Department of Electronic & Telecommunication Engineering (EN)
Department of Biomedical Engineering (BM)
University of Moratuwa

API Service Documentation

EN4440 - Embedded Systems Engineering

Last Modified: Sep 16, 2025

TABLE OF CONTENTS

1. Modbus Frame Introduction

Modbus RTU Request Frame Format

[Slave Address] [Function Code] [Data Field(s)] [CRC (2 bytes)]

- **Slave Address**: 1 byte → ID of the target device (1–247).
- **Function Code**: 1 byte → Defines the action (e.g., 0x03 = Read Holding Registers, 0x06 = Write Single Register, etc.).
- **Data Field**: N bytes → Depends on the function (e.g., start address/register address, number of registers, values).
- **CRC**: 2 bytes (LSB first) → Cyclic Redundancy Check for error detection.

Example: Read Holding Registers (Function 0x03)

Suppose we want to read **2 registers** starting from address 0x006B from slave 0x11.

Request Frame:

11 03 00 6B 00 02 C4 0B

Breakdown:

- 11 → Slave Address (17 decimal).
- 03 → Function Code (Read Holding Registers).
- 00 6B → Start Address = 107 decimal.
- 00 02 → Number of Registers = 2.
- C4 0B → CRC (little-endian: LSB C4, MSB 0B).

Example: Write Single Register (Function 0x06)

Suppose we want to write value 16 to address 0x0008 from slave 0x11.

Request Frame:

11 06 00 08 00 10 0B 54

Breakdown:

- 11 → Slave Address (17 decimal).
- 06 → Function Code (Write Single Register).
- 00 08 → Register Address = 8.
- 00 10 → Value = 16.
- 0B 54 → CRC (little-endian: LSB 0B, MSB 54).

Modbus RTU Response Frame Format

[Slave Address] [Function Code] [Byte Count] [Data] [CRC (2 bytes)]

Response Frame (example reply from slave):

11 03 04 02 2B 00 64 85 DB

Breakdown:

- 11 → Slave Address.
- 03 → Function Code.
- 04 → Byte Count (2 registers × 2 bytes = 4).
- 02 2B → First Register = 555 decimal.
- 00 64 → Second Register = 100 decimal.
- 85 DB → CRC.

Modbus RTU CRC Function

This is a java function that can be used to calculate the low and high bytes of a given frame.

```
public int calculateCRC(byte[] data, int length) {
        int crc = 0xFFFF;

        for (int i = 0; i < length; i++) {
            crc ^= (data[i] & 0xFF);
            for (int j = 0; j < 8; j++) {
                if ((crc & 0x0001) != 0) {
                    crc >>= 1;
                    crc ^= 0xA001;
                } else {
                    crc >>= 1;
                }
            }
        }

        return crc & 0xFFFF;
}

int crc = calculateCRC(frameWithoutCRC, length);

byte low = (byte) (crc & 0xFF);
byte high = (byte) ((crc >> 8) & 0xFF);
```

2. Read API

Description

Fetches a Modbus frame for the specified register addresses. Requires an API key in the request headers for authentication.

Endpoint

http://20.15.114.131:8080/api/inverter/read

Method

POST

Request Body

Format: JSON

```
{
  "frame": "110300000002C69B"
}
```

Response Body

Format: JSON

```
{
   "frame": "11030409040002A2870"
}
```

Notes

- If the read request is valid and successful API will respond with the requested data in a Modbus frame.
- If the request frame is invalid API will send a blank response.
- If the request frame is valid but requested information (registry, function code, etc) is invalid, a Modbus frame containing an error code will be sent.

Example

```
curl -X 'POST' \
  'http://20.15.114.131:8080/api/inverter/read' \
  -H 'accept: */*' \
  -H 'Authorization: <your api key>' \
  -H 'Content-Type: application/json' \
  -d '{"frame": "110300000002C69B"}'
```

3. Write API

Description

Writes data from the Modbus frame to the inverter. Requires an API key in the request headers for authentication.

Endpoint

http://20.15.114.131:8080/api/inverter/write

Method

POST

Request Body

Format: JSON

```
{
  "frame": "1106000800100B54"
}
```

Response Body

Format: JSON

```
{
   "frame": "1106000800100B54"
}
```

Notes

- If the write request is valid and successful, the API will respond with the same requested frame.
- If the request frame is invalid API will send a blank response.
- If the request frame is valid but requested information (registry, function code, etc) is invalid Modbus frame containing an error code will be sent.

Example

```
curl -X 'POST' \
  'http://20.15.114.131:8080/api/inverter/write' \
  -H 'accept: */*' \
  -H 'Authorization: <your api key>' \
  -H 'Content-Type: application/json' \
  -d '{"frame": "1106000800100B54"}'
```

4. Modbus Data Registers

Description

These are all the available data registers that can be accessed through APIs.

Address	English Name	Function	Gain	Unit
0	Vac1 /L1 Phase voltage	Read	10	V
1	Iac1 /L1 Phase current	Read	10	A
2	Fac1 /L1 Phase frequency	Read	100	Hz
3	Vpv1 /PV1 input voltage	Read	10	V
4	Vpv2 /PV2 input voltage	Read	10	V
5	Ipv1 /PV1 input current	Read	10	A
6	Ipv2 /PV2 input current	Read	10	A
7	Inverter internal temperature	Read	10	°C
8	Set the export power percentage	Read/Write	1	%
9	Pac L /Inverter current output power	Read	1	W

Notes

- Register **8** is writable. It controls the **export power percentage**.
- The allowed range is **0–100**.

5. Exception Codes and Meaning

Description

These are the exception codes that will be returned when invalid Modbus frames are received or when API service issues occur.

A Modbus exception response has this format:

Byte	Description
1	Slave Address
2	Function Code + 0x80 (indicates error)
3	Exception Code (e.g., 01 = Illegal Function, 02 = Illegal Data Address, etc.)
4-5	CRC (Low byte first in Modbus RTU)

Code	Meaning
01	Illegal Function (function not supported)
02	Illegal Data Address (address not valid)
03	Illegal Data Value (value out of range)
04	Slave Device Failure
05	Acknowledge (request accepted, processing delayed)
06	Slave Device Busy
08	Memory Parity Error
0A	Gateway Path Unavailable
0B	Gateway Target Device Failed to Respond

6. Resources

API Keys

EN4440 API Service Keys

7. Add Error Flag API

Description

This API allows setting a flag to determine whether the response to the next request from the EcoWatt Device to the Inverter SIM triggers an error. When the flag is set, your next read or write API call (sent from EcoWatt Device to Inverter SIM) receives a response with an error or exception based on the flag's status. Below are the error types and issues that can be queued (one at a time by setting the flag) by the API. Once the error frame is sent to the EcoWatt device, the API will return to its normal behaviour.

- Exception (Any type of exception with an exception code)
- CRC errors
- Corrupt responses
- Packet Drops
- Delays

Endpoint

http://20.15.114.131:8080/api/user/error-flag/add

Method

POST

Request Body

Format: JSON

```
{
  "errorType": "EXCEPTION",
  "exceptionCode": 1,
  "delayMs": 0
}
```

Response Body

Status: 200 OK

Error Type Enum

- EXCEPTION
- CRC_ERROR
- CORRUPT
- PACKET_DROP
- DELAY

Notes

- The exception code must be sent if the selected error type is Exception. To check possible exception codes, refer to Section 5: Exception Codes and Meaning.
- The delay (“delayMs”) must be sent if the selected error type is Delay.
- If the necessary information is not provided correctly, the API will respond with an empty frame with a bad request status.

Example

```
curl -X 'POST' \
  'http://20.15.114.131:8080/api/user/error-flag/add' \
  -H 'accept: */*' \
  -H 'Authorization: <your api key>' \
  -H 'Content-Type: application/json' \
  -d '{
  "errorType": "DELAY",
  "exceptionCode": 0,
  "delayMs": 10000
}'
```

8. Error Emulation API

Description

This API can be used to simulate possible errors that can happen during read-write API calls. Users should provide a slave address, function code and what type of error should be produced. Below are the error types and issues that can be produced by the API.

- Exception (Any type of exception with an exception code)
- CRC errors
- Corrupt responses
- Packet Drops
- Delays

Endpoint

http://20.15.114.131:8080/api/inverter/error

Method

POST

Request Body

Format: JSON

```
{
  "slaveAddress": 1,
  "functionCode": 1,
  "errorType": "EXCEPTION",
  "exceptionCode": 1,
  "delayMs": 0
}
```

Response Body

Format: JSON

```
{
   "frame": "0181018190"
}
```

Error Type Enum

- EXCEPTION
- CRC_ERROR
- CORRUPT
- PACKET_DROP
- DELAY

Notes

- No need to include the authorization header.
- The exception code must be sent if the selected error type is Exception. To check possible exception codes, refer to Section 5: Exception Codes and Meaning.
- The delay (“delayMs”) must be sent if the selected error type is Delay.
- If the necessary information is not provided correctly, the API will respond with an empty frame with a bad request status.
- In addition to the given “Add Error Flag API”, which mimics the actual behavior, you can use this (“Error Emulation API”) to easily test your error handling capabilities without setting flags that trigger errors in subsequent requests.

Example

```
curl -X 'POST' \
  'http://20.15.114.131:8080/api/inverter/error' \
  -H 'accept: */*' \
  -H 'Content-Type: application/json' \
  -d '{
  "slaveId": 1,
  "functionCode": 1,
  "errorType": "EXCEPTION",
  "exceptionCode": 1,
  "delayMs": 0,
  "dropPacket": false,
  "corruptPacket": false
}'
```