31100510

LOVATO ELECTRIC S.P.A.
24020 GORLE (BERGAMO) ITALIA
VIA DON E. MAZZA, 12
TEL. 035 4282111
E-mail info@LovatoElectric.com
Web www.LovatoElectric.com

I721GBIO_25

(GB) **ENERGY METER FOR DC ELECTRIC VEHICLE CHARGING STATIONS**
Communication manual

(I) **CONTATORE DI ENERGIA PER STAZIONI DI RICARICA VEICOLI ELETTRICI**
Manuale per la comunicazione

DMED4…

CE

INTRODUCTION
The DMED4… series energy meters support the modbus protocol in the variants RTU, ASCII and TCP and rest/API communication over http/https protocol.

MODBUS
The protocols differ mainly in the structure of the messages, although the information content is equivalent, and in some constraints which make them suitable for different communication buses.

RTU
Message structure:

Pause	Modbus node	Function	Data	CRC16	Pause
3,5 characters	1 byte	1 byte	2N bytes	2 bytes	3,5 characters

Bit timing is critical, therefore the RTU variant is suitable for serial buses (RS485).

ASCII
Message structure:

Character	Modbus node	Function	Data	CRC16	Characters
:	2 chars	2 chars	2N chars	2 chars	CR LF

The beginning and end of a message are marked by specific bytes and there are no time constraints, so the ASCII variant is suitable for buses with non-deterministic timings (for example, modems).

TCP
Message structure:

Transaction ID	Protocol ID	Length	Modbus node	Function	Data
2 bytes	00 00 (2 bytes)	2 bytes	1 byte	1 byte	2N bytes

The messages are marked by an identifier which lets the association between a specific query of the master and the relevant response of the slave, therefore the TCP variant is suitable for buses in which the sequence of messages is not guaranteed (ethernet).

MODBUS NODE NUMBER
The modbus node number identifies the device within modbus RTU and ASCII networks, while it is only an information value (therefore not significant) in modbus TCP networks. Consequently, the power analyzers behave differently when they receive a message with a modbus node value equal to 0 (broadcast).

- Modbus RTU: no response, only messages with function 0x06 are processed.
- Modbus TCP: response provided normally (node 0 is considered a non-significant value, being considered the IP address to distinguish servers).
- Modbus TCP/RTU gateway: response normally provided without propagation to the slaves on the RS485 bus.

PROTOCOL SPECIFICATIONS

- Byte and word order: big endian (high word first, high byte first), except for CRC which is a little endian (low byte first) register.
- A maximum of 120 registers can be contained in the data.
- Max connection number supported on Modbus TCP: 2.
- Supported functions:

Function	Query data content	Reply data content
0x03 (Read holding register) 0x04 (Read input register)	Address (2 bytes) Register number R (2 bytes)	Replied registers byte number (1 byte) Registers (2R bytes)
0x06 (Preset single register)	Address (2 bytes) Register (2 bytes)	Address (2 bytes) Register (2 bytes)
0x10 (Preset multiple registers)	Address (2 bytes) Register number R (2 bytes) Registers (2R bytes)	Address (2 bytes) Writen bytes number
0x11 (Slave ID)	-	Replied registers byte number (1 byte) Model code (1 byte) Firmware revision (1 byte) Hardware revision (1 byte) Parameter revision (1 byte) 0x02 Ethernet firmware revision (1 byte) 0x00 0x00

INTRODUZIONE
I contatori di energia serie DMED4… supportano il protocollo modbus nelle tre varianti RTU, ASCII e TCP e la comunicazione rest/API su protocollo http e https.

MODBUS
I tre protocolli si differenziano principalmente per la struttura dei messaggi, benché il contenuto informativo sia equivalente, e per alcuni vincoli che li rendono adatti a diversi bus di comunicazione.

RTU
Struttura messaggio:

Pausa	Nodo modbus	Funzione	Dati	CRC16	Pausa
3,5 caratteri	1 byte	1 byte	2N byte	2 byte	3,5 caratteri

La temporizzazione dei bit è fondamentale, perciò la variante RTU è adatta a bus seriali (RS485).

ASCII
Struttura messaggio:

Carattere	Nodo modbus	Funzione	Dati	LRC	Caratteri
:	2 char	2 char	2N char	2 char	CR LF

L'inizio e la fine di un messaggio sono scanditi da byte specifici e non ci sono vincoli temporali, dunque la variante ASCII è adatta a bus con tempistiche non deterministiche (ad esempio modem).

TCP
Struttura messaggio:

ID transazione	ID protocollo	Lunghezza	Nodo modbus	Funzione	Dati
2 byte	00 00 (2 byte)	2 byte	1 byte	1 byte	2N byte

I messaggi sono marcati da un identificatore che permette l'associazione tra una specifica query del master e la risposta relativa dello slave, perciò la variante TCP è adatta a bus in cui non è garantita la sequenza dei messaggi (ethernet).

NUMERO DI NODO MODBUS
Il numero di nodo modbus è identificativo del dispositivo all'interno di reti modbus RTU e ASCII, mentre è solo una valore informativo (quindi non significativo) nelle reti modbus TCP. Di conseguenza gli analizzatori di rete si comportano in modo diverso quando ricevono un messaggio con valore nodo modbus uguale a 0 (broadcast).

- Modbus RTU: nessuna risposta, vengono processati solo i messaggi con funzione 0x06.
- Modbus TCP: risposta fornita normalmente (nodo 0 è considerato un non-significant value, essendo considerato l'indirizzo IP per distinguere i server).
- Gateway modbus TCP/RTU: risposta fornita normalmente senza propagazione agli slave su bus RS485.

SPECIFICHE PROTOCOLLO

- Ordine byte e word: big endian (high word first, high byte first), tranne CRC che è un registro little endian (low byte first).
- Nei dati possono essere contenuti al massimo 120 registri.
- Massimo numero di connessioni supportate su Modbus TCP: 2.
- Funzioni supportate:

Funzione	Contenuto Dati Query	Contenuto Dati Reply
0x03 (Read holding register) 0x04 (Read input register)	Indirizzo (2 byte) Numero registri R (2 byte)	Numero byte registri restituiti (1 byte) Registri (2R byte)
0x06 (Preset single register)	Indirizzo (2 byte) Registro (2 byte)	Indirizzo (2 byte) Registro (2 byte)
0x10 (Preset multiple registers)	Indirizzo (2 byte) Numero registri R (2 byte) Registri (2R byte)	Indirizzo (2 byte) Numero byte scritti
0x11 (Slave ID)	-	Numero byte registri restituiti (1 byte) Codice modello (1 byte) Revisione firmware (1 byte) Revisione hardware (1 byte) Revisione parametri (1 byte) 0x02 Revisione firmware ethernet (1 byte) 0x00 0x00

	Model code
DMED404R1500	0x4A
DMED404R0600	0x48
DMED404R0400	0x47
DMED404R0150	0x46
DMED404D1500	0x4A
DMED404D0600	0x48
DMED404D0400	0x47
DMED404D0150	0x46
DMED404R1500	0x45
DMED403R0600	0x43
DMED403R0400	0x42
DMED403R0150	0x41
DMED404D1500	0x45
DMED403D0600	0x43
DMED403D0400	0x42
DMED403D0150	0x41
DMED4DC1	0x0C
DMED4DC2	0x0D

In the event of an error, the reply involves modifying the function code by raising the most significant bit (for example, if the error occurs with function 0x04, the function code in the response is 0x84) and the data consists only of 1 byte for the exception code:

Error code	**Description**
0x01	Function is not valid
0x02	Address is not valid
0x03	Value is out of range
0x04	Operation not valid
0x06	Slave busy

CRC COMPUTATION EXAMPLE

Frame = 0207h

CRC initialization	1111	1111	1111	1111
Load the first byte			0000	0010
Execute xor with the first Byte of the frame	1111	1111	1111	1101
Execute 1st right shift	0111	1111	1111	1110 **1**
Carry=1, load polynomial	1010	0000	0000	0001
Execute xor with the polynomial	1101	1111	1111	1111
Execute 2nd right shift	0110	1111	1111	1111 **1**
Carry=1, load polynomial	1010	0000	0000	0001
Execute xor with the polynomial	1100	1111	1111	1110
Execute 3rd right shift	0110	0111	1111	1111 **0**
Execute 4th right shift	0011	0011	1111	1111 **1**
Carry=1, load polynomial	1010	0000	0000	0001
Execute xor with the polynomial	1001	0011	1111	1110
Execute 5th right shift	0100	1001	1111	1111 **0**
Execute 6th right shift	0010	0100	1111	1111 **1**
Carry=1, load polynomial	1010	0000	0000	0001
Execute xor with the polynomial	1000	0100	1111	1110
Execute 7th right shift	0100	0010	0111	1111 **0**
Execute 8th right shift	0010	0001	0011	1111 **1**
Carry=1, load polynomial	1010	0000	0000	0001
Load the second byte of the frame			0000	0111
Execute xor with the Second byte of the frame	1000	0001	0011	1001
Execute 1st right shift	0100	0000	1001	1100 **1**
Carry=1,load polynomial	1010	0000	0000	0001
Execute xor with the polynomial	1110	0000	1001	1101
Execute 2nd right shift	0111	0000	0100	1110 **1**
Carry=1,load polynomial	1010	0000	0000	0001
Execute xor with the polynomial	1101	0000	0100	1111
Execute 3rd right shift	0110	1000	0010	0111 **1**
Carry=1,load polynomial	1010	0000	0000	0001
Execute xor with the polynomial	1100	1000	0010	0110
Execute 4th right shift	0110	0100	0001	0011 **0**
Execute 5th right shift	0010	0100	0000	1001 **1**
Carry=1,load polynomial	1010	0000	0000	0001
Execute xor with the polynomial	1001	0010	0000	1000
Execute 6th right shift	0100	1001	0000	0100 **0**
Execute 7th right shift	0010	0100	1000	0010 **0**
Execute 8th right shift	0001	0010	0100	0001 **0**
CRC Result	0001	0010	0100	0001
	0x12		**0x41**	

	Codice modello
DMED404R1500	0x4A
DMED404R0600	0x48
DMED404R0400	0x47
DMED404R0150	0x46
DMED404D1500	0x4A
DMED404D0600	0x48
DMED404D0400	0x47
DMED404D0150	0x46
DMED404R1500	0x45
DMED403R0600	0x43
DMED403R0400	0x42
DMED403R0150	0x41
DMED404D1500	0x45
DMED403D0600	0x43
DMED403D0400	0x42
DMED403D0150	0x41
DMED4DC1	0x0C
DMED4DC2	0x0D

In caso di errore, la replica prevede la modifica del codice funzione alzando il bit più significativo (ad esempio se l'errore avviene con la funzione 0x04, il codice funzione nella risposta è 0x84) e i dati sono costituiti solo da 1 byte per il codice eccezione:

Codice errore	**Descrizione**
0x01	Funzione non valida
0x02	Indirizzo non valido
0x03	Valore fuori range
0x04	Operazione non valida
0x06	Slave occupato

ESEMPIO DI CALCOLO CRC

Frame = 0x0207

Inizializzazione CRC	1111	1111	1111	1111
Carica primo byte			0000	0010
Esegue xor con il primo Byte della frame	1111	1111	1111	1101
Esegue primo shift a dx	0111	1111	1111	1110 **1**
Carry=1,carica polinomio	1010	0000	0000	0001
Esegue xor con il polinomio	1101	1111	1111	1111
Esegue secondo shift dx	0110	1111	1111	1111 **1**
Carry=1,carica polinomio	1010	0000	0000	0001
Esegue xor con il polinomio	1100	1111	1111	1110
Esegue terzo shift	0110	0111	1111	1111 **0**
Esegue quarto shift	0011	0011	1111	1111 **1**
Carry=1, carica polinomio	1010	0000	0000	0001
Esegue xor con il Polinomio	1001	0011	1111	1110
Esegue quinto shift dx	0100	1001	1111	1111 **0**
Esegue sesto shift dx	0010	0100	1111	1111 **1**
Carry=1, carica polinomio	1010	0000	0000	0001
Esegue xor con polinomio	1000	0100	1111	1110
Esegue settimo shift dx	0100	0010	0111	1111 **0**
Esegue ottavo shift dx	0010	0001	0011	1111 **1**
Carry=1, carica polinomio	1010	0000	0000	0001
Carica secondo byte della frame			0000	0111
Esegue xor con il Secondo byte della frame	1000	0001	0011	1001
Esegue primo shift dx	0100	0000	1001	1100 **1**
Carry=1, carica polinomio	1010	0000	0000	0001
Esegue xor con il polinomio	1110	0000	1001	1101
Esegue secondo shift dx	0111	0000	0100	1110 **1**
Carry=1, carica polinomio	1010	0000	0000	0001
Esegue xor con il polinomio	1101	0000	0100	1111
Esegue terzo shift dx	0110	1000	0010	0111 **1**
Carry=1, carica polinomio	1010	0000	0000	0001
Esegue xor con il polinomio	1100	1000	0010	0110
Esegue quarto shift dx	0110	0100	0001	0011 **0**
Esegue quinto shift dx	0010	0100	0000	1001 **1**
Carry=1, carica polinomio	1010	0000	0000	0001
Esegue xor con il polinomio	1001	0010	0000	1000
Esegue sesto shift dx	0100	1001	0000	0100 **0**
Esegue settimo shift dx	0010	0100	1000	0010 **0**
Esegue ottavo shift dx	0001	0010	0100	0001 **0**
Risultato CRC	0001	0010	0100	0001
	0x12		**0x41**	

LRC COMPUTATION EXAMPLE

Address	01	00000001
Function	04	00000100
Start address hi.	00	00000000
Start address lo.	00	00000000
Register number	08	00001000
	Sum	00001101
	Complement to 1	11110010
	+ 1	00000001
	Complement to 2	11110101

LRC result **F5**

ESEMPIO DI CALCOLO LRC

Indirizzo	01	00000001
Funzione	04	00000100
Start address hi.	00	00000000
Start address lo.	00	00000000
Numero registri	08	00001000
	Somma	00001101
	Complemento a 1	11110010
	+ 1	00000001
	Complemento a 2	11110101

Risultato LRC **F5**

MODBUS REGISTERS
FUNCTION 0x03 – 0x04

The modbus protocol is 1-based: subtract 1 to the register value to get the address to be used in the modbus query. Example:
"Total imported energy" query to node 8: 08 06 *00 01* 00 04 D9 50

REGISTRI MODBUS
FUNZIONI 0x03 – 0x04

Il protocollo modbus è 1-based: sottrarre 1 al valore del registro per ottenere l'indirizzo da utilizzare nella query modbus. Esempio:
Query "Energia importata totale" al nodo 8: 08 06 *00 01* 00 04 D9 50

REGISTER REGISTRO	REGISTER	REGISTRO	WORD	TYPE TIPO	R/W
0002h	Total imported energy (0.000 kWh)	Energia importata totale (0.000 kWh)	4	Unsigned long	R
0006h	Last charge cycle energy (0.000 kWh)	Consumo energia ultimo ciclo di carica (0.000 kWh)	4	Unsigned long	R
000Ah	Voltage (0.00 V)	Tensione (0.00 V)	2	Unsigned long	R
000Ch	Current (0.0000 A)	Corrente (0.0000 A)	2	Unsigned long	R
000Eh	Power (0.000 kW)	Potenza (0.000kW)	4	Signed long	R
0012h	I1 temperature (0.0°C)	Temperatura I1 (0.0°C)	1	Signed int	R
0013h	I1 temperature (0.0°F)	Temperatura I1 (0.0°F)	1	Signed int	R
0014h	I2 temperature (0.0°C)	Temperatura I2 (0.0°C)	1	Signed int	R
0015h	I2 temperature (0.0°F)	Temperatura I2 (0.0°F)	1	Signed int	R
0016h	Total exported energy (0.000 kWh)	Energia esportata totale (0.000 kWh)	4	Unsigned long	R
001Ah	Last cycle exported energy (0.000 kWh)	Energia esportata ultimo ciclo (0.000 kWh)	4	Unsigned long	R
001Eh	Total exchanged energy (0.000 kWh)	Energia scambiata totale (0.000 kWh)	4	Unsigned long	R
0030h	Total hour counter	Contaore totale	4	Unsigned long	R
0034h	Working hour counter	Contaore lavoro	4	Unsigned long	R
0200h	Voltage (0.00 V) - fast update: 100ms	Tensione (0.00 V) – aggiornamento rapido: 100ms	2	Unsigned long	R
0202h	Current (0.0000 A) - fast update: 100ms	Corrente (0.0000 A) – aggiornamento rapido: 100ms	2	Unsigned long	R
0204h	Power (0.000 kW) - fast update: 100ms	Potenza (0.000 kW) – aggiornamento rapido: 100ms	4	Signed long	R
0208h	Date-Time of the last sample	Data ora ultimo campione	2	Unsigned long	R
020Ah	Time offset	Offset di tempo	1	Unsigned int	R
020Dh	Max number of metrological events records	Numero massimo di record per eventi metrologici	1	Unsigned int	R
020Eh	Number of metrological events saved records	Numero di record per eventi metrologici salvati	1	Unsigned int	R
0220h	Max number of OCMF records	Numero massimo di record OCMF	1	Unsigned int	R
0221h	Number of OCMF saved records	Numero di record OCMF salvati	1	Unsigned int	R

Charge cycle management
Gestione ciclo di carica

ADDRESS INDIRIZZO	REGISTER	REGISTRO	WORD	TYPE TIPO	R/W
4000h	Error dword 1 (see details below)	dword errore 1 (vedere dettagli in seguito)	2	Unsigned long	R
4002h	Error dword 2 (see details below)	dword errore 2 (vedere dettagli in seguito)	2	Unsigned long	R
4004h	FW checksum	Checksum firmware	2	Unsigned long	R
4080h	Date and time synchronization	Sincronizzazione ora e data	2	Unsigned long	R/W
4200h	Measurement status (see details below)	Stato misure (vedere dettagli in seguito)	1	Unsigned int 0 = NOT INITIALIZED 1 = IDLE 2 = ACTIVE 3 = ACTIVE AFTER POWER FAILURE	R
4201h	Digital signature status (see details below)	Stato firma digitale (vedere dettagli in seguito)	1	Unsigned int	R
4202h	Output message length	Lunghezza messaggio output	1	Unsigned int	R
4203h	Binary output message length	Lunghezza messaggio output binario	1	Unsigned int	R
4204h	Digitale signature length	Lunghezza firma digitale	1	Unsigned int	R
4280h	Command word	Comandi	1	Unsigned int	R/W
4281h	Input message length	Lunghezza messaggio input	1	Unsigned int	R/W
4282h	Digital signature format	Formato firma digitale	1	Unsigned int 0 = hex 1 = base64	R
4283h	Date and time synchronization timeout	Timeout sincronizzazione ora e data	1	Unsigned int	R
4284h	Timezone UTC DST Offset	Offset per gestione fusi orari ed ora legale/solare	1	Unsigned int	R/W
4400h	Output message JSON	Messaggio JSON di output	512	ASCII String	R
4600h	Public key raw	Chiave pubblica	32	ASCII String	R
4620h	Signature raw	Messaggio firmato	32	ASCII String	R
4640h	Signature ANS. 1 DER	Messaggio firmato	128	ASCII String	R
46C0h	Binary output message	Messaggio binario di output	512	Binary String	R
4A00h	Input message JSON binary	Messaggio JSON di input	512	ASCII String	R
4F10h	Display management Write 06h with value 00h: off Write 06h with value 01h: on	Gestione display Write 06h con valore 00h: spegnimento Write 06h con valore 01h: accensione	1	Unsigned int	W
4F11h	Display selection N (see measure list)	Selezione pagina N (vedi lista misure)	1	Unsigned int	W

Error dword 1	Meaning - Significato
Bit 0 (LSB)	ERROR_JSON_SYNTAX
Bit 1	ERROR_JSON_NOT_FULL_PACKET
Bit 2	ERROR_JSON_SEQUENCE
Bit 3	ERROR_JSON_VALUE
Bit 4…8	Reserved
Bit 9	ERROR_ELLIPTIC_CURVE_INIT,
Bit 10	ERROR_PUBKEY_INIT
Bit 11	ERROR_PRIVKEY_INIT
Bit 12	ERROR_GENERATION_KEY_PAIR
Bit 13	ERROR_ECDSA_SIGN_STRUCT_INIT
Bit 14	ERROR_SHA_FAILED
Bit 15	ERROR_ECDSA_SIGN
Bit 16	ERROR_ECDSA_VERIFY_SIGN
Bit 17…31	Reserved

Error dword 2	Meaning - Significato
Bit 0 (LSB)	ERROR_EEPROM_READ
Bit 1	ERROR_DMED_COMM
Bit 2	ERROR_DMED_FRAME
Bit 3…31	Reserved

Digital signature status	Meaning - Significato
Bit 0 (LSB)	SIGN NOT INITIALIZED
Bit 1	CORRECT JSON RECEIVED
Bit 2	SIGNATURE IN PROGRESS
Bit 3	SIGNATURE OK
Bit 4	SIGNATURE ERROR
Bit 5	TIMESTAMP SYNCRONIZED
Bit 6	METER ROLL OVER
Bit 7…14	Reserved
Bit 15 (MSB)	ERROR

If bit 6 is set, the master must consider that the maximum value which can be displayed on the energy meter has been exceeded. For example, supposing to have at the beginning of the transaction the counter of DMED4… *at 99999990.5kWh and at the end 00000023.7kWh, in the JSON file 100000023.7kWh is indicated as the end value, but bit 6 is set. At the start of the next transaction, 00000023.7kWh is indicated in the JSON file as the counter value and bit 6 is reset.*

Se il bit 6 viene attivato, il master deve considerare che è stato superato il valore massimo visualizzabile sul contatore di energia. Ad esempio, supponendo di avere all'inizio della transazione il contatore del DMED4… a 99999990.5kWh e alla fine 00000023.7kWh, nel file JSON viene indicato come valore di fine 100000023.7kWh, ma il bit 6 viene attivato. All'inizio della successiva transazione, nel file JSON viene indicato come valore del contatore 00000023.7kWh e il bit 6 viene resettato.

PARAMETERS SETUP
FUNCTION 0x06 – 0x10
The parameters are read and modified according to the following rules.

IMPOSTAZIONE PARAMETRI
FUNZIONI 0x06 – 0x10
I parametri vengono letti e modificati applicando la seguente regola.

Address ***Indirizzo***	**Word**	**Meaning** ***Significato***	**Function** ***Funzione***	**Example** ***Esempio***
0x5000	1	Menu number selection *Selezione numero menu*	0x04 read 0x06 write	Write value 1 to select the menu number 1 *Per selezionare il menu 1 scrivere il valore 1*
0x5001	1	Submenu number selection *Selezione numero sottomenu*	0x04 read 0x06 write	Write value 4 to select the submenu number 4. If the submenu number is not required, write 0. *Per selezionare il sottomenu 4 scrivere il valore 4. Se il sottomenu non è presente, scrivere 0.*
0x5002	1	Parameter number selection *Selezione numero parametro*	0x04 read 0x06 write	Write value 2 to select the parameter number 2 *Per selezionare il parametro 2 scrivere il valore 2*
0x5004	1…28	Parameter value Valore parametro	0x04 read 0x06 write 0x10 multi-write	
0x2F03	1	Save to flash memory Salvataggio in memoria	0x06 write	Value=5 *Valore=5*

Example: setting for menu M02 – OBIS Mode, P02.01
Menu 02: 01 06 4F FF 00 02 2E EF
Submenu: not necessary
Parameter P02.01 (OBIS Mode): 01 06 50 01 00 01 08 CA
Parameter value (ON): 01 06 50 03 00 01 A9 0A
Save
01 06 2F 02 00 05 E0 DD
The device saves and reboots (no response modbus protocol message will be received).

Esempio: impostazione menu M02 – Modalità OBIS, P02.01
Menu 02: 01 06 4F FF 00 02 2E EF
Sottomenu: non necessario
Parametro P02.01 (Modalità OBIS): 01 06 50 01 00 01 08 CA
Valore parametro (ON): 01 06 50 03 00 01 A9 0A
Salvataggio
01 06 2F 02 00 05 E0 DD
Il dispositivo effettua il salvataggio dei parametri ed esegue il reboot (non si riceve nessuna risposta da modbus).

API COMMUNICATION OVER HTTP/HTTPS PROTOCOL

Default IP (static): 192.168.1.1
Port: 80 when TLS is OFF, 443 when TLS is ON
URI (with default IP address): http://192.168.1.1:80/
URI (with default IP address and TLS ON): https://192.168.1.1:443/
The IP address can be changed using buttons, API or Web page.
The Web page is available only if TLS is OFF.

COMUNICAZIONE API SU PROTOCOLLO HTTP/HTTPS

Default IP (statico): 192.168.1.1
Porta: 80 con TLS OFF, 443 con TLS ON
URI (con IP address di default): http://192.168.1.1:80/
URI (con IP address di defaulr e TLS ON): https://192.168.1.1:443/
L'indirizzo IP può essere modificato con la tastiera frontale, via API oppure dalla pagina WEB.
La pagina WEB è disponibile solo se TLS è OFF.

REQUEST HEADER

GET request

```
GET <PATH> HTTP/1.1
Host: <DMED4 IP>
```

POST request

```
POST <PATH> HTTP/1.1
Host: <DMED4 IP>
Content-Type: application/json
Content-Length: <calculated when request is sent>
<BODY>
```

PUT request

```
POST <PATH> HTTP/1.1
Host: <DMED4 IP>
Content-Length: <calculated when request is sent>
```

RESPONSE HEADER

Success case

```
HTTP/1.1 200 OK
Connection: close
Content-Type: application/json
Transfer-Encoding: chunked
```

Failing case

```
HTTP/1.1 <ERROR CODE> <STATUS>
Connection: close
```

Where error code and status can be, for example, “400 Bad request” or “501 Not Implemented”

CHUNCKED TRANSFER
Typical http chuncked response is with max block size of 256 bytes, with length indicated at beginning of the data in exadecimal format without “0x” prefix. The last block is identified with

CHUNCKED TRANSFER
La tipica risposta http chunked è con dimensione massima del blocco di 256 byte, con lunghezza indicata all'inizio dei dati in formato esadecimale senza prefisso "0x". L'ultimo blocco è identificato con

```
<REMAINING_LENGTH_IN_HEXADECIMAL_FORMAT>
<LAST_BODY_CHUNK>
0
```

v1/measure
To request the measures (data are refreshed every 1 second).
The URI used to get the data is (with default IP address) **http://192.168.1.1/v1/measure** and the only supported method is GET.
The JSON response body is organized as follow:

v1/measure
Per richiedere le misure (i dati vengono aggiornati ogni 1 secondo).
L'URI utilizzato per ottenere i dati è (con indirizzo IP predefinito) **http://192.168.1.1/v1/measure** e l'unico metodo supportato è GET.
Il corpo della risposta JSON è organizzato come segue:

```
{
“voltage”: number,
“current”: number,
“power”: number,
“energy”: number,
“energy_neg”: number,
“temperature_i1”: number,
“temperature_i1_fahrenheit”: number,
“temperature_i2”: number,
“temperature_i2_fahrenheit”: number,
“energyPos”: number,
“energyNeg”: number
}
```

TAG	MEASURE	MISURA	UNIT UNITA' DI MISURA
voltage	Voltage	Tensione	V/10
current	Current	Corrente	A/10
power	Power	Potenza	kW/10
energy	Last charge cycle energy	Consumo energia ultimo ciclo di carica	Wh
energy_neg	Last cycle exported energy	Esportazione energia ultimo ciclo	Wh
temperature_i1	I1 Temperature	Temperatura I1	°C/10
temperature_i1_fahrenheit	I1 Temperature	Temperatura I1	°F/10
temperature_i2	I2 Temperature	Temperatura I2	°C/10
temperature_i2_fahrenheit	I2 Temperature	Temperatura I2	°F/10
energyPos	Total imported energy	Energia importata totale	Wh
energyNeg	Total exported energy	Energia esportata totale	Wh

v1/charge_settings
This API allows configuration of the charge settings parameters.
Some fields are read only. Freely settable fields are “datetime”, “datetime_offset” and “command”.
Field “datetime” and “datetime_offset" can be updated only when the charging status is not “active”
The URI used is (with default IP address) **http://192.168.1.1/v1/charge_settings** and it support GET, PUT and POST method.
The JSON response body for a GET request is organized as follow:

v1/charge_settings
Questa API consente la configurazione dei parametri per il ciclo di carica.
Alcuni campi sono di sola lettura. I campi liberamente impostabili sono "datetime", "datetime_offset" e "command". I campi "datetime" e "datetime_offset" possono essere aggiornati solo quando lo stato di carica non è "attivo".
L'URI utilizzato è (con indirizzo IP predefinito) **http://192.168.1.1/v1/charge_settings** e supporta i metodi GET, PUT e POST.
Il corpo della risposta JSON per una richiesta GET è organizzato come segue:

```
{
“sync_timeout”: number,
“signature_format”: number,
“public_key”: string,
“res_cable”: number,
“datetime_offset”: number,
“datetime”: number,
“tariff”: number
}
```

The JSON response body for a POST request is organized as follow:

Il corpo della risposta JSON per una richiesta POST è organizzato come segue:

```
{
“datetime”: number,
“datetime_offset”: number,
“tariff”: number,
“command”: string
}
```

TAG	DESCRIPTION DESCRIZIONE	TIPO TYPE	U.M.	METHODS METODI	NOTES NOTE
sync_timeout	Date and time synchronization timeout Scadenza sincronizzazione data/ora	number	Minutes	GET	
signature_format	The signature format code Formato firma digitale	number		GET	“0”: “HEX” format “1”: “BASE64” format
public_key	Public key Chiave pubblica	string	String of 128 hex values	GET	
res_cable	The cable resistance compensation Resistenza cavo per compensazione	number		GET/PUT/POST	
datetime_offset	Time zone UTC DST offset	number	Minutes	GET/PUT/POST	-900…900. Step of 15 – Multipli di 15
datetime	Date and time Data e ora	number	Seconds	GET/PUT/POST	“1695970042” = GMT, 29 September 2023 06:47:22
tariff	Tariff code number Numero tariffa	number		GET/PUT/POST	
command	The command word Comando	string		PUT/POST	See command table Vedi tabella comandi

COMMAND - COMANDO	DESCRIPTION	DESCRIZIONE
B	Begin of the charging cycle. Creation of the dedicated OCMF file	Inizio ciclo di ricarica. Creazione del file OCMF
C	Intermediate read. Creation of the dedicated OCMF file	Lettura intermedia. Creazione del file OCMF
E	End of the charging cycle. Creation of the dedicated OCMF file	Fine ciclo di ricarica. Creazione del file OCMF
r	Begin and End of the charging cycle and creation of the OCMF file containing both begin and end informations.	Recupero dati inizio + fine ciclo di ricarica. Creazione del file OCMF
X	Dword error reset	Reset errori

PUT/POST requests involving “res_cable”, “tariff”, “datetime” and “datetime_offset” parameters are accepted only f the system is not in “charging” status.

Le richieste PUT/POST che coinvolgono i parametri "res_cable", "tariff", "datetime" e "datetime_offset" sono accettate solo se il sistema non è in stato "charging" .

The parameters for a PUT request can be organized as follow:

I parametri per una richiesta PUT possono essere organizzati come segue:

http://192.168.1.1/v1/charge_settings?datetime=[number]&datetime_offset=[number] &tariff=[number]&command=[string]

v1/charge_status
To request the status of the charging cycle and of the signature and the presence of any errors in the signature process.
The URI used to get the data is (with default IP address) http://192.168.1.1/v1/charge_status and the only supported method is GET.
The JSON response body is organized as follow:

v1/charge_status
Per richiedere lo stato del ciclo di ricarica e della firma e la presenza di eventuali errori nel processo di firma.
L'URI utilizzato per ottenere i dati è (con indirizzo IP predefinito) **http://192.168.1.1/v1/charge_status** e l'unico metodo supportato è GET.
Il corpo della risposta JSON è organizzato come segue:


```
{
"measurement_status": number,
"signature_status": number,
"error_status_1": number,
"error_status_2": number
}
```

TAG	DESCRIPTION DESCRIZIONE	TIPO TYPE
measurement_status	The measurement status register value Registro stato delle misure	number
signature_status	The signature status register value Registro stato firma digitale	number
error_status_1	The DWORD1 error register value Registro errori DWORD1	number
error_status_2	The DWORD2 error register value Registro errori DWORD2	number

measurement_status	Meaning
0	NOT INITIALIZED
1	IDLE
2	ACTIVE
3	ACTIVE AFTER POWER FAILURE

Digital signature status	Meaning - Significato
Bit 0 (LSB)	SIGN NOT INITIALIZED
Bit 1	CORRECT JSON RECEIVED
Bit 2	SIGNATURE IN PROGRESS
Bit 3	SIGNATURE OK
Bit 4	SIGNATURE ERROR
Bit 5	TIMESTAMP SYNCRONIZED
Bit 6	METER ROLL OVER
Bit 7…14	Reserved
Bit 15 (MSB)	ERROR

Error dword 1	Meaning - Significato
Bit 0 (LSB)	ERROR_JSON_SYNTAX
Bit 1	ERROR_JSON_NOT_FULL_PACKET
Bit 2	ERROR_JSON_SEQUENCE
Bit 3	ERROR_JSON_VALUE
Bit 4…8	Reserved
Bit 9	ERROR_ELLIPTIC_CURVE_INIT,
Bit 10	ERROR_PUBKEY_INIT
Bit 11	ERROR_PRIVKEY_INIT
Bit 12	ERROR_GENERATION_KEY_PAIR
Bit 13	ERROR_ECDSA_SIGN_STRUCT_INIT
Bit 14	ERROR_SHA_FAILED
Bit 15	ERROR_ECDSA_SIGN
Bit 16	ERROR_ECDSA_VERIFY_SIGN
Bit 17…31	Reserved

Error dword 2	Meaning - Significato
Bit 0 (LSB)	ERROR_EEPROM_READ
Bit 1	ERROR_DMED_COMM
Bit 2	ERROR_DMED_FRAME
Bit 3…31	Reserved

v1/ocmf
This API must be used to set and get OCMF file.
At the beginning of the charging cycle (before sending BEGIN command) the OCMF has to be communicated to the device to set fixed fields. After every command, DMED4xx generate an OCMF signed file that can be read using this API.
The URI used is (with default IP address) **http://192.168.1.1/v1/ocmf** and it support GET and POST method.
The JSON response body for a GET request is organized as follow:

v1/ocmf
Questa API deve essere utilizzata per impostare e ottenere il file OCMF.
All'inizio del ciclo di carica (prima di inviare il comando BEGIN) l'OCMF deve essere comunicato al dispositivo per impostare i campi fissi. Dopo ogni comando, DMED4xx genera un file firmato OCMF che può essere letto utilizzando questa API.
L'URI utilizzato è (con indirizzo IP predefinito) **http://192.168.1.1/v1/ocmf** e supporta i metodi GET e POST.
Il corpo della risposta JSON per una richiesta GET è organizzato come segue:

```
{
"OCMF_BUFF": string
}
```

OCMF_BUFF:

```
OCMF|{
"FV": string,
"GI": string,
"GS": string,
…
"ID": string,
"TT": string,
"RD": [
        {
          "TM": string,
          "TX": string,
          "RV": number,
          "RI": string,
          "RU": string,
          "RT": string,
          "EF": string,
          "ST": string
        },
        {
          "RV": number,
          "RI": string,
          "RU": string,
          "ST": string
        },
        {
          "RV": number,
          "RI": string,
          "RU": string,
          "ST": string
        },
        {
          "TM": string,
          "TX": string,
          "RV": number,
          "RI": string,
          "RU": string,
          "RT": string,
          "EF": string,
          "ST": string
        },
        {
          "RV": number,
          "RI": string,
          "RU": string,
          "ST": string
        },
        {
          "RV": number,
          "RI": string,
          "RU": string,
          "ST": string
        }
        ]
 }
 |{
"SA": string,
"SE": string,
"SD": string
```

The complete list of the JSON field is described at the following web site:
https://github.com/SAFE-eV/OCMF-Open-Charge-Metering-Format

DMED4 requires that the RD, SA, SE, SD fields are managed and placed at the bottom of the file as shown in the schema above.

The “RD” field is composed of 6 tuples, always present in the OCMF file and with this meaning:

L'elenco completo dei campi JSON è descritto sul seguante sito web:
https://github.com/SAFE-eV/OCMF-Open-Charge-Metering-Format

DMED4 richiede che I campi RD, SA, SE, SD siano gestiti e posizionati in fondo al file come indicato nello schema sopra.

Il campo “RD” è composto da 6 tuple, sempre presenti nel file OCMF e con questo significato:

- First reading tuple: total imported energy value at start of the transaction
- Second reading tuple: total exported energy value at start of the transaction
- Third reading tuple: last charge cycle energy value at start of the transaction
- Fourth reading tuple: total imported energy value at end of transaction or during the transaction, depending on the command received
- Fifth reading tuple: total exported energy value at end of transaction or during the transaction, depending on the command received
- Sixth reading tuple: last charge cycle energy value at end of transaction or during the transaction, depending on the command received.

The JSON body for a POST request is the same but without RD content and without signature data.

v1/logbook/info
To request the status of the logbook.
The URI used to get the data is (with default IP address) **http://192.168.1.1/v1/logbook/info** and the only supported method is GET.
The JSON response body is organized as follow:

- Tupla di prima lettura: valore totale dell'energia importata all'inizio della transazione
- Tupla di seconda lettura: valore totale dell'energia esportata all'inizio della transazione
- Tupla di terza lettura: valore dell'energia dell'ultimo ciclo di carica all'inizio della transazione
- Tupla di quarta lettura: valore totale dell'energia importata alla fine della transazione o durante la transazione, a seconda del comando ricevuto
- Tupla di quinta lettura: valore totale dell'energia esportata alla fine della transazione o durante la transazione, a seconda del comando ricevuto
- Tupla di sesta lettura: valore dell'energia dell'ultimo ciclo di carica alla fine della transazione o durante la transazione, a seconda del comando ricevuto.

Il corpo JSON per una richiesta POST è lo stesso, ma senza contenuto RD e senza dati di firma.

v1/logbook/info
Per richiedere lo stato del logbook.
L'URI utilizzato per ottenere i dati è (con indirizzo IP predefinito) **http://192.168.1.1/v1/logbook/info** e l'unico metodo supportato è GET.
Il corpo della risposta JSON è organizzato come segue:

```
{
"ocmf_max_number": number,
"ocmf_total_number": number,
"ocmf_last_internal_progressive": number,
"ocmf_first_internal_progressive": number,
"events_max_number": number,
"events_total_number": number,
"log_search_status": number,
"log_search_position": number
}
```

TAG	DESCRIPTION DESCRIZIONE	TIPO TYPE
ocmf_max_number	The max number of ocmf files that can be stored	number
ocmf_total_number	The total number of ocmf files stored	number
ocmf_last_internal_progressive	The last internal ocmf log progressive number	number
Ocmf_first_internal_progressive	The first internal ocmf log progressive number saved	number
events_max_number	The max number of metrological events that can be saved	number
events_total_number	The total number of metrological events recorded	number
Log_search_status	The log search status	number
Log_search_poisition	The position of last file searched by position or transaction ID	number

v1/logbook/OCMF
To request an OCMF signed saved file.
The URI used to get the data is (with default IP address) **http://192.168.1.1/v1/logbook/OCMF** and the only supported method is GET.
This API can support 3 parameters: index_type, index_reference and index_value.
These parameters can be used to identify a specific OCMF log saved file.
The parameters can assume values specified in the following tables:

v1/logbook/OCMF
Per richiedere un file salvato firmato OCMF.
L'URI utilizzato per ottenere i dati è (con indirizzo IP predefinito) **http://192.168.1.1/v1/logbook/OCMF** e l'unico metodo supportato è GET.
Questa API può supportare 3 parametri: index_type, index_reference e index_value.
Questi parametri possono essere utilizzati per identificare uno specifico file salvato del registro OCMF.
I parametri possono assumere valori specificati nelle tabelle successive:

index_type

Value **Valore**	**Meaning** **Significato**
POS	Search by position (see index_reference) Ricerca per posizione (vedere index_reference)
PROGR	Search by internal progressive Ricerca per numero progressivo
TRANS_ID	Search by transaction ID Ricerca per ID di transazione

index_reference

Value **Valore**	**Meaning** **Significato**
LAST	Last OCMF Ultimo OCMF
LAST_BUT	Search by position starting from the last OCMF. This value needs a value in "index_value" parameter. Ricerca per posizione partendo dall'ultimo OCMF. Necessario un valore in "index_value".
FIRST	Oldest OCMF saved OCMF più vecchio
FIRST_BUT	Search by position starting from the first OCMF. This value needs a value in "index_value" parameter. Ricerca per posizione partendo dal primo OCMF. Necessario un valore in "index_value".

index_value
It's a string data that identify the value (numeric or transaction ID) used as reference for the search.
Stringa che identifica il valore (numerico o di ID transazione) utilizzato come riferimento per la ricerca.

The JSON response body for a GET request is organized as follow:

Il corpo della risposta JSON per una richiesta GET è organizzato come segue:

```
{
"OCMF_LOG": string
}
```

where the OCMF string is the same of **v1/ocmf** case.

Dove la stringa OCMF + la stessa del caso **v1/ocmf**

v1/logbook/events
To request a metrological event saved.
The URI used to get the data is (with default IP address) **http://192.168.1.1/v1/logbook/events** and the only supported method is GET.
This API can support 3 parameters: index_type, index_reference and index_value.
These parameters can be used to identify a specific event log saved.
The parameters can assume values specified in the following tables:

v1/logbook/events
Per richiedere un evento metrologico salvato.
L'URI utilizzato per ottenere i dati è (con indirizzo IP predefinito) **http://192.168.1.1/v1/logbook/events** e l'unico metodo supportato è GET.
Questa API può supportare 3 parametri: index_type, index_reference e index_value.
Questi parametri possono essere utilizzati per identificare uno specifico registro eventi salvato.
I parametri possono assumere valori specificati nelle tabelle successive:

index_type

Value Valore	Meaning Significato
POS	Search by position (see index_reference) Ricerca per posizione (vedere index_reference)

index_reference

Value Valore	Meaning Significato
LAST	Last event Ultimo evento
LAST_BUT	Search by position starting from the last event. This value needs a value in “index_value” parameter. Ricerca per posizione partendo dall’ultimo evento. Necessario un valore in “index_value”.
FIRST	Oldest event saved Evento più vecchio
FIRST_BUT	Search by position starting from the first event. This value needs a value in “index_value” parameter. Ricerca per posizione partendo dal primo evento. Necessario un valore in “index_value”.

index_value
It’s a string data that identify the value (numeric or transaction ID) used as reference for the search.
Stringa che identifica il valore (numerico o di ID transazione) utilizzato come riferimento per la ricerca.

The JSON response body for a GET request is organized as follow:

Il corpo della risposta JSON per una richiesta GET è organizzato come segue:

```
{
“EVENT_LOG”: string
}
```

EVENT_LOG:

```
EVENT|{
“date”: string,
“time”: string,
“event_code”: string,
“arguments”: string
}
```

Date: yyyy-mm-dd
Time: hh:mm:ss

v1/system_status
To request the status of the system.
The URI used to get the data is (with default IP address) **http://192.168.1.1/v1/system_status** and the supported methods are GET, PUT and POST.
PUT and POST operations are possible only if the user is logged in correctly. The “logged” status can be checked looking at the field “logged” of the GET answer.

The JSON response body is organized as follow:

v1/system_status
Per richiedere lo stato del sistema.
L'URI utilizzato per ottenere i dati è (con indirizzo IP predefinito) **http://192.168.1.1/v1/system_status** e i metodi supportati sono GET, PUT e POST.
Le operazioni PUT e POST sono possibili solo se l'utente ha effettuato l'accesso correttamente. Lo stato "logged" può essere verificato guardando il campo "logged" della risposta GET.

Il corpo della risposta JSON è organizzato come segue:

```
{
“meas_fw_rev”: string,
“meas_fw_cks”: string,
“pow_fw_rev”: string,
“pow_fw_cks”: string,
“serial_number”: number,
“model_code”: number,
“operative_status”: string,
“cable_res_locked”: number,
“system_status”: number,
“board_1_status”: number,
“board_1_status_bits”: {
        “hw_test_ongoing”: number,
        “flash_err”: number,
        “mac_err”: number,
        “flash_init_err”: number,
        “udp_client_err”: number,
        “meas_not_comm_err”: number
},
“board_2_status”: number,
“board_2_status_bits”: {
       “fram_init_err”: number,
```

```
                "flash_init_err": number,
                "adconv_init_err": number,
                "display_init_err": number,
                "temp_i1_init_err": number,
                "temp_i2_init_err": number,
                "signature_err": number,
                "timer_irq_err": number,
                "fw_crc_err": number,
                "k_crc_err": number,
                "init_uncorrect_err": number,
                "temp_i1_err": number,
                "temp_i2_err": number,
                "adconv_err": number,
                "flash_err": number,
                "wiring_err": number,
                "osc_err": number,
                 "fatal_err": number,
                 "fatal_err_bits": {
                  "e_1": number,
                  "e_2": number,
                  "e_3": number,
                  "e_4": number,
                  "e_5": number,
                  "e_6": number,
                  "e_7": number,
                  "e_8": number,
                  "e_9": number,
                  "e_10": number,
                  "e_11": number,
                          "e_12": number
                 }
        },
        "logged": number
        }
```

TAG	DESCRIPTION DESCRIZIONE	TIPO TYPE	**Example/note**
meas_fw_rev	The measurement board firmware revision Revisione firmware microcontrollore principale	number	
meas_fw_cks	The measurement board firmware checksum Checksum firmware microcontrollore principale	string	The hexadecimal 4 bytes checksum value.
pow_fw_rev	The power board firmware revision Revisione firmware microcontrollore di comunicazione	number	
pow_fw_cks	The power board firmware checksum Checksum firmware microcontrollore di comunicazione	string	The hexadecimal 4 bytes checksum value.
serial_number	The device serial number Numero seriale dispositivo	number	The serial number assigned to the device
model_code	The device model code Modello dispositivo	number	0: DMED404R1500 1: DMED404R0600 2: DMED404R0400 3: DMED404R0150 4: DMED404D1500 5: DMED404D0600 6: DMED404D0400 7: DMED404D0150
operative_status	The system operative status Stato operativo	string	Ready: parameters and clock can be modified – orologio e parametri possono essere modificati Charging: clock and parameters are locked – orologio e parametri non possono essere modificati Operating: parameters are locked – I parametri non possono essere modificati.
cable_res_locked	The cable resistance loss status Stato resistenza per perdite del cavo	number	1 resistance value is locked – valore resistenza bloccata 0 otherwise – altrimenti
system_status	The system status code Stato sistema	number	Status code meaning is detailed in following table Vedere tabelle di seguito.
board_1_status	The power board error status Stato microcontrollore di comunicazione	number	See detailed table Vedere tabelle di seguito.
board_1_status_bits	The power board error flags Flag microcontrollore di comunicazione	number	See detailed table Vedere tabelle di seguito.
board_2_status	The measure board error status Stato microcontrollore di misura	number	See detailed table Vedere tabelle di seguito.
board_2_status_bits	The measure board error status Flag microcontrollore di misura	number	See detailed table Vedere tabelle di seguito.
logged	The login status Stato credenziali	number	1 if the user is logged, 0 otherwise 1 con utente accreditato, 0 altrimenti

system_status (device error status)

Value	**Meaning**
0	OK
1	MEASURE BOARD ERROR
10	COMMUNICATION (POWER) BOARD ERROR

11	SYSTEM ERROR (COMMUNICATION BOARD ERROR AND MEASURE BOARD ERROR)
22	HW TEST ON

board_1_status_bits (power board error status)

Tag	Meaning
hw_test_ongoing	HW TEST ON
flash_pattern_error	Flash memory corrupted or not hw tested
flash_init_err	Error in flash initialization procedure
udp_client_err	Unable to start the udp client
meas_not_comm_error	Unable to communicate with measure board

board_2_status_bits (measurement board error status)

Tag	Meaning
Fram_init_err	Detail of init error: error in fram initialization procedure
Flash_init_err	Detail of init error: error in flash initialization procedure
Adconv_init_err	Detail if init error: error in adconverter initialization procedure
Display_init_err	Detail of init error: error in display initialization procedure
Temp_i1_init_err	Detail of init error: error in temperature sensor i1 initialization
Temp_i2_init_err	Detail of init error: error in temperature sensor i2 initialization
Signature_err	Error in signature procedure
Timer_irq_err	Error on fw timers
Fw_crc_err	Attempt to manipulate the application. System not working
K_crc_err	Calibration constants compromised
init_uncorrect_flag	Error during initialization procedure
Temp_i1_err	Error of temperature sensor i1
Temp_i2_err	Error of temperature sensor i2
Adconv_err	Error of adconverter
Flash_err	Communication error with external flash memory
Wiring_err	Negative power revealed in a non-reverse model code
Osc_err	Fault revealed on external oscillator
hw_test_ongoing	HW TEST ON
Fatal_err	Measure board in fatal err status. Fatal err is detailed in fatal_err_bits tag
Fatal_err_bits	Fatal err flags. See detailed table

Fatal_err_bits (measurement board fatal err detail)

Tag	Meaning
e_1	fatal err 1. Flash memory compromised
e_2	Fatal err 2. Calibration constant compromised
e_3	Fatal err 3. Wiring error revealed after commissioning done
e_4	Fatal err 4. FRAM memory compromised
e_5	Fatal err 5. FRAM memory compromised in energy data position
e_6	Fatal err 6. Unmanaged operative status
e_7	Fatal err 7. Oscillators error
e_8	Fatal err 8. Internal ram failure
e_9	Fatal err 9. Application compromised
e_10	Fatal err 10. Power board not communicating
e_11	Fatal err 11. System in commissioning and log events full
e_12	Fatal err 12. Unreliable measurements

The only parameter admitted for the PUT/POST request is “operative_status” and the only accepted value is “operating”. Operative status change is accepted only if the system is not in “charging” status and the user is logged.
For the PUT request, the requested format is:
http://192.168.1.1/v1/system_status?operative_status=[string]
while the JSON body for a POST request shall be organized as follow:

```
{
“operative_status”: string
}
```

L'unico parametro ammesso per la richiesta PUT/POST è "operative_status" e l'unico valore accettato è "operating". Il cambio di stato operativo è accettato solo se il sistema non è in stato "in carica" e l'utente è loggato.
Per la richiesta PUT, il formato richiesto è:
http://192.168.1.1/v1/system_status?operative_status=[string]
mentre il corpo JSON per una richiesta POST deve essere organizzato come segue:

v1/login
To log a user to the system.
The URI used to put the password is (with default IP address) **http://192.168.1.1/v1/login** and the only supported method is PUT.
The only parameter that can be set is “password”. The requested password value is the advanced password value.
After three attempts to login with an incorrect password, there is a 15-minute window during which no more attempts to login are allowed.
When the correct password has been received, the “logged” field of v1/system_status GET answer becomes 1.

v1/system_parameters
To get/set all system parameters.
The URI used to get/set data is (with default IP address) **http://192.168.1.1/v1/system_parameters** and the supported method are GET and POST.
Parameters data are all numeric values.
Operations (both get and set) are possible only if the user is logged in correctly.
Parameters must be written using the POST method as a complete block. For the changes to take effect, the body of the message must contain all the fields.
After system parameters change, the system reboots.
The JSON response body is organized as follow:

v1/login
Per autorizzare un utente ad accedere al sistema.
L'URI utilizzato per inserire la password è (con indirizzo IP predefinito) **http://192.168.1.1/v1/login** e l'unico metodo supportato è PUT.
L'unico parametro che può essere impostato è "password". Il valore della password richiesta è il valore della password avanzata.
Dopo tre tentativi di accesso con una password errata, c'è una finestra di 15 minuti durante la quale non sono consentiti altri tentativi di accesso.
Quando è stata ricevuta la password corretta, il campo "logged" della risposta GET v1/system_status diventa 1.

v1/system_parameters
Per ottenere/impostare tutti i parametri di sistema.
L'URI utilizzato per ottenere/impostare i dati è (con indirizzo IP predefinito)
http://192.168.1.1/v1/system_parameters e i metodi supportati sono GET e POST.
I dati dei parametri sono tutti valori numerici.
Le operazioni (sia get che set) sono possibili solo se l'utente ha effettuato l'accesso correttamente.
I parametri devono essere scritti utilizzando il metodo POST come blocco completo. Affinché le modifiche abbiano effetto, il corpo del messaggio deve contenere tutti i campi.
Dopo la modifica dei parametri di sistema, il sistema si riavvia.
Il corpo della risposta JSON è organizzato come segue:

```
{
“P01_01”: number,
“P02_01”: number,
```

```
"P02_02": number,
"P02_03": number,
"P02_04": number,
"P02_05": number,
"P03_01": number,
"P03_02": number,
"P04_01": number,
"P04_02": number,
"P04_03": number,
"P04_04": number,
"P04_05": number,
"P04_06": number,
"P05_01": number,
"P05_02": number,
"P05_03": number,
"P05_04": number,
"P05_05": number,
"P05_06": number,
"P05_07": number,
"P05_08": number,
"P05_09": number,
"P05_10": number,
"P05_11": number,
"P06_01": number,
"P06_02": number
}
```

P02_01, P02_02, P02_03, P02_04, P03_01, P04_06, P05_11

0	OFF
1	ON

P02_05

0	°C
1	°F

P04_02

0	1200 bps
1	2400 bps
2	4800 bps
3	9600 bps
4	19200 bps
5	38400 bps
6	57600 bps
7	115200 bps

P04_03

0	8 bit – none
1	8 bit – odd
2	8 bit – even
3	7 bit – odd
4	7 bit - even

P04_05

0	RTU
1	ASCII

P05_05

0	Slave
1	Gateway

P05_10

0	RTU
1	TCP

P06_01

0	HEX
1	BASE64

The JSON file for POST is like the one read by the GET, but the IP address, which must be divided per bytes. For example:
P05_03 becomes:

```
{
 "P05_03_01": number,
 "P05_03_02": number,
 "P05_03_03": number,
 "P05_03_04": number
}
```

v1/force_display
To change the displayed page.
The URI used is (with default IP address and TLS NOT activated) **http://192.168.1.1/v1/force_display**

Il file JSON per POST è come quello letto da GET, a parte l'indirizzo IP, che deve essere diviso per byte.
Ad esempio:
P05_03 diventa:

```
{
 "P05_03_01": number,
 "P05_03_02": number,
 "P05_03_03": number,
 "P05_03_04": number
}
```

v1/force_display
Per cambiare la pagina visualizzata.
L'URI utilizzato è (con indirizzo IP predefinito e TLS NON attivato) http://192.168.1.1/v1/force_display e

and it support PUT and POST method.
The JSON parameters for a PUT request can be organized as follow:
http://192.168.1.1/v1/force_display?display_on_off=[number]&set_page=[number]

The JSON body for a POST request can be organized as follow:

supporta i metodi PUT e POST.
I parametri JSON per una richiesta PUT possono essere organizzati come segue:
http://192.168.1.1/v1/force_display?display_on_off=[numero]&set_page=[numero]

Il corpo JSON per una richiesta POST può essere organizzato come segue:

```
{
"display_on_off": number,
"set_page": number
}
```


TAG	DESCRIPTION DESCRIZIONE	TIPO TYPE	
Display_on_off	Set display status ON or OFF Stato display ON o OFF	number	0: display OFF 1: display ON
Set_page	Force the display to show a page Forza il display a mostrare una pagina	number	1: Total imported energy 2: Last charge cycle energy 3: Voltage 4: Current 5: Power 6: Date 7: Time 8: I1 temperature 9: I2 temperature 10: Total hour counter 11: Working hour counter 12: Total exported energy 13: Last discharge cycle energy

API with DMED4DC1
If you want to use the energy meter APIs through the DMED4DC1 ethernet port, the channel number of the remote display to which the energy meter is connected must be inserted between v1 and the specific call suffix:

v1/<n>/<api specific suffix>, with n=1, 2, 3, 4.

For example, if the energy meter is connected to channel 2, the API

v1/force_display

becomes

v1/**2**/force_display

API con DMED4DC1
Nel caso si volesse utilizzare le API del contatore di energia attraverso la porta ethernet di DMED4DC1, occorre interporre tra v1 e il suffisso della chiamata specifica il numero di canale del display remoto a cui è connesso il contatore di energia:

v1/<n>/<api specific suffix>, con n=1, 2, 3, 4.

Se ad esempio il contatore di energia è connesso al canale 2, l'API

v1/force_display

diviene

v1/**2**/force_display