Whatsminer API
V2.0.3

MicroBT Electronics Technology Co.,Ltd

Content

1. Summary

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

Using WhatMinerTool gains 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"
}
```

Return:

```
{
        "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 -> miner(text flow): "update_firmware"

JSON:

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

}

Miner -> client(text flow): "ready"
JSON:
{
"STATUS":"S",
"When":1594179080,
"Code":131,"Msg":"ready",
"Description":""
}

Client -> miner(binary flow): file_size(4Bytes) 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 one-time TCP connections.

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 8 bytes.

Notice: you must regain token from miner for encrypted transmission.
JSON:
{
"token":"str",

```
        "cmd":"update_pwd",
        "old":"str",                    use letter,number,underline
        "new":"str"                     use letter,number,underline
}
```

2.11 Modify network configuration

Notice: after modifying the configuration, Miner will restart.

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 -> miner(text flow):

JSON:

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

Miner -> client(text flow):

JSON:

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

}
Miner -> client(binary flow):
The miner sends the file contents after 10ms delay.

2.13 Set target frequency

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

2.14 Enable btminer fast boot

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

2.15 Disable btminer fast boot

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

2.16 Enable web pools

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

2.17 Disable web pools

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

}

2.18 Set hostname

JSON:

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

2.19 Set zone

JSON:

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

2.20 Load log

JSON:

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

2.21 Set power percent

JSON:

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

2.22 Pre power on

JSON:

{
"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.
"wait for adjust temp": The temperature adjustment of the miner is not completed.
"adjust complete": The temperature adjustment of the miner is completed, and miner can be power on.
"adjust continue": Miner is adjusting the temperature while waiting to end.

The value of "complete" is true after the temperature adjustment is complete.

3.Readable API

3.1 Summary

Contains fan speed, power info, etc.

JSON:
{
"cmd":"summary"
}

Return:
{
"STATUS":[{"STATUS":"S","Msg":"Summary"}],
"SUMMARY":[
{
"Elapsed":2648,
"MHS av":84983730.62, *Average hash rate of miner(MHS)*
"MHS 5s":102423869.64,
"MHS 1m":86361423.06,
"MHS 5m":84941366.02,
"MHS 15m":84969424.09,
"HS RT":84941366.02,
"Accepted":804,
"Rejected":0,
"Total MH":225043191209.0000,
"Temperature":80.00,

```
                "freq_avg":646,
                "Fan Speed In":4530,                    Air outlet fan speed(RPM)
                "Fan Speed Out":4530,                   Air inlet Fan speed(RPM)
                "Power":3593,                           Input power(W)
                "Power Rate":42.31,
                "Pool Rejected%":0.0000,
                "Pool Stale%":0.0000,
                "Last getwork":0,
                "Uptime":20507,                         System up time(second)
                "Security Mode":0,
                "Hash Stable":true,
                "Hash Stable Cost Seconds":17569,
                "Hash Deviation%":0.1398,
                "Target Freq":574,
                "Target MHS":76157172,
                "Env Temp":32.00,
                "Power Mode":"Normal",              Power mode (Low/Normal/High)
                "Factory GHS":84773,                Factory hash rate(GHS)
                "Power Limit":3600,
                "Chip Temp Min":75.17,
                "Chip Temp Max":101.25,
                "Chip Temp Avg":89.60,
                "Debug":"-0.0_100.0_354",
                "Btminer Fast Boot":"disable"
        }
        ]
}
```

3.2 Pools

Contains pool miner information.

JSON:

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

Return:

```
{
        "STATUS":[{"STATUS":"S","Msg":"1 Pool(s)"}],
        "POOLS":[
                {
                "POOL":1,
                "URL":"stratum+tcp://btc.ss.poolin.com:443",    Pool address and port
```

```
                "Status":"Alive",                                   Pool status
                "Priority":0,                                       Pool priority(0 highest)
                "Quota":1,                                          Pool default strategy is 1
                "Long Poll":"N",
                "Getworks":1,
                "Accepted":0,                                       Accepted nonces by the pool
                "Rejected":0,                                       Rejected nonces by the pool
                "Works":0,
                "Discarded":0,
                "Stale":0,
                "Get Failures":0,
                "Remote Failures":0,
                "User":"microbtinitial",                            Miner name
                "Last Share Time":0,                                Last nonce submission time
                "Diff1 Shares":0,
                "Proxy Type":"",
                "Proxy":"",
                "Difficulty Accepted":0.00000000,
                "Difficulty Rejected":0.00000000,
                "Difficulty Stale":0.00000000,
                "Last Share Difficulty":0.00000000,
                "Work Difficulty":0.00000000,
                "Has Stratum":1,
                "Stratum Active":true,                              Pool stratum status
                "Stratum URL":"btc-vip-3dcoa7jxu.ss.poolin.com",    Pool address
                "Stratum Difficulty":65536.00000000,                Pool difficulty
                "Best Share":0,
                "Pool Rejected%":0.0000,                            Pool rejection percent
                "Pool Stale%":0.0000,
                "Bad Work":0,
                "Current Block Height":0,                           Current Block Height
                "Current Block Version":536870916                   Current Block Version
                }
        ]
}
```

3.3 Edevs/devs

Contains information for each hash board.

JSON:

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

Return:


```
{
        "STATUS":[{"STATUS":"S","Msg":"3 ASC(s)"}],
        "DEVS":[
        {
                "ASC":0,
                "Slot":0,                                   Hash board slot number
                "Enabled":"Y",
                "Status":"Alive",
                "Temperature":80.00,                Board temperature at air outlet (℃)
                "Chip Frequency":587,             Average frequency of chips in hash board (MHz)
                "MHS av":10342284.80,             Average hash rate of hash board (MHS)
                "MHS 5s":5298845.66,
                "MHS 1m":8508905.30,
                "MHS 5m":10351110.56,
                "MHS 15m":10296867.74,
                "HS RT":10351110.56,
                "Accepted":18,
                "Rejected":0,
                "Last Valid Work":1643183296,
                "Upfreq Complete":0,
                "Effective Chips":156,
                "PCB SN":"HEM1EP9C400929K60003",        PCB serial number
                "Chip Data":"K88Z347-2039     BINV01-195001D",
                "Chip Temp Min":80.56,
                "Chip Temp Max":97.00,
                "Chip Temp Avg":89.89,
                "chip_vol_diff":9
        },
        {
                "ASC":1,
                "Slot":1,
                "Enabled":"Y",
                "Status":"Alive",
                "Temperature":80.00,
                "Chip Frequency":590,
                "MHS av":10259948.84,
                "MHS 5s":5413853.90,
                "MHS 1m":8577249.68,
                "MHS 5m":10441143.92,
                "MHS 15m":10214893.36,
                "HS RT":10441143.92,
                "Accepted":16,
```

```
                "Rejected":0,
                "Last Valid Work":1643183291,
                "Upfreq Complete":0,
                "Effective Chips":156,
                "PCB SN":"HEM1EP9C400929K60001",
                "Chip Data":"K88Z347-2039      BINV01-195001D",
                "Chip Temp Min":77.94,
                "Chip Temp Max":96.50,
                "Chip Temp Avg":88.23,
                "chip_vol_diff":9
        },
        {
                "ASC":2,
                "Slot":2,
                "Enabled":"Y",
                "Status":"Alive",
                "Temperature":80.00,
                "Chip Frequency":590,
                "MHS av":10258829.89,
                "MHS 5s":5571781.71,
                "MHS 1m":8675316.17,
                "MHS 5m":10479953.41,
                "MHS 15m":10213779.32,
                "HS RT":10479953.41,
                "Accepted":19,
                "Rejected":0,
                "Last Valid Work":1643183296,
                "Upfreq Complete":0,
                "Effective Chips":156,
                "PCB SN":"HEM1EP9C400929K60002",
                "Chip Data":"K88Z347-2039      BINV01-195001D",
                "Chip Temp Min":80.50,
                "Chip Temp Max":97.44,
                "Chip Temp Avg":90.91,
                "chip_vol_diff":9
        }
        ]
}
```


3.4 Devdetails

JSON:


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

}

Return:


```
{
        "STATUS":[{
                "STATUS":"S",
                "When":1643181852,
                "Code":69,
                "Msg":"Device Details",
                "Description":"btminer"
        }],
        "DEVDETAILS":[                                  Hashboard detail
                {
                        "DEVDETAILS":0,
                        "Name":"SM",
                        "ID":0,
                        "Driver":"bitmicro",
                        "Kernel":"",
                        "Model":"M30S+VE40"
                },
                {
                        "DEVDETAILS":1,
                        "Name":"SM",
                        "ID":1,
                        "Driver":"bitmicro",
                        "Kernel":"",
                        "Model":"M30S+VE40"
                },
                {
                        "DEVDETAILS":2,
                        "Name":"SM",
                        "ID":2,
                        "Driver":"bitmicro",
                        "Kernel":"",
                        "Model":"M30S+VE40"
                }
        ]
}
```


3.5 Get PSU

Contains power information.

JSON:

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

Return:

```
{
        "STATUS":"S",
        "When":1643182793,
        "Code":131,
        "Msg":{
                "name":"P221B",
                "hw_version":"V01.00",
                "sw_version":"V01.00.V01.03",
                "model":"P221B",
                "iin":"8718",                          Current in
                "vin":"22400",                         Voltage in
                "fan_speed":"6976",                    Power fan speed
                "version":"-1",
                "serial_no":"A1232B0120100049",
                "vender":"7"
        },
        "Description":""
}
```

3.6 Get version

Get miner API version.

JSON:

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

Return:

```
{
        "STATUS":"S",
        "When":1643187652,
        "Code":131,
        "Msg":{"api_ver":"2.0.3","fw_ver":"20220125.13.Rel"},
        "Description":""
}
```

3.7 Get token

You must use plaintext, and miner will return plaintext.

JSON:

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

Return:

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

3.8 Status

Get btminer status and firmware version.

JSON:

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

Return:

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

Notice: miner supports 16 IP clients, one IP can get 32 tokens, and token keepalive is 30min.

3.9 Get miner info

JSON:

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

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

Return:

```
{
```

"STATUS":"S",
"When":1618212903,
"Code":131,
"Msg":{"ip":"192:168:2:16","proto":"dhcp","netmask":"255.255.255.0","dns":"114.114.114.114","mac":"C6:07:20:00:1E:C2","ledstat":"auto","gateway":"192.168.2.1"},
"Description":""
}

4.0 Get error code

JSON:

{
"cmd":"get_error_code",
}

Return:

{
"STATUS":"S",
"When":1642392343,
"Code":131,
"Msg":{"error_code":["329":"2022-01-17 11:28:11"]},
"Description":""
}

API:

client -> miner： "get_token"

miner -> client： $time $salt $newsalt

e.g.： "1592555626 BQ5hoXV9 jbzkfQls"

JSON:

client -> miner： {"cmd":"get_token"}

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

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

$time $salt $newsalt are separated by space

time: timestamp, This count starts at the Unix Epoch on January 1st, 1970 at UTC.

salt: a random Salt is generated for each password

new_slat: new salt for sign

Token calculation method:

Get token from miner: 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：
First, Get those values from miner: $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}${time:0-4}"|cut -f 4 -d '$'`

The default user name and password are 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.

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, Writable API only supports ciphertext.

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

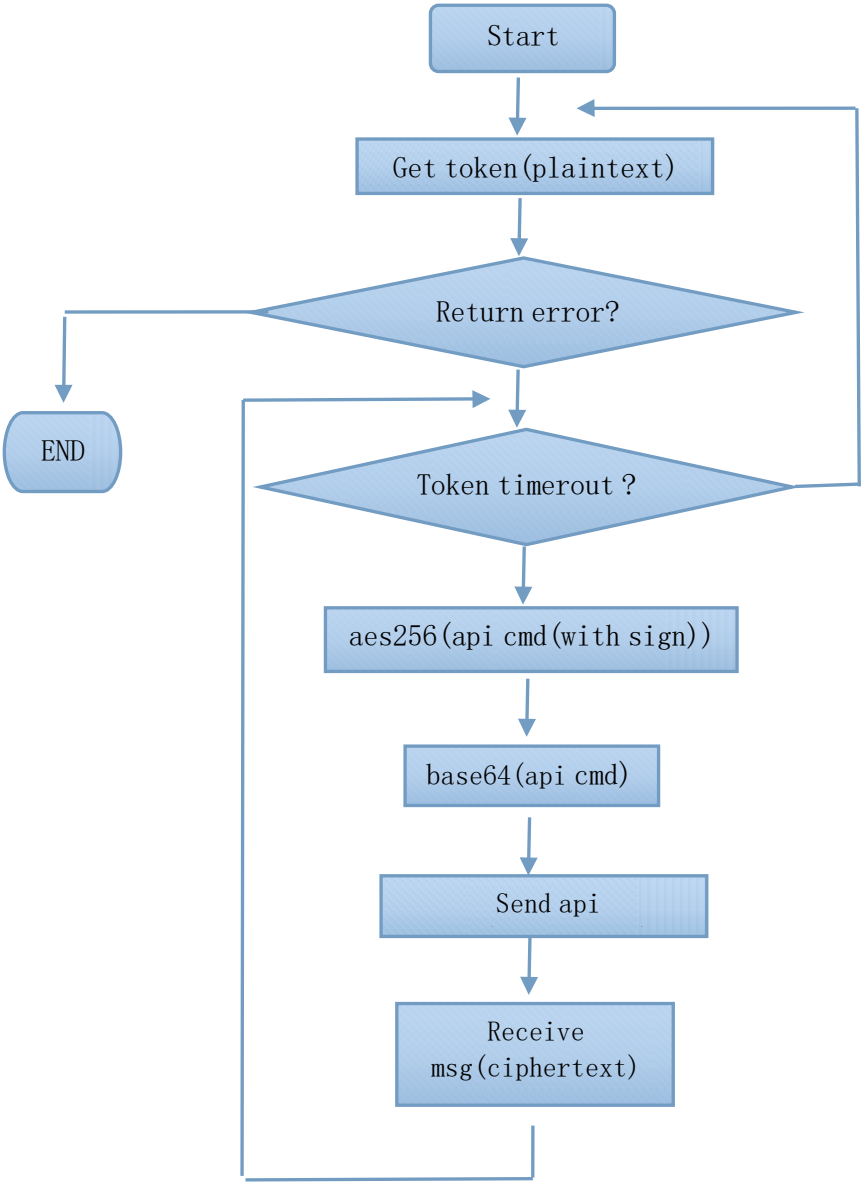
Steps as follows:
(1)api_cmd = token,$sign|api_str
(2)enc_str = aes256(api_cmd, $key)
(3)tran_str = base64(enc_str)
api_str is API command plaintext

Generate aeskey step：
(1)Get token from miner: $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 error?
END
Token timerout ?
aes256(api cmd(with sign))
base64(api cmd)
Send api
Receive
msg(ciphertext)