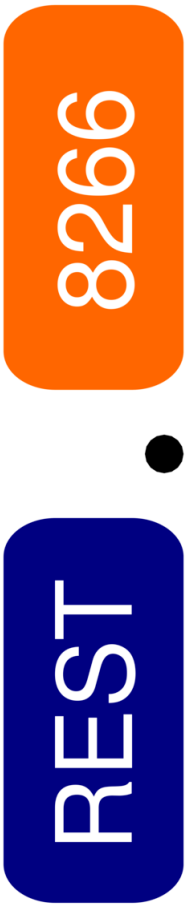
API Documentation

Release 0.1.2



Abhi-1U

May 8, 2021

Contents

1

Builtin LED Methods

Note:

1.1 /led/status

1. **Operation Category :** Inbuilt LED status
2. **Description :** Returns the current status of Builtin LED of ESP board.
3. **Method :** GET
4. **Input Type :** None(No Input)
5. **Output Type :** JSON Text.
6. **Expected Error Codes :**
7. **Sample :**

</> jQuery Snippet

```
1 var settings = {
2   "url": "http://192.168.43.97/led/status",
3   "method": "GET",
4   "timeout": 0,
5 };
6 $.ajax(settings).done(function (response) {
7   console.log(response);
8 });
```

</> JSON Output

```
1 {"LED":"ON"}
```

GET

1.2 /led/config

1. **Operation Category :** Inbuilt LED configuration
2. **Description :** Toggle LED ON/OFF, Returns the current status of Builtin LED of ESP board.
3. **Method :** POST,GET,PATCH,PUT
4. **Input Type :**
   POST,PUT,PATCH: x-www-form-urlencoded, raw, none
   GET: Query Strings, none
5. **Output Type :** JSON Text,No Body (Depends on Method)
6. **Expected Error Codes :**
7. **Sample :**

</> jQuery Snippet

```
var settings = {
  "url": "http://192.168.43.97/led/config",
  "method": "POST",
  "timeout": 0,
  "headers": {
    "Content-Type": "application/x-www-form-urlencoded"
  },
  "data": {
    "led": "ON"
  }
};
$.ajax(settings).done(function (response) {
  console.log(response);
});
```

</> JSON Output

```
{"LED":"ON"}
```

GET, POST, PUT, PATCH

2

CPU Methods

ESP8266 supports Changing CPU clock Frequency, However it is not dynamic, nor very ambitious in overclock speeds.

Frequency States

1. **OC-OverClock**: 160 MHz
2. **PS-PowerSaving** : 80 Mhz

Processor

Processor Info

- **Model :** Tensilica L106(also called LX106)
- **Base Frequency :** 80 MHz
- **Max Frequency :** 160 MHz
- **Architecture :** Xtensa (Proprietary)
- **Bit Width :** 32 Bit
- **Instruction Type :** RISC
- **Endianess :** little endian

Processor

2.1 /cpu/status

1. **Operation Category :** CPU frequency status
2. **Description :** Returns the current CPU frequency (in Hz) of ESP board.
3. **Method :** GET
4. **Input Type :** None(No Input)
5. **Output Type :** JSON
6. **Expected Error Codes :**
7. **Sample :**

</> jQuery Snippet

```
1 var settings = {
2   "url": "http://192.168.43.97/cpu/status",
3   "method": "GET",
4   "timeout": 0,
5 };
6 $.ajax(settings).done(function (response) {
7   console.log(response);
8 });
```

</> JSON Output

```
1 {"freq": 80000000}
```

GET

2.2 /cpu/config

1. **Operation Category :** CPU frequency configuration
2. **Description :** Activate Over-Clock(OC) or Power-Saving(PS) state, Returns the current frequency of CPU (in Hz) of ESP board.
3. **Method :** POST,PUT,PATCH,GET
4. **Input Type :**
   POST,PUT,PATCH: x-www-form-urlencoded, raw, none
   GET : Query Strings, none
5. **Output Type :** JSON Text.
6. **Expected Error Codes :**
7. **Sample :**

</> jQuery Snippet

```jquery
var settings = {
  "url": "http://192.168.43.97/cpu/config",
  "method": "POST",
  "timeout": 0,
  "headers": {
    "Content-Type": "application/x-www-form-urlencoded"
  },
  "data": {
    "freq": "PS"
  }
};
$.ajax(settings).done(function (response) {
  console.log(response);
});
```

</> JSON Output

```
{"freq": 160000000}
```

GET, POST, PUT, PATCH

3

Memory Methods

ESP8266EX has the following breakdown of memory

Note:

3.1 /heap/status

1. **Operation Category :** Heap Memory status
2. **Description :** Returns the allocated and free heap memory value (in Bytes) of ESP board.
3. **Method :** GET
4. **Input Type :** None(No Input)
5. **Output Type :** JSON
6. **Expected Error Codes :**
7. **Sample :**

</> jQuery Snippet

```
var settings = {
  "url": "http://192.168.43.97/heap/status",
  "method": "GET",
  "timeout": 0,
};
$.ajax(settings).done(function (response) {
  console.log(response);
});
```

</> JSON Output

```
{"allocated": 18528, "free": 19424}
```

GET

3.2 /stack/status

1. **Operation Category :** Stack Memory status
2. **Description :** Returns the total, used and free stack units (in relative Quantity) of ESP board.
3. **Method :** GET
4. **Input Type :** None(No Input)
5. **Output Type :** JSON
6. **Expected Error Codes :**
7. **Sample :**

</> jQuery Snippet

```
1 var settings = {
2   "url": "http://192.168.43.97/stack/status",
3   "method": "GET",
4   "timeout": 0,
5 };
6 $.ajax(settings).done(function (response) {
7   console.log(response);
8 });
```

</> JSON Output

```
1 {
2     "total": 8192,
3     "used": 3600,
4     "free": 4592
5 }
```

GET

3.3 /garbage/collect

1. **Operation Category :** Invoke GC status
2. **Description :** Invokes GC to collect of ESP board.
3. **Method :** GET
4. **Input Type :** None(No Input)
5. **Output Type :** JSON
6. **Expected Error Codes :**
7. **Sample :**

</> jQuery Snippet

```
1 var settings = {
2   "url": "http://192.168.43.97/garbage/collect",
3   "method": "GET",
4   "timeout": 0,
5 };
6 $.ajax(settings).done(function (response) {
7   console.log(response);
8 });
```

</> JSON Output

```
1 {
2     "OK_GC_1": "Garbage Collector Initiated Successfully"
3 }
```

GET

4

Device Methods

Note:

4.1 /device/software

1. **Operation Category :** Device Software Details
2. **Description :** Returns the software details such as nodename,sysname,version etc. of ESP board.
3. **Method :** GET
4. **Input Type :** None(No Input)
5. **Output Type :** JSON
6. **Expected Error Codes :**
7. **Sample :**

</> jQuery Snippet

```
var settings = {
  "url": "http://192.168.43.97/device/software",
  "method": "GET",
  "timeout": 0,
};
$.ajax(settings).done(function (response) {
  console.log(response);
});
```

</> JSON Output

```
{
    "nodename": "esp8266",
    "release": "2.2.0-dev(9422289)",
    "version": "v1.13 on 2020-09-11",
    "machine": "ESP module with ESP8266",
    "sysname": "esp8266"
}
```

GET

4.2 /device/hardware

1. **Operation Category :** Device Hardware Details
2. **Description :** Returns the hardware details such as firmware,Flash Size,Board etc. of ESP board.
3. **Method :** GET
4. **Input Type :** None(No Input)
5. **Output Type :** JSON
6. **Expected Error Codes :**
7. **Sample :**

</> jQuery Snippet

```
var settings = {
  "url": "http://192.168.43.97/device/hardware",
  "method": "GET",
  "timeout": 0,
};
$.ajax(settings).done(function (response) {
  console.log(response);
});
```

</> JSON Output

```
{
    "Board": "ESP8266EX",
    "Arch": "Xtensa 32bit",
    "Firmware": "MicroPython",
    "External-ROM": 4194304,
    "RAM": 98304
}
```

GET

4.3 /device/uid

1. **Operation Category :** Device Unique ID
2. **Description :** Returns the Unique Identifier String of ESP board.
3. **Method :** GET
4. **Input Type :** None(No Input)
5. **Output Type :** JSON
6. **Expected Error Codes :**
7. **Sample :**

</> jQuery Snippet

```
var settings = {
  "url": "http://192.168.43.97/device/uid",
  "method": "GET",
  "timeout": 0,
};
$.ajax(settings).done(function (response) {
  console.log(response);
});
```

</> JSON Output

```
{
    "Unique ID": "4aa85e00"
}
```

GET

4.4 /device/reset

1. **Operation Category :** Device Reset/Reboot
2. **Description :** Reboots ESP board.
3. **Method :** POST,PUT,PATCH,GET
4. **Input Type :**
   POST,PUT,PATCH: x-www-form-urlencoded, raw
   GET : Query Strings
5. **Output Type :** JSON Text.
6. **Expected Error Codes :**
7. **Sample :**

</> jQuery Snippet

```
var settings = {
  "url": "http://192.168.43.97/device/reset",
  "method": "POST",
  "timeout": 0,
  "headers": {
    "Content-Type": "application/x-www-form-urlencoded"
  },
  "data": {
    "reset": "True"
  }
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

</> JSON Output

```

```

GET, POST, PUT, PATCH

4.5 /device/webrepl

1. **Operation Category :** Device WebREPL
2. **Description :** Activates WebREPL ESP board.
3. **Method :** POST,PUT,PATCH,GET
4. **Input Type :**
   POST,PUT,PATCH: x-www-form-urlencoded, raw
   GET : Query Strings
5. **Output Type :** JSON Text.
6. **Expected Error Codes :**
7. **Sample :**

</> jQuery Snippet

```
var settings = {
  "url": "http://192.168.43.97/device/webrepl?repl=True",
  "method": "GET",
  "timeout": 0,
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

</> JSON Output

```
{
    "OK_WR_1": "WebREPL Initialized"
}
```

GET, POST, PUT, PATCH

5

Network Methods

ESP8266 Has Wifi 4 capabilities.

> **Note:**

5.1 /network/ifconfig

1. **Operation Category :** Network Details
2. **Description :** Returns the ifconfig details such as IP address,gateway,DNS etc. of ESP board.
3. **Method :** GET
4. **Input Type :** None(No Input)
5. **Output Type :** JSON
6. **Expected Error Codes :**
7. **Sample :**

</> jQuery Snippet

```
var settings = {
  "url": "http://192.168.44.97/network/ifconfig",
  "method": "GET",
  "timeout": 0,
};

$.ajax(settings).done(function (response) {
  console.log(response);
});
```

</> JSON Output

```
{
    "IP": "192.168.44.97",
    "Mask": "255.255.255.0",
    "Gateway": "192.168.44.4",
    "DNS": "192.168.44.4"
}
```

GET

5.2 /network/scan

1. **Operation Category :** Network Details
2. **Description :** Scans and returns a list of available wlan networks in the vicinity.
3. **Method :** GET
4. **Input Type :** None(No Input)
5. **Output Type :** JSON
6. **Expected Error Codes :**
7. **Sample :**

</> jQuery Snippet

```
var settings = {
  "url": "http://192.168.43.97/network/scan",
  "method": "GET",
  "timeout": 0,
};
$.ajax(settings).done(function (response) {
  console.log(response);
});
```

</> JSON Output

```
{
    "available-networks": [
    ]
}
```

GET

5.3 /network/status

1. **Operation Category :** Network Details
2. **Description :** Returns details about current network connection.
3. **Method :** GET
4. **Input Type :** None(No Input)
5. **Output Type :** JSON
6. **Expected Error Codes :**
7. **Sample :**

</> jQuery Snippet

```
var settings = {
  "url": "http://192.168.43.97/network/status",
  "method": "GET",
  "timeout": 0,
};
$.ajax(settings).done(function (response) {
  console.log(response);
});
```

</> JSON Output

```
{
    "active": true,
    "HW_protocol": "IEEE 802.11n",
    "status": "STAT_GOT_IP",
    "strength": "Fair",
    "connected": true
}
```

GET

5.4 /network/rssi

1. **Operation Category :** Network Details
2. **Description :** Returns Received Signal Strength Indicator(rssi) value for current connection .
3. **Method :** GET
4. **Input Type :** None(No Input)
5. **Output Type :** JSON
6. **Expected Error Codes :**
7. **Sample :**

</> jQuery Snippet

```
var settings = {
  "url": "http://192.168.43.97/network/rssi",
  "method": "GET",
  "timeout": 0,
};
$.ajax(settings).done(function (response) {
  console.log(response);
});
```

</> JSON Output

```
{
    "rssi": -56
}
```

GET

6

GPIO Methods

ESP8266EX has 17 GPIO pins which can be assigned to various functions by programming the appropriate registers. Each GPIO can be configured with internal pull-up or pull-down, or set to high impedance. When GPIO is configured as an input, the data are stored in software registers. The input can also be set to edge-trigger or level-trigger CPU interrupts. In short, the IO pads are bidirectional, non-inverting and tristate, which includes input and output buffer with tristate control inputs.

These pins can be multiplexed with other functions, such as I2C, I2S, UART, PWM, IR Remote Control, LED Light and Button, etc.

For low-power operations, the GPIOs can also be set to hold their state. For instance, when the chip is powered down, all output enable signals can be set to hold low.

Note:

6.1 /gpio/readadc

1. **Operation Category :** GPIO Read
2. **Description :** Returns the ADC level(0-1023) from ADC0 pin in ESP board.
3. **Method :** GET
4. **Input Type :** None(No Input)
5. **Output Type :** JSON
6. **Expected Error Codes :**
7. **Sample :**

</> jQuery Snippet

```
var settings = {
  "url": "http://192.168.43.97/gpio/readadc",
  "method": "GET",
  "timeout": 0,
};
$.ajax(settings).done(function (response) {
  console.log(response);
});
```

</> JSON Output

```
{
    "adc-level": 13
}
```

GET

7

File System Methods

MicroPython File System

Note:

7.1 /fsys/ls

1. **Operation Category :** File System Operations
2. **Description :** Returns a list of directory path.
3. **Method :** POST,PUT,PATCH,GET
4. **Input Type :**
   POST,PUT,PATCH: x-www-form-urlencoded, raw, none
   GET : Query Strings, none
5. **Output Type :** JSON Text.
6. **Expected Error Codes :**
7. **Sample :**

</> jQuery Snippet

```jquery
var settings = {
  "url": "http://192.168.43.97/fsys/ls",
  "method": "POST",
  "timeout": 0,
  "headers": {
    "Content-Type": "application/x-www-form-urlencoded"
  },
  "data": {
    "path": "/"
  }
};
$.ajax(settings).done(function (response) {
  console.log(response);
});
```

</> JSON Output

```json
{
    "/": [
        "boot.py",
        "controller",
        "main.py",
        "userver",
        "webrepl_cfg.py"
    ]
}
```

GET, POST, PUT, PATCH

7.2 /fsys/getcwd

1. **Operation Category :** File System Operations
2. **Description :** Returns the current working directory path of ESP board.
3. **Method :** GET
4. **Input Type :** None(No Input)
5. **Output Type :** JSON
6. **Expected Error Codes :**
7. **Sample :**

</> jQuery Snippet

```
var settings = {
  "url": "http://192.168.43.97/fsys/getcwd",
  "method": "GET",
  "timeout": 0,
};
$.ajax(settings).done(function (response) {
  console.log(response);
});
```

</> JSON Output

```
{
    "cwd": "/"
}
```

GET

7.3 /fsys/chdir

1. **Operation Category :** File System Operations
2. **Description :** Changes current working directory
3. **Method :** POST,PUT,PATCH,GET
4. **Input Type :**
   POST,PUT,PATCH: x-www-form-urlencoded, raw
   GET : Query Strings
5. **Output Type :** JSON Text.
6. **Expected Error Codes :**
7. **Sample :**

</> jQuery Snippet

```
var settings = {
  "url": "http://192.168.43.97/fsys/chdir",
  "method": "POST",
  "timeout": 0,
  "headers": {
    "Content-Type": "application/x-www-form-urlencoded"
  },
  "data": {
    "path": "/"
  }
};
$.ajax(settings).done(function (response) {
  console.log(response);
});
```

</> JSON Output

```
{
    "OK_FS_3": "Working Directory Changed"
}
```

GET, POST, PUT, PATCH

8

Clock Methods

The clock methods are re implemented with time zone based offset addition support and the time is no longer GMT time. There is support for Network Time sync using NTP protocol and this will refresh the time in the ESP8266.

> **Note:** One might face issues with default clock and RTC in ESP8266.

8.1 /clock/time

1. **Operation Category :** Clock Time
2. **Description :** Returns the Current Time on ESP board.
3. **Method :** GET
4. **Input Type :** None(No Input)
5. **Output Type :** JSON
6. **Expected Error Codes :**
7. **Sample :**

</> jQuery Snippet

```
var settings = {
  "url": "http://192.168.43.97/clock/time",
  "method": "GET",
  "timeout": 0,
};
$.ajax(settings).done(function (response) {
  console.log(response);
});
```

</> JSON Output

```
{
    "TimeZone": "+5:30",
    "Day Of Month": 15,
    "Year": 2021,
    "Month": "March",
    "Hour": 22,
    "Minute": 49,
    "Second": 20,
    "WeekDay": "Monday",
    "Year-Day": 74
}
```

GET

8.2 /clock/ntpsync

1. **Operation Category :** Clock Time
2. **Description :** Syncs NTP Time and Returns the Current Time on ESP board.
3. **Method :** GET
4. **Input Type :** None(No Input)
5. **Output Type :** JSON
6. **Expected Error Codes :**
7. **Sample :**

</> jQuery Snippet

```
var settings = {
  "url": "http://192.168.43.97/clock/ntpsync",
  "method": "GET",
  "timeout": 0,
};
$.ajax(settings).done(function (response) {
  console.log(response);
});
```

</> JSON Output

```
{
    "TimeZone": "+5:30",
    "Day Of Month": 15,
    "Year": 2021,
    "Month": "March",
    "Hour": 22,
    "Minute": 49,
    "Second": 20,
    "WeekDay": "Monday",
    "Year-Day": 74
}
```

GET

9 Error Codes

REST.8266 Supports Various Error Codes, these will act as a Negative feedback from the micro-controller to the end user or API client. Some of these will also act as a safety valve and prevent user from the hassle of working out the reason for no or empty response.

</> Response Schema

```
{
    "<Error_code>":"<description>"
}
```

Response code is made of 3 components:

- **Error Identifier :** This will distinguish the Error code from Success codes.
- **Operation Category/Relation :** This will help in relating the Error to any specific method or operation type.
- **Index :** This will be helpful for keeping track of multiple Error codes for one category.

Note: Error Codes are limited to very few API Methods.

9.1 ERR_ME_1

1. **Operation Category :** HTTP Method
2. **Description :** Wrong or unsupported HTTP method call.
3. **Method :** POST,GET,PATCH,PUT
4. **Output Type :** JSON Text,No Body (Depends on Method)

Error

9.2 ERR_FS_1

1. **Operation Category :** File System
2. **Description :** File Path Not Valid.
3. **Method :** POST,GET,PATCH,PUT
4. **Output Type :** JSON Text,No Body (Depends on Method)

Error

9.3 ERR_FS_2

1. **Operation Category :** File System
2. **Description :** Access Denied.
3. **Method :** POST,GET,PATCH,PUT
4. **Output Type :** JSON Text,No Body (Depends on Method)

Warning

10 Success Codes

REST.8266 Supports Various Success Codes, these will act as a positive feedback from the micro-controller to the end user or API client.

</> Response Schema

```
{
    "<Success_Code>":"<description>"
}
```

Response code is made of 3 components:

- **Success Identifier :** This will distinguish the Success code from Error codes.
- **Operation Category/Relation :** This will help in relating the Success to any specific method or operation type.
- **Index :** This will be helpful for keeping track of multiple Success codes for one category.

Note: Success Codes are limited to very few API Methods.

10.1 OK_FS_3

1. **Operation Category :** File System
2. **Description :** Working Directory changed.
3. **Method :** POST,GET,PATCH,PUT
4. **Output Type :** JSON Text,No Body (Depends on Method)

Success

10.2 OK_GC_1

1. **Operation Category :** Garbage Collector/ Memory
2. **Description :** Garbage Collector Initiated.
3. **Method :** POST,GET,PATCH,PUT
4. **Output Type :** JSON Text,No Body (Depends on Method)

Success

10.3 OK_WR_1

1. **Operation Category :** WebREPL
2. **Description :** WebREPL Initialized.
3. **Method :** POST,GET,PATCH,PUT
4. **Output Type :** JSON Text,No Body (Depends on Method)

Success