": "load_log",
        "ip": "str",
        "port": "str",
        "proto": "str",              //tcp/udp
        "token": "str"
}
```

2.21 Set power pct

```
{
        "cmd": "set_power_pct",
        "percent": "str",            //range: 0 ~ 100
        "token": "str"
}
```

2.22 Pre power on

```
{
        "cmd": "pre_power_on",
        "complete": "str",                  //true/false
        "msg": "str",     //"wait for adjust temp"/"adjust complete"/"adjust continue"
        "token": "str"
}
```

- The miner can be preheated by "pre_power_on" before "power on", so that the machine can quickly enter the full power state when "power on" is used.You can also use this command to query the pre power on status.Make sure power_off btminer before pre_power_on.

3.Readable API

3.1 API summary

Contains error code, fan speed, power info, etc.

Json:

```
{
        "cmd":"summary"
}
```

```
[MHS av] => 67761775.18          Average hash rate of miner(MHS)
[Fan Speed In] => 6060           Air outlet fan speed(RPM)
[Fan Speed Out] => 6000          Air inlet Fan speed(RPM)
[Power] => 3470                  Input power(W)
[Uptime] => 91051                System up time(second)
[Power Fanspeed] => 9060         Power fan speed
[Error Code Count] => 1          Error code number
[Error Code 0] => 250            Error code value
[Power Mode] => Low              Power mode (Low/Normal/High)
[Firmware Version] => '20200207.23.1'        Firmware version
[MAC] => C4:05:08:00:02:3B                   Network MAC address
[Factory GHS] => 44397                       Factory hash rate(GHS)
```

3.2 API pools

Contains pool miner information.

Json:

```
{
        "cmd":"pools"
}
```

```
[POOL] => 0
[URL] => stratum+tcp://btc.ss.poolin.com:443    Pool address and port
[Status] => Alive                               Pool status
[Priority] => 0                                 Pool priority(0 highest)
[Accepted] => 4615                              Accepted nonces by the pool
[Rejected] => 3                                 Rejected nonces by the pool
[User] => microbtinit                           Miner name
[Last Share Time] => 1587278148                 Last nonce submission time
[Stratum Active] => true                        Pool stratum status
[Stratum URL] => btc-vip-3dcoa7jxu.ss.poolin.com        Pool address
[Stratum Difficulty] => 262144.00000000         Pool difficulty
[Pool Rejected%] => 0.0667                      Pool rejection percent
```

[Current Block Height] => 626663 Current Block Height
[Current Block Version] => 536870912 Current Block Version

3.3 API edevs/devs

Contains information for each hash board.

Json:

```
{
        "cmd":"edevs"
}

{
        "cmd":"devs"
}
```

[Slot] => 0 Hash board slot number
[Temperature] => 71.50 Board temperature at air outlet (℃)
[Chip Frequency] => 608 Average frequency of chips in hash board (MHz)
[MHS av] => 30991182.21 Average hash rate of hash board (MHS)
[PCB SN] => Z5M1ES94100115K90371 PCB serial number

3.4 API devdetails

Json:

```
{
        "cmd":"devdetails"
}
```

[DEVDETAILS0] =>
[Model] => M20SV10 Miner type

3.5 API get psu

Contains power information.

Json:

```
{
        "cmd":"get_psu"
}
```

3.6 API get version

Get whatsminer api version

Json:

```
{
        "cmd":"get_version"
}
```

3.7 Get token

You must use plaintext,and whatsminer will return plaintext.

Json:

```
{
"cmd":"get_token"
}
```

Return:

```
{
"STATUS":"string",
"When":12345678,                    //inter
"Code":133,
"Msg":{"time":"str","salt":"str","newsalt":"str"},
"Description":"string",
}
```

3.8 Status

Get btminer status and firmware version

Json:

```
{
        "cmd":"status"
}
```

Return:

```
{
        "btmineroff":"str",                 //"true"/"false"
        "Firmware Version":"str"
}
```

Notice：whatsminer support 16 IP clients, one IP can get 32 tokens, token keepalive is 30min.

3.9 Get miner info

Json:

```
{
```

```
        "cmd":"get_miner_info",
        "info":"ip,proto,netmask,gateway,dns,hostname,mac,ledstat"
}
```

You can select the fields in "info" that you want to return.

Return:

```
{
"STATUS":"S",
"When":1618212903,
"Code":131,
"Msg":{"netmask":"255.255.255.0","ledstat":"auto","gateway":"192.168.2.1"},
"Description":"whatsminer v1.4.0"
}
```

API:

client -> whatsminer： "get_token"

whatsminer -> client： $time $salt $newsalt

e.g.： "1592555626 BQ5hoXV9 jbzkfQls"

JSON:

client -> whatsminer： {"cmd":"get_token"}

whatsminer -> client： {"time":"str","salt":"str","newsalt":"str"}

e.g.： {"time":"5626","salt":"BQ5hoXV9","newsalt":"jbzkfQls"}

$time $salt $newsalt are separated by space

time: timestamp(sec)（Epoch Time since 1970-01-01 00:00:00 UTC）

salt:

new_slat: new salt for sign

Token calculation method：

get token form whatsminer machine: time salt newsalt.

1.calculate key use admin’s password and salt.

2.timesec is the last four characters of time .

key = md5(salt + admin_password)

sign = md5(newsalt + key + timesec)

The reference code in Ubuntu：

Frist,Get those values from whatsminer: $time $salt $newsalt.

Ubuntu Shell command:
key = `openssl passwd -1 -salt $salt "${admin_password}"|cut -f 4 -d '$'`
sign=`openssl passwd -1 -salt $newsalt "${key}${time:0-4}"|cut -f 4 -d '$'`

Default user password：admin:admin

The API command can be used for two joins.
Eg.

```
{
        "cmd":"summary+pools"
}
```

4.Others

The whatsminer API TCP port is 4028.
Notice: If no data is received within 10 seconds after the port is opened, it will time out and close.
Return MSG :
STATUS="",When="",Code="",Msg="",Description=""
STATUS: E (error) S (success)
When: timestamp
Code: type of message
Description: details of the message

JSON API RETURN format:

```
{
        "STATUS":"string",
        "When":12345678,                //inter
        "Code":133,
        "Msg":"string",          //string or object
        "Description":"string",
}
```

Message Code:

14	invalid api command or data
23	invalid json message
45	permission denied
131	command OK
132	command error
134	get token message OK
135	check token error

136 token over max times
137 base64 decode error

4.1 API ciphertext

Notice: readable API supports two-way communication plaintext and ciphertext, Writeable API only supports ciphertext.

Encryption algorithm:
Ciphertext = aes256(plaintext)，ECB mode
Encode text = base64(ciphertext)

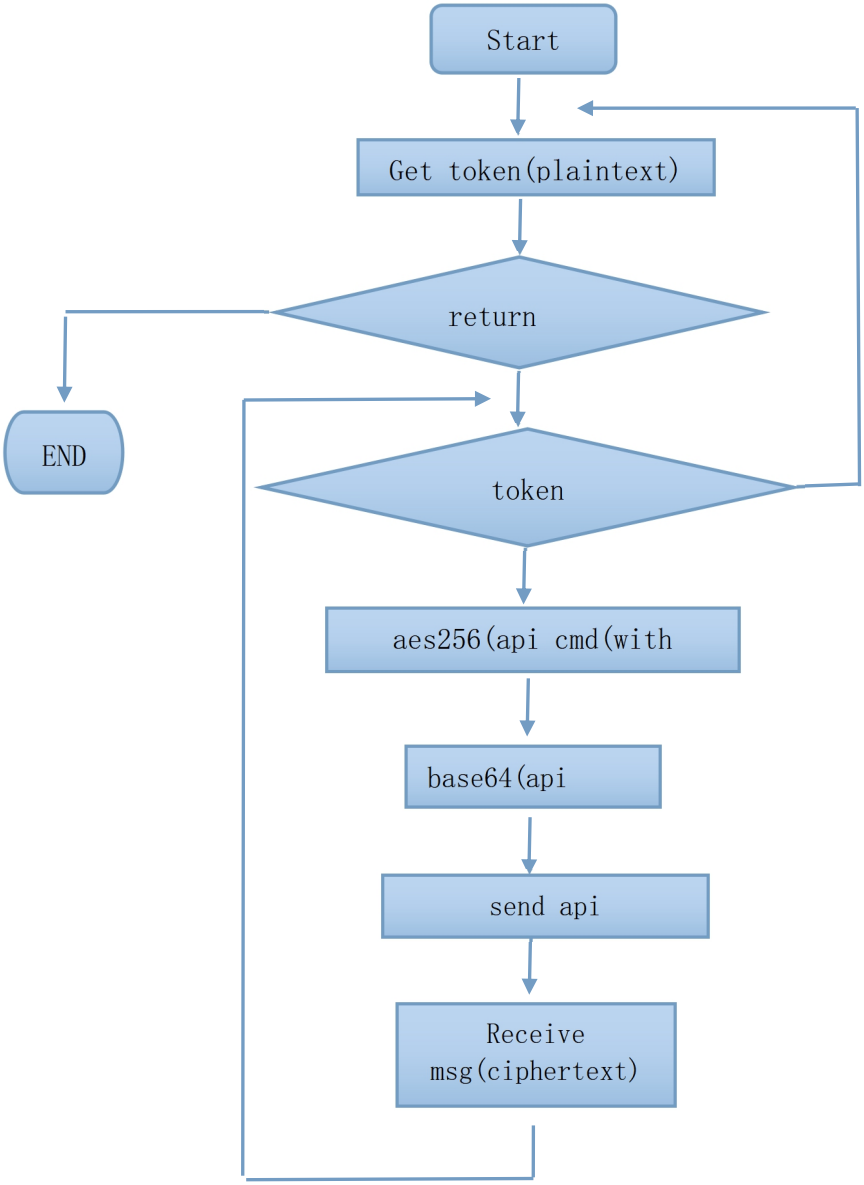
Steps as follows:
(1)api_cmd = token,$sign|api_str
(2)enc_str = aes256(api_cmd, $key)
(3)tran_str = base64(enc_str)
api_str is API command plaintext

Generate aeskey step：
(1)Get token from whatsminer: $time $salt $newsalt
(2)Generate key:
key = md5(salt + admin_password)
Reference code:
key = `openssl passwd -1 -salt $salt "${admin_password}"`
(3)Generate aeskey:
aeskey = sha256($key)

e.g.:
set_led|auto ->
token,$sign|set_led|auto ->
ase256("token,sign|set_led|auto", $aeskey) ->
base64(ase256("token,sign|set_led|auto", $aeskey)) ->
enc|base64(ase256("token,sign|set_led|auto", $aeskey))

Json:
```
{
    "enc":1,          //inter
    "data":"base64 str"
}
```

5.The flow


Start
Get token(plaintext)
return
END
token
aes256(api cmd(with
base64(api
send api
Receive
msg(ciphertext)