OSDPETH

Connecting an OSDP reader over RS485 to Ethernet

OSDP to Ethernet Converter



Version 1

October 19, 2026

Table of Contents

1 Revision History 2
2 About the Device 3
2.1 Network Traffic Security 3
3 Connections 4
3.1 Ethernet, Power Supply and RS485 4
3.2 BIAS Resistors and IMP Termination 4
4 Commissioning 5
5 Web Interface 6
5.1 Network 6
5.2 Security 7
5.3 HTTP / MQTT 7
5.4 OSDP Protocol 7
5.5 Other Settings and Buttons 8
6 Data Communication 9
6.1 Spinel over TCP 9
6.2 Reading the Latest Event over HTTP 13
6.3 HTTP and MQTT 13
7 Diagnostics and Restoring Defaults 16
7.1 Hardware Reset 16
8 Technical Parameters 17

1 Revision History

Version	Date	Description
1	19 October 2026	Device description, connections, configuration and communication

2 About the Device

OSDPETH connects an OSDP reader on an RS485 interface to an Ethernet network. It forwards card and keypad data to a host application.

OSDPETH acts as a control panel for one reader: it polls the reader and receives card or keypad data. An Ethernet application obtains events using Spinel over TCP, HTTP POST or MQTT. Control of reader LEDs, buzzers and outputs can be added on request.

2.1 Network Traffic Security

The device is intended exclusively for operation on a trusted local area network (LAN), behind a network router or firewall. It is not intended for direct connection to the public Internet or for direct access from external networks, for example by using a public IP address or port forwarding.

Web configuration uses unencrypted HTTP. The built-in authentication provides only basic protection against inadvertent access on the local network and does not protect against attacks from public networks. For remote access, use a VPN or a comparable security mechanism.

The OSDP Secure Channel option applies to communication with the reader over RS485. It does not by itself secure the Ethernet network interface.

3 Connections

3.1 Ethernet, Power Supply and RS485

Connect Ethernet to a switch using the RJ45 connector and a standard straight-through cable. Power and RS485 connections use a removable screw terminal block.

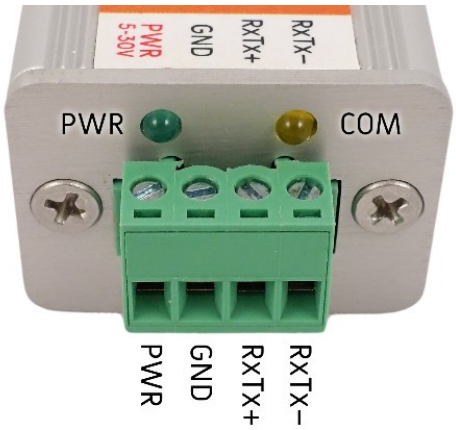


Fig. 1: power supply and RS485 terminals

Label	Meaning
PWR	Positive supply terminal, 5 to 30 V DC
GND	Negative supply terminal
RxTx+	Positive RS485 signal
RxTx-	Negative RS485 signal

Table 1: Power and RS485 connections

3.2 BIAS Resistors and IMP Termination

Internal jumpers enable the 680 Ω bias resistors (BIAS) and the 120 Ω termination resistor (IMP). The 22 kΩ bias resistors are permanently connected.



Fig. 2: BIAS and IMP jumpers inside the converter

Termination resistors are connected at opposite ends of the RS485 bus, with no more than two on one bus. Enable BIAS jumpers on one device on the bus.

4 Commissioning

Default network settings:

- IP address: **192.168.1.254**.
- Netmask: **255.255.255.0**.
- Gateway IP address: **0.0.0.0**.

Set the IP address using Papouch Device Finder:

1. Connect OSDPETH to the local network and power supply. The computer must be connected to the same LAN.
2. Start Papouch Device Finder, select the network interface connected to this LAN and click **Refresh**.
3. Identify OSDPETH by the MAC address on the device label.
4. Select **Set IP Address...** in the device menu and enter the IP address, netmask and gateway for your network. The IP address must be unique on the network.
5. Open the web interface in a browser at the configured IP address.

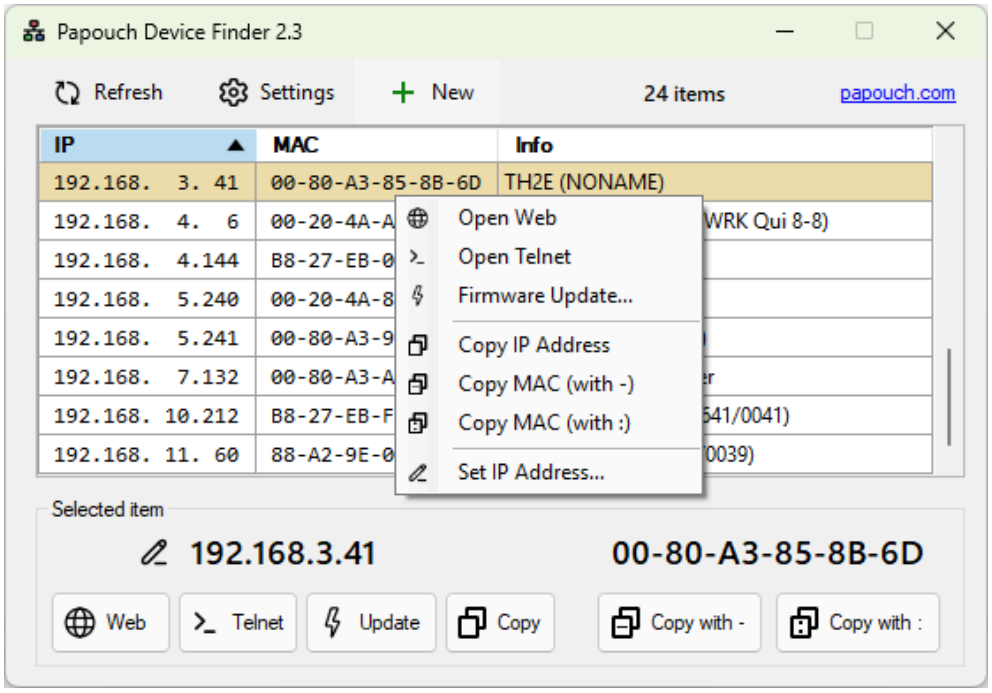


Fig. 3: Papouch Device Finder – example device menu

The illustration shows the program menu for another Papouch.com device.

The default web port is 80; the Spinel data port is 10001. No administrator password is set by default. Once a password is configured, the login username is `admin`.

Configure the reader for **38,400 Bd, 8 data bits, no parity, 1 stop bit**.

The default baud rate depends on the reader model. If it differs, configure the reader for the rate above before connecting it, following the reader manufacturer's documentation.

In the **OSDPETH** section of the web interface, set **Reader ID** to the OSDP address of the connected reader and save the settings. The addresses configured in the converter and reader must match.

5 Web Interface

The interface is available in Czech and English. It provides Network, Security, HTTP / MQTT, OSDPETH and Other sections. The top area shows device identification and the latest reader event.

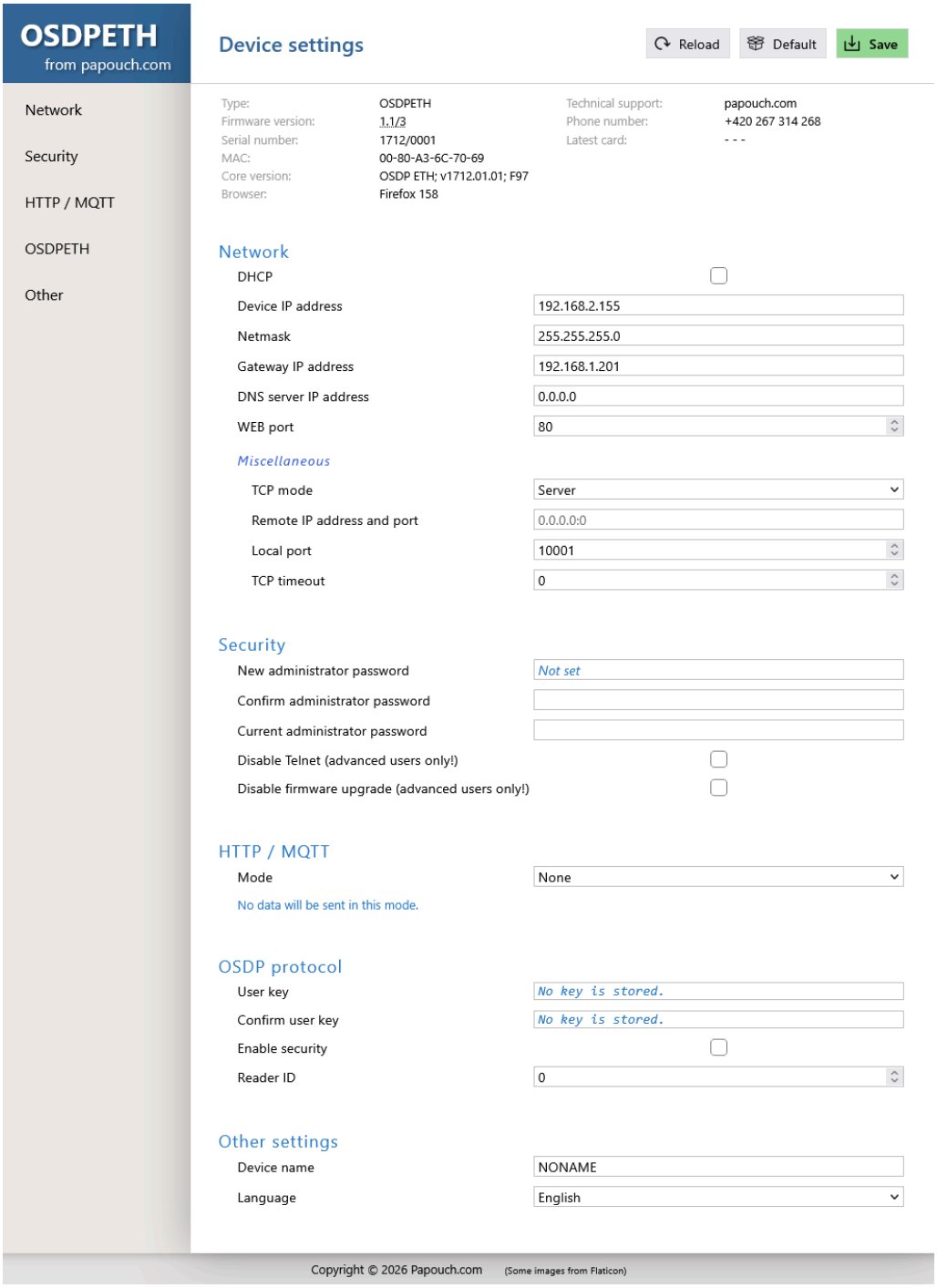


Fig. 4: OSDPETH English web interface home page

5.1 Network

Settings include DHCP, Device IP address, Netmask, Gateway IP address, DNS server IP address and WEB port. Miscellaneous settings include TCP mode Server/Client, Remote IP address and port, Local port and TCP timeout.

TCP timeout is specified in seconds. A value of 0 disables disconnection based on this timeout.

5.2 Security

The interface contains New administrator password, Confirm administrator password and Current administrator password fields. Separate options disable Telnet and firmware upgrades.

The login username is `admin`.

5.3 HTTP / MQTT

Select **None**, **HTTP** or **MQTT**. None disables delivery to a server or broker; HTTP sends events using POST, and MQTT publishes them to the selected topic. Spinel and `fresh.json` reads remain available independently of this selection.

Setting	Meaning
Server	IP address or domain name and port, for example `192.168.1.10:80` for HTTP or `192.168.1.10:1883` for MQTT. Do not include an `http://` prefix or a path.
Path / Topic	In HTTP, the receiving script path, for example `osdp/events`, up to 100 characters. In MQTT, the publishing topic, for example `papouch/osdpeth/events`, up to 50 characters.
Encryption Key	HTTP only: optional key of exactly 16 characters using a–z, A–Z and 0–9; enter it again in the confirmation field. This is a separate key, distinct from the OSDP key.
QoS	MQTT only: level 0
User name	MQTT only: broker login name, up to 15 characters
Password and Retype password	MQTT only: broker login password, up to 15 characters

Table 2: HTTP and MQTT settings

When using a domain name, configure an accessible DNS server under Network. If the target server is on another network, also configure the correct default gateway.

After changing parameters, click **Save**. Then use **Test message** to send a TEST message. Check reception on the target server or with an MQTT subscriber. Data formats and configuration examples are described under HTTP and MQTT.

5.4 OSDP Protocol

User key and **Confirm user key**: enter the same 16-byte key in both fields as exactly 32 hexadecimal characters. The interface converts letters to uppercase. A stored key is not displayed when settings are loaded; the fields indicate whether a key is stored.

The user key serves as a Master key. OSDPETH derives the SCBK for Secure Channel from this key and the reader UID. The reader must use the corresponding key and the same derivation method; configure it according to the reader manufacturer's documentation. Do not enter an already derived SCBK in these fields.

Enable security: enables OSDP Secure Channel. If no key is stored, the interface requires a valid new key when enabling security.

Reader ID: reader address from 0 to 126.

Supported reader events:

Type	Meaning	Supported length
0x50 RAW	Raw card data	1 to 64 valid bits; byte count rounded up
0x51 FRM	Formatted card data	1 to 16 bytes
0x53 KEY	Keypad data	1 to 16 bytes

Table 3: Supported reader events

5.5 Other Settings and Buttons

The device name and interface language can be configured. **Reload** reads settings again; **Save** sends settings to the device. **Default** opens a confirmation dialogue for restoring defaults.

Restoring defaults through the web preserves the device IP address but restores the web port to 80 and clears user settings including the OSDP key. For a complete reset including the IP address, use the hardware reset.

6 Data Communication

6.1 Spinel over TCP

OSDPETH uses the binary Spinel protocol, format 97. The default data port is TCP **10001**. In TCP Server mode, the host application connects to OSDPETH; in TCP Client mode, OSDPETH connects to the configured remote IP address and port. Configure the mode and ports under Network.

The Spinel Terminal application can be used to compose and send requests. The hexadecimal examples represent individual binary bytes; spaces and `|` separators are for readability and are not transmitted.

Format 97

Request:

PRE	FRM	NUM HI	NUM LO	ADR	SIG	INST	DATA	SUM	CR

Response:

PRE	FRM	NUM HI	NUM LO	ADR	SIG	ACK	DATA	SUM	CR

Field	Length [B]	Meaning
PRE	1	Prefix `0x2A`, character `*`
FRM	1	Format 97, `0x61`
NUM	2	Number of bytes from ADR through CR inclusive; high byte first, then low byte
ADR	1	Spinel address; use `0x31`, or the universal address `0xFE`. Responses contain `0x31`. This is not the reader's OSDP address.
SIG	1	Request signature, `0x00` to `0xFF`; the same value is returned in the response
INST / ACK	1	Instruction code in a request, or acknowledgement code in a response
DATA	N	Instruction-specific data; this field may be empty
SUM	1	Checksum
CR	1	End character `0x0D`

Table 4: Spinel frame fields

For N data bytes, NUM = N + 5 and the complete frame contains N + 9 bytes. Two-byte numbers in instructions and responses are transmitted high byte first unless a field description states otherwise.

Checksum: add all bytes from PRE through the last DATA byte, retain the low eight bits of the sum and subtract the result from `0xFF`. SUM and the final CR are excluded from this sum.

```
SUM = 0xFF - ((PRE + FRM + NUM_HI + NUM_LO + ADR + SIG + INST/ACK + DATA) & 0xFF)
```

The device does not respond to a frame with an incorrect checksum or end character. TCP is a byte stream: a frame may arrive in several parts, and a single read may contain multiple frames. Use NUM when parsing; `0x0D` alone does not identify the end of a frame because this byte can also occur in the data.

Example: identification request, NUM = 5, address `0x31`, signature `0x02`:

```
→  2A 61 00 05 31 02 F3 49 0D
```

Acknowledgement Codes

ACK	Meaning
`0x00`	Instruction completed; response data follows where applicable
`0x02`	Unknown instruction
`0x03`	Invalid request data length or value
`0x0C`	Automatic message containing a new reader event

Table 5: Spinel acknowledgement codes

ACK describes instruction processing. The status flag within event data has a separate meaning and does not indicate reader availability.

Instruction Overview

Instruction	Code	Category	Page
Read identification	`0xF3`	Additional	10
Read the latest event	`0xA2`	Basic	11
Read automatic message setting	`0xA4`	Configuration	11
Set automatic messages	`0xB4`	Configuration	12

Table 6: Spinel instructions

The descriptions below list INST, ACK and data fields separately. Complete frame examples use address `0x31` and signature `0x02` unless stated otherwise.

Read Identification

Returns the device type, core firmware version and supported communication format designation.

Request: `0xF3`
Response: `ACK 0x00` `string`

Legend:

Field	Length [B]	Description
string	Variable	ASCII string, for example `OSDP ETH; v1712.01.01; F97`; no terminating null byte

Example:

```
→  2A 61 00 05 31 02 F3 49 0D
```

```
←  2A 61 00 1F 31 02 00 4F 53 44 50 20 45 54 48 3B 20 76 31 37 31 32 2E 30 31 2E 30 31 3B 20
46 39 37 20 0D
```

Read the Latest Event

Reads the latest card or keypad event. After sending the response, the device marks the event as already read. A new event overwrites the previously stored value.

Request: `0xA2` `slot`
Response: `ACK 0x00` `slot` `status` `type` `representation` `bits` `bytes` `data`

Legend:

Field	Length [B]	Description
slot	1	`0x01` in both request and response; internal slot, not the reader's OSDP address
status	1	`0x00`: new valid event; `0x01`: no valid event yet, or already read using A2
type	1	`0x50`: RAW, `0x51`: FRM, `0x53`: KEY; `0x00` before the first event
representation	1	`0x01`: RAW data with a valid bit count; `0x00`: FRM or KEY byte data
bits	2	Number of valid bits, high byte first
bytes	1	Number of following data bytes N
data	N	Event data; up to 8 bytes for RAW, or 16 bytes for FRM and KEY

Table 7: Spinel event data

RAW bytes are transmitted in the order received from the reader; their meaning and conversion to a card number depend on the reader format. The RAW byte count is the bit count divided by eight, rounded up. For FRM and KEY, the bit count equals eight times the byte count. Control characters sent by the reader are included in the data.

Example: new 26-bit RAW event, four data bytes `01 02 03 40`:

```
→  2A 61 00 06 31 02 A2 01 98 0D
```

```
←  2A 61 00 10 31 02 00 01 00 50 01 00 1A 04 01 02 03 40 7B 0D
```

Example: response containing keypad input `1234` terminated by CR (`0x0D`), five bytes / 40 bits in total:

```
←  2A 61 00 11 31 02 00 01 00 53 00 00 28 05 31 32 33 34 0D D8 0D
```

Repeated reads without a new event return the retained data with status `0x01`. Before the first event, type, bits and bytes are zero and the data field is empty. The status flag cannot be used to determine the reader's RS485 communication state.

Read Automatic Message Setting

Request: `0xA4`
Response: `ACK 0x00` `enabled`

Legend:

Field	Length [B]	Description
enabled	1	`0x00`: automatic messages disabled; `0x01`: enabled

Example: automatic messages are enabled:

```
→ 2A 61 00 05 31 02 A4 98 0D
```

```
← 2A 61 00 06 31 02 00 01 3A 0D
```

Set Automatic Messages

Enables or disables individual event delivery without repeated A2 requests. The setting is stored in the device and retained after restart. Automatic messages are disabled at the factory.

Request: `0xB4` `enabled`
Response: `ACK 0x00`

Legend:

Field	Length [B]	Description
enabled	1	`0x00`: disable; `0x01`: enable

Example: enable automatic messages:

```
→ 2A 61 00 06 31 02 B4 01 86 0D
```

```
← 2A 61 00 05 31 02 00 3C 0D
```

Example: disable automatic messages:

```
→ 2A 61 00 06 31 02 B4 00 87 0D
```

```
← 2A 61 00 05 31 02 00 3C 0D
```

Automatic Messages

When automatic transmission is enabled, a new reader event generates a message with ACK `0x0C`. The message arrives over an established TCP connection without a preceding request.

Message: `ACK 0x0C` `slot` `status` `type` `representation` `bits` `bytes` `data`

The data has the same structure as an A2 response. The SIG signature increments, wrapping from `0xFF` to `0x00`. It is not copied from a request and is not a persistent event identifier across restarts.

Example: new 26-bit RAW data, signature `0x00`:

```
← 2A 61 00 10 31 00 0C 01 00 50 01 00 1A 04 01 02 03 40 71 0D
```

A new presentation of the same card produces a new automatic message containing the same card data. Automatic messages can be used alongside A2 requests and the web interface. Sending an automatic message does not itself change the latest event's read status; an A2 read changes this status. The application must distinguish ACK `0x0C` messages from responses to its own requests.

6.2 Reading the Latest Event over HTTP

A `GET /fresh.json` request returns the latest event. Example response containing 26-bit RAW card data:

```
{
  "sn": "01712/00001",
  "slot": "1",
  "status": "0",
  "type": "50",
  "bits": "26",
  "bytes": "4",
  "data": "01020340"
}
```

Field	Meaning
sn	Converter serial number; preserve the string type including leading zeros
slot	Internal reader slot, currently "1"; this is not the OSDP address
status	"0": new valid data; "1": no valid event yet or already read using A2
type	Hexadecimal event code: "50" RAW, "51" FRM, "53" KEY
bits	Valid bit count as a decimal string
bytes	Data byte count as a decimal string
data	Data as a hexadecimal string, two characters per byte

Table 8: fresh.json response fields

Reading `fresh.json` does not change the event status. After an A2 read, the JSON response retains the data and status changes to "1". Do not treat data with this status as a new event. The web interface uses the same endpoint.

The response represents the latest state, not an event list. Repeated presentations of the same card can return identical JSON, including status "0". To process individual events, including repeated card presentations, use automatic Spinel messages.

The response does not report reader communication status. A successful HTTP read confirms converter availability, not reader connectivity on RS485.

6.3 HTTP and MQTT

OSDPETH sends events to the configured server or MQTT broker. The event name `CARD` is shared by card and keypad events; the `type` field identifies the actual data type.

Event Format

```
{
  "sn": "01712/00001",
  "event": "CARD",
  "slot": "1",
  "status": "0",
  "type": "53",
  "bits": "40",
  "bytes": "5",
  "data": "313233340D"
}
```

The example contains keypad data, five bytes `31 32 33 34 0D`, or text `1234` terminated by CR. The sn, slot, status, type, bits, bytes and data fields have the same meanings as in a fresh.json response. Numeric JSON values are strings. A RAW event contains up to 8 data bytes, and FRM and KEY contain up to 16; the length of the data HEX string always equals twice the bytes value.

The **Test message** button sends a message without reader data:

```
{
  "sn": "01712/00001",
  "event": "TEST"
}
```

The receiver must distinguish TEST from CARD events. TEST checks the path to the server or broker; it does not confirm reader availability on RS485.

Sending HTTP POST

Example configuration for a server receiving data at `http://192.168.1.10/osdp/events`:

1. Select **HTTP** mode.
2. Enter `192.168.1.10:80` in **Server**.
3. Enter `osdp/events` in **Path / Topic**, without a leading slash. The device adds the slash when composing the request.
4. Leave body encryption disabled to send plain JSON. If encryption is required, configure it as described in the following section.
5. Save the settings and use **Test message** to send a TEST message.

The device uses HTTP/1.1, the POST method and the `Content-Type: application/json` header. The receiver reads JSON from the request body. After processing, return **HTTP 200 OK**; the device treats this response as successful sending. The HTTP response is not used to submit commands for reader LEDs, the buzzer or outputs.

HTTP Body Encryption

To enable encryption, enter the same 16-character key in both HTTP encryption fields and save the settings. The key consists of the 16 bytes of the entered characters, not a hexadecimal representation. The stored key is not displayed. To disable stored encryption, clear both fields and save the change.

When encryption is enabled, the standard CARD or TEST object is encrypted using AES-128 in CFB mode without padding (`aes-128-cfb`). The request body contains a JSON object with a single `crypt` field. Its string consists of a hexadecimal 16-byte initialization vector followed by the hexadecimal ciphertext.

The receiver processes the `crypt` field in this order:

1. Convert the HEX string to bytes.
2. Separate the first 16 bytes as the initialization vector.
3. Decrypt the remaining bytes using AES-CFB, the configured 16-byte key and the separated vector.
4. Process the decrypted content as a standard CARD or TEST event JSON object.

The PAPAGO AES decryption examples for PHP, Node.js and Node-RED can serve as a starting point for the receiver application. For OSDPETH, adapt the input handling: instead of the GET parameter `encrypted_data`, read JSON from the HTTP POST body and extract its `crypt` field. This field already contains a HEX string without `%` characters; after conversion to bytes, use the same vector separation and decryption. The result is a JSON object; process it using a JSON parser.

Only the outgoing HTTP request body is encrypted. This does not secure web configuration, HTTP headers, fresh.json reads or the Spinel data connection. This feature does not replace HTTPS or peer authentication.

MQTT Publishing

Example configuration for a broker at `192.168.1.10`, port 1883, topic `papouch/osdpeth/events`:

1. Select **MQTT** mode.
2. Enter `192.168.1.10:1883` in **Server**.
3. Enter `papouch/osdpeth/events` in **Path / Topic**. Do not use the `+` or `#` wildcards for publishing.
4. If the broker requires authentication, enter the username, password and password confirmation. Leave these fields empty for a broker without authentication.
5. Save the settings. In the subscriber application, connect to the same broker and subscribe to the same topic, then send a TEST message.

The payload is a standard CARD or TEST JSON object. The device publishes with **QoS 0** and **retain = 0**. The broker therefore does not acknowledge individual publications using PUBACK, and messages are not stored as retained values for new subscribers. Connect the subscriber before generating an event. This mode does not provide confirmation that every event has reached the target application.

In this mode, OSDPETH publishes data; it does not receive reader control commands from a topic. The MQTT connection is not encrypted using TLS, and the HTTP encryption key does not apply to it. For transmission over untrusted networks, use a secure network path as described under Network Traffic Security.

7 Diagnostics and Restoring Defaults

7.1 Hardware Reset

A hardware reset restores defaults including the IP address 192.168.1.254:

1. Disconnect the power supply.
2. Press and hold the RST button in the opening below and to the left of the Ethernet connector.
3. Connect the power supply and wait 10 seconds.
4. Release the button.
5. After restart, connect using the default IP address.

A reset also erases the user OSDP key and security settings. Reconfigure the network, reader and required security before returning the device to service.

8 Technical Parameters

Ethernet

Parameter	Value
Connection	RJ45 Ethernet 10/100BASE-T
Interface compatibility	Ethernet version 2.0 / IEEE 802.3

Power Supply and RS485

Parameter	Value
Supply voltage	5 to 30 V DC
Current consumption at 12 V	Typically 80 mA
Current consumption at 5 V	Typically 190 mA
Connector	Removable screw terminal block
RS485 signals	RxTx+, RxTx-
RS485 settings	38,400 Bd, 8 data bits, no parity, 1 stop bit
Galvanic isolation of RS485 port	Yes
Galvanic isolation of power supply	No
Termination	120 Ω, enabled by the IMP jumper
Bias	22 kΩ permanently connected; 680 Ω enabled by BIAS jumpers

Mechanical and Environmental Parameters

Parameter	Value
Weight	60 g
Ingress protection	IP30
Operating temperature	−25 to +75 °C
Operating humidity	0 to 90 %RH, non-condensing
Dimensions with terminal block removed	54 mm × 24 mm × 33 mm
Dimensions with terminal block connected	63 mm × 24 mm × 33 mm

Notes:

Notes:

Papouch s.r.o.

Industrial data transfer, line and protocol converters, RS232, RS485, RS422, USB, Ethernet, LTE, WiFi, measurement modules, intelligent temperature sensors, I/O modules, custom development and manufacturing.

Address: Strašnická 3164/1a, 102 00 Prague 10, Czech Republic
Phone: +420 267 314 268
Internet: www.papouch.com
E-mail: papouch@papouch.com

© 2026 Papouch s.r.o.