

Panduit ES2P PDU REST API Documentation

Version 1.0

Table of Contents

Table of Contents	2
Authentication	5
POST /api/login	5
Request Body:	5
Responses	5
Password Strength Requirements:	6
Session Management	6
Unit Parameter	7
Bulk query commands	7
Configuration and control commands	7
Examples:	7
Naming and case	7
Bulk Status Queries	8
POST /api/bulk	8
Request Body:	8
Response:	8
Supported Status Query Items	9
Device Status	9
Phase Input Status	9
Bank Status	10
Outlet Status	10
Sensor Status	10
System Information	11
TCP/IP Configuration	11
SNMP Configuration	12
Modbus Status	12
Email Configuration	12
Syslog Configuration	13
Web Protocol Status	13
IPv6 Configuration	13
Date/Time Status	14
Daylight Saving Time (DST) Status	14
User Accounts Status	14
Outlet Group Status	15
Supported Threshold Query Items	15

Device Thresholds	15
Phase Thresholds	16
Bank Thresholds	16
Outlet Thresholds	17
Sensor Thresholds	17
Configuration Management	18
POST /api/config	18
TCP/IP Configuration	18
SNMP Configuration	18
Modbus Configuration	19
HTTP/HTTPS Configuration	19
Email Configuration	19
Syslog Configuration	19
IPv6 Configuration	19
System Information Configuration	20
Threshold Configuration	20
Sensor Naming Configuration	21
Date/Time Configuration	21
Daylight Saving Time (DST) Configuration	22
User Account Management	23
Outlet Group Configuration	24
Device Control	25
POST /api/control	25
Request Body:	25
Device actions	25
Outlet operations	26
Certificate Management	27
POST /api/certificate/upload	27
Add Certificate:	27
Delete Certificate	27
Firmware Management	28
POST /api/firmware/upload	28
Notes	29
Error Codes	30
Example Usage	31
Complete Workflow	31

1. Login:.....	31
2. Query Device Status:.....	31
3. Update Configuration:.....	31
4. Control Outlet:	31
5. Upload Certificate:	31
6. Upload Firmware:	31

Authentication

POST /api/login

Authenticates a user and creates a session.

Request Body:

```
{
  "username": "string",
  "password": "string",
  "newpassword": "string", // Optional, for password change
  "confirmpassword": "string" // Required when newpassword is provided
}
```

Responses

Success

```
{
  "status": "success",
  "passwordexpired": false,
  "csrfToken": "token_string",
  "access": ["admin", "ctrl", "base", "changeOwnPass"],
  "config/sessionMgmt": {
    "signInRetriesLimited": 1,
    "numRetries": 3,
    "sessionTimeoutValue": "30 minutes",
    "lockoutTime": "10 minutes"
  }
}
```

Password Change Required:

Returned when the default password has not been changed. A session is not created in this case; csrfToken may be empty or unusable. Login again with newpassword and confirmpassword after changing the password.

```
{
  "status": "success",
  "passwordexpired": true,
  "csrfToken": "token_string",
  "access": ["changeOwnPass"]
}
```

Error - Authentication Failed:

Returned when the password authentication fails.

```
{
  "code": 401,
  "message": "Invalid User Name or Password."
}
```

Error - Confirm Password Missing or Mismatch:

When new password is provided but confirm password is missing or does not match.

```
{
  "code": 401,
  "message": "Confirm password missing or mismatch."
}
```

Error - Password Does Not Meet Requirements:

When newpassword does not meet the strength requirements:

```
{
```

```
"code": 401,  
"message": "New password does not meet requirements."  
}
```

Password Strength Requirements:

When changing password via the newpassword field, the new password must meet the following requirements:

- Minimum Length At least 8 characters
- Maximum Length Maximum 20 characters
- Number Must include at least one digit (0-9)
- Lowercase Letter Must include at least one lowercase letter (a-z)
- Uppercase Letter Must include at least one uppercase letter (A-Z)
- Special Character Must include at least one special character

Allowed special characters: ~!@#\$%^&*()_+<>?:\{"},./;'[]

Notes:

Missing or invalid session/CSRF returns HTTP 401. If the client IP is locked out, the device returns HTTP 423. Login credential errors still return HTTP 200 with code 401 in the JSON body.

If default password is not updated, user must change password on first login. There is no session until that succeeds.

When changing password, password, newpassword, and confirmpassword must all be provided newpassword and confirmpassword must be identical. If password validation fails, the current password remains unchanged.

Session Management

- All API endpoints except /api/login require a valid session cookie and CSRF header
- Session ID is provided via Set-Cookie: connect.sid=... in the login response
- CSRF token is included in JSON responses as csrfToken and must be sent as request header X-CSRF-Token
- Session timeout: 30 minutes
- Maximum retries: 3 failed login attempts
- Lockout time: 10 minutes after failed attempts (HTTP 423 while locked)
- Unauthenticated or invalid CSRF: HTTP 401
- Access levels (Administrator / Super-User / Read-Only) are not enforced on REST calls; a valid session is sufficient

Unit Parameter

All status and threshold queries use the format with unit{N} where N is the device unit ID. The number after "unit" corresponds to the device unit ID (0-based indexing).

Bulk query commands

All query commands need unit-scoped IDs must include numeric unit{N} (for example unit0, unit2). Offline units will return DB_DEVICE_OFFLINE. Invalid unit numbers return DB_INVALID_DEVICEID. You cannot use a wildcard (*) for the unit id.

Configuration and control commands

Configuration and control commands need unit-scoped IDs (for example unit0, unit2) or "all." The keyword "all" applies the action to all online units.

Examples:

- status/unit0/device - Query device unit 0
- status/unit1/phaseInput1 - Query phase 1 of device unit 1
- status/unit2/bank2 - Query bank 2 of device unit 2
- threshold/unit0/device - Query thresholds of device unit 0
- threshold/unit0/device - Sets thresholds of all devices
- status/unit3/outlet5 - Query outlet 5 of device unit 3

Naming and case

Use the exact form documented for each endpoint. All identifiers are case-sensitive.

- **Threshold query (POST /api/bulk):** lowercase id, e.g. threshold/unit0/outlet1. Query does not support outlet* or bank*.
- **Threshold set (POST /api/config):** lowercase id, including threshold/unit0/outlet* and threshold/unit0/bank* for all outlets or banks on that unit.
- **Outlet control (POST /api/control):** lowercase id values like unit0/outlet1 or unit0/outlet*
- Unit numbers are 0-based in request paths. Phase, bank, outlet, and sensor indexes in IDs are 1-based (e.g., phaseInput1, outlet2).

Note: For "all outlets", threshold set uses threshold/unit0/outlet*; outlet control uses unit0/outlet*. Threshold queries and status queries must list specific outlets.

Bulk Status Queries

POST /api/bulk

Query multiple status items in a single request. Maximum 5 items per request.

Status * wildcards: only status/unit{N}/phaseInput* is supported. status/.../bank*, outlet*, and sensor* are not accepted. Threshold query wildcards are listed under Threshold identity and naming.

Request Body:

```
{
  "items": [
    {
      "id": "status/unit0/phaseInput*" // Example: unit0 for all phase input.
    },
    {
      "id": "status/unit1/device"      // Example: unit1 for device ID 1
    }
  ]
  // ... up to 5 items
}
```

Response:

```
{
  "csrfToken": "token_string",
  "query": {
    "status/unit0/phaseInput1": {
      "voltage": "220.00",
      "current": "10.50",
      "powerFactor": "0.95",
      "apparentPower": "2310.00",
      "activePower": "2194.50",
      "energy": "1000.00"
    },
    "status/unit0/phaseInput2": {
      "voltage": "220.00",
      "current": "10.50",
      "powerFactor": "0.95",
      "apparentPower": "2310.00",
      "activePower": "2194.50",
      "energy": "1000.00"
    },
    "status/unit0/phaseInput3": {
      "voltage": "220.00",
      "current": "10.50",
      "powerFactor": "0.95",
      "apparentPower": "2310.00",
      "activePower": "2194.50",
      "energy": "1000.00"
    },
    "status/unit1/device": {
      "current": "30.00",
      "powerFactor": "0.92",
      "apparentPower": "6600.00",
      "activePower": "6072.00",
      "frequency": "50.00",
      "energy": "5000.00"
    }
  }
}
```

```
"errors": {}
}
```

Supported Status Query Items

Device Status

Query ID:

- status/unit{N}/device

Where:

- N is the device unit ID (0, 1, 2, ...)

Query Example:

```
{
  "items": [
    {"id": "status/unit0/device"} // Device unit 0
  ]
}
```

Response Data (individual item):

```
{
  "current": "30.00",           // Total current (A)
  "powerFactor": "0.92",       // Power factor
  "apparentPower": "6600.00",   // VA
  "activePower": "6072.00",     // Watts (W)
  "frequency": "50.00",        // Hz
  "energy": "5000.00"          // kWh
}
```

Phase Input Status

Query ID:

- status/unit{N}/phaseInput* (all phases)
- status/unit{N}/phaseInput{M} (specific phase)

Where:

- N is the device unit ID (0, 1, 2, ...) - e.g., unit0, unit1, unit2
- M is the phase number (1, 2, 3, ...) - e.g., phaseInput1, phaseInput2

Query Example:

```
{
  "items": [
    {"id": "status/unit0/phaseInput*"} // All phases of device unit 0
  ]
}
```

Response Data (individual item):

```
{
  "voltage": "220.00",         // Volts (V)
  "current": "10.50",          // Amperes (A)
  "powerFactor": "0.95",       // Power factor
  "apparentPower": "2310.00",   // VA
  "activePower": "2194.50",     // Watts (W)
  "energy": "1000.00"          // kWh
}
```

Bank Status

Query ID:

- status/unit{N}/bank{M}

Where:

- N is the device unit ID (0, 1, 2, ...) - e.g., unit0, unit1, unit2
- M is the bank number (1, 2, 3, ...) - 1-based

Query Example:

```
{
  "items": [
    {"id": "status/unit0/bank1"} // Bank 1 of device unit 0
  ]
}
```

Response Data (individual item):

```
{
  "voltage": "220.00",
  "current": "15.00",
  "powerFactor": "0.90",
  "apparentPower": "3300.00",
  "activePower": "2970.00",
  "energy": "2000.00",
  "status": "On" // Optional, if bank has breaker control
}
```

Outlet Status

Query ID:

- status/unit{N}/outlet{M}

Where:

- N is the device unit ID (0, 1, 2, ...)
- M is the outlet number (1, 2, 3, ...) - 1-based

Query Example:

```
{
  "items": [
    {"id": "status/unit0/outlet1"}
  ]
}
```

Response Data (individual item):

```
{
  "current": "5.00",           // Amperes (A) - if metered outlet
  "apparentPower": "1100.00", // VA - if metered outlet
  "activePower": "990.00",    // Watts (W) - if metered outlet
  "energy": "500.00",         // kWh - if metered outlet
  "status": "On"              // On/Off - if switchable outlet
}
```

Sensor Status

Query ID:

- status/unit{N}/sensor{M}

Where:

- N is the device unit ID (0, 1, 2, ...)

- M is the sensor number (1, 2, 3, ...) - 1-based

Query Example:

```
{
  "items": [
    {"id": "status/unit0/sensor1"} // Sensor 1 of device unit 0
  ]
}
```

Response Data (individual item):

```
{
  "name": "sensor1",
  "temperature": "25.50", // Celsius (°C), or "--" if the sensor is offline
  "humidity": "60.00" // Percentage (%), or "--" if the sensor is offline
}
```

System Information

Query ID:

- status/unit0/systemInfo

Query Example:

```
{
  "items": [
    {"id": "status/unit0/systemInfo"} // System info of device unit 0
  ]
}
```

Response Data (individual item):

```
{
  "appVer": "1.0.0",
  "bootVer": "0.9.0",
  "interVer": "1.2.0",
  "hardwareVer": "0.1",
  "contactName": "John Doe",
  "location": "Data Center A",
  "deviceName": "PDU-001"
}
```

TCP/IP Configuration

Query ID:

- status/tcpip

Query Examples:

```
{
  "items": [
    {"id": "status/tcpip"}
  ]
}
```

Response:

```
{
  "bootmode": "dhcp", // "dhcp" or "static"
  "ipv4addr": "192.168.1.100",
  "subnetmask": "255.255.255.0",
  "defaultgw": "192.168.1.1",
  "defaultdns": "8.8.8.8"
}
```

SNMP Configuration

Query ID:

- status/snmp

Query Examples:

```
{
  "items": [
    {"id": "status/snmp"}
  ]
}
```

Response:

```
{
  "trapserver": "192.168.1.10",
  "version1": "enable",          // "enable" or "disable"
  "version2": "enable",
  "version3": "enable",
  "readcommunity": "public",
  "writecommunity": "private",
  "username": "snmpuser",
  "authprotocol": "sha",        // "sha", "md5", or "none"
  "privprotocol": "aes",        // "aes", "des", or "none"
  "authkey": "authkey123",
  "privkey": "privkey123"
}
```

Modbus Status

Query ID:

- status/modbus

Query Examples:

```
{
  "items": [
    {"id": "status/modbus"}
  ]
}
```

Response:

```
{
  "status": "enable" // "enable" or "disable"
}
```

Email Configuration

Query ID:

- status/email

Query Examples:

```
{
  "items": [
    {"id": "status/email"}
  ]
}
```

Response:

```
{
  "smtpserver": "smtp.example.com",
  "smtpport": "587",
}
```

```
"smtpoverssl": "enable",          // "enable" or "disable"
"smtpauthen": "enable",          // "enable" or "disable"
"smtpusername": "user@example.com",
"smtppassword": "password",
"smtpaddrfrom": "sender@example.com",
"smtpaddrto": "recipient@example.com"
}
```

Syslog Configuration

Query ID:

- status/syslog

Query Examples:

```
{
  "items": [
    {"id": "status/syslog"}
  ]
}
```

Response:

```
{
  "status": "enable",              // "enable" or "disable"
  "syslogserver": "192.168.1.20",
  "syslogport": "514"
}
```

Web Protocol Status

Query ID:

- status/web

Query Examples:

```
{
  "items": [
    {"id": "status/web"}
  ]
}
```

Response:

```
{
  "http_enabled": "enable",        // "enable" or "disable"
  "https_enabled": "disable"       // "enable" or "disable"
}
```

IPv6 Configuration

Query ID:

- status/ipv6

Query Examples:

```
{
  "items": [
    {"id": "status/ipv6"}
  ]
}
```

Response:

```
{
  "mode": "manual",                // "manual" or "auto"
  "ipv6_address": "2001:db8::1/64",
}
```

```
"gateway": "2001:db8::1"
}
```

Date/Time Status

Query ID:

- status/datetime

Query Example:

```
{
  "items": [
    {"id": "status/datetime"}
  ]
}
```

Response Data:

```
{
  "mode": "auto",           // "auto" (NTP) or "manual"
  "current_date": "08/20/2026", // MM/DD/YYYY
  "current_time": "14:30:00",  // HH:MM:SS
  "timezone": 800,           // UTC offset × 100 (e.g., 800 = UTC+8, 450 = UTC+4.5, -500 = UTC-5)
  "ntp_server": "pool.ntp.org", // NTP server hostname
  "ntp_update_freq": 8       // NTP update interval in hours (1-8760, default 8) JSON number, not a string
}
```

Daylight Saving Time (DST) Status

Query ID:

- status/dst

Query Example:

```
{
  "items": [
    {"id": "status/dst"}
  ]
}
```

Response Data:

```
{
  "enable": true,           // DST enabled flag
  "offset": 60,             // DST offset in minutes
  "start_month": 3,         // Start month (1-12)
  "start_week": 2,         // Start week of month (1-5)
  "start_day": 0,          // Start day of week (0=Sunday, 6=Saturday)
  "start_time": "02:00:00", // Start time (HH:MM:SS)
  "end_month": 11,         // End month (1-12)
  "end_week": 1,           // End week of month (1-5)
  "end_day": 0,            // End day of week (0=Sunday, 6=Saturday)
  "end_time": "02:00:00"   // End time (HH:MM:SS)
}
```

User Accounts Status

Query ID:

- status/user

Query Example:

```
{
  "items": [
    {"id": "status/user"}
  ]
}
```

```
]
}
```

Response Data:

```
{
  "users": [
    {
      "index": "0",
      "username": "admin",
      "access_level": "Administrator" // "Administrator", "Super-User", "Read-Only"
    },
    {
      "index": "1",
      "username": "user1",
      "access_level": "Read-Only"
    }
  ]
}
```

Outlet Group Status**Query ID:**

- status/outletgroup/{N}

Where:

- N is the group number (1-based, 1-10).

Query Example:

```
{
  "items": [
    {"id": "status/outletgroup/1"} // Query outlet group 1
  ]
}
```

Response Data:

```
{
  "group": 1,
  "pdus": [
    {
      "pdu": 0,
      "outlets": [1, 2, 3] // Outlet numbers included in this group
    },
    {
      "pdu": 1,
      "outlets": [4, 5]
    }
  ]
}
```

Supported Threshold Query Items**Device Thresholds****Query ID:**

- threshold/unit{N}/device

Where:

- N is the device unit ID (0, 1, 2, ...)

Query Example:

```
{
```

```

"items": [
  {"id": "threshold/unit0/device"} // Thresholds for device unit 0
]
}

```

Response Data (individual item):

```

{
  "load_lower_warning": "10.0", // Amperes (A)
  "load_upper_critical": "80.0", // Amperes (A)
  "freq_lower_critical": "49.0", // Hz
  "freq_upper_critical": "51.0" // Hz
}

```

Phase Thresholds

Query ID:

- status/unit{N}/phaseInput* (all phases)
- status/unit{N}/phaseInput{M} (specific phase)

Where:

- N is the device unit ID (0, 1, 2, ...) - e.g., unit0, unit1, unit2
- M is the phase number (1, 2, 3, ...) - e.g., phaseInput1, phaseInput2

Query Example:

```

{
  "items": [
    {"id": "threshold/unit0/phase*"} // All phase thresholds of device unit 0
  ]
}

```

Response Data (individual item):

```

{
  "current_lower_warning": "5.0", // Amperes (A)
  "current_upper_critical": "20.0", // Amperes (A)
  "voltage_lower_warning": "200.0", // Volts (V)
  "voltage_upper_critical": "240.0" // Volts (V)
}

```

Bank Thresholds

Query ID:

- threshold/unit{N}/bank{M}

Where:

- N is the device unit ID (0, 1, 2, ...) - e.g., unit0, unit1, unit2
- M is the bank number (1, 2, 3, ...) - 1-based

Query Example:

```

{
  "items": [
    {"id": "threshold/unit0/bank1"} // Bank 1 thresholds of device unit 0
  ]
}

```

Response Data (individual item):

```

{
  "current_lower_warning": "10.0", // Amperes (A)

```

```
"current_upper_critical": "30.0" // Amperes (A)
}
```

Outlet Thresholds

Query ID:

- threshold/unit{N}/outlet{M}

Where:

- N is the device unit ID (0, 1, 2, ...)
- M is the outlet number (1, 2, 3, ...) - 1-based

Query Example:

```
{
  "items": [
    {"id": "threshold/unit0/outlet1"} // Outlet 1 thresholds of device unit 0
  ]
}
```

Response Data (individual item):

```
{
  "load_lower_warning": "2.0", // Amperes (A)
  "load_upper_critical": "10.0" // Amperes (A)
}
```

Sensor Thresholds

Query ID:

- status/unit{N}/sensor{M}

Where:

- N is the device unit ID (0, 1, 2, ...)
- M is the sensor number (1, 2, 3, ...) - 1-based

Query Example:

```
{
  "items": [
    {"id": "threshold/unit0/sensor*"} // All sensor thresholds of device unit 0
  ]
}
```

Response Data (individual item):

```
{
  "temperature_upper_critical": "40", // Celsius (°C)
  "humidity_lower_critical": "20" // Percentage (%)
}
```

Configuration Management

POST /api/config

Update device configuration. Returns updateResult: "OK" on success or "Failed" on error.

Request Body:

```
{
  "tcpip": { /* TCP/IP config */ },
  "snmp": { /* SNMP config */ },
  "modbus": { /* Modbus config */ },
  "https": { /* HTTPS config */ },
  "http": { /* HTTP config */ },
  "email": { /* Email config */ },
  "syslog": { /* Syslog config */ },
  "ipv6": { /* IPv6 config */ },
  "systemInfo": { /* System info */ },
  "thresholds": [ /* Threshold configs */ ],
  "sensorNames": [ /* Sensor name configs */ ],
  "datetime": { /* Date/time or NTP config */ },
  "dst": { /* Daylight saving time config */ },
  "users": { /* User account management */ },
  "outletgroups": { /* Outlet group config */ }
}
```

Response:

```
{
  "csrfToken": "token_string",
  "updateResult": "OK" // or "Failed"
}
```

TCP/IP Configuration

```
{
  "tcpip": {
    "bootmode": "dhcp", // "dhcp" or "static"
    "ipv4addr": "192.168.1.100",
    "subnetmask": "255.255.255.0",
    "defaultgw": "192.168.1.1",
    "defaultdns": "8.8.8.8"
  }
}
```

Note: Device reboot is required for TCP/IP address changes to take effect.

SNMP Configuration

```
{
  "snmp": {
    "trapserver": "192.168.1.10",
    "version1": "enable", // "1", "enable", "true" or "0", "disable", "false"
    "version2": "enable",
    "version3": "enable",
    "readcommunity": "public",
    "writecommunity": "private",
    "username": "snmpuser",
    "authprotocol": "sha", // "sha", "md5", or "none"
    "privprotocol": "aes", // "aes", "des", or "none"
    "authkey": "authkey123", // Minimum 8 characters
    "privkey": "privkey123" // Minimum 8 characters
  }
}
```

```
}
```

Note: Device reboot is required for SNMP configuration changes to take effect.

Modbus Configuration

```
{
  "modbus": {
    "enable": "enable" // "1", "enable", "true" or "0", "disable", "false"
  }
}
```

HTTP/HTTPS Configuration

```
{
  "https": {
    "status": "enable" // "1", "enable", "true" or "0", "disable", "false"
  },
  "http": {
    "status": "enable" // "1", "enable", "true" or "0", "disable", "false"
  }
}
```

Note: Enabling HTTP disables HTTPS and vice versa.

Note: Device reboot is required for HTTP/HTTPS mode changes to take effect.

Email Configuration

```
{
  "email": {
    "smtpserver": "smtp.example.com",
    "smtpport": "587",
    "smtpoverssl": "enable", // "1", "enable", "true" or "0", "disable", "false"
    "smtpauthen": "enable", // "1", "enable", "true" or "0", "disable", "false"
    "smtpusername": "user@example.com", // Written if provided; missing values do not fail the request
    "smtppassword": "password", // Written if provided; missing values do not fail the request
    "smtpaddrfrom": "sender@example.com",
    "smtpaddrto": "recipient@example.com"
  }
}
```

Note: Device reboot is required for Email Configuration changes to take effect.

Syslog Configuration

```
{
  "syslog": {
    "status": "enable", // "1", "enable", "true" or "0", "disable", "false"
    "syslogserver": "192.168.1.20",
    "syslogport": "514"
  }
}
```

Note: Device reboot is required for Syslog Configuration changes to take effect.

IPv6 Configuration

```
{
  "ipv6": {
    "mode": "manual", // "manual", "static", or "auto"
    "ipv6_address": "2001:db8::1/64", // Required if mode is "manual"
    "gateway": "2001:db8::1" // Required if mode is "manual"
  }
}
```

```
}
```

Note: Device reboot is required for IPv6 configuration changes to take effect.

System Information Configuration

```
{
  "systemInfo": {
    "unit": "unit0",           // Optional: "unit0", "unit1", ..., "all", or omit for default unit
    "contactName": "John Doe",
    "location": "Data Center A",
    "deviceName": "PDU-001"
  }
}
```

Threshold Configuration

Use the set id format from Threshold identity and naming: threshold/unit{N}/{name} or threshold/all/{name}. Set wildcards include phase*, bank*, outlet*, temp*, and hum*. Query wildcards are a subset (phase* and sensor* only).

```
{
  "thresholds": [
    {
      "id": "threshold/unit0/device",
      "type": "load_lower_warning",
      "value": "10.0"
    },
    {
      "id": "threshold/unit0/phase*",
      "type": "current_upper_critical",
      "value": "20.0"
    }
  ]
}
```

Configuration ID:

- threshold/unit{N}/{object}
- threshold/all/{object}

Where:

- N is the device unit ID (0, 1, 2, ...)
- Object can be the following:
 - device;
 - phase1-phase3 or phase*
 - bank1-bank12 or bank*
 - outlet1-outlet42 or outlet*;
 - temp1-temp4 or temp*
 - hum1–hum4 or hum* (map to sensor index; see Threshold identity and naming).

Threshold attribute by object:

The threshold attribute is different based on the object.

- **device:** load_lower_warning, load_upper_critical, freq_lower_critical, freq_upper_critical
- **phase:** current_lower_warning, current_upper_critical, voltage_lower_warning, voltage_upper_critical
- **bank:** current_lower_warning, current_upper_critical
- **outlet:** load_lower_warning, load_upper_critical
- **temp/hum:** temperature_upper_critical, humidity_lower_critical

Note: Value is multiplied by 10 internally (e.g., "10.0" becomes 100). For temperature/humidity thresholds, value is divided by 10.

Sensor Naming Configuration

Configuration ID:

- status/unit{N}/sensor{M}

Where:

- N is the device unit ID (0, 1, 2, ...)
- M is the sensor number (1, 2, 3, ...) - 1-based

Configuration Example:

```
{
  "sensornames": [
    {
      "id": "sensorname/unit0/sensor1",
      "name": "new sensor name"
    },
    {
      "id": "sensorname/unit0/sensor2",
      "name": "new sensor name"
    }
  ]
}
```

Configuration Example (All Units):

```
{
  "sensornames": [
    {
      "id": "sensorname/all/sensor1",
      "name": "new sensor name"
    }
  ]
}
```

Date/Time Configuration

Configure device date and time settings. Supports manual time setting or NTP automatic synchronization.

Time zone format:

Time zone is expressed as UTC offset × 100 (integer). For example:

- 800 = UTC+8:00
- 450 = UTC+4:30 (half-hour offsets supported)
- -500 = UTC-5:00
- 1200 = UTC+12:00

Range: -1200 to 1200 (UTC-12:00 to UTC+12:00)

Manual Mode - Set Date and Time:

```
{
  "datetime": {
    "mode": "manual",
    "date": "08/20/2026",           // MM/DD/YYYY format
    "time": "14:30:00"            // HH:MM:SS format (24-hour)
  }
}
```

Auto Mode - NTP Synchronization:

```
{
  "datetime": {
    "mode": "auto",
    "timezone": 800,           // UTC offset × 100 (e.g., 800 = UTC+8, 450 = UTC+4.5, -500 = UTC-5)
    "ntp_server": "pool.ntp.org", // NTP server hostname
    "ntp_update_freq": 8,      // Update interval in hours (1-8760, default 8)
    "ntp_update_now": 1        // Optional: trigger immediate NTP sync
  }
}
```

Response:

```
{
  "csrfToken": "token_string",
  "updateResult": "OK"
}
```

Error Responses

On failure, updateResult is "Failed" and message is one of:

- Date and time are required for manual mode.
- Invalid date/time format. Expected MM/DD/YYYY and HH:MM:SS.
- Timezone must be between -1200 and 1200.
- Unknown datetime mode. Use 'manual' or 'auto'.
- Mode field is missing or empty.
- Values outside 1-8760 for ntp_update_freq are clamped to that range.

Daylight Saving Time (DST) Configuration

Configure daylight saving time rules. When enabled, the device automatically adjusts time based on DST rules.

Enable DST:

```
{
  "dst": {
    "enable": true,
    "offset": 60,           // DST offset in minutes (typically 60)
    "start_month": 3,       // Start month (1-12)
    "start_week": 2,       // Start week of month (1-5)
    "start_day": 0,        // Start day of week (0=Sunday through 6=Saturday)
    "start_time": "02:00:00", // Start time (HH:MM:SS)
    "end_month": 11,       // End month (1-12)
    "end_week": 1,         // End week of month (1-5)
    "end_day": 0,          // End day of week (0=Sunday through 6=Saturday)
    "end_time": "02:00:00" // End time (HH:MM:SS)
  }
}
```

Disable DST:

```
{
  "dst": {
    "enable": false
  }
}
```

Response:

```
{
  "csrfToken": "token_string",
  "updateResult": "OK"
}
```

Error Responses

On failure, updateResult is "Failed" and message is:

- All DST fields are required when enabled.

User Account Management

Manage user accounts: add, delete, and update user passwords. Maximum 6 users supported.

Add User:

```
{
  "users": {
    "action": "add",
    "username": "newuser",
    "password": "Password123!",      // Must meet password requirements
    "access_level": 2               // 0=Administrator, 1=Super-User, 2=Read-Only
  }
}
```

Delete User:

```
{
  "users": {
    "action": "delete",
    "username": "user_to_remove"
  }
}
```

Update User Password:

```
{
  "users": {
    "action": "update",
    "username": "existinguser",
    "password": "NewPassword123!",  // New password meeting requirements
    "confirmpassword": "NewPassword123!" // Required: must match password
  }
}
```

Response (Success):

```
{
  "csrfToken": "token_string",
  "updateResult": "OK"
}
```

Error Responses

On failure, updateResult is "Failed" and message is one of:

- Username and password are required.
- Duplicate username.
- Maximum number of users reached.
- Invalid parameters.
- Username is required.
- Cannot remove admin user.

- User not found.
- Confirm password missing or mismatch.
- Password must be at least 8 characters and contain at least one digit, one uppercase letter, one lowercase letter, and one special character.

Outlet Group Configuration

Configure outlet groups for batch control across multiple PDUs. Maximum 10 groups supported.

Set Outlet Group:

```
{
  "outletgroups": {
    "action": "set",
    "group": 1,                // Group number (1-based)
    "pdus": [
      {
        "pdu": 0,              // PDU unit ID
        "outlets": [1, 2, 3]   // Outlet numbers to include
      },
      {
        "pdu": 1,
        "outlets": [4, 5]
      }
    ]
  }
}
```

Delete Outlet Group:

```
{
  "outletgroups": {
    "action": "delete",
    "group": 1                // Group number to delete (1-based)
  }
}
```

Response:

```
{
  "csrfToken": "token_string",
  "updateResult": "OK"
}
```

Error Responses:

On failure, **updateResult** is **"Failed"** and **message** is one of:

- group and pdus array are required.
- Invalid group number.
- group is required.
- Unknown action type.
- Action field is missing or empty.

Device Control

POST /api/control

Control device actions and outlets. Supports applying multiple control operations in one request.

Request Body:

```
{
  "control": [
    {
      "id": "unit0/outlet1",
      "action": "on"
    },
    {
      "id": "unit0/outlet*",
      "action": "off"
    },
    {
      "id": "unit0/device",
      "action": "reboot"
    }
  ]
}
```

Device actions

reboot: Reboots the device (2 second delay)

resetall: Resets all device and network configuration to defaults

Control ID:

- unit{N}/device

Where:

- N is the device unit ID (0, 1, 2, ...)

Example – Reboot the device.

Will delay two seconds and then reboot.

```
{
  "control": [
    {
      "id": "unit0/device",
      "action": "reboot"
    }
  ]
}
```

Example – Reset the device.:

```
{
  "control": [
    {
      "id": "unit0/device",
      "action": "resetall"
    }
  ]
}
```

Response:

If at least one control item is applied:

```
{
  "csrfToken": "token_string",
  "updateResult": "OK"
}
```

```
}
```

Outlet operations

Control ID:

- unit{N}/outlet{M} (1-based outlet number)
- unit{N}/outlet* (all outlets)

Where:

- N is the device unit ID (0, 1, 2, ...)
- M is the outlet number (1, 2, 3, ...) - 1-based

Example - Turn on outlet 1:

```
{
  "control": [
    {
      "id": "unit0/outlet1",
      "action": "on"          // "on" or 1
    }
  ]
}
```

Example - Turn off all outlets:

```
{
  "control": [
    {
      "id": "unit0/outlet*",
      "action": "off"        // "off" of 0
    }
  ]
}
```

Example - Reboot device:

```
{
  "control": [
    {
      "id": "unit0/device",
      "action": "reboot"
    }
  ]
}
```

Response:

If at least one control item is applied:

```
{
  "csrfToken": "token_string",
  "updateResult": "OK"
}
```

Certificate Management

POST /api/certificate/upload

Upload SSL certificate and private key for HTTPS encryption.

Add Certificate:

Request:

- **Content-Type:** typically multipart/form-data
- **Body:** PEM content containing a certificate (-----BEGIN CERTIFICATE) and a private key (-----END RSA PRIVATE KEY----- or -----END EC PRIVATE KEY-----)
- **Authentication:** Required (cookie + X-CSRF-Token)

The handler looks for PEM markers in the POST body; a specific form field name is not required.

Response (Success):

```
{
  "csrfToken": "token_string",
  "updateResult": "Update Successful, the device needs to reboot for New Certificate to take effect."
}
```

Error Responses:

On failure, updateResult is "Failed" and message is one of:

- Wrong file format
- Parse files Error
- Wrong Cert file
- or a certificate verification description from the TLS stack

Delete Certificate

To delete the existing SSL certificate, send a POST request to **/api/certificate/upload** whose body contains the substring remove (for example mode=remove). The device searches the body for remove; Content-Type and field name are not strictly checked.

Request:

- **Authentication:** Required

Response (Success):

```
{
  "csrfToken": "token_string",
  "updateResult": "Certificate removed successfully, waiting for device reboot."
}
```

Notes:

- Certificate must be in PEM format.
- File must contain both certificate and private key (RSA or EC)
- Device reboot is required for a new certificate to take effect
- Deleting the certificate reverts to the default self-signed certificate

Firmware Management

POST /api/firmware/upload

Upload firmware file for device upgrade.

Request:

- **Content-Type:** typically multipart/form-data
- **Body:** firmware binary
- **Authentication:** Required (cookie + X-CSRF-Token)

Success Response:

```
{
  "csrfToken": "token_string",
  "updateResult": "Firmware upload successfully, waiting for device reboot to complete upgrading."
}
```

Failure examples:

```
{
  "csrfToken": "token_string",
  "updateResult": "Firmware upload failed with wrong file."
}
```

Possible updateResult values:

- Firmware upload successfully, waiting for device reboot to complete upgrading.
- Firmware upload failed with wrong file.
- Firmware upload failed with flash write error.
- Firmware upload failed with same version.
- Firmware upload failed with CRC error.
- Firmware signature is invalid.

Notes:

- Firmware must be in the correct signed binary format
- After a successful upload the device reboots after about 2 seconds
- Do not power off during upgrade

Notes

- Most measurement and threshold fields in bulk responses are strings. Exceptions:
 - status/datetime: timezone, ntp_update_freq are numbers
 - status/dst: enable is boolean
 - month/week/day/offset: are numbers)
 - status/outletgroup/{N}: group, pdu, outlets[] are numbers
- Login, bulk, config, and control use JSON. Certificate and firmware upload use a file POST body (or a body containing remove to delete a certificate).
- After authentication:
 - Successful responses use HTTP 200
 - Missing session/CSRF is HTTP 401;
 - lockout is HTTP 423.
 - Login password errors are HTTP 200 with JSON code 401.
- Maximum 5 items per bulk query request
- Unit numbers are 0-based in request paths. Phase, bank, outlet, and sensor indexes in IDs are 1-based (e.g., phaseInput1, outlet2).
- Threshold values in configuration are specified as strings and converted internally
- Password change is required on first login if the default password is not updated; no session is created until that succeeds
- File config endpoint (/files/config) is registered but not implemented
- Maximum 6 user accounts and 4 sensors per unit; outlet groups are numbered 1-10

Error Codes

The following error codes may appear in the **errors** object of bulk query responses:

- **DB_INVALID_ID:** Invalid query ID or parameter. This includes unit* (for example status/unit*/device), unsupported wildcards such as outlet* / bank* on query, out-of-range indexes, and outlet queries on a unit that has no outlets (or no metered outlets for threshold query).
- **DB_INVALID_DEVICEID:** Invalid device/unit ID
- **DB_DEVICE_OFFLINE:** Specified device is offline
- **DB_WILDCARD_UNIT_NOT_ALLOWED:** Defined in firmware for a unit wildcard, but current bulk ID parsing rejects unit* earlier and returns DB_INVALID_ID instead.
- **DB_DEVICE_OFFLINE:** The unit is offline

Example Usage

Complete Workflow

1. Login:

POST /api/login

```
{
  "username": "admin",
  "password": "password"
}
```

2. Query Device Status:

POST /api/bulk

Cookie: connect.sid=<session_id>

X-CSRF-Token: <csrfToken>

```
{
  "items": [
    {"id": "status/unit0/device"},
    {"id": "status/unit1/phaseInput*"},
    {"id": "status/unit2/bank2"},
    {"id": "status/unit0/outlet1"}
  ]
}
```

3. Update Configuration:

POST /api/config

Cookie: connect.sid=<session_id>

X-CSRF-Token: <csrfToken>

```
{
  "tcpip": {
    "bootmode": "static",
    "ipv4addr": "192.168.1.100"
  }
}
```

4. Control Outlet:

POST /api/control

Cookie: connect.sid=<session_id>

X-CSRF-Token: <csrfToken>

```
{
  "control": [
    { "id": "unit0/outlet1", "action": "on" }
  ]
}
```

5. Upload Certificate:

POST /api/certificate/upload

Cookie: connect.sid=<session_id>

X-CSRF-Token: <csrfToken>

6. Upload Firmware:

POST /api/firmware/upload

Cookie: connect.sid=<session_id>

X-CSRF-Token: <csrfToken>