

RPET Programming Guide

v0.1



Table of Contents

Introduction	5
RPET Service Supported Operating Systems	5
RPET Service Installation	5
RPET Service Installation for windows	5
Library Requirement	5
Package Download Links	6
Installation Steps	6
RPET Service Installation Steps for Linux	7
Library Requirement	7
Installation Steps	8
Check RPET Service Installation	8
RPET command Overview	9
Usage	9
Command Options	9
API	10
POST /pet/api/login	11
POST /pet/api/logout	12
POST /pet/api/permission/add	12
POST /pet/api/permission/delete	12
GET /pet/api/permission	13
POST /pet/api/permission/off	14
POST /pet/api/permission/on	14
POST /pet/api/permission/update	14
GET /pet/api/version	15
BoardInfo	15
GET /pet/board/bios	16
GET /pet/board/ec	16
GET /pet/board	17
GET /pet/board/platform	18
ECI2C	18
GET /pet/eciic/byte/{addr}/{offset}/{length}	18
POST /pet/eciic/byte/{addr}/{offset}/{length}	19
POST /pet/eciic/frequency/{frequency}	20
GET /pet/eciic/quick/{addr}	21
POST /pet/eciic/quick/{addr}	22
GET /pet/eciic/receive/{addr}	22
POST /pet/eciic/send/{addr}/{data}	23
GET /pet/eciic/word/{addr}/{offset}/{length}	24
POST /pet/eciic/word/{addr}/{offset}/{length}	24
FirmwareUpdate	25
POST /pet/firmware/delete/{type}	26
GET /pet/firmware	26
GET /pet/firmware/{type}	27
POST /pet/firmware/update/bios	28
POST /pet/firmware/update/ec	28
POST /pet/firmware/upload/{type}	29
GPIO	30
GET /pet/gpio	30
GET /pet/gpio/{set}	31

GET /pet/gpio/{set}/{pin}	32
POST /pet/gpio/{set}/{pin}	33
POST /pet/gpio/{set}	34
HardwareMonitor	35
GET /pet/hwm/fan	35
GET /pet/hwm/fan/speed/{idx}	36
GET /pet/hwm	36
GET /pet/hwm/temperature	37
GET /pet/hwm/temperature/{idx}	38
GET /pet/hwm/voltage	39
GET /pet/hwm/voltage/{idx}	40
Logs	40
GET /pet/log/{idx}	41
PowerControl	41
POST /pet/power/off	41
POST /pet/power/reboot	42
SCAN	43
GET /scan/{scan-key}	43
SMBus	44
GET /pet/smbus/block/{addr}/{cmd}/{count}	44
POST /pet/smbus/block/{addr}/{cmd}/{count}	45
POST /pet/smbus/byte/{addr}/{cmd}/{data}	46
GET /pet/smbus/byte/{addr}/{cmd}	46
GET /pet/smbus	47
POST /pet/smbus/word/{addr}/{cmd}/{data}	48
GET /pet/smbus/word/{addr}/{cmd}	49
Watchdog	50
GET /pet/wdt	50
POST /pet/wdt	50
JSON Models	51
Err	51
ErrArray	51
FWStatus	52
GPIOSettings	52
PinAll	52
PinOne	53
Set	53
api_login_body	53
cmd_count_body	54
gpio_set_body	54
RspBodySystemInfo	54
RspBodyAPIKey	54
RspBodyVoltage	54
RspBodyFanAll	55
RspBodyFan	55
RspBodyGPIOAll	55
RspBodyWDT	55
RspBodySMBusSta	56
RspBodySMBusByteRd	56
RspBodySMBusWdRd	56
RspBodySMBusBlkRd	56

RspBodyFWFile	56
RspBodyPerm	57
RspBodyApiVer	57
RspBodyBrdInfo	57
RspBodyBiosInfo	57
RspBodyECInfo	58
RspBodyTempInfo	58
RspBodyTempData	58
RspBodyVoltageInfo	58
offset_length_body	59
offset_length_body_1	59
permission_add_body	59
permission_delete_body	60
permission_update_body	60
pet_wdt_body	60
power_off_body	61
power_reboot_body	61
update_bios_body	61
upload_type_body	61
Troubleshooting	63
Unable to Connect	63



Introduction

Remote PET represents the next evolutionary step in hardware management, bringing remote functionality to the Portwell Engineering Toolkits. This enhancement allows developers to control and monitor Portwell hardware effortlessly across various use cases, leveraging the widely adopted RESTful API architecture and JSON format. By integrating these modern standards, Remote PET provides a seamless and efficient interface for hardware management, making it accessible and easy to use for developers of all skill levels.

This guide includes an API list, documenting each available endpoint along with its parameters and expected responses. This API reference is crucial for developers looking to integrate Remote PET into their existing systems or to build new applications that require robust hardware control and monitoring capabilities. The APIs are designed to be intuitive and flexible, allowing for easy implementation in various programming environments.

By following this guide, developers will be equipped with the knowledge and tools necessary to effectively utilize Remote PET, ensuring efficient and reliable hardware management across a wide range of applications.

RPET Service Supported Operating Systems

Windows 7, Windows 10, Windows 11
Ubuntu 18.04 and later
CentOS 7 and later

RPET Service Installation

RPET Service Installation for windows

Library Requirement

Microsoft Visual C++ Redistributable packages for Visual Studio 2015, 2017, 2019, and 2022. For more information about these packages, visit the Microsoft website:

<https://docs.microsoft.com/en-us/cpp/windows/latest-supported-vc-redist?>



view=msvc-170

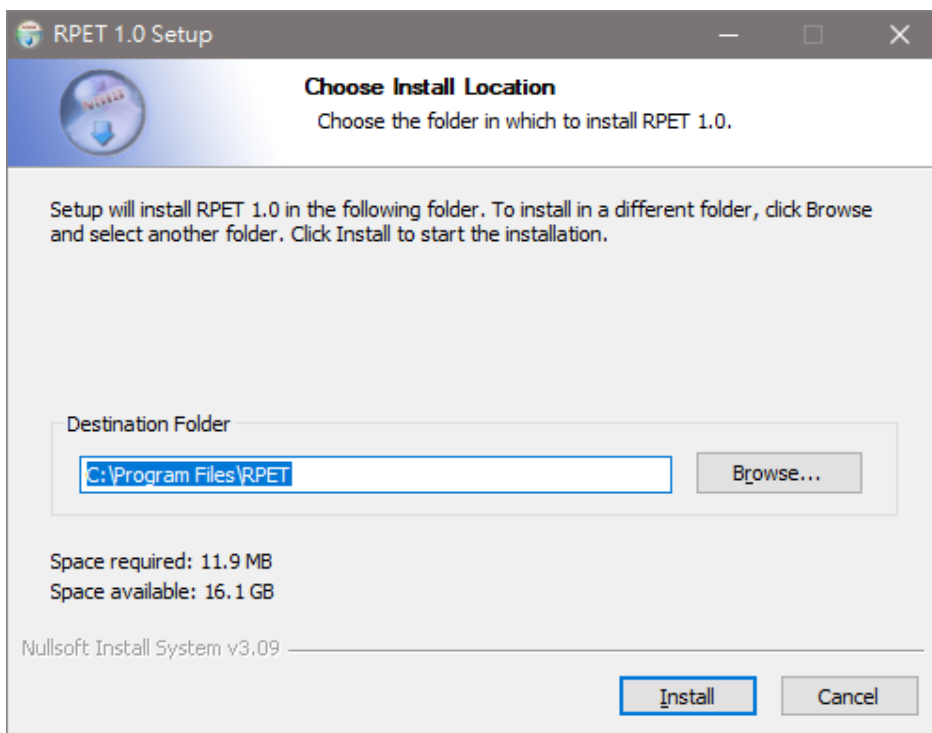
Package Download Links

https://aka.ms/vs/17/release/vc_redist.x64.exe

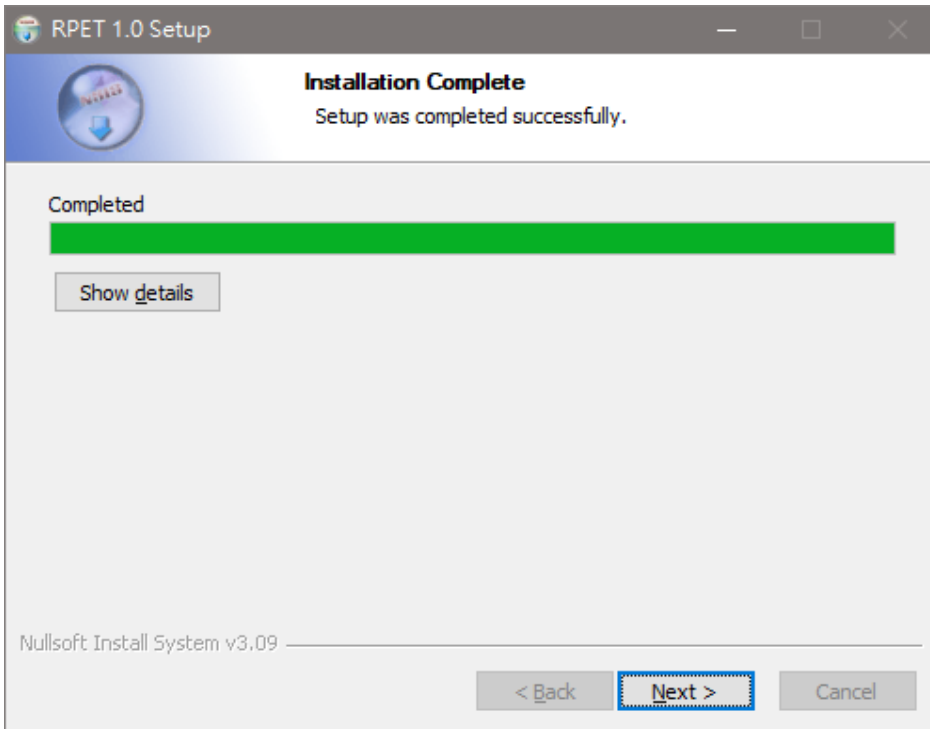
https://aka.ms/vs/17/release/vc_redist.x86.exe

Installation Steps

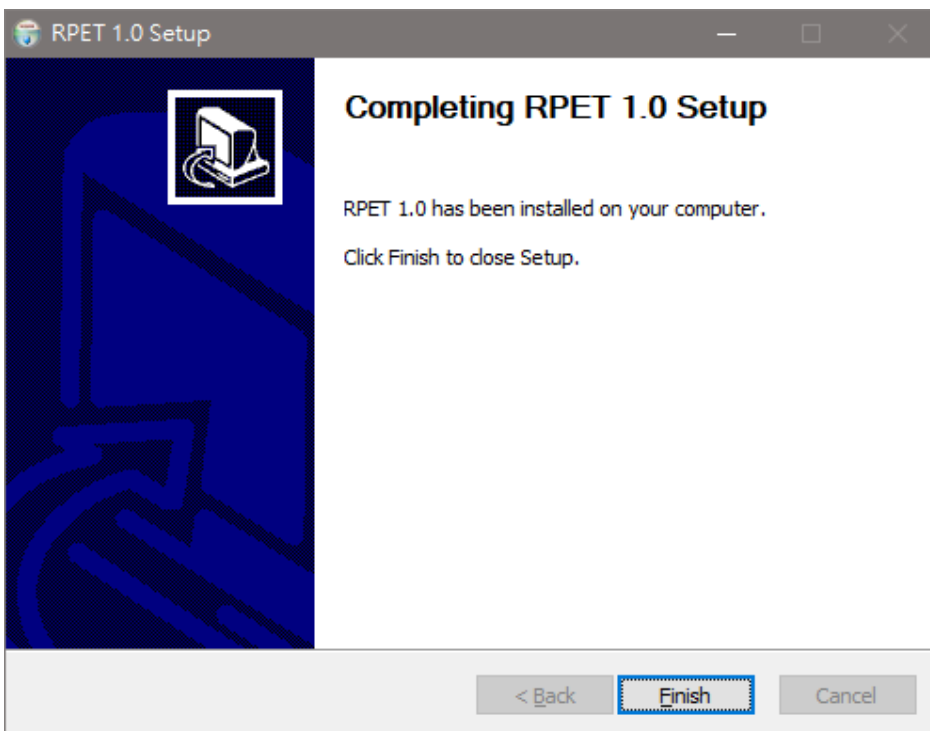
1. Double-click "RPET-1.0-Setup.exe"



2. click "Next."



3. Click "Finish" to complete the installation.



RPET Service Installation Steps for Linux

Library Requirement

Before installing RPET, ensure you have installed bwm-ng by command "sudo



`apt -y install bwm-ng"`

```
pw@pet:~$ sudo apt -y install bwm-ng
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following NEW packages will be installed:
  bwm-ng
0 upgraded, 1 newly installed, 0 to remove and 70 not upgraded.
Need to get 34.2 kB of archives.
After this operation, 94.2 kB of additional disk space will be used.
Get:1 http://tw.archive.ubuntu.com/ubuntu noble/universe amd64 bwm-ng amd64 0.6.3-2 [34.2 kB]
Fetched 34.2 kB in 0s (111 kB/s)
Selecting previously unselected package bwm-ng.
(Reading database ... 137023 files and directories currently installed.)
Preparing to unpack .../bwm-ng_0.6.3-2_amd64.deb ...
Unpacking bwm-ng (0.6.3-2) ...
Setting up bwm-ng (0.6.3-2) ...
Processing triggers for man-db (2.12.0-4build2) ...
Scanning processes...
Scanning processor microcode...
Scanning linux images...

Running kernel seems to be up-to-date.

The processor microcode seems to be up-to-date.

No services need to be restarted.

No containers need to be restarted.

No user sessions are running outdated binaries.

No VM guests are running outdated hypervisor (qemu) binaries on this host.
pw@pet:~$
```

Installation Steps

1. Install RPET by command "`sudo dpkg -i RPET_1.0_amd64.deb`"

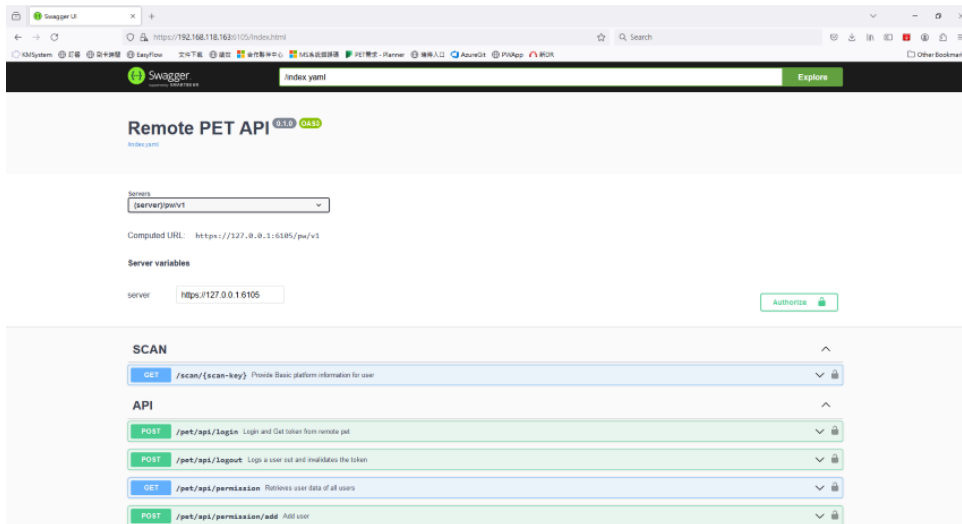
```
pw@pet:~$ sudo dpkg -i RPET_1.0_amd64.deb
Selecting previously unselected package rpet.
(Reading database ... 137000 files and directories currently installed.)
Preparing to unpack RPET_1.0_amd64.deb ...
Unpacking rpet (1.0) ...
Setting up rpet (1.0) ...
pw@pet:~$
```

Check RPET Service Installation

To check the RPET installation, follow these steps:

1. Open a web browser.
2. Connect to the following URL:
`https://:6105/index.html`

If you encounter an SSL certificate warning, as the server uses a self-signed certificate, click "Continue" to proceed. You should then see the following page:



This page is the online manual for RPET. You can now begin testing the tool.

RPET command Overview

This section provides an overview of the RPET command-line tool, which is used for managing the RPET service. Below are the commands and options available for user management and other operations.

Usage

To use the RPET command-line tool, open a command prompt (on Windows) or terminal (on Linux), navigate to the directory where the tool is located, and use the following command format:

```
RPET <option>
```

Command Options

- Display Help

-h: Show the help message with a list of available options and their descriptions.

```
$ RPET -h
Options:
  -h                show this message
  -s [<port>]       start remote pet
  -x [<port>]       start remote pet in a
                   non-authenticated state
  -l                list all users
```



```
-a <username> <passwd> <permission>  add user
-d <username>                          delete user
-p <username> <passwd>                  change password
-m <username> <permission>             change user permission
```

Parameters:

```
<username>:  username. max length is 12
<passwd>:    password. max length is 64
<permission>: user permission, 0=none, 1=read, 2=write, 3=read/write,
```

- Start the RPET Service

-s [<port>]: Start the remote pet service. If no port is specified, the default port 6105 will be used.

- Start the RPET Service in Non-Authenticated State

-x [<port>]: Start the remote pet service in a non-authenticated state. If no port is specified, the default port will be used.

- List All Users

-l: List all users in the system along with their permissions.

- Add a New User

-a <username> <passwd> <permission>: Add a new user with specified permissions.

- Delete a User

-d <username>: Delete an existing user from the system.

- Change User Password

-p <username> <passwd>: Change the password for an existing user.

- Change User Permissions

-m <username> <permission>: Change the permission level for an existing user.

API



POST /pet/api/login

Login and Get token from remote pet (`petApiLoginPost`)

Consumes

This API call consumes the following media types via the Content-Type request header:

- application/json

Request body

body [api_login_body](#)
(required)

Body Parameter –

Return type

[RspBodyAPIKey](#)

Example data

Content-Type: application/json

```
{"apiKey" : "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiIxMjM0NTY3ODkwIiwibmFtZSI6IkpvaG4gRG9lIiwiaWF0IjoxNTE2MjM5MDIyfQ.SflKxwRJSMe2QT4fwpMeJf36POk6yJV_adQssw5c"}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

Login and API init OK [RspBodyAPIKey](#)

401

Invalid credentials

500

An error occurred while login [Err](#)



POST /pet/api/logout

Logs a user out and invalidates the token (`petApiLogoutPost`)

Responses

204

logout ok

POST /pet/api/permission/add

Add user (`petApiPermissionAddPost`)

Consumes

This API call consumes the following media types via the Content-Type request header:

- application/json

Request body

body [permission_add_body](#)

(required)

Body Parameter –

Responses

200

Successfully create the user.

401

Unauthorized to create user.

500

An error occurred while create the user.

POST /pet/api/permission/delete

Delete user (`petApiPermissionDeletePost`)

Consumes

This API call consumes the following media types via the Content-Type request header:

- application/json



Request body

body [permission_delete_body](#)

(required)

Body Parameter –

Responses

200

Successfully delete the user.

401

Unauthorized to disable permission control.

500

An error occurred while turn off permission control.

GET /pet/api/permission

Retrieves user data of all users ([petApiPermissionGet](#))

Return type

array[[RspBodyPerm](#)]

Example data

Content-Type: application/json

```
[ {
  "username" : "admin",
  "permission" : [ "admin", "read", "write" ]
}, {
  "username" : "gpiouser",
  "permission" : [ "read" ]
} ]
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

Successfully get user data.

**401**

Unauthorized to get user data.

500

An error occurred while get user data.

POST /pet/api/permission/off

Turn off permission control. (petApiPermissionOffPost)

Responses

200

Successfully turned off permission control.

401

Unauthorized to disable permission control.

500

An error occurred while turn off permission control.

POST /pet/api/permission/on

Turn on permission control. (petApiPermissionOnPost)

Responses

200

Successfully turned on permission control.

500

An error occurred while turn on permission control.

POST /pet/api/permission/update

Modify permission data of target user (petApiPermissionUpdatePost)

Consumes

This API call consumes the following media types via the Content-Type request header:

- application/json

Request body

body [permission_update_body](#)
(required)



Body Parameter –

Responses

200

Successfully update the user data.

401

Unauthorized to update the user data.

500

An error occurred while update the user data.

GET /pet/api/version

Retrieves API version (`petApiVersionGet`)

Return type

[RspBodyApiVer](#)

Example data

Content-Type: application/json

```
{
  "api_version" : "api_version"
}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

API version retrieved successfully. [RspBodyApiVer](#)

500

An error occurred while retrieving API version [Err](#)

BoardInfo



GET /pet/board/bios

Retrieves BIOS information for a board (`petBoardBiosGet`)

Return type

[RspBodyBiosInfo](#)

Example data

Content-Type: application/json

```
{
  "bios_vendor" : "bios_vendor",
  "bios_version" : "bios_version"
}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

BIOS information retrieved successfully. [RspBodyBiosInfo](#)

500

An error occurred while retrieving bios information [ErrArray](#)

GET /pet/board/ec

Retrieves EC information for a board (`petBoardEcGet`)

Return type

[RspBodyECInfo](#)

Example data

Content-Type: application/json

```
{
  "ec_version" : "ec_version"
}
```



Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

EC firmware information retrieved successfully. [RspBodyECInfo](#)

500

An error occurred while retrieving EC information [Err](#)

GET /pet/board

Retrieves all board firmware information ([petBoardGet](#))

Return type

[RspBodyBrdInfo](#)

Example data

Content-Type: application/json

```
{
  "bios_vendor" : "bios_vendor",
  "ec_version" : "ec_version",
  "hw_version" : "hw_version",
  "serialnumber" : "serialnumber",
  "name" : "name",
  "type" : "type",
  "bios_version" : "bios_version"
}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

**200**Board information retrieved successfully. [RspBodyBrdInfo](#)**500**An error occurred while retrieving board information [ErrArray](#)

GET /pet/board/platform

Retrieves platform information for a board (petBoardPlatformGet)

Return type

[RspBodySystemInfo](#)

Example data

Content-Type: application/json

```
{
  "hw_version" : 0,
  "serialnumber" : "serialnumber",
  "name" : "name",
  "type" : "type"
}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200Board platform information retrieved successfully. [RspBodySystemInfo](#)**500**An error occurred while retrieving platform information [Err](#)

ECI2C

GET /pet/eciic/byte/{addr}/{offset}/{length}

Send EC I2C Byte Read Command (petEciicByteAddrOffsetLengthGet)



Path parameters

addr (required)

Path Parameter – Address parameter in hexadecimal format

offset (required)

Path Parameter – Offset parameter in hexadecimal format

length (required)

Path Parameter – Number of bytes to read

Return type

[cmd_count_body](#)

Example data

Content-Type: application/json

```
{
  "data" : [ "data", "data" ]
}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

Byte read successfully [cmd_count_body](#)

500

An error occurred while send EC I2C byte read Command. [Err](#)

POST /pet/eciic/byte/{addr}/{offset}/{length}

Send EC I2C Byte Write Command (petEciicByteAddrOffsetLengthPost)

Path parameters

addr (required)



Path Parameter – Address parameter in hexadecimal format

offset (required)

Path Parameter – Offset parameter in hexadecimal format

length (required)

Path Parameter – Number of bytes to write

Consumes

This API call consumes the following media types via the Content-Type request header:

- application/json

Request body

body [offset_length_body](#)
(required)

Body Parameter –

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

Byte sent successfully

500

An error occurred while send EC I2C byte write Command. [Err](#)

POST /pet/eciic/frequency/{frequency}

Set EC I2C Frequency (`petEciicFrequencyFrequencyPost`)

Set the Frequency of EC I2C.

Path parameters



frequency (required)

Path Parameter – Address parameter in hexadecimal format

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

Successful Set EC I2C Frequency.

500

An error occurred while retrieving EC I2C Frequency. [Err](#)

GET /pet/eciic/quick/{addr}

Send EC I2C Quick Read Command (`petEciicQuickAddrGet`)

Path parameters

addr (required)

Path Parameter – Address parameter in hexadecimal format

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

EC I2C Quick Command sent successfully

500

An error occurred while send EC I2C Quick Read Command. [Err](#)



POST /pet/eciic/quick/{addr}

Send EC I2C Quick Write Command (`petEciicQuickAddrPost`)

Path parameters

addr (required)

Path Parameter – Address parameter in hexadecimal format

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

EC I2C Quick Command sent successfully

500

An error occurred while send EC I2C Quick Write Command. [Err](#)

GET /pet/eciic/receive/{addr}

Command for receive byte from EC I2C (`petEciicReceiveAddrGet`)

Path parameters

addr (required)

Path Parameter – Address parameter in hexadecimal format

Return type

[RspBodySMBusByteRd](#)

Example data

Content-Type: application/json

```
{
  "data" : "0x12"
}
```



Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

Byte received successfully [RspBodySMBusByteRd](#)

500

An error occurred while send EC I2C received byte Command. [Err](#)

POST /pet/eciic/send/{addr}/{data}

Command for send byte to EC I2C Device ([petEciicSendAddrDataPost](#))

Path parameters

addr (required)

Path Parameter – Address parameter in hexadecimal format

data (required)

Path Parameter – Data parameter in hexadecimal format

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

Byte sent successfully

500

An error occurred while send EC I2C send byte Command. [Err](#)



GET /pet/eciic/word/{addr}/{offset}/{length}

Send EC I2C Word Read Command (`petEciicWordAddrOffsetLengthGet`)

Path parameters

addr (required)

Path Parameter – Address parameter in hexadecimal format

offset (required)

Path Parameter – Offset parameter in hexadecimal format

length (required)

Path Parameter – Number of bytes to read

Return type

[offset_length_body](#)

Example data

Content-Type: application/json

```
{
  "data" : [ "data", "data" ]
}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

Word read successfully [offset_length_body](#)

500

An error occurred while send EC I2C word read Command. [Err](#)

POST /pet/eciic/word/{addr}/{offset}/{length}

Send EC I2C Word Write Command (`petEciicWordAddrOffsetLengthPost`)



Path parameters

addr (required)

Path Parameter – Address parameter in hexadecimal format

offset (required)

Path Parameter – Offset parameter in hexadecimal format

length (required)

Path Parameter – Number of bytes to write

Consumes

This API call consumes the following media types via the Content-Type request header:

- application/json

Request body

body [offset_length_body_1](#)
(required)

Body Parameter –

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

Word sent successfully

500

An error occurred while send EC I2C word write Command. [Err](#)

FirmwareUpdate



POST /pet/firmware/delete/{type}

Delete uploaded firmware binary file (petFirmwareDeleteTypePost)

Path parameters

type (required)

Path Parameter –

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

File upload success.

410

Target file was gone. Cannot delete it.

500

An error occurred while upload file. [Err](#)

GET /pet/firmware

Returns firmware update status. (petFirmwareGet)

Return type

[RspBodyFWFile](#)

Example data

Content-Type: application/json

```
{
  "bios" : {
    "file_version" : "file_version",
    "file_md5" : "file_md5",
    "current_version" : "current_version",
    "status" : "File Not Exist"
  }
}
```



Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

Firmware update status retrieved successfully. [RspBodyFWFile](#)

500

An error occurred while get firmware update status. [Err](#)

GET /pet/firmware/{type}

Returns firmware update status. (petFirmwareTypeGet)

Path parameters

type (required)

Path Parameter –

Return type

[FWStatus](#)

Example data

Content-Type: application/json

```
{
  "file_version" : "file_version",
  "file_md5" : "file_md5",
  "current_version" : "current_version",
  "status" : "File Not Exist"
}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json



Responses

200

Firmware update status retrieved successfully. [FWStatus](#)

500

An error occurred while get firmware update status. [Err](#)

POST /pet/firmware/update/bios

update firmware (`petFirmwareUpdateBiosPost`)

Consumes

This API call consumes the following media types via the Content-Type request header:

- application/json

Request body

body [update_bios_body](#)

(required)

Body Parameter –

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

firmware update started

500

An error occurred while start firmware update. [Err](#)

POST /pet/firmware/update/ec

update firmware (`petFirmwareUpdateEcPost`)

Produces

This API call produces the following media types according to the Accept



request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

firmware update started

500

An error occurred while start firmware update. [Err](#)

POST /pet/firmware/upload/{type}

upload firmware binary file (petFirmwareUploadTypePost)

Path parameters

type (required)

Path Parameter –

Consumes

This API call consumes the following media types via the Content-Type request header:

- multipart/form-data

Form parameters

binfile (required)

Form Parameter – format: binary

version (required)

Form Parameter –

md5sum (required)

Form Parameter –

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json



Responses

200

Delete file success.

409

Conflict due to file exists.

500

An error occurred while upload file. [Err](#)

GPIO

GET /pet/gpio

Get GPIO Settings (**petGpioGet**)

Retrieve the current state of all GPIO pins.

Return type

[GPIOSettings](#)

Example data

Content-Type: application/json

```
{
  "totalset" : 2,
  "sets" : [ {
    "setnum" : 1,
    "totalpins" : 3,
    "pins" : [ {
      "pinnum" : 1,
      "direction" : "in",
      "value" : 1
    }, {
      "pinnum" : 2,
      "direction" : "in",
      "value" : 0
    }, {
      "pinnum" : 3,
      "direction" : "in",
      "value" : 0
    } ]
  } ]
}
```



```

}, {
  "setnum" : 2,
  "totalpins" : 2,
  "pins" : [ {
    "pinnum" : 1,
    "direction" : "out",
    "value" : 1
  }, {
    "pinnum" : 2,
    "direction" : "out",
    "value" : 0
  } ]
} ]
}

```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

Successful response with current GPIO settings. [GPIOSettings](#)

400

Invalid request body format

500

An error occurred while retrieving GPIO data. [Err](#)

GET /pet/gpio/{set}

Get the direction and value of GPIO pins in a set ([petGpioSetGet](#))

Path parameters

set (required)

Path Parameter – The set of GPIO pins

Return type

[RspBodyGPIOAll](#)



Example data

Content-Type: application/json

```
{
  "pins" : [ {
    "pinnum" : 1,
    "direction" : "in",
    "value" : 1
  }, {
    "pinnum" : 2,
    "direction" : "out",
    "value" : 0
  }, {
    "pinnum" : 3,
    "direction" : "in",
    "value" : 1
  } ]
}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

OK [RspBodyGPIOAll](#)

500

An error occurred while retrieving GPIO data. [Err](#)

GET /pet/gpio/{set}/{pin}

Get the direction of a GPIO pin in a set (**petGpioSetPinGet**)

Path parameters

set (required)

Path Parameter – The set of GPIO pins

pin (required)



Path Parameter – The GPIO pin number

Return type

[PinOne](#)

Example data

Content-Type: application/json

```
{
  "value" : 0,
  "direction" : "out"
}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

OK [PinOne](#)

500

An error occurred while retrieving GPIO data. [Err](#)

POST /pet/gpio/{set}/{pin}

Set the direction of a GPIO pin in a set (**petGpioSetPinPost**)

Path parameters

set (required)

Path Parameter – The set of GPIO pins

pin (required)

Path Parameter – The GPIO pin number

Consumes

This API call consumes the following media types via the Content-Type request header:

- application/json



Request body

body [PinOne](#) (required)

Body Parameter –

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

The direction of the specified pin was successfully set

400

Invalid request body format

500

An error occurred while set direction. [Err](#)

POST /pet/gpio/{set}

Set the direction and/or value of some GPIO pins in a set ([petGpioSetPost](#))

Path parameters

set (required)

Path Parameter – The set of GPIO pins

Consumes

This API call consumes the following media types via the Content-Type request header:

- application/json

Request body

body [gpio_set_body](#)
(required)

Body Parameter –

Produces

This API call produces the following media types according to the Accept



request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

The direction of the specified pin was successfully set

400

Invalid request body format

500

An error occurred while set direction. [Err](#)

HardwareMonitor

GET /pet/hwm/fan

Returns fan pwm. (petHwmFanGet)

Return type

[RspBodyFanAll](#)

Example data

Content-Type: application/json

```
{
  "system2" : 1.4658129805029452,
  "system" : 6.027456183070403,
  "aux2" : 5.637376656633329,
  "psu2" : 7.061401241503109,
  "aux" : 5.962133916683182,
  "cpu" : 0.8008281904610115,
  "psu1" : 2.3021358869347655
}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json



Responses

200

Fan PWM retrieved successfully. [RspBodyFanAll](#)

500

An error occurred while retrieving fan pwm. [Err](#)

GET /pet/hwm/fan/speed/{idx}

Get the pwm of the specified fan (petHwmFanSpeedIdxGet)

Path parameters

idx (required)

Path Parameter –

Return type

[RspBodyFan](#)

Example data

Content-Type: application/json

```
{
  "PWM" : 1234
}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

Fan data retrieved successfully. [RspBodyFan](#)

500

An error occurred while retrieving fan data. [Err](#)

GET /pet/hwm

Returns hardware monitor datas. (petHwmGet)



Return type

[RspBodyTempInfo](#)

Example data

Content-Type: application/json

```
{
  "system2" : 1.4658129805029452,
  "system" : 6.027456183070403,
  "system4" : 5.637376656633329,
  "psu2" : 7.061401241503109,
  "system3" : 5.962133916683182,
  "cpu" : 0.8008281904610115,
  "psu1" : 2.3021358869347655
}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

hardware monitor datas retrieved successfully. [RspBodyTempInfo](#)

500

An error occurred while retrieving temperature. [Err](#)

GET /pet/hwm/temperature

Returns temperature data for CPU, System, and PSU. (`petHwmTemperatureGet`)

Return type

[RspBodyTempInfo](#)

Example data

Content-Type: application/json

```
{
  "system2" : 1.4658129805029452,
```



```
"system" : 6.027456183070403,
"system4" : 5.637376656633329,
"psu2" : 7.061401241503109,
"system3" : 5.962133916683182,
"cpu" : 0.8008281904610115,
"psu1" : 2.3021358869347655
}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

Temperature data retrieved successfully. [RspBodyTempInfo](#)

500

An error occurred while retrieving temperature. [Err](#)

GET /pet/hwm/temperature/{idx}

Get the temperature of the specified device (`petHwmTemperatureIdxGet`)

Path parameters

idx (required)

Path Parameter –

Return type

[RspBodyTempData](#)

Example data

Content-Type: application/json

```
{
  "temperature" : 0.8008282
}
```

Produces

This API call produces the following media types according to the Accept



request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

Temperature data retrieved successfully. [RspBodyTempData](#)

500

An error occurred while retrieving temperature. [Err](#)

GET /pet/hwm/voltage

Returns voltage data. (petHwmVoltageGet)

Return type

[RspBodyVoltageInfo](#)

Example data

Content-Type: application/json

```
{
  "vccddr" : 6.027456183070403,
  "psuvin" : 4.145608029883936,
  "24v" : 2.3021358869347655,
  "12v" : 5.637376656633329,
  "psuvout" : 7.386281948385884,
  "vcc" : 5.962133916683182,
  "vcore" : 0.8008281904610115,
  "3p3v" : 1.4658129805029452,
  "vbat" : 7.061401241503109,
  "avsb" : 2.027123023002322,
  "3vsb" : 3.616076749251911,
  "ccm" : 9.301444243932576
}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.



- application/json

Responses

200

Voltage data retrieved successfully. [RspBodyVoltageInfo](#)

500

An error occurred while retrieving voltage data. [Err](#)

GET /pet/hwm/voltage/{idx}

Get the temperature of the specified device (`petHwmVoltageIdxGet`)

Path parameters

`idx` (required)

Path Parameter –

Return type

[RspBodyVoltage](#)

Example data

Content-Type: application/json

```
{
  "voltage" : 0.8008282
}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

Voltage data retrieved successfully. [RspBodyVoltage](#)

500

An error occurred while retrieving voltage. [Err](#)

Logs



GET /pet/log/{idx}

Get RPET logfile (petLogIdxGet)

Path parameters

idx (required)

Path Parameter – The index of logfile

Return type

byte[]

Example data

Content-Type: application/json

```
""
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- text/plain
- application/json

Responses

200

Get the logfile success. [byte\[\]](#)

404

Logfile not found.

500

An error occurred while get logfile [Err](#)

PowerControl

POST /pet/power/off

Shutdown the system (petPowerOffPost)

send shutdown command to the system

Consumes

This API call consumes the following media types via the Content-Type request



header:

- application/json

Request body

body [power_off_body](#)

(required)

Body Parameter –

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

Successful start reboot process.

500

An error occurred while start reboot process. [Err](#)

POST /pet/power/reboot

Reboot the system (**petPowerRebootPost**)

send reboot command to the system

Consumes

This API call consumes the following media types via the Content-Type request header:

- application/json

Request body

body [power_reboot_body](#)

(required)

Body Parameter –

Produces

This API call produces the following media types according to the Accept



request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

Successful start reboot process.

500

An error occurred while start reboot process. [Err](#)

SCAN

GET /scan/{scan-key}

Provide Basic platform information for user (`scanScanKeyGet`)

Path parameters

scan-key (required)

Path Parameter –

Return type

[RspBodySystemInfo](#)

Example data

Content-Type: application/json

```
{
  "hw_version" : 0,
  "serialnumber" : "serialnumber",
  "name" : "name",
  "type" : "type"
}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json



Responses

200

Basic platform information retrieved successfully. [RspBodySystemInfo](#)

500

An error occurred while retrieving Basic information [Err](#)

SMBus

GET /pet/smbus/block/{addr}/{cmd}/{count}

Send SMBus Block Read Command (`petSmbusBlockAddrCmdCountGet`)

Path parameters

addr (required)

Path Parameter – Address parameter in hexadecimal format

cmd (required)

Path Parameter – Command parameter in hexadecimal format

count (required)

Path Parameter – Number of bytes to read

Return type

[RspBodySMBusBlkRd](#)

Example data

Content-Type: application/json

```
{
  "data" : [ "data", "data" ]
}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json



Responses

200

Block read successfully [RspBodySMBusBlkRd](#)

500

An error occurred while send SMBus byte read Command. [Err](#)

POST /pet/smbus/block/{addr}/{cmd}/{count}

Send SMBus Block Write Command (`petSmbusBlockAddrCmdCountPost`)

Path parameters

addr (required)

Path Parameter – Address parameter in hexadecimal format

cmd (required)

Path Parameter – Command parameter in hexadecimal format

count (required)

Path Parameter – Number of bytes to write

Consumes

This API call consumes the following media types via the Content-Type request header:

- application/json

Request body

body [cmd_count_body](#)
(required)

Body Parameter –

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json



Responses

200

Block write successfully

500

An error occurred while send SMBus byte write Command. [Err](#)

POST /pet/smbus/byte/{addr}/{cmd}/{data}

Send SMBus Byte Write Command (`petSmbusByteAddrCmdDataPost`)

Path parameters

addr (required)

Path Parameter – Address parameter in hexadecimal format

cmd (required)

Path Parameter – Command parameter in hexadecimal format

data (required)

Path Parameter – Data parameter in hexadecimal format

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

Byte sent successfully

500

An error occurred while send SMBus byte write Command. [Err](#)

GET /pet/smbus/byte/{addr}/{cmd}

Send SMBus Byte Read Command (`petSmbusByteAddrCmdGet`)

Path parameters

**addr (required)**

Path Parameter – Address parameter in hexadecimal format

cmd (required)

Path Parameter – Command parameter in hexadecimal format

Return type

[RspBodySMBusByteRd](#)

Example data

Content-Type: application/json

```
{
  "data" : "0x12"
}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses**200**

Byte read successfully [RspBodySMBusByteRd](#)

500

An error occurred while send SMBus byte read Command. [Err](#)

GET /pet/smbus

Get SMBus Status ([petSmbusGet](#))

Retrieve the current state of SMBus.

Return type

[RspBodySMBusSta](#)

Example data

Content-Type: application/json



```
{
  "status" : "BUSY"
}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

Successful response with current SMBus status. [RspBodySMBusSta](#)

500

An error occurred while retrieving SMBus status. [Err](#)

POST /pet/smbus/word/{addr}/{cmd}/{data}

Send SMBus Word Write Command (petSmbusWordAddrCmdDataPost)

Path parameters

addr (required)

Path Parameter – Address parameter in hexadecimal format

cmd (required)

Path Parameter – Command parameter in hexadecimal format

data (required)

Path Parameter – Data parameter in hexadecimal format

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

**200**

Word sent successfully

500

An error occurred while send SMBus word write Command. [Err](#)

GET /pet/smbus/word/{addr}/{cmd}

Send SMBus Word Read Command (`petSmbusWordAddrCmdGet`)

Path parameters

addr (required)

Path Parameter – Address parameter in hexadecimal format

cmd (required)

Path Parameter – Command parameter in hexadecimal format

Return type

[RspBodySMBusWdRd](#)

Example data

Content-Type: application/json

```
{
  "data" : "0x1234"
}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

Word read successfully [RspBodySMBusWdRd](#)

500

An error occurred while send SMBus word read Command. [Err](#)



Watchdog

GET /pet/wdt

get watchdog status (petWdtGet)

Return type

[RspBodyWDT](#)

Example data

Content-Type: application/json

```
{
  "timeunit" : "second",
  "timeout" : 0,
  "enabled" : true
}
```

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

Watchdog config and status retrieved successfully. [RspBodyWDT](#)

500

An error occurred while retrieving watchdog status. [Err](#)

POST /pet/wdt

Set timeout and enable/disable watchdog. (petWdtPost)

Consumes

This API call consumes the following media types via the Content-Type request header:

- application/json



Request body

body [pet_wdt_body](#)

(required)

Body Parameter –

Produces

This API call produces the following media types according to the Accept request header; the media type will be conveyed by the Content-Type response header.

- application/json

Responses

200

Watchdog settings updated successfully.

400

Invalid request body format.

500

An error occurred while config watchdog. [Err](#)

JSON Models

Err

path

[String](#)

example: path of request uri

code

[Integer](#)

example: 8

message

[String](#)

example: READ_ERROR

ErrArray

errors

[array\[Err\]](#)



FWStatus

file_version (optional)

[String](#)

current_version (optional)

[String](#)

file_md5 (optional)

[String](#)

status (optional)

[String](#)

Enum:

File Not Exist

File Failure

Old Version File

The Same Version File

Read for update

Rollback

Updating

Wait for Reboot

Error

GPIOSettings

All of the GPIO pin sets with their current settings.

totalset (optional)

[Integer](#) Number of GPIO sets.

sets (optional)

[array\[Set\]](#)

PinAll

pinnum (optional)

[Integer](#) The GPIO pin number.

example: 1

direction (optional)

[String](#) The current direction of the



GPIO pin ("in" or
"out").

example: in

value (optional)

[*Integer*](#) The current value of the GPIO
pin (0 or 1).

example: 0

PinOne

direction (optional)

[*String*](#) The current direction of the
GPIO pin ("in" or
"out").

example: out

value (optional)

[*Integer*](#) The current value of the GPIO
pin (0 or 1).

example: 0

Set

A set of GPIO pins.

setnum (optional)

[*Integer*](#) The GPIO set number.

totalpins (optional)

[*Integer*](#) Number of GPIO pins in this
set.

pins (optional)

[*array\[PinAll\]*](#)

api_login_body

username (optional)

[*String*](#) Username in remote pet config
file

example: admin



voltage (optional)

[Float](#) format: float

RspBodyFanAll

cpu (optional)

[BigDecimal](#)

system (optional)

[BigDecimal](#)

system2 (optional)

[BigDecimal](#)

aux (optional)

[BigDecimal](#)

aux2 (optional)

[BigDecimal](#)

psu1 (optional)

[BigDecimal](#)

psu2 (optional)

[BigDecimal](#)

RspBodyFan

PWM (optional)

[Integer](#)

example: 1234

RspBodyGPIOAll

GPIO pins in the set.

pins (optional)

[array\[PinALL\]](#)

RspBodyWDT

timeout (optional)

[Integer](#) Timeout value. If no trigger



is received within this time, the system will be restarted.

timeunit (optional)

[*String*](#) Time unit of timeout value.

Enum:

second

minute

enabled

[*Boolean*](#) Specifies whether the watchdog is currently enabled or not.

RspBodySMBusSta

status

[*String*](#)

example: BUSY

RspBodySMBusByteRd

data (optional)

[*String*](#)

example: 0x12

RspBodySMBusWdRd

data (optional)

[*String*](#)

example: 0x1234

RspBodySMBusBlkRd

data (optional)

[*array\[String\]*](#)

RspBodyFWFile

bios (optional)



[FWStatus](#)

ec (optional)

[FWStatus](#)

RspBodyPerm

username (optional)

[String](#) The username of new user.

example: testuser

permission (optional)

[String](#) permission for new user.

example: ["admin", "read", "write"]

RspBodyApiVer

api_version (optional)

[String](#)

RspBodyBrdInfo

bios_version

[String](#)

bios_vendor

[String](#)

ec_version (optional)

[String](#)

name

[String](#)

serialnumber (optional)

[String](#)

type (optional)

[String](#)

hw_version (optional)

[String](#)

RspBodyBiosInfo



bios_version

[*String*](#)

bios_vendor

[*String*](#)

RspBodyECInfo

ec_version

[*String*](#)

RspBodyTempInfo

cpu

[*BigDecimal*](#)

system (optional)

[*BigDecimal*](#)

system2 (optional)

[*BigDecimal*](#)

system3 (optional)

[*BigDecimal*](#)

system4 (optional)

[*BigDecimal*](#)

psu1 (optional)

[*BigDecimal*](#)

psu2 (optional)

[*BigDecimal*](#)

RspBodyTempData

temperature (optional)

[*Float*](#) format: float

RspBodyVoltageInfo

vcore

[*BigDecimal*](#)



vccddr

[*BigDecimal*](#)

3p3v (optional)

[*BigDecimal*](#)

vcc (optional)

[*BigDecimal*](#)

12v (optional)

[*BigDecimal*](#)

24v (optional)

[*BigDecimal*](#)

vbat (optional)

[*BigDecimal*](#)

ccm (optional)

[*BigDecimal*](#)

3vsb (optional)

[*BigDecimal*](#)

avsb (optional)

[*BigDecimal*](#)

psuvin (optional)

[*BigDecimal*](#)

psuvout (optional)

[*BigDecimal*](#)

offset_length_body

data (optional)

[*array\[String\]*](#)

offset_length_body_1

data (optional)

[*array\[String\]*](#)

permission_add_body

username (optional)



String The username of new user.

example: testuser

password (optional)

String password for new user.

example: p@ssw0rd

permission (optional)

String permission for new user.

example: ["admin", "read", "write"]

permission_delete_body

username (optional)

String The username to be deleted

example: testuser

permission_update_body

username (optional)

String The username of target user.

example: testuser

password (optional)

String new password for target user.

example: p@ssw0rd

permission (optional)

String new permission for target user.

example: ["admin", "read", "write"]

pet_wdt_body

timeout (optional)

Integer Timeout value. If no trigger is received within this time, the system will be restarted.

example: 30

timeunit (optional)

String Time unit of timeout value.

Enum:



second
minute

enabled (optional)

[*Boolean*](#) Enable or disable watchdog. If disabled, the watchdog will be completely disabled and will not reboot the system.

example: true

power_off_body

timer (optional)

[*Integer*](#) Specifies the time, in seconds, before initiating a shutdown. Must be 3~60 seconds.

example: 30

power_reboot_body

timer (optional)

[*Integer*](#) Specifies the time, in seconds, before initiating a reboot. Must be 3~60 seconds.

example: 30

update_bios_body

updatedmi (optional)

[*Boolean*](#)

doreboot (optional)

[*Boolean*](#)

upload_type_body

binfile (optional)

[*byte\[\]*](#) format: binary

version (optional)



String The version of the firmware

md5sum (optional)

String MD5 checksum of the firmware
binary



Troubleshooting

Unable to Connect

- **Ensure Server is Running:** Verify that the RPET server is running and accessible from your network.
- **Correct IP and Port:** Double-check that you are using the correct IP address and port number.
- **Firewall Settings:** Check your firewall settings to ensure they are not blocking the connection.
- **SSL Certificate Warnings:** The server's default SSL certificate is self-signed. If the certificate is not trusted, the client-side PWApp.html will be unable to connect. Follow these steps to resolve the issue:
 1. Open your web browser.
 2. Navigate to `https://<server-ip>:<server-port>/index.html`.
 3. When prompted, choose to trust the self-signed certificate.

By following these steps, you should be able to bypass the SSL warnings and ensure that PWApp.html can connect to the RPET Management UI server.