



# WhatsMiner API

## User's Manual

V2.2.2




Shenzhen MicroBT Electronics Technology Co., Ltd.

# Forward

## About this Document

This Document includes instructions to how to start the API, how to use the API, and functions that the API can achieve.

## Symbol instruction

Symbol	Instruction
	Provides additional information to supplement the main text.

## Revision history

Version	Revision Content	Release Time
V2.2.1	Updated description	20250724

## Legal information

© 2024 Shenzhen MicroBT Electronics Technology Co., Ltd. All rights reserved.

Any part of this Document, including text, images, graphics, etc., belongs to Shenzhen MicroBT Electronics Technology Co., Ltd. or its affiliated companies (hereinafter referred to as "MicroBT"). Without written permission, no unit or individual may extract, copy, translate, or modify all or part of this Document in any way. Unless otherwise agreed, MicroBT makes no express or implied representations or warranties with respect to this Document. The information contained in this Document is subject to change, without notice, due to product updates or other reasons. Please find the latest version of this Document on the MicroBT website.



and other MicroBT' s trademarks and logos are properties of MicroBT in various jurisdictions.

Other trademarks, logos and company names in this Document are properties of their respective owners.

# Contents

Forward .....	2
Contents .....	3
1. Summary .....	6
2. Protocol .....	7
2.1 API Ciphertext .....	7
2.2 Flow .....	9
3. Writable API .....	10
3.1 Update Mining Pool Information .....	11
3.2 Restart btminer .....	11
3.3 Power off Hashboard .....	12
3.4 Power on Hashboard .....	12
3.5 Manage LED Lights .....	12
3.6 Switch Power Mode .....	13
3.7 Upgrade Firmware .....	13
3.8 Reboot System .....	14
3.9 Restore to Factory Setting .....	15
3.10 Modify Password of Admin Account .....	15
3.11 Modify Network Configuration .....	15
3.12 Download Logs .....	16
3.13 Set Target Frequency .....	17
3.14 Enable btminer Fast Boot .....	17
3.15 Disable btminer Fast Boot .....	18
3.16 Enable Web Mining Pool .....	18

3.17 Disable Web Mining Pool .....	18
3.18 Set Hostname .....	19
3.19 Set Time Zone .....	19
3.20 Load log .....	19
3.21 Set Power Percent (Fast Mode) .....	20
3.22 Set Power Percent V2 (Normal Mode) .....	21
3.23 Set Power Value .....	21
3.24 Restore Power PCT .....	22
3.25 Set Temp Offset .....	22
3.26 Adjust Power Limit .....	22
3.27 Adjust Upfreq Speed .....	23
3.28 Set Poweroff Cool .....	23
3.29 Set Fan Zero Speed .....	24
3.30 Set Heat Mode .....	24
3.31 Disable btminer Init .....	24
3.32 Enable btminer Init .....	25
3.33 Set Customer Message .....	25
3.34 Get Customer Message .....	25
3.35 Set Fast Mining .....	26
4. Readable API .....	27
4.1 Summary .....	27
4.2 Pools .....	28
4.3 Edevs/Devs .....	29
4.4 Devdetails .....	31
4.5 Get PSU .....	33
4.6 Get Version .....	34

4.7 Get Token .....	35
4.8 Status .....	36
4.9 Get miner info .....	36
4.10 Get error code .....	37

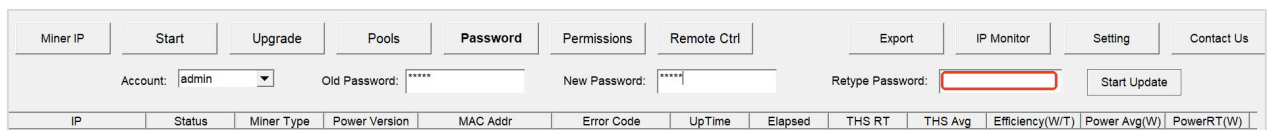
# 1. Summary

This Document describes how to use WhatsMiner API (Hereinafter referred to as “API”). API enables miner management software developers to replicate functionalities comparable to WhatsMinerTool, including real-time monitoring and miner control operations. Functions of miner management software comparable to WhatsMinerTool can be achieved via API.

API read permission is enabled by default. As for API write permission, it requires activation through WhatsMinerTool. Follow the following steps.

Step 1 Open WhatsMinerTool, click **Password**, and then change the default password.

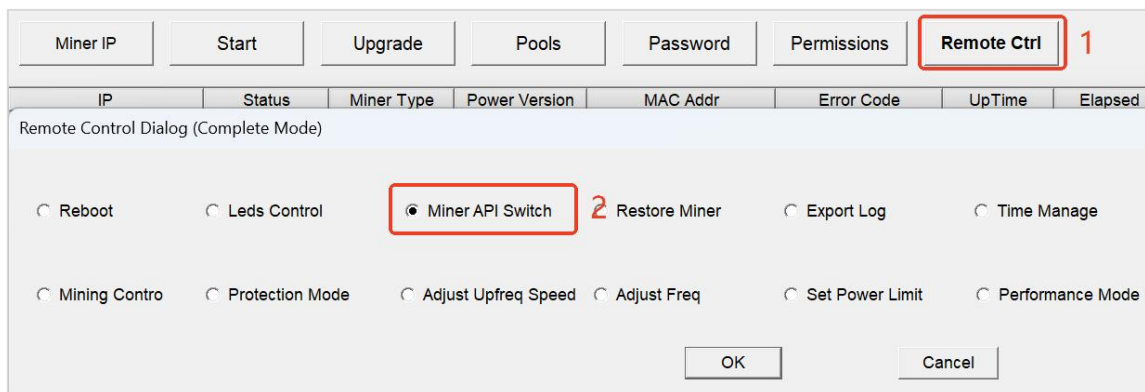
The default password is admin.



The screenshot shows the 'Password' tab in the WhatsMinerTool interface. It includes fields for 'Account' (set to 'admin'), 'Old Password' (masked with asterisks), 'New Password' (masked with asterisks), and 'Retype Password' (masked with asterisks). A red box highlights the 'Retype Password' field. Below the password fields is a 'Start Update' button. At the bottom, there is a table with columns: IP, Status, Miner Type, Power Version, MAC Addr, Error Code, UpTime, Elapsed, THS RT, THS Avg, Efficiency(W/T), Power Avg(W), and PowerRT(W).

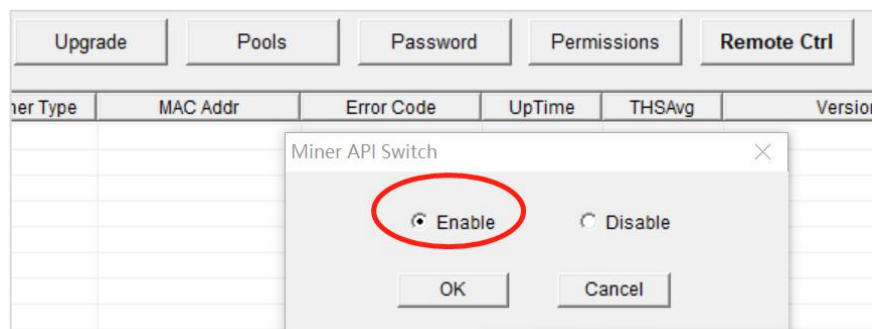
Figure 1-1

Step 2 Click **Remote Ctrl**, click **Miner API Switch**, and then enable API.



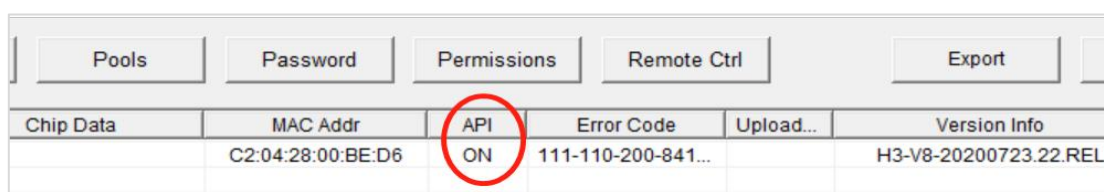
The screenshot shows the 'Remote Ctrl' tab in the WhatsMinerTool interface. A red box highlights the 'Remote Ctrl' tab with a red '1' next to it. Below the tab is a table with columns: IP, Status, Miner Type, Power Version, MAC Addr, Error Code, UpTime, and Elapsed. Below the table is a 'Remote Control Dialog (Complete Mode)' window. Inside this dialog, the 'Miner API Switch' option is selected with a radio button, and a red box highlights it with a red '2' next to it. Other options include Reboot, Leds Control, Restore Miner, Export Log, Time Manage, Mining Control, Protection Mode, Adjust Upfreq Speed, Adjust Freq, Set Power Limit, and Performance Mode. At the bottom of the dialog are 'OK' and 'Cancel' buttons.

Figure 1-2



The screenshot shows the 'Miner API Switch' dialog box. It has two radio buttons: 'Enable' and 'Disable'. The 'Enable' option is selected, and a red box highlights it. Below the radio buttons are 'OK' and 'Cancel' buttons. The dialog box is overlaid on a table with columns: Miner Type, MAC Addr, Error Code, UpTime, THSAvg, and Version.

Figure 1-3



The screenshot shows the 'Permissions' tab in the WhatsMinerTool interface. A red box highlights the 'API' status, which is 'ON'. The table below has columns: Chip Data, MAC Addr, API, Error Code, Upload..., and Version Info. The data row shows: C2-04:28:00:BE:D6, ON, 111-110-200-841..., and H3-V8-20200723.22.REL.

Figure 1-4

## 2. Protocol

API TCP port is 4028.



Note:

1. If no data is received within 10 seconds after the API TCP port is connected, connection will time out and be closed.
2. Miner supports up to 100 tokens.

JSON API return format:

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

The following introduces meanings represented by different codes.

Code	Message
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

### 2.1 API Ciphertext

Readable API supports two communication modes: plaintext and ciphertext, and writable API only supports one communication mode: ciphertext.

Encryption algorithm:

Ciphertext = aes256(plaintext), ECB mode

Encode text = base64(ciphertext)

Steps are 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

Steps for generating aeskey are as follows.

(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))

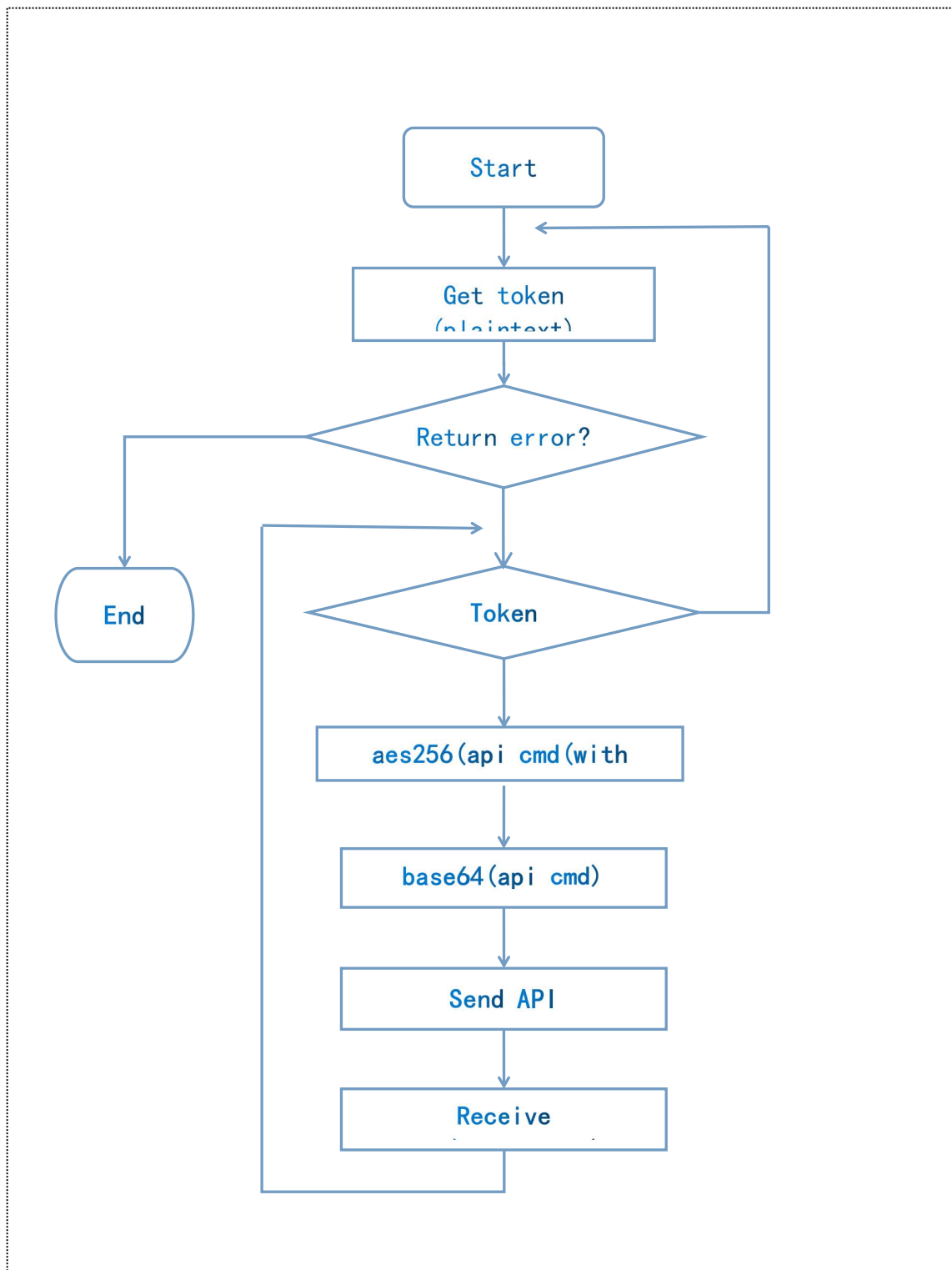
JSON:

```
{  
  "enc":1,          # integer  
  "data":"base64 str"
```



}

## 2.2 Flow



### 3. Writable API

Writable API only supports ciphertext.

Writable API must first obtain the token as follows.

JSON:

```
client -> miner:
{"cmd": "get_token"}

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

e. g. :
{"time": "5626", "salt": "BQ5hoXV9", "newsalt": "jbz kfQls"}
```

**Time:** Time (timestamp) starting from the Unix Epoch (January 1, 1970 UTC).

**Salt:** Random salt generated for each password.

**newsalt:** Newsalt for sign.

#### Token Calculation Method

Get token from miner: time, salt, newsalt.

1. Calculate key using admin' s password and salt.

2. Time is last four characters of time.

key = md5(salt + admin\_password)

sign = md5(newsalt + key + time)

Reference code in Ubuntu:

Firstly, get those values from miner: \$time \$salt \$newsalt.

Secondly, input Ubuntu Shell commands:

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



Note: Default user name and password are admin

### 3.1 Update Mining Pool Information

This operation can update mining pool configuration. After this operation, the new configuration set for the mining pool will take effect immediately.

JSON:

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

### 3.2 Restart btminer

This operation only restarts the btminer process not the control board.

JSON:

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

### 3.3 Power off Hashboard

This operation simply stops the mining process and cuts the power of hashboards.

JSON:

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

### 3.4 Power on Hashboard

This operation simply initiates the mining process and powers on hashboards.

JSON:

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

### 3.5 Manage LED Lights

This operation restores LED lights to flash in an automatic control mode.

JSON:

```
{
  "token": "str",
  "cmd": "set_led",
  "param": "auto"  #The LED flashes in auto mode
}
```

This operation enables LED lights to flash in a manual setting mode.

JSON:

```
{
  "token": "str",
  "cmd": "set_led",
  "color": "str",      # str must be "red" or "green"
  "period": integer,  # flash cycle ms
  "duration": integer, # led on time in cycle(ms)
  "start": integer     # led on time offset in cycle(ms)
}
```

### 3.6 Switch Power Mode

After the power mode of the miner is successfully configured, btminer will be restarted.

JSON:

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

{
  "token": "str",
  "cmd": "set_high _power"
}

{
  "token": "str",
  "cmd": "set_normal _power"
}
```

### 3.7 Upgrade Firmware

Upgrade flow is as follows.

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 whether upgrade is successful by a value of "Firmware Version" returned by summary.



Note: All interactions are one-time TCP connections.

### 3.8 Reboot System

This operation restarts the control board.

JSON:

```
{
```

```
"token": "str",  
"cmd": "reboot"  
}
```

### 3.9 Restore to Factory Setting

This operation restores the network configuration, system passwords, user permission, power mode, power limit, and other settings to factory setting.

JSON:

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

### 3.10 Modify Password of Admin Account

The maximum password length is 8 bytes.



Note: The token must be acquired again from the miner for encrypted transmission.

JSON:

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

### 3.11 Modify Network Configuration

After modifying the network configuration, the 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",      # e.g. : "114.114.114.114 192.168.0.1" , divide by
space
  "host": "str"
}
```

### 3.12 Download Logs

This operation exports log files of the miner.

Download flow is as follows.

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 content after a 10-millisecond delay.

### 3.13 Set Target Frequency

This operation sets a new target mining frequency for the miner, which adjusts a certain percentage based on a mining frequency in normal power mode (a percentage of 0 means no adjustment).



Note: All of air cooling miners, hydro-cooling miners, and immersion cooling miners support overclocking and underclocking. After successfully setting the target frequency, the miner will automatically restart.

JSON:

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

### 3.14 Enable btminer Fast Boot

This operation enables the mining service to quickly increase power at startup, which is useful for mining farm load power scenarios that demand



quick activation. Note: This operation prioritizes rapid power target achievement over quick hashrate stabilization, resulting in minimal effect on hashrate stabilization time. Such configuration will

take effect upon the next startup of the miner.

JSON:

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

### 3.15 Disable btminer Fast Boot

This operation disables btminer fast boot mode.



Note: Such configuration will take effect upon the next startup of the miner.

JSON:

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

### 3.16 Enable Web Mining Pool

This operation enables you to configure pools via a webpage, with changes taking effect immediately.

JSON:

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

### 3.17 Disable Web Mining Pool

This operation disables the pool configuration feature via a webpage, with changes taking effect immediately.

JSON:

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

### 3.18 Set Hostname

This configuration does not take effect until the network is restarted.

JSON:

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

### 3.19 Set Time Zone

This configuration does not take effect until the network is restarted.

JSON:

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

### 3.20 Load log

This operation configures the rsyslog log server.

JSON:

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

### 3.21 Set Power Percent (Fast Mode)

The dynamic adjustment of the power percentage is rooted in the initially stable mining power, and this adjustment process is accomplished within a single second. It's important to note that only an approximate power percentage can be reached. The lowest percentage that allows for stable operation is contingent upon the miner's inherent characteristics, the environmental temperature, and the cooling conditions.

If the target percentage is set too low, the miner may become unstable, triggering an automatic restart of the miner. Meanwhile, the maximum percentage must not exceed 100%. This adjustment method is only advisable for temporary use, as long - term operation might result in instability.

When compared with Set Power Percent V2 (Normal Mode), it can adjust the power percentage more rapidly. Nevertheless, this speed comes at the cost of greater performance loss. If the speed of adjustment is not your top priority, it is recommended that you opt for Set Power Percent V2 (Normal Mode).

JSON:

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

## 3.22 Set Power Percent V2 (Normal Mode)

The dynamic adjustment of the power percentage is rooted in the initially stable mining power, and this adjustment process spans a few minutes and might lead to a minor, yet negligible, decline in the hashrate.

It is important to understand that only an approximate power percentage can be reached. The lowest percentage that allows for stable operation is contingent upon the miner's inherent characteristics, the environmental temperature, and the cooling conditions. Setting the target percentage too low may destabilize the miner, prompting an automatic restart of the miner.

The maximum power percentage for air cooling miners cannot exceed 100%, while for water cooling and liquid cooling miners, this threshold is set at 200%.

Although its impact on performance is less pronounced than that of Set Power Percent (Fast Mode), the miner still fails to reach its peak performance and stability. This dynamic adjustment is well-suited for scenarios where power costs fluctuate frequently, necessitating continuous power adjustments.

If there is no requirement for frequent power changes and long-term operation at a fixed power level is desired, it is advisable to opt for **Adjust Power Limit** or **Set Target Frequency**.

JSON:

```
{
  "cmd": "set_power_pct_v2",
  "percent": "str",      #range: 0~100(air cooling), 0~200(other)
  "token": "str"
}
```

## 3.23 Set Power Value

This operation can set the power value of the miner.

JSON:

```
{
```

```
"cmd": "set_power",  
"power": "str"  
"token": "str"  
}
```

### 3.24 Restore Power PCT

This operation restores configurations made in **Set Power Percent V2** and **Set Power Percent**.

JSON:

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

### 3.25 Set Temp Offset

This operation sets the target temperature offset for miner' s hashboards (range: -30 °C to 0 °C). This will increase the fan speed of the miner, reducing the strain on the mining farm's cooling system. This is only effective for the air cooling miner. Such configuration will take effect upon the next startup of the miner.

JSON:

```
{  
  "cmd": "set_temp_offset",  
  "temp_offset": "str",           #range: -30 ~ 0  
  "token": "str"  
}
```

### 3.26 Adjust Power Limit

This operation sets the upper power limit of the miner. It is recommended that the power limit should be not higher than the ordinary power of the miner. The miner will restart when such configuration take effect.

JSON:

```
{
  "cmd": "adjust_power_limit",
  "power_limit": "str",           #range: 0 ~ 99999
  "token": "str"
}
```

### 3.27 Adjust Upfreq Speed

This operation configures the startup speed of the miner, which also pertains to the speed at which the miner's frequency is adjusted and stabilized. A higher speed setting results in a shorter time to achieve a stable hashrate. The speed is adjustable on a scale from 0 to 9, where 0 indicates the normal speed and 9 represents the fastest.

However, it's important to note that as the speed increases, there will be a larger deviation between the target hashrate and power. This will have a more significant impact on the miner's performance and stability. Moreover, Fast Boot mode cannot be used concurrently.

JSON:

```
{
  "cmd": "adjust_upfreq_speed",
  "upfreq_speed": "str",         #range: 0 ~ 9
  "token": "str"
}
```

### 3.28 Set Poweroff Cool

This operation configures whether to cool the miner when stopping mining. This is only effective for the air cooling miner.

JSON:

```
{
  "cmd": "set_poweroff_cool",
  "poweroff_cool": "str",          #type: bool, range: [0,1]
  "token": "str"
}
```

### 3.29 Set Fan Zero Speed

This operation configures whether the fan speed can be set to a minimum of 0. This is only effective for the air cooling miner.

JSON:

```
{
  "cmd": "set_fan_zero_speed",
  "fan_zero_speed": "str",          #type: bool, range: [0,1]
  "token": "str"
}
```

### 3.30 Set Heat Mode

The hydro-cooling miner is default to be anti-freezing mode.

The power-keeping mode ensures that the miner maintains full power during network outages.

JSON:

```
{
  "cmd": "set_heat_mode",
  "mode": "str",                    #anti-icing(anti-freezing), heating
  "token": "str"
}
```

### 3.31 Disable btminer Init



This operation disables the miner from initiating mining at startup.

JSON:

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

### 3.32 Enable btminer Init

This operation enables the miner to initiate mining at startup.

JSON:

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

### 3.33 Set Customer Message

This operation writes the customer-defined SN.

JSON:

```
{
  "cmd": "set_customer_msg",
  "key": "str",    # "CustomerSn", "msg0" ~ "msg9".key length is less
than 128.
  "val": "str",    # val length is less than 128.
  "cmd": "set_customer_msg",
  "token": "str"
}
```

### 3.34 Get Customer Message

This operations read the customer-defined SN.

JSON:

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

### 3. 35 Set Fast Mining

1978 (VK) miner: 80%~90% of the hashrate is achieved up frequency.

1980 (VL)/1990 (VM) miner: 85%~95% of the hashrate is achieved up frequency.

Other miners: do not support this function.

The two commands set\_fast\_mining and set\_fast\_hash have the same function.

JSON:

```
{  
  "cmd": "set_fast_mining",  
  "fast_mining": "str"    # 1 enable, 0 disable  
}
```

```
{  
  "cmd": "set_fast_hash",  
  "fast_hash": "str"    # 1 enable, 0 disable  
}
```

## 4. Readable API

### 4.1 Summary

This operation can obtain the summary of the miner, including fan speed, power information, and the like.

JSON:

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

Return:

```
{
  "STATUS": "S",
  "When": 15158,
  "Code": 131,
  "Msg": {
    "Elapsed": 1707.7506103515625,
    "MHS av": 201774992,           #Average hash rate of miner (MHS)
    "MHS 1m": 201208000,
    "MHS 15m": 201338752,
    "HS RT": 201208000,
    "freq_avg": 544.21868896484375,
    "Fan Speed In": 5601,         #Air outlet fan speed (RPM)
    "Fan Speed Out": 5601,       #Air inlet Fan speed (RPM)
    "Power": 4177.1733333333332,   #Input power (W)
    "Power Rate": 20.760604858398438,
    "Pool Rejected%": 0,
    "Uptime": 2967,               #System up time (second)
    "Hash Stable": "true",
    "Hash Stable Cost Seconds": 1212,
  }
}
```

```

    "Target Freq": 0,
    "Env Temp": 32.312,
    "Power Mode": "Normal",      #Power mode (Low/Normal/High)
    "Factory GHS": 84773,       #Factory hash rate(GHS)
    "Power Limit": 4200,
    "Chip Temp Min": 59.166667938232422,
    "Chip Temp Max": 95.766670227050781,
    "Chip Temp Avg": 77.333328247070312,
    "Debug": "62.4 2.0/1.3 2.6/1.3 1.9/1.4 A:99.5/0.14/20.7/1.1",
    "Btminer Fast Boot": "disable"
  },
  "Description": ""
}

```

## 4.2 Pools

The operation can obtain the mining pool information of the miner, including pool address, pool difficulty, and the like.

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
    "Getworks": 1,
    "Accepted": 0,                                   #Accepted nonces by the pool
    "Rejected": 0,                                   #Rejected nonces by the pool
    "Discarded": 0,
    "Stale": 0,
    "Get Failures": 0,
    "Remote Failures": 0,
    "User": "microbtinitial",                        #Miner name
    "Last Share Time": 0,                            #Last nonce submission time
    "Stratum Active": true,                           #Pool stratum status
    "Stratum URL": "btc-vip-3dcoa7jxu.ss.poolin.com", #Pool address
    "Stratum Difficulty": 65536.00000000,            #Pool difficulty
    "Pool Rejected%": 0.0000,                         #Pool rejection percent
    "Pool Stale%": 0.0000,
    "Bad Work": 0,
  }
]
}

```

### 4.3 Edevs/Devs

This operation can obtain the information of each hashboard, including temperature, chip frequency, and the like.

JSON:

```

{
  "cmd": "edevs"
}

```

```
}
```

Return:

```
{
  "STATUS": [{"STATUS": "S", "Msg": "3 ASC(s)"}],
  "DEVS":
  [
    {
      "Slot": 0,                                #Hash board slot number
      "Temperature": 80.00,                      #Board temperature at air outlet
      (°C)
      "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 15m": 10296867.74,
      "HS RT": 10351110.56,
      "Factory GHS": 28836,                    #Factory marking(GHS)
      "Upfreq Complete": 0,
      "Effective Chips": 156,
      "PCB SN": "HEM1EP9C400929K60003",      #PCB serial number
      "Chip Data": "K88Z347-2039   BINV01-195001D",
    },
    {
      "Slot": 1,
      "Temperature": 80.00,
      "Chip Frequency": 590,
      "MHS av": 10259948.84,
      "MHS 5s": 5413853.90,
      "MHS 1m": 8577249.68,
      "MHS 15m": 10214893.36,
```

```

    "HS RT":10441143.92,
    "Factory GHS":47761,
    "Upfreq Complete":0,
    "Effective Chips":156,
    "PCB SN":"HEM1EP9C400929K60001",
    "Chip Data":"K88Z347-2039   BINV01-195001D",
  },
  {
    "Slot":2,
    "Temperature":80.00,
    "Chip Frequency":590,
    "MHS av":10258829.89,
    "MHS 5s":5571781.71,
    "MHS 1m":8675316.17,
    "MHS 15m":10213779.32,
    "HS RT":10479953.41,
    "Factory GHS":47761,
    "Upfreq Complete":0,
    "Effective Chips":156,
    "PCB SN":"HEM1EP9C400929K60002",
    "Chip Data":"K88Z347-2039   BINV01-195001D",
  }
]
}

```

## 4.4 Devdetails

This function is not recommended for use.

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" #Renamed to minor\_type in get\_version
    },
    {
      "DEVDETAILS": 1,
      "Name": "SM",
      "ID": 1,
      "Driver": "bitmicro",
      "Kernel": "",
      "Model": "M30S+VE40"
    },
  ]
}
```



```

    {
      "DEVDETAILS":2,
      "Name":"SM",
      "ID":2,
      "Driver":"bitmicro",
      "Kernel":"",
      "Model":"M30S+VE40"
    }
  ]
}

```

## 4.5 Get PSU

This operation can obtain the PSU information, including model, temperature, and the like.

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, unit: 1mA
        "vin": "22400",                                #Voltage in, unit: 10mV
        "pin": "3000",
        "fan_speed": "6976",                            #Power fan speed
        "version": "-1",
        "serial_no": "A1232B0120100049",
        "vendor": "7",
        "temp0": "33.5"                                #Temperature of power (Celsius degree)
    },
    "Description": ""
}

```

## 4.6 Get Version

This operation can get the API version of the miner.

JSON:

```

{
    "cmd": "get_version"
}

```

Return:

```

{
    "STATUS": "S",
    "When": 1643187652,
    "Code": 131,
    "Msg":
    {
        "api_ver": "2.0.3",
        "fw_ver": "20220125.13.Rel",
        "platform": "H60S",
        "chip": "K88Z001-2039 BINV01-195001B",
    }
}

```

```
    "miner_type": "M30S+VE40"
  },
  "Description": ""
}
```

## 4.7 Get Token

Plaintext must be used, and the miner returns plaintext.

JSON:

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

Return:

```
{
  "STATUS": "string",
  "When": 12345678,
  "Code": 133,
  "Msg":
  {
    "timeout": "str",    #"timeout" == "0":disable timeout. "timeout" >
    "0":timeout is the time difference from the last API received to the
    current time. When this field is not defined, the timeout is fixed at 30
    minutes.
    "time": "str",
    "salt": "str",
    "newsalt": "str"
  },
  "Description": ""
}
```

## 4.8 Status

This operation can obtain the btminer status and the firmware version.

JSON:

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

Return:

```
{
  "btminerooff": "str",           #"true"/"false"
  "Firmware Version": "str",
  "power_mode": "str",
  "power_limit_set": "str",      #Empty string is the default
  "hash_percent": "str"         #changed by set_target_freq
}
```

## 4.9 Get miner info

This operation can obtain the miner information, including IP address, netmask, gateway, hostname, and the like.

JSON:

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

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",
    "minersn": "str",
    "powersn": "str",
    "ut_speed": "5"
  },
  "Description": ""
}
```

#### 4.10 Get error code

This operation can obtain the error code of the miner.

JSON:

```
{
  "cmd": "get_error_code",
}
```

Return:

```
{
  "STATUS": "S",
```

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

The API command can be used for two joins.

e. g.

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