BlueNet
BN PRO
Reference Manual
CLI

BACHMANN
We power your life.

BluNet

BACHMANN

BN PRO

Reference Manual CLI

Content:

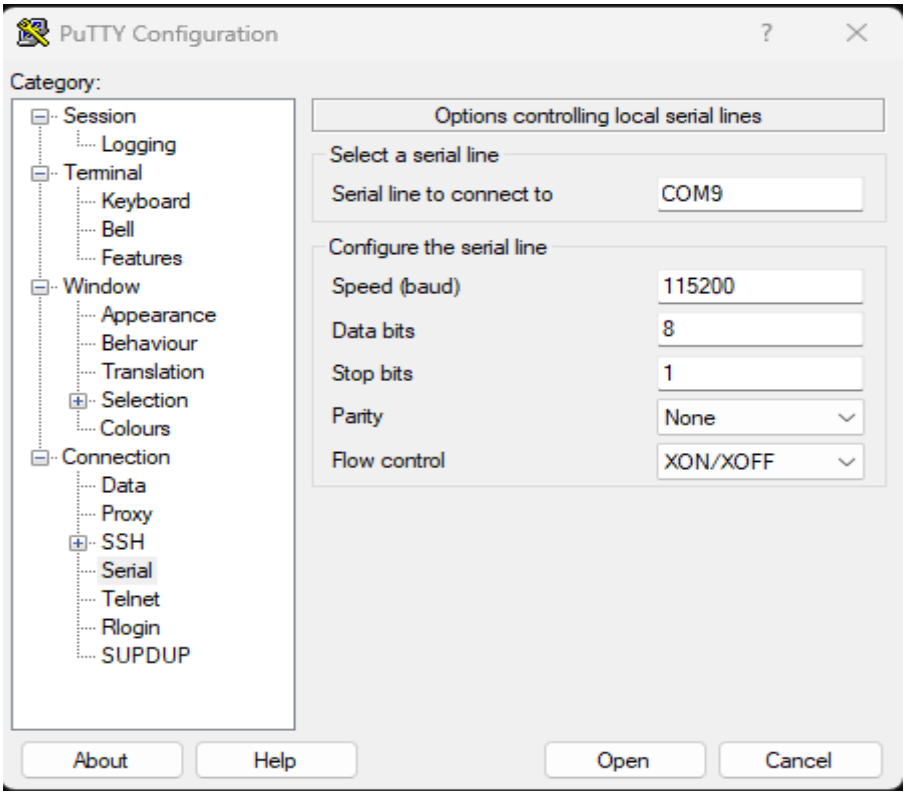
1 Accessing and Navigating the CLI

This document describes the usage of the Command Line Interface (CLI).

Using the CLI interface the user can access and configure all parts of the device.

1.1 Accessing the CLI (Local and Remote)

CLI can be accessed locally, the PDU features a USB-C port located on its Control Unit (CU), which supports serial communication. Users can access the Command Line Interface (CLI) by connecting a Micro-USB to serial cable to this port. Once connected, use a terminal emulator (e.g., PuTTY, Tera Term) with the appropriate serial settings (e.g., baud rate: 115200, data bits: 8, stop bits: 1, no parity) to log into the device.



CLI can be accessed remotely through SSH if enabled. For example, using Power Shell the user can give the following command:

`ssh admin@<PDU-IP-Address>` using the corresponding IP address of the PDU that can be read from the local GUI

By default, the login credentials are:

- **Username:** admin
- **Password:** admin

If the password has not been changed in the web GUI, the first login to the CLI will always be with admin / admin.

1.2 CLI Key combinations

The history of already entered commands is kept. It can be accessed using the Up and Down keys.

The following key combinations are available:

- CTRL-a: moves the cursor the start of the line
- CTRL-b: moves the cursor left one character
- CTRL-c: clears the current line and refreshes the prompt
- CTRL-d: deletes the character under the cursor; if pressed on an empty line, exits the current mode or the CLI session when already in the main commands
- CTRL-e: moves the cursor to the end of the line
- CTRL-f: moves the cursor right one character
- CTRL-k: deletes from the cursor to the end of the line
- CTRL-l: refreshes the current line
- CTRL-u: deletes from the cursor to the beginning of the line
- CTRL-r: reverse search through command history
- CTRL-y: pastes the last item deleted
- CTRL-_: undo
- ALT-b: moves the cursor backward one word
- ALT-f: moves the cursor forward one word
- ALT-Backspace: deletes the word left of the cursor

Tab key Behavior:

- The **Tab** key in the PDU CLI is primarily used for **path and system command autocompletion**.
  - File and Directory Paths:
    When entering file paths (e.g., for logs or configuration files), pressing **Tab** will autocomplete the path if it is unambiguous. If multiple matches exist, pressing **Tab** twice will list available options.
  - System Commands:
    For commands like systemctl, journalctl, or others that interact with the Linux system, **Tab** can help autocomplete service names or options.
  - CLI Commands:
    Autocompletion for built-in CLI commands is **also supported**.
    - Example:

    The user writes “cli” in the command line then presses the “Tab” → should look like this:

```
BN-PRO-7A93CFE2:~$ cli
auth            element         ldap-group       outlet-group     role
cluster         export          license-info     outlet-sequence  set
config          get             license-manager  product-info     signalchain
cronjob         help            log              rcm-selftest     user
do              import          monitor          reset
```

1.3 Using CLI Help and Autocompletion

The Command-Line Interface (CLI) provides a consistent and intuitive help system across all commands and subcommands. This allows users to explore available functionalities, understand command structures, and learn how to use each feature effectively.

General Behavior:

- **Every CLI command supports the -h or --help option.**
- When appended to a command, it displays:
  - A short description of the command's purpose.
  - A list of available options and arguments.
  - Example usage to guide the user.

General Help Access

Users can access help information in two ways:

```
cli -h
```

Displays a list of all top-level commands along with a short description of each.

```
cli help
```

Provides the same overview as cli -h, listing all available subcommands and their functions.

Subcommand Help

Every subcommand supports the -h or --help option to display detailed usage instructions. This includes:

- **Structure of the command**
- **Available options and arguments**
- **Example usage**

`cli config -h` # Lists all configuration modules

`cli config dns set -h` # Shows how to set DNS parameters

`cli auth -h` # Displays authentication-related commands

1.3.1 Command Options Syntax

Some CLI commands support additional **options** to customize their behavior. Options can usually be written in **short form** (single letter with a dash) or **long form** (full word with two dashes). Both forms are equivalent.

Examples:

- -p or --password → specify a password
- -u or --username → specify a username

General rules:

- Options are placed **after the command** and before the required arguments.

Example:

```
cli user add -u user1 -p secret123 -r operator
```

- Use -h or --help with any command to see all available options and their descriptions.

1.4 Handling Errors and Invalid Input

The CLI is designed to provide clear and informative feedback when a command is entered incorrectly or when required parameters are missing. This helps users quickly identify and correct mistakes without needing external support.

Common Error Scenarios

1. Unknown or Misspelled Options

If a user enters an unsupported option, the CLI will display an error and suggest valid alternatives.

- **Example:**

```
BN-PRO-7AD4B474:~$ cli config dns set --server 13 ''

Usage: cli config dns set [OPTIONS]

Try 'cli config dns set -h' for help.
```

```
Error: No such option: --server (Possible options: --server1, --server2,
--server3)
```

2. Missing Required Values

- **Example:**

```
BN-PRO-7AD4B474:~$ cli config dns set --mode manual --server1 ''
Error: HTTP status code 422 (Unprocessable Entity)
    At least one DNS server has to be defined: __root__
```

Always check the help output before using a new command

- **Example:**

```
BN-PRO-7AD4B474:~$ cli config dns set -h
Usage: cli config dns set [OPTIONS]

  Configure DNS settings

  Example:

  # configure DNS mode
  cli config dns set --mode automatic

Options:
  --mode [manual|automatic]  Set DNS mode
                                 manual - use specified DNS server
                                 automatic - use DNS server provided by
DHCP server
  --server1 TEXT             Set DNS server 1
  --server2 TEXT             Set DNS server 2
  --server3 TEXT             Set DNS server 3
```

```
  -h, --help                  Show this message and exit.
```

1.5 Logging Out

There are 2 possibilities to log out from the current session.

`logout` and `exit`

1.6 File Transfer via SCP

SCP (Secure Copy Protocol) allows users to securely transfer files between their PC and the PDU over SSH. The command is executed **on the user's PC**, and the behavior depends on the **current working directory** and **path formatting**.

- Copying from PDU to PC on Linux:

  # Save to current directory

```
scp user@:/home/user/log.txt ./
```

  # Save to a specific directory

```
scp user@:/home/user/log.txt /home/username/logs/
```

  # Save to a path with spaces

```
scp user@:"/home/user/log files/log.txt" "./My Documents/"
```

- Copying from PDU to PC on Windows (PowerShell):

  # Save to current directory

```
scp user@:/home/user/log.txt .\
```

  # Save to a specific directory

```
scp user@:/home/user/log.txt "C:\Users\Romina\Documents\logs\"
```

  # Save to a path with spaces

```
scp user@:"/home/user/log files/log.txt" "C:\Users\Romina\My Documents\"
```

- Copying from PC to PDU on Linux

  # Upload from current directory

```
scp ./config.txt user@:/home/user/
```

  # Upload from a specific path

```
scp "/home/romina/Documents/config.txt" user@:/home/user/
```

  # Upload and rename

```
scp ./config.txt user@:/home/user/new_config.txt
```

- Copying from PC to PDU on Windows (PowerShell)

  # Upload from current directory

```
scp .\config.txt user@:/home/user/
```

  # Upload from a specific path

```
scp "C:\Users\Romina\Documents\config.txt" user@:/home/user/
```

  # Upload and rename

```
scp .\config.txt user@:/home/user/new_config.txt
```

2 Handling Errors and Operation Results

When working with the CLI, each command provides direct feedback in the console. This ensures that admin users can immediately see whether an operation was successful or if an error occurred.

General behavior:

- **Successful operations** → The console prints a confirmation message such as:
  - Successfully created new user
  - Successfully created new role
  - Cluster mode changed successfully
  - Event log written to: /run/media/sd/eventLogs_SD_admin.csv
  - RCM Selftest triggered …
- **Errors or invalid input** → The CLI prints an error message describing the issue. **Examples:**
  - Error: [Errno 21] Is a directory: '/home/admin/' (invalid file path)
  - Error: HTTP status code 404 (Not Found)

    No such measurement
  - Access denied (user management) for user operator

Notes:

- The CLI will not silently ignore mistakes — any incorrect parameter or invalid syntax triggers an error message.
- For **interactive prompts** (e.g., cli user add without full arguments), the CLI asks step by step. If input is the process stops and displays the corresponding error.
- Confirmation messages always use clear wording such as *"Successfully created new user"*, *"Successfully deleted user: maria"*, so the administrator knows the command was executed correctly.

3 System Settings Configuration

The `cli config -h` command provides access to a structured list of configuration subcommands available in the CLI. These commands allow users to **get and set system settings** directly from the terminal, offering full control over network, security, and service parameters.

When entered, the command displays:

- A short description of the config module.
- A list of available subcommands, each targeting a specific configuration area (e.g., ssh, dns, hostname, modbus, etc.).
- Usage examples to guide the user in applying settings correctly.

3.1 dns

The cli config dns command allows users to **view and modify DNS settings** on the device. This is useful for managing how the system resolves domain names, especially in environments with custom or static network configurations.

- **View Current DNS Settings**

  To check the current DNS configuration, use:

```
cli config dns get
```

  This displays the current DNS mode (automatic or manual) and the configured DNS servers (dns_server1, dns_server2, dns_server3).

- **Modify DNS Settings**

  To change the DNS configuration, use:

  Available Options:

  - --mode [manual|automatic]
    - **manual**: Use custom DNS servers specified by the user.
    - **automatic**: Use DNS servers provided by the DHCP server.
  - --server1 TEXT
    - Set the primary DNS server.

- --server2 TEXT
  - Set the secondary DNS server.
- --server3 TEXT
  - Set the tertiary DNS server.

Example Commands:

```
BN-PRO-7A93CFE2:~$ cli config dns set --mode manual --server1 10.130.33.22

DNS settings changed successfully
```

3.2 eth1

The cli config eth1 command allows users to **view and modify the network configuration** of the ETH1 interface. This includes both IPv4 and IPv6 settings, DHCP modes, manual IP assignment, and gateway configuration.

- **View Current ETH1 Settings**

To check the current ETH1 configuration, use:

```
cli config eth1 get
```

This displays the current IPv4 and IPv6 settings, including DHCP status, manual IP address, and gateway information.

- **Modify ETH1 Settings**
  To change the ETH1 configuration, use:

```
cli config eth1 set [OPTIONS]
```

Example Commands:

```
cli config eth1 set --v4-dhcp-enable

cli config eth1 set --v4-dhcp-disable --v4-manual-ip-address
10.180.98.13/24 --v4-gateway 10.180.98.254
```

!! For a full list of options, run:

```
cli config eth1 set -h
```

3.3 eth2

The cli config eth2 command is used to view and modify the network settings of the ETH2 interface. It functions in the same way as the cli config eth1 command, with identical options and structure. The only difference is that the configuration applies to the ETH2 interface instead of ETH1.

For details on available options and example usage, refer to section 3.2

3.4 hostname

The cli config hostname command allows users to view and modify the system hostname. This is useful for identifying the device on a network or assigning a meaningful name for easier management.

- **View Current Hostname**

To check the current system hostname, use:

```
BN-PRO-7A93CFE2:~$ cli config hostname get

Key       Value

--------  -----------------

hostname  "BN-PRO-7A93CFE2"
```

This displays the hostname currently assigned to the device.

- **Modify Hostname**

To change the system hostname, use:

```
BN-PRO-7A93CFE2:~$ cli config hostname set --new-hostname Bachmann

Hostname set to Bachmann successfully
```

3.5 ldapconfig

The cli config ldapconfig command allows users to configure **LDAP (Lightweight Directory Access Protocol)** settings for centralized user authentication. This enables the

PDU to authenticate users against an external LDAP server, such as **OpenLDAP** or **Microsoft Active Directory**, instead of relying solely on local accounts.

Purpose of LDAP in the PDU

LDAP integration allows:

- Centralized user management
- Role-based access control
- Easier integration into enterprise IT environments
- Secure login using credentials stored on an LDAP server

The following options for configuring LDAP are:

```
Commands:
  get                 Get LDAPCONFIG settings
  set                 Configure LDAPCONFIG settings
  testconnection      Test LDAP connection
  uploadcertificate   Upload a certificate
  uploadldaproles     upload ldaproles txt file
```

- View Current LDAP Settings:

```
BN-PRO-7A93CFE2:~$ cli config ldapconfig get
Key                   Value
--------------------  ----------------------------
enableLdap            False
serverType            "openLdap"
server                "ldap://example.com"
bindDn                "cn=admin,dc=example,dc=com"
bindPassword          "******"
sslTls                False
port                  "389"
connection_timeout    "5"
search_timeout        "5"
baseDn                "dc=example,dc=com"
searchFilter          "(objectClass=inetOrgPerson)"
searchBase            ""
userSearchScope       "SUBTREE"
groupsSearchScope     "SUBTREE"
userAttributeId       "uid"
groupFilter           "(objectClass=groupOfNames)"
groupMemberAttribute  "member"
```

The system displays the current LDAP configuration. If LDAP has not been customized yet, this output reflects the **default settings** provided by the system.

Notes

- These defaults are **not active** until enableLdap is set to true.
- You must replace placeholder values (e.g., server address, bind DN) with actual values from your LDAP infrastructure.
- SSL/TLS should be enabled (--enable-ssl true) if your LDAP server supports secure connections.

- Modify LDAP Settings:

Example Commands:

```
cli config ldapconfig set --enable-ldap True --server-type "MS Active Directory" --server "INTERNAL.SERVER..COM" --bind-dn "CN=Administrator,CN=Users,DC=internal,DC=server,DC=com" --bind-password "password143 --port "636" --enable-ssl True --connection_timeout "30" --search_timeout "30" --base-dn "DC=interanl,DC=server,DC=com" --search-filter "(objectClass=user)" --user-search-scope "SUBTREE" --group-search-scope "SUBTREE" --user-attribute-id "sAMAccountName" --group-filter "objectClass=groupOfNames" --group-member-attribute "member" enable

LDAP settings changed successfully
```

- Check ldap connection:

```
BN-PRO-7A93CFE2:~$ cli config ldapconfig testconnection

connection successful.
```

- Upload certificate:

```
BN-PRO-7A93CFE2:~$  cli config ldapconfig uploadcertificate /home/admin/ldapMs_cert.crt

certificate uploaded successfully
```

- Upload ldap roles:

```
BN-PRO-7A93CFE2:~$  cli config ldapconfig uploadldaproles /home/admin/ldap-
Mapping_Ms.txt

LDAP roles uploaded successfully
```

- The text file for the ldap roles should look like this:

```
LDAP Group,Role
BlueNetGen2_noReboot_fwReset_userManag,MS_ldapRole
```

3.6 localgui

The cli config localgui command allows users to configure settings related to the **Local Graphical User Interface (GUI)** of the PDU. This interface is accessed via the four buttons on the Control Unit (CU) and is used for local monitoring and control.

Purpose of Local GUI Configuration

This configuration controls how the **local display behaves**, including:

- **Security**: Setting a PIN to restrict access
- **Power-saving**: Automatically turning off the display after inactivity
- **Session management**: Automatically logging out inactive users
- **Display orientation**: Adjusting the screen layout based on mounting position

- View Current localgui Settings:

```
BN-PRO-7A93CFE2:~$ cli config localgui get

Key                  Value

-------------------  -------

display_timeout_min  5

logout_timeout_min   5

orientation          "auto"
```

The system displays the current localgui configuration. This output reflects the **default settings** provided by the system.

- Modify localgui Settings:

```
cli config localgui set [OPTIONS]
```

Example Commands:

```
BN-PRO-7A93CFE2:~$ cli config localgui set --pin 1994 --display-timeout 10
--logout-timeout 10 --orientation angle90

Local GUI pin changed successfully

Localgui settings changed successfully
```

!! For a full list of options, run:

```
cli config localgui set -h
```

3.7 modbus

The cli config modbus command allows users to enable or disable the **Modbus TCP server** on the PDU and configure the port it listens on. This server provides access to internal data for external systems using the Modbus protocol.

- View Current modbus Settings:

```
BN-PRO-7A93CFE2:~$ cli config modbus get

Key     Value

------  -------

active  False

port    502
```

The system displays the current modbus configuration. This output reflects the **default settings** provided by the system.

- Modify modbus Settings:

```
cli config modbus set [OPTIONS]
```

Example Commands:

```
BN-PRO-7A93CFE2:~$ cli config modbus set --enable

Modbus settings changed successfully
```

!! For a full list of options, run:

```
cli config modbus set -h
```

3.8 rcm

The cli config rcm command allows users to configure **RCM (Residual Current Monitoring)** settings on the PDU. RCM is a safety feature that performs **automatic self-tests** to detect leakage currents, helping ensure electrical safety and compliance with standards.

- View Current rcm Settings:

```
BN-PRO-7A93CFE2:~$ cli config rcm get
Key                  Value
-------------------  -------
automatic_execution  True
interval_days        365
```

The system displays the current rcm configuration. This output reflects the **default settings** provided by the system.

- Modify rcm Settings:

```
cli config rcm set [OPTIONS]
```

Example Commands:

```
BN-PRO-7A93CFE2:~$ cli config rcm set --enable-auto-selftest --interval 20
RCM settings changed successfully
```

```
BN-PRO-7A93CFE2:~$ cli config rcm set --disable-auto-selftest
RCM settings changed successfully
```

!! For a full list of options, run:

```
cli config modbus set -h
```

3.9 smtp

The cli config smtp command allows users to configure **SMTP (Simple Mail Transfer Protocol)** settings on the PDU. This enables the device to send **email notifications** to designated recipients.

Email Notifications via Signal Chain

The PDU can send **email notifications** for system events such as **alarms** and **warnings**.

To receive these notifications, the user must also configure the **Signal Chain** to include an email recipient. Specifically:

If the **email address used in the Signal Chain configuration** matches the one set in the SMTP configuration, the PDU will send notifications to that address when relevant events occur.

This means:

- SMTP handles the **sending mechanism**
- Signal Chain defines **who receives** the notifications and **under what conditions**

- View Current smtp Settings:

```
BN-PRO-7A93CFE2:~$ cli config smtp get
Key                   Value
--------------------  -------
enabled               False
host
port
user
password
sender_email_address
authentication_active  False
encryption            "none"
```

The system displays the current smtp configuration. This output reflects the **default settings** provided by the system.

- Modify smtp Settings:

```
cli config smtp set  [OPTIONS]
```

Example Commands:

```
BN-PRO-7A93CFE2:~$ cli config smtp set --enable --server smtp.1und1.de --
port 587 --sender user@noOne.de --enable-auth --username user@noOne.de --
password passwordNoOne --encryption starttls

SMTP settings changed successfully
```

```
BN-PRO-7A93CFE2:~$ cli config smtp set --disable

SMTP settings changed successfully
```

!! For a full list of options, run:

```
cli config smtp set -h
```

3.10 snmp

Simple Network Management Protocol (SNMP) is a standard protocol used to monitor and manage network devices such as switches, routers, and PDUs (Power Distribution Units). It allows administrators to query device status, receive alerts, and configure settings remotely.

SNMP supports three versions:

- **SNMPv1/v2c**: Community-based, simple to configure but less secure.
- **SNMPv3**: Adds authentication and encryption for secure communication.

- View Current snmp Settings:

```
BN-PRO-7A93CFE2:~$ cli config snmp get

Key                  Value

-------------------  -------

active_v1v2          False

active_v3            False

community_string_ro

community_string_rw

community_source_ro

community_source_rw
```

The system displays the current snmp configuration. This output reflects the **default settings** provided by the system.

- Modify snmp Settings:

```
cli config snmp set  [OPTIONS]
```

SNMPv1 and SNMPv2c use **community strings** as a form of simple authentication.

1. --community-string-ro TEXT

- **What it is**: The **read-only** community string.
- **What it does**: Allows SNMP managers to **read** data (e.g., power usage, temperature) from the device, but **not change anything**.
- **What to enter**: A string like public, monitor, or any custom word.
- **Why it matters**: This is the most common setup for monitoring tools. If someone knows this string, they can query the device.
- **Example**:
  --community-string-ro public

2. --community-source-ro TEXT

- **What it is**: The **IP address or subnet** from which SNMP read-only requests are accepted.
- **What it does**: Restricts which devices can use the read-only string.
- **What to enter**:
  - 0.0.0.0 → allow **any IP** (not secure)
  - 192.168.1.100 → allow only **one specific IP**
  - 192.168.1.0/24 → allow all devices in a **subnet**
- **Why it matters**: Helps prevent unauthorized access even if someone knows the community string.
- **Example**:
  --community-source-ro 192.168.1.0/24

3. --community-string-rw TEXT

- **What it is**: The **read-write** community string.
- **What it does**: Allows SNMP managers to **read and modify** settings on the device.

- **What to enter**: A strong, unique string like private, admin123, etc.
- **Why it matters**: This is powerful and potentially dangerous. Only use it if you need to change settings remotely via SNMP.
- **Example**:
  --community-string-rw private

4. **--community-source-rw TEXT**

- **What it is**: The **IP address or subnet** allowed to use the read-write string.
- **What it does**: Restricts which devices can send SNMP write commands.
- **What to enter**: Same format as --community-source-ro.
- **Why it matters**: Critical for security. You should **always** limit this to trusted IPs.
- **Example**:
  --community-source-rw 192.168.1.100

Example Commands:

```
BN-PRO-7A93CFE2:~$ cli config snmp set --enable-v1v2 --enable-v3 --commu-
nity-string-ro public --community-string-rw private

SNMP settings changed successfully



BN-PRO-7A93CFE2:~$ cli config snmp set --community-string-ro '' --commu-
nity-string-rw '' --community-source-ro '' --community-source-rw '' --disa-
ble-v1v2 --disable-v3

SNMP settings changed successfully
```

!! For a full list of options, run:

```
cli config snmp set -h
```

3.11 ssh

SSH (Secure Shell) is a secure way to **remotely access the device's command-line interface (CLI)** over the network. It allows engineers, administrators, or automation tools to log in and execute commands without needing physical access.

This is especially useful for:

- **Remote diagnostics** (e.g., checking logs, system status)
- **Configuration changes** (e.g., setting SNMP, network settings)
- **Scripted automation** (e.g., using Ansible or Python scripts)

SSH ensures that all communication is **encrypted**, protecting credentials and commands from being intercepted.

- View Current ssh Settings:

```
BN-PRO-7A93CFE2:~$ cli config ssh get
Key     Value
------  -------
active  True
port    22
```

The system displays the current configuration. This output reflects the **default settings** provided by the system.

- Modify ssh Settings:

```
cli config ssh set  [OPTIONS]
```

Example Commands:

```
BN-PRO-7A93CFE2:~$ cli config ssh set --disable
SSH settings changed successfully
```

!! For a full list of options, run:

```
cli config smtp set -h
```

3.12 time

The time settings are critical for:

- **Timestamping logs and events** (e.g., power failures, SNMP traps)
- **Synchronizing with monitoring systems**
- **Ensuring consistency across distributed systems**

You can either set the time manually or configure the device to sync with **NTP (Network Time Protocol)** servers.

- View Current time Settings:

```
BN-PRO-7A93CFE2:~$ cli config time get

Key             Value

--------------  --------------------------------

timezone        "UTC"

ntp             False

ntp_server1

ntp_server2

local_datetime  2025-11-27 08:49:54.927918+00:00
```

The system displays the current configuration. This output reflects the **default settings** provided by the system.

- Modify ssh Settings:

```
cli config time set  [OPTIONS]
```

Example Commands:

```
BN-PRO-7A93CFE2:~$ cli config time set --enable-ntp --server1 web.de

Time settings changed successfully
```

```
BN-PRO-7A93CFE2:~$ cli config time set --timezone Europe/Berlin --enable-
ntp  --server1 ptbtime1.ptb.de --server2 ptbtime2.ptb.de

Time settings changed successfully
```

!! For a full list of options, run:

```
cli config time set -h
```

3.13 web

This section controls the **embedded web server** running on the device. The web server allows users to access the device's **Web UI** via a browser using either **HTTP** or **HTTPS**.

This is useful for:

- **Visual configuration** of the device
- **Monitoring status** via a graphical interface
- **Secure remote access** over HTTPS

- View all fields for a complete configuration of the web Settings:

```
BN-PRO-7A93CFE2:~$ cli config web

Usage: cli config web [OPTIONS] COMMAND [ARGS]...

  Configuration Web settings

Options:

  -h, --help  Show this message and exit.

Commands:

  get                Get Web settings

  set                Configure Web settings

  uploadcertificate  Upload a certificate

  uploadkey          Upload a Key
```

To use HTTPS, the device needs a valid **SSL/TLS certificate** and **private key**.

!!!IMPORTANT!!!!

The system **only supports certificates with RSA or EC (Elliptic Curve) keys**. Other key types (e.g., DSA) are **not supported**.

- Default Web Settings:

```
BN-PRO-7A93CFE2:~$ cli config web get
Key           Value
------------  -------
http_active   False
http_port     80
https_active  True
https_port    443
```

The system displays the current configuration. This output reflects the **default settings** provided by the system.

- Modify ssh Settings:

```
cli config web set  [OPTIONS]
```

Example Commands:

```
BN-PRO-7A93CFE2:~$ cli config web uploadkey ca.key
Key uploaded successfully,after uploading the key and cert, please save the change with runing the 'cli config web set'
```

```
BN-PRO-7A93CFE2:~$ cli config web uploadkey ca.key
Key uploaded successfully,after uploading the key and cert, please save the change with runing the 'cli config web set'
```

```
BN-PRO-7A93CFE2:~$ cli config web set
Web settings changed successfully
```

!! For a full list of options, run:

```
cli config web set -h
```

4 Device Operations and Maintenance

This chapter explains how to perform essential actions that help you keep the device running smoothly and up to date. These commands are useful when you need to restart the system, update the firmware, back up or restore settings, or collect logs and diagnostic data.

You'll also learn how to:

- Reboot the device remotely without unplugging it
- Reset the system to factory settings if something goes wrong
- Install firmware updates directly from the CLI
- Saving and transferring the files that have export options
- Export logs and diagnostic files to share with support
- Back up your current configuration and restore it later if needed
- Clear measurement data (for peak values)

These actions are especially helpful during setup, maintenance, or when preparing the device for handover or support.

4.1 Rebooting the device

The cli do reboot command allows you to **restart the PDU remotely** via the command line. This is useful when applying certain configuration changes, recovering from a non-critical issue, or preparing the system for maintenance — without needing to physically access the device.

Basic Usage:

```
cli do reboot
```

After running the command, the system will prompt you for confirmation:

```
Do you really want to reboot the PDU? [y/N]:
```

- Type y and press **Enter** to proceed with the reboot.
- Press **Enter** or type n to cancel.

Note: Rebooting the PDU will temporarily interrupt access to the CLI, Web UI, and any ongoing communication (e.g., SNMP, Modbus). Power output to connected devices is **not affected**.

Rebooting a Specific PDU in a Cluster:

The command also supports the --pdu-id option:

```
cli do reboot --pdu-id <PDU-ID>
```

This allows the user to reboot a **specific PDU** in a clustered setup. However, this option is only relevant when **cluster mode(main-link)** is configured and multiple PDUs are managed together

If the user has a cluster configuration the reboot can be used as follow:

```
BN-PRO-7AD4B474:~$ cli cluster list -m

  Index   PDU-ID    Mode

-------  --------  ------

      1   AABBCCDQ  main

      2   AAABMS0R  link

      3   a0000012  link

BN-PRO-7AD4B474:~$ cli do reboot --pdu-id a0000012

Do you really want to reboot PDU a0000012? [y/N]: y

PDU reboot initiated!
```

4.2 Factory reset

The cli do fw-reset command allows the user to reset the device to its original factory settings. This will erase all user configurations, including network settings, SNMP, SSH, time settings, user accounts, and any other customizations.

Important Warning:

A factory reset is irreversible. All settings will be lost, and the device will return to its default state. Make sure to export your configuration first if you want to restore it later (see Section 3.5 Backup and restore configuration).

Basic Usage:

```
cli do fw-reset
```

After running the command, the system will prompt you for confirmation:

```
Reset all settings to factory defaults? [y/N]:
```

- Type y and press **Enter** to proceed with the reset.
- Press **Enter** or type n to cancel.

Resetting a Specific PDU in Cluster:

Like the reboot command, cli do fw-reset also supports the --pdu-id option:

```
cli do fw-reset --pdu-id <PDU-ID>
```

This allows you to reset a **specific PDU** in a clustered environment.

4.3 Saving and Transferring Files

The device supports multiple options for saving exported files such as **diagnostic archives, settings backups, event logs, and RCM logs**. This chapter explains the general storage locations and transfer methods, which apply to all export commands.

4.3.1 Storage Location

Exported files can be written to three different types of storage:

1. **Local storage**
    - Example path: `/home/admin/`
    - File is stored directly on the device's internal storage.
    - Requires SCP or similar tools to transfer to a PC.
2. **MicroSD card**
    - Path: `/run/media/sd/`
    - The file is written to an inserted microSD card.
    - The card can be removed and read directly on a computer.

3. **USB storage**
    - Path: `/run/media/usb/`
    - The file is written to an attached USB drive.
    - The drive can be disconnected and read on a computer.

Important: Always provide a **full file path** including the filename (e.g., `/run/media/usb/eventLogs_USB_admin.csv`). If only a folder is specified, the export command will fail with an error.

4.3.2 File Transfer with SCP

When saving to **local storage** (/home/admin/), the file must be copied to the user's computer using **SCP (Secure Copy Protocol)**.

- **From device to computer (exported file):**

```
scp admin@192.168.1.10:/home/admin/device_backup.zip "C:\Users\YourName\Backups"
```

    - Replace the IP address and paths as needed.
    - Enter the device password when prompted.
    - The file will be saved in the chosen folder on your PC.

- **From computer to the device (imported files)**

```
scp "C:\Users\YourName\Backups\device_backup.zip" admin@192.168.1.10:/home/admin/
```

4.3.3 Supported Export Types

The following commands can export data to the locations mentioned above

The cli helper can help

```
cli export -h
```

Output from cli:

```
diagnosis  Export diagnostic file as a ZIP archive
```

```
  log              Export event log
  rcm-log    Export rcm selftest log
  settings   Export settings file as an AES encrypted ZIP archive
```

Each export command follows the same pattern:

```
cli export <command> <options> <target file path>
```

Examples:

- Save event log to USB drive:

```
cli export log /run/media/usb/eventLogs_USB_admin.csv
```

- Save diagnostic data to microSD card:

```
cli export diagnosis /run/media/sd/diagnostics_admin.zip
```

- Save settings to local storage:

```
cli export settings /home/admin/device_backup.zip -p MySecurePassword123
```

- Save rcm-log to USB drive

```
cli export rcm-log /home/admin/rcmLogs_USB_admin.csv
```

4.4 Firmware update

Firmware updates are essential for keeping the device secure, stable, and feature complete. The CLI provides a straightforward way to install a new firmware image using the cli do fw-update command.

Before running the update, the firmware file must be transferred to the device — typically using **SCP (Secure Copy Protocol)** see section ***4.3.2 File Transfer with SCP***

*S***tep 1:** Once the file is on the device, run:

```
cli do fw-update /home/admin/firmware.raucb
```

This command starts the upload process. The progress message will appear as follows:

```
Starting firmware image upload
Upload completed
  0% installing
  0% Installing
  0% Determining slot states
 20% Determining slot states done.
…
99% Updating slots done.
100% Installing done.
Successfully installed update. Device will restart now...
```

After installation, the device will **automatically reboot** to apply the update.

During the Update:

- The system verifies the firmware signature and bundle contents.
- It determines the correct installation slot.
- The image is copied to the appropriate root filesystem partition.
- Once complete, the device reboots and runs the new firmware.

4.5 Exporting diagnostic data

The cli export diagnosis command allows generating a **diagnostic archive** containing system logs, configuration, and other technical data. This file is useful for troubleshooting, internal analysis, or sharing with support teams.

Step1: check available options

```
cli export diagnosis -h
```

The options are

--encrypted / --unencrypted Export diagnostic file as encrypted/unencrypted

ZIP archive

Step 2: Export the Diagnostic File locally

```
cli export diagnosis --unencrypted /home/admin/diagnosis.zip.gpg
```

- --unencrypted creates a readable ZIP file (with .gpg extension).
- Users can also use --encrypted to protect the contents

Note: A **full file path must be provided**, not just a directory. If only a folder is specified (e.g., /home/admin/), the command will fail with an error.

Once successful, it should appear:

```
Diagnostic file written to: /home/admin/diagnosis.zip.gpg
```

Step 3: Export the diagnostic file on SD card or USB drive

See section 4.3.3 Supported Exported types

Step 4: Transfer the File to the Local Machine

See section 4.3.2 File Transfer with SCP

4.6 Backup and restore configuration

The CLI provides commands to back up and restore the device configuration. Configuration backups are stored as **AES-encrypted ZIP archives**, ensuring that sensitive settings remain protected. These archives can later be re-imported to restore the configuration to a known working state.

4.6.1 Export settings

The `cli export settings` command allows generating a backup file containing all current configuration settings.

Step 1: Check available options:

```
cli export settings -h
```

➢ The option will be listed in more detail, and the user will be informed of what is needed, a path and a password:

Step 2: Export the settings file:

- A **full file path must be provided** (not just a folder).
- Use the --password (or -p) option to set the AES encryption password.

Example:

```
cli export settings –password 1111 /home/admin/settingsCLI_admin.zip
```

The answer to the output should be:

```
Settings written to: /home/admin/settingsCLI_admin.zip
```

Notes:

- The password must be remembered, as it is required when importing the backup.
- If only a directory is specified (e.g., /home/admin/), the command will fail with an error: “Error: [Errno 21] Is a directory: '/home/admin/'”

Step 3: Transfer the File to The Local Machine:

See section ***4.3.2*** File Transfer with SCP

Step 4: Export the settings file on SD card or USB drive

See section 4.3.3 Supported Export Types

Step 5: Viewing Backup Contents on Windows

- The exported archive is **AES-encrypted** and cannot be opened with the default Windows Explorer ZIP utility.
- Attempting to open with Explorer results in error *0x80004005*
- To extract the file, use one of the following tools:
  - 7-Zip → Right-click file → *7-Zip* → *Extract Here* → enter password.
  - WinRAR → Right-click file → *Extract to …* → enter password.
- **Tip:** Extraction is not required if you only intend to restore the configuration on the device. You can directly use the .zip file for cli import settings.

4.6.2 Import settings

The `cli export settings` command restores configuration from a previously exported backup file.

Step 1: Check Available Options

- The option will be listed in more detail, and the user will be informed of what is needed, a path and a password:
  `cli export settings -h`

Step 2: Import the Settings File

- Run the command with the correct file path and password:

```
cli import settings /home/admin/settingsCLI_admin.zip -p 1111
```

4.7 Export log

The command allows exporting of the device event log for analysis, troubleshooting, or archiving. The log can be saved in **CSV** (default) or **JSON** format.

Step 1: Check available options:
`cli export log -h`

- The option will be listed in more detail, and the user will be informed of what is needed: a **path** (full file path, not just a folder) and optionally the **--json flag** if JSON format is preferred.

Step 2: Export the event log

- **A full path must be provided not just a folder**
  Example:

  1. Export to local storage:

```
cli export log /home/admin/eventLogs_local.csv
```

  The answer to the output should be:

```
Event log written to: /home/admin/eventLogs_local.csv
```

  2. Export on the SD card or USB Drive:

  See section 4.3.3 Supported Export Types

Step 3: Transfer the file to the local machine:

See section **4.3.2** File Transfer with SCP

Notes:

- Use CSV format for spreadsheets (Excel, LibreOffice) and JSON format for log analysis tools.

4.8 Export rcm-log

The command allows exporting the device **RCM selftest log** for diagnostic and verification purposes. The log can be saved in **CSV** (default) or **JSON** format.

Step 1: Check available options:

```
cli export rcm-log -h
```

- The option will be listed in more detail, and the user will be informed of what is needed: a **path** (full file path, not just a folder) and optionally the **--json flag** if JSON format is preferred.

Step 2: Export the rcm selftest log

- **A full path must be provided not just a folder**
- **Use the -j (or --json) option if JSON format is required, otherwise the file will be exported as CSV by default.**

Example:

1. Export to local storage:

```
cli export rcm-log /home/admin/rcmLogs_local.csv
```

The answer to the output should be:

```
RCM selftest log written to: /home/admin/rcmLogs_local.csv
```

2. Export on the SD card or USB Drive:

See section 4.3.3 Supported Export Types

Step 3: Transfer the file to the local machine:

See section **4.3.2** File Transfer with SCP

4.9 Resetting Measurements

The command “cli export” allows resetting individual **measurement values** for logical elements in the device. This is useful when the user wants to reset peak values.

Step 1: Check available options:

The following command can be used: “cli export -h”

- The option will be listed in more detail, helping the user to understand how the syntax for resetting a measurement is used

Step 2: List available measurements

The following command can be used:

```
cli get -a -
```

This will print all elements with their measurement values.

!!Tip:
The resettable values are listed in cli with the word “max”, the user can search only after these measurements using the following command:

```
cli get -a - | grep max
```

This will highlight all measurements that can be reset (e.g., current_max, voltage_max, power_active_max).

Notes:

- **Resetting a measurement only clears the stored value; it does not disable future monitoring.**
- **Measurements without a resettable counter will return an error if used with cli reset.**
- **Using the grep max trick helps identify all measurements that are safe to be reset.**

Tip on grep:
The grep command is a standard Linux tool used to filter text output.

- **Example:**
- **cli get -a | grep max**

Here, cli get -a lists all measurements, and grep max only shows the lines that contain the word max (e.g., current_max, voltage_max).

- **This is useful for quickly finding all resettable peak counters without scrolling through the full measurement list.**

Step 3: Run the Reset Command

Example – reset the peak current of phase 1 on inlet 1:

```
BN-PRO-7A93CFE2:~$ cli reset /pdu[AAABMS2O]/core/inlet[1]/phase[1] cur-
rent_max

Reset successful
```

5 User and Role Management

The CLI provides commands to configure **users, roles, and LDAP groups**, which together form the device's access control and security model. These features allow administrators to manage **who can log in** and **what permissions they have** when interacting with the PDU.

- **Users** represent individual accounts that log in to the device. Each user is assigned to one role that defines their permissions. Typical actions include creating new users, editing account details, changing passwords, and removing accounts when no longer needed.
- **Roles** define sets of permissions (called *rights*) that control access to different device functions. For example, a role may allow only viewing measurements, or it may grant full control over firmware updates, outlet configuration, or security settings. Administrators (***admin*** users) can create, edit, and delete roles, then assign them to users or LDAP groups.
- **LDAP Groups** integrate with external directory services (such as Microsoft Active Directory or OpenLDAP). By mapping LDAP groups to roles, organizations can centralize authentication and authorization, reducing the need to manage local accounts for every user.

Together, these three components ensure that access to the device is **secure, granular, and flexible**, supporting both local standalone management and centralized enterprise environments.

5.1 User management

The cli user command is used to manage local user accounts on the device. Local accounts define who can log in and what **roles** (permissions) they have.

Available actions:

- Add a new user (cli user add)
- List all users (cli user list)
- Edit an existing user (cli user edit)
- Delete a user (cli user delete)
- Change a password (cli user passwd)

The system provides a built-in mechanism for managing authentication and authorization through **users**, **roles**. Each user is assigned to a role, and roles define the rights available to that user.

Predefined Accounts and Roles

The system includes two predefined roles and one predefined user:

Name	Type	Rights / Description	Editable	Deletable	Notes
admin	CLI User	Full rights (linked to the admin role)	Password only	❌	Default user account; cannot be renamed or removed
admin	CLI Role	Grants all available rights (full system access)	❌	❌	Always present; cannot be renamed or removed
operator	CLI Role	Limited rights for basic operations	✅	✅	Can be customized or deleted
SNMP users	SNMP Credentials	Used for SNMPv3 authentication (not for CLI login)	✅ (via cli config snmp user set)	✅	Managed separately with cli config snmp user commands

Important distinction:

- The **CLI admin user** is system-protected and only its password can be changed.
- The **SNMP user admin** (or any SNMP user) is different: it can be created, updated, or deleted using cli config snmp user.

5.1.1 user add

The cli user add command creates a new account. There are two ways to add a user:

- **Method 1: Interactive Creation (Prompt-Based)**

  Run the command with only the --username parameter (or no parameters at all).

  The CLI will prompt for each field one by one.

  Example:

```
cli user add
```

The system then asks for the details interactively:

```
Username: userOperator
Password:
Repeat for confirmation:
Role: admin
Description []: hello
Mail []: bc.ro@yahoo.com
Successfully created new user
```

This method is convenient when the user wants to enter details step by step while the password is unrevealed when typing.

- **Method 2: One-Line Creation (All Arguments Provided)**

  The user can also provide all necessary fields directly in a single command. This is useful for scripting or automation.

  Example:

```
cli user add -u user1 -p password -r operator -d "main user" -m mail.u-
ser@de.com --ssh-key 'ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAIJq0slSbeWdBAoR-
WXUqY3axnjanrxpuWnRE9vToor/Vb mail.user@DEBWS1-NW0428'

Successfully created new user
```

Notes:

- The –role, --user, password options are **mandatory**. A user without a role cannot be created.
- The --ssh-key option allows adding one or multiple SSH keys for secure, passwordless login. In the example shown there is only one ssh key added. The user can add multiple ssh keys by simply accessing as much as it is needed the command –ssh-key in the creation of the user.
- If no description or mail is provided, these fields remain empty.

5.1.2 user edit

The `cli user edit` command modifies an existing user account. This can include changing the password, role, description, email, or managing SSH keys.

Step1: Check available Options:

```
cli user edit -h
```

Step 2: Run the edit command

Examples:

```
cli user edit user1 --password user1pass

cli user edit user1 -d "Project Owner" -m owner@mail.com

cli user edit user1 --clear-ssh-key "ssh-ed25519
AAAAC3NzaC1lZDI1NTE5AAAA..."

cli user edit user1 --clear-ssh-keys
```

Notes:

- The username to edit is always required as the first argument.
- Multiple changes can be applied in one command (e.g., update password, role, and description simultaneously).
- SSH key management supports fine control: add new keys, remove selected key, or clear them all.
- If no option is given, the command does nothing and displays *"No changes were specified"*.

5.1.3 user list and user delete

The CLI allows administrators to remove existing user accounts. This is useful when a user no longer requires access to the device.

Step 1: Check Existing Users (user list)

Before deleting, the current user can check which users are already in the system using the command:

```
cli user list
```

Step 2: Delete the user

Provide the username of the account you want to remove:

```
BN-PRO-7A93CE08:~$ cli user delete user1

Successfully deleted user: user1
```

Notes:

- The **default admin user cannot be deleted**.
- The **operator user can be deleted**.

- Deleting a user is **irreversible**. To restore access for the same person, you must create the account again with cli user add.

5.1.4 user passwd

The `cli user passwd` command allows the **currently logged-in user** to change their own password. This ensures secure management of credentials without requiring administrator intervention.

Step 1: Run the command

Execute the command without arguments to start an interactive password change:

```
cli user passwd
```

The system will prompt for:

- **Password** → Enter the new password.
- **Repeat for confirmation** → Re-enter the same password.

If successful, the system responds with:

Password changed successfully

```
BN-PRO-7A93CE08:~$ cli user passwd
Password:
Repeat for confirmation:
Password changed successfully
```

Step 2: Run the command with Command-Line Argument

The user can specify the new password directly in the command:

```
BN-PRO-7A93CE08:~$ cli user passwd -p "admin"
Password changed successfully
```

5.1.5 auth login

➢ **Purpose:**

The `cli auth login` command is used to authenticate a user and obtain a new **login session token (JWT)**.
A valid login is required before executing commands that change configuration or access restricted system functions.

This command confirms your username and password and establishes a secure session for CLI access.

- *Basic Interactive Login*

If the user enters the commands without arguments, the system will ask for credentials:

```
cli auth login
Username: admin
Password:
Logged in successfully
```

This method is recommended for local or direct console use and keeps the password hidden while typing.

- *Login with Parameters (non-interactive)*

The user can provide the username and the password directly in the command:

```
cli auth login -u admin -p admin
Logged in successfully
```

Or in reversed option order (both are valid):

```
cli auth login -p admin -u admin
Logged in successfully
```

- *Login Output (JSON Mode)*

  If the -j option is used during login, the CLI returns a **JSON formatted response** instead of a standard success message.

This response contains a **JWT authentication token**, which represents the active login session.

Example:

```
BN-PRO-7A93CFE2:~$ cli auth login -p admin -u admin -j

{"token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJhZG1pbiIsIm-
lhdCI6MTc2NDE2NzA1MCwiaXNzIjpudWxsfQ.6tzsZIFsCkJnokef9GUE-
AEht77LE3Mc3PCTgNAqCUx0"}
```

This means:

- The user **logged in successfully**
- Instead of the text *"Logged in successfully"*, the system prints a **token**
- This token is proof of authentication and can be used for automated access

What is a Token?

- The token is a **temporary digital proof of authentication**
- It replaces the need to enter username/password repeatedly
- It can be used for CLI automations or REST API commands

  The token should be treated like a password.

When should the JSON login (-j) be used?

The JSON output mode is useful mainly for automation, scripting, and API usage. Normal users logging in through the CLI do **not** need it.

5.2 Role management

Roles define the permissions available to users within the system. A role groups together multiple rights, such as reading measurements, editing configuration, or

performing maintenance actions. Each user must be assigned to exactly one role, and the rights of the user are determined by the role they belong to.

The system includes two predefined roles:

- **admin** → grants full system access with all available rights.
- **operator** → provides limited rights suitable for basic operations.

In addition, the system contains one predefined user account:

- **user admin** → linked to the admin role, always present in the system. This account cannot be renamed or deleted. Only the password can be changed.

SNMP credentials are configured separately and apply only to SNMPv3 authentication. They do not provide CLI login access but are managed in a similar way to users.

The `cli role` command is used to manage role-based access control. Roles define what operations a user can perform and on which parts of the system. Each user must be assigned to a role, and the rights within that role determine the user's permissions.

Available Rights Table

The following rights can be assigned to a role:

Right	Description
get-phys	Read logical element and all subelements
set-phys	Control logical element and all subelements
conf-phys	Configure logical element and all subelements
conf-alarm	Configure alarms for logical element and all subelements
conf-signal	Configure signal chains (add, edit, delete)
reset-values	Reset measurements for logical element and all subelements

Right	Description
test-rcm	Execute RCM selftest
conf-group	Configure outlet groups for logical element and all subelements
conf-sequence	Configure power-up sequence
conf-critical	Configure outlets as critical
conf-system	Configure system configuration
update-fw	Perform firmware updates
settings	Backup and restore settings
reboot	Reboot the PDU
factory-reset	Restore factory defaults
conf-cluster	Configure PDU cluster
access-console	Access via SSH/Debug console
access-web	Access the web frontend
access-snmp	Access SNMPv3
get-log	Read event log
reset-log	Reset event log
get-diag	Download diagnostic file
conf-cron	Configure cron jobs

Right	Description
conf-security	Configure security settings (e.g., password rules)
user-management	Manage users and roles (create, edit, delete, assign)
self-profile-management	Self profile management such as edit password, ssh keys, language ,email, Description

Available Subcommands

- add → Create a new role
- edit → Modify an existing role
- delete → Remove an existing role
- list → Display all roles and their assigned rights

5.2.1 role add

The `cli role add` command is used to create new roles and assign specific rights to them. Each role defines what operations a user assigned to that role is permitted to perform.

General Syntax:

```
cli role add [OPTIONS] ROLE_NAME
```

- ROLE_NAME is mandatory. It defines the identifier of the new role.
- If no ROLE_NAME is provided, the CLI will return an error.
- If a role is created **with only a name** and without additional options, it will exist but contain **no rights** until further configured.

Option	Description
-a, --allow-address TEXT	Restrict the rights to one or more logical element addresses (all child elements included). If omitted, rights apply globally. Multiple addresses can be specified by repeating the option.
-e, --enable-rights <rights>	Comma-separated list of rights to grant (see Section 5.2 for all available rights). If omitted, the role will have no permissions.
-h, --help	Display help for this command.

Examples:

1. **Create a role with no rights (base role only):**

```
BN-PRO-7A93CFE2:~$ cli role add mainRole

Successfully created new role
```

The role mainRole is created but has no rights assigned.

2. Create a role with more rights and all elements enabled:

```
BN-PRO-7A93CFE2:~$ cli role add CLI_role1 -a /pdu[AAABMS2O] -e get-
phys,set-phys,conf-p

hys,conf-alarm,reset-values,test-rcm,conf-group,conf-sequence,conf-criti-
cal,conf-system

,update-fw,settings,reboot,factory-reset,conf-cluster,access-console,ac-
cess-web,access-

snmp,get-log,reset-log,get-diag,conf-security,user-management,conf-signal

Successfully created new role
```

Notes:

- A role name is the starting point: it defines the role object in the system. Without it, the command fails.

- Rights (-e) and addresses (-a) can be combined to build granular access models.

- A role with no rights is valid but grants no access until updated.
- The `cli role list` command can be used afterward to verify role creation and rights.
- Check the “Available Rights Table” to spot all rights that can be grant to a role

Restriction on conf-cron Right

- The right conf-cron (configure cronjobs) is **reserved exclusively for the built-in admin user**.
- Even if this right is included when creating or editing other roles, it will **not take effect**.
- No other user accounts can be assigned conf-cron.

5.2.2 role list

The `cli role list` command displays all defined roles, along with their associated rights and the element addresses where these rights apply.

Example output:

```
Name:                admin
--------------------------
Elements:
    /
Rights:
    get-phys: True
    set-phys: True
    ...
```

```
    user-management: True
    self-profile-management: True

Name:                operator
-----------------------------
Elements:
    /
Rights:
    get-phys: True
    set-phys: False
    access-web: True
    ...

Name:                mainRole
-----------------------------
Elements:
Rights:
    get-phys: False
    ...
    self-profile-management: False

Name:                CLI_role1
------------------------------
Elements:
    /pdu[a0000017]
Rights:
    get-phys: True
```

```
    set-phys: True
    ...
    user-management: True
    conf-cron: False
```

Notes on interpretation:

- **admin** is predefined and always has all rights enabled.
- **operator** is predefined with very limited rights. This can be deleted or edited
- Custom roles (e.g., mainRole, CLI_role1) are defined by the administrator and can have any combination of rights and addresses.
- If no rights are specified during creation, the role will exist but without permissions (as shown for mainRole).

5.2.3 role edit

The `cli role edit` command modifies an existing role.

It can enable or disable rights or add/remove addresses that define the scope of rights.

Example:

```
cli role edit example_role --disable-rights set-phys
```

```
cli role edit example_role --delete-address /pdu[ESIzRKpV]/core/inlet[1]
```

Important:

- If both enable and disable flags are given for the same right, the **disable** action takes precedence.

- Predefined role admin cannot be renamed or deleted.

5.2.4 role delete

The `cli role delete` command removes an existing role from the system.

Example:

```
BN-PRO-7A93CE08:~$ cli role delete operator
Successfully deleted role: operator
```

Constraints and Notes:

- Predefined role admin **cannot be deleted**.
- If a role is still assigned to one or more users, the system prevents deletion and returns an error:

```
BN-PRO-7A93CE08:~$ cli role delete CLI_role1
Error: HTTP status code 403 (Forbidden)
    The following users are still assigned to the role: ['user2']
```

5.3 Ldap-group

The `cli ldap-group` command is used to manage LDAP groups on the device. LDAP groups allow centralized user management and role assignment by linking users in an external LDAP directory to specific roles on the system.

General Usage:

```
cli ldap-group [OPTIONS] COMMAND [ARGS]...
```

Available Commands:

Command	Description
add	Create a new LDAP group
delete	Delete an existing LDAP group
edit	Edit an existing LDAP group
list	List all LDAP groups and their associated roles

Notes:

- The cli ldap-group menu only manages group definitions and role assignments. User authentication and membership are handled by the external LDAP server.
- LDAP groups are particularly useful in environments where multiple users share common roles or permissions. Assigning roles via LDAP groups ensures consistent access control without configuring individual users manually.

5.3.1 ldap-group add

The `cli ldap-group add` command is used to create a new LDAP group on the device and assign a role to it. This allows users belonging to the LDAP group to inherit the permissions of the assigned role

General Syntax:

```
cli ldap-group add [OPTIONS] GROUP_NAME
```

Examples:

```
BN-PRO-7A93CE08:~$ cli ldap-group add BNG2_readOnlyUser --role admin
Successfully created new LDAP group: BNG2_readOnlyUser
```

```
BN-PRO-7A93CE08:~$ cli ldap-group add BNG2_readOnlyUser
Usage: cli ldap-group add [OPTIONS] GROUP_NAME
Try 'cli ldap-group add -h' for help.
Error: Missing option '--role'.
```

```
BN-PRO-7A93CE08:~$ cli ldap-group add --role admin
Usage: cli ldap-group add [OPTIONS] GROUP_NAME
Try 'cli ldap-group add -h' for help.
Error: Missing argument 'GROUP_NAME'.
```

5.3.2 ldap-group list

The `cli ldap-group list` command displays all LDAP groups currently defined on the device along with the roles assigned to each group. This is useful to quickly verify group existence and role assignments.

Exmple:

```
BN-PRO-7A93CE08:~$ cli ldap-group list
Group Name:         BNG2_readOnlyUser
Role:               admin
```

Notes:

- Each LDAP group is listed with its corresponding role.
- Roles assigned to LDAP groups determine the permissions inherited by users who are members of those groups.
- If no groups exist, the command will return an empty list:

```
BN-PRO-7A93CE08:~$ cli ldap-group list
No LDAP groups found.
```

5.3.3 ldap-group edit

The `cli ldap-group edit` command allows modifying an existing LDAP group on the device, such as changing the role assigned to the group.

Example:

```
BN-PRO-7A93CE08:~$ cli ldap-group edit BNG2_readOnlyUser --role CLI_user1
Successfully edited LDAP group: BNG2_readOnlyUser with new role: CLI_user1
```

Notes:

- Editing a group only changes its assigned role; it does not affect the users in the external LDAP server.
- The new role must exist on the device. Use cli role list to verify available roles.
- Changes take effect immediately for all users who are members of the LDAP group.

5.3.4 ldap-group delete

The command removes an existing LDAP group from the device.

Example:

```
BN-PRO-7A93CE08:~$ cli ldap-group delete BNG2_readOnlyUser

Successfully deleted LDAP group: BNG2_readOnlyUser
```

Notes:

- The LDAP group must exist; otherwise, the CLI will return an error.

```
BN-PRO-7A93CE08:~$ cli ldap-group delete BNG2_r

Error: HTTP status code 404 (Not Found)

    LDAP group BNG2_r does not exist!
```

- Deleting a group does **not** affect the users in the external LDAP server, but they will no longer inherit the assigned role on the device.

6 Cronjob management

Cronjobs provide a mechanism to schedule and automate recurring tasks directly on the device. This feature is available **only through CLI** and is not supported in the web frontend.

With cronjobs, only the predefined admin user can define commands that run at fixed times, dates, or intervals, such as:

- Exporting log or diagnostic data on a regular basis.
- Scheduling automated reboots outside business hours.
- Running periodic monitoring or reporting commands.

Each cronjob is defined by three key elements:

- **Schedule** – A time-based expression (minute hour day month weekday) that specifies when the job should run.
- **Command** – The exact CLI command or script to be executed.
- **Identifier** – A unique label to manage the cronjob (used when editing, deleting, or running cronjobs manually).

Cronjobs follow standard scheduling syntax, offering full flexibility.

Key characteristics:

- **Automation** – Reduce manual effort and ensure consistent execution of tasks.
- **Logging (optional)** – Output of commands can be redirected to files for auditing or troubleshooting.
- **Persistence** – Cronjobs remain configured across device reboots until explicitly deleted.

- **CLI-only** – Configuration, management, and execution of cronjobs are possible only from the CLI.

This chapter covers the following operations:

- Adding new cronjobs.
- Listing existing cronjobs.
- Editing cronjobs.
- Deleting cronjobs.
- Running cronjobs manually.

6.1 cronjob add

The `cli cronjob add` command allows scheduling automated tasks on the device. Each cronjob requires a schedule, a command to execute, and a unique identifier. Optionally, the command output can be redirected to a file.

Usage:

```
cli cronjob add --schedule SCHEDULE --command COMMAND --identifier IDENTI-
FIER
```

See the command helper for much understanding

cli cronjob add -h

Examples:

1. This job executes the cli user list command every minute.

```
BN-PRO-7A93CE08:~$ cli cronjob add --schedule "* * * * *" --command "cli
user list >> /home/admin/admin_list.log 2>&1" --identifier "UserList"

Cron job 'UserList' added successfully.
```

- The output is saved to /home/admin/admin_list.log.
- Redirection (>> /path/to/file 2>&1) is optional.

2. This job executes the reboot command at 15:30 (3:30 PM) every Monday and Wednesday.

```
BN-PRO-7A93CE08:~$ cli cronjob add --schedule "30 15 * * 1,3" --command
"echo y | cli do reboot " --identifier "rebootThePDU"

Cron job 'rebootThePDU' added successfully.Cron job 'UserList' added suc-
cessfully.
```

- No logs are saved for this job.

Notes:

- Cron schedules use the standard format: minute hour day-of-month month day-of-week.
- Each identifier must be unique. Attempting to add a job with an existing identifier will return an error.
- Commands are executed by the system shell; you can use standard shell redirections to store logs.

6.2 cronjob list

The cli cronjob list command shows all configured cron jobs on the device.

- If no cron jobs are present, the output will indicate this clearly.
- If cron jobs are configured, each job is displayed with its **command**, **schedule**, and **identifier**.

Example – No Jobs Configured:

```
BN-PRO-7A93CE08:~$ cli cronjob list
```

```
There are no jobs.
```

Example – Jobs Configured:

```
BN-PRO-7A93CE08:~$ cli cronjob list

----------------------------------------

Command:   cli user list >> /home/admin/admin_list.log 2>&1

Schedule:  * * * * *

Identifier:   UserList

----------------------------------------

Command:   echo y | cli do reboot

Schedule:  30 15 * * 1,3

Identifier:   rebootThePDU

----------------------------------------
```

Each cron job is separated by a dashed line (----------).

- **Command**: The CLI or shell command that will be executed.
- **Schedule**: The execution schedule in cron format.
- **Identifier**: Unique name of the cron job, used for editing or deleting.

6.3 cronjob edit

The `cli cronjob edit` command allows modifying the schedule or command of an existing cronjob. The cronjob is identified by its **identifier** (name).

See helper command for more information how it can be used

```
cli cronjob edit -h
```

Example – Updating Schedule:

The following example updates the schedule of the job UserList to run only at **midnight every day**:

```
BN-PRO-7A93CE08:~$ cli cronjob edit --identifier UserList --new-schedule "0 0 * * *"
Cron job updated successfully.
```

6.4 cronjob delete

The `cli cronjob delete` command permanently removes a cronjob from the system. Jobs are identified by their **identifier** (name).

Syntax:

```
cli cronjob delete --identifier <name>
```

--identifier (required): The unique identifier of the cronjob to delete.

Example:

```
BN-PRO-7A93CE08:~$ cli cronjob delete --identifier UserList
Cron job deleted successfully.
```

Note:

- Deletion cannot be undone. To restore a job, it must be created again with cli cronjob add.
- Identifiers are unique; if the given identifier does not exist, an error will be shown.

6.5 cronjob run

The cli cronjob run command allows manual execution of an existing cron job without waiting for its scheduled time. This is especially useful for testing whether the job is configured correctly.

Syntax:

```
cli cronjob run --identifier <name>
```

--identifier (required): The unique identifier of the cronjob to run.

Example:

Running the cronjob rebootThePDU:

```
BN-PRO-7A93CE08:~$ cli cronjob run --identifier rebootThePDU
Connection to 10.150.37.69 closed by remote host.
Connection to 10.150.37.69 closed.
```

This confirms that the PDU was rebooted as defined in the cronjob's command.

Notes:

- Manual execution respects the command defined in the cronjob exactly as scheduled.
- Effects are immediate. For example, if the cronjob is a reboot task, the reboot will occur instantly when executed manually.

7 Get and Set Commands

The `cli get` and `cli set` commands are **core functionalities of the CLI** that allow the user to:

- **Read data** from the PDU (cli get) such as measurements, statuses, or alarm conditions.
- **Control outputs** of the PDU (cli set) such as switching outlets or digital outputs on or off.

These commands are widely used for:

- **Monitoring** live measurement values (e.g., current, voltage, power).
- **Checking alarms** and their active/inactive status.
- **Performing control actions** (e.g., enabling/disabling outputs).
- **Automation** by combining them with signal chains or scripts.

Key difference:

- cli get works in a **read-only mode**, fetching information without altering the PDU state.
- cli set is **write-oriented**, changing the state of a control and therefore affecting the behavior of the device.

Both commands accept a **logical element address** as input (see Chapter 7: Element management), which ensures that actions are applied precisely to the intended measurement or control point.

7.1 Get measurements and alarm status (cli get)

The `cli get` command is used to **read measurements and check alarm status** from logical elements of the PDU.

It is the primary command for retrieving live system data, such as:

- Electrical measurements (voltage, current, power, energy, frequency).
- Alarm states (active/okay).
- Group measurements (aggregated over outlet groups or other logical elements).

Syntax:

```
cli get [OPTIONS] PREFIX [MEASUREMENT]
```

- **PREFIX** – Path to the logical element (see Chapter 7: Element Management). Use - to request measurements from all elements.
- **MEASUREMENT** – Optional filter to return only one measurement type (e.g., current, voltage).

Examples:

1. Get all available measurements

This returns all available measurements across the PDU.

```
BN-PRO-7A93CFE2:~$ cli get –
Address                        Value                    Unit
/pdu[AAABMS20]/core/inlet[1]/energy_active          16.0     kWh
/pdu[AAABMS20]/core/inlet[1]/frequency               50.1     Hz
…….                            …….                      …….
/pdu[AAABMS20]/status          okay (active)
```

2. Get all measurements with alarm status

This adds the **Alarm** column to the output:

```
BN-PRO-7A93CFE2:~$ cli get -a -
Address                         Value  Unit           Alarm
/pdu[AAABMS20]/core/inlet[1]/current 0.0 A      warning low
/pdu[AAABMS20]/core/inlet[1]/current_neutral 0.0  A okay
……….                              ….       ….        ….

/pdu[AAABMS20]/status                           alarm in children
```

3. Get measurements of a specific element

```
BN-PRO-7A93CFE2:~$ cli get -s /pdu[AAABMS20]/core/inlet[1]/phase[1] voltage
Address                                 Value             Unit

-----------------------                   -------           ------
/pdu[AAABMS20]/core/inlet[1]/phase[1]/voltage        232.3  V
```

4. Get a specific measurement type for an element

```
BN-PRO-7A93CFE2:~$ cli get /pdu[AAABMS20]/core/inlet[1]/phase[1]/outlet[4]
Address
                                Value     Unit
-------------------------------------------------------------------
-------------  ------
/pdu[AAABMS20]/core/inlet[1]/phase[1]/outlet[4]/current
0.0      A
/pdu[AAABMS20]/core/inlet[1]/phase[1]/outlet[4]/current_max
0.0    A
/pdu[AAABMS20]/core/inlet[1]/phase[1]/outlet[4]/energy_active
0.1    kWh
/pdu[AAABMS20]/core/inlet[1]/phase[1]/outlet[4]/energy_active2
0.0    kWh
```

```
/pdu[AAABMS2O]/core/inlet[1]/phase[1]/outlet[4]/energy_apparent
6.6   kVAh

/pdu[AAABMS2O]/core/inlet[1]/phase[1]/outlet[4]/energy_reactive
0.1  kvarh

/pdu[AAABMS2O]/core/inlet[1]/phase[1]/outlet[4]/power_active
0   W

/pdu[AAABMS2O]/core/inlet[1]/phase[1]/outlet[4]/power_active_max
0  W

/pdu[AAABMS2O]/core/inlet[1]/phase[1]/outlet[4]/power_apparent
0  VA

/pdu[AAABMS2O]/core/inlet[1]/phase[1]/outlet[4]/power_factor
1.000

/pdu[AAABMS2O]/core/inlet[1]/phase[1]/outlet[4]/power_factor_suffix

/pdu[AAABMS2O]/core/inlet[1]/phase[1]/outlet[4]/power_reactive
0  var

/pdu[AAABMS2O]/core/inlet[1]/phase[1]/outlet[4]/status            okay (ac-
tive)
```

Notes:

- Using -a is particularly useful during **troubleshooting**, since alarms (e.g., thresholds exceeded) are displayed alongside the measurements.
- The **JSON output** option is recommended if the results are processed by scripts or monitoring systems.

7.2 Set state of control (cli set)

While "`cli get`" is used to **read measurements**, the `cli set` command is used to **control outputs and device states**.
It allows the user to change the operational state of logical elements in the PDU, such as **outlets, digital outputs and to turn on or off the identification for the outlets**

Syntax:

```
cli set [OPTIONS] PREFIX CONTROL VALUE
```

- **PREFIX** – Path to the logical element.
- **CONTROL** – The control to modify (e.g., switch, identify).
- **VALUE** – The new value to apply: on or off.

Examples:

```
cli set /pdu[AAABMS2O]/core/inlet[1]/phase[2]/outlet[1] identify on

Control set successfully
```

```
cli set /pdu[AAABMS2O]/local/do[1] switch off

Control set successfully
```

Verification with cli get

The status of digital inputs (DI) and digital outputs (DO) can be checked with the cli get command.

1. Digital outputs in OFF state

```
BN-PRO-7A93CFE2:~$ cli set /pdu[AAABMS2O]/local/do[1] switch off

Control set successfully

BN-PRO-7A93CFE2:~$ cli set /pdu[AAABMS2O]/local/do[2] switch off

Control set successfully

BN-PRO-7A93CFE2:~$ cli get -a -| grep local

/pdu[AAABMS2O]/local/di[1]/status
okay (inactive)

/pdu[AAABMS2O]/local/di[2]/status
okay (inactive)

/pdu[AAABMS2O]/local/do[1]/status                                    okay
(inactive)
```

```
/pdu[AAABMS2O]/local/do[2]/status                                    okay
(inactive)
```

Here, both digital outputs are **inactive**.

2. Switching a digital output ON

```
BN-PRO-7A93CFE2:~$ cli set /pdu[AAABMS2O]/local/do[1] switch on

Control set successfully

BN-PRO-7A93CFE2:~$ cli set /pdu[AAABMS2O]/local/do[2] switch on

Control set successfully

BN-PRO-7A93CFE2:~$ cli get -a -| grep local

/pdu[AAABMS2O]/local/di[1]/status                                    okay
(inactive)

/pdu[AAABMS2O]/local/di[2]/status                                    okay
(inactive)

/pdu[AAABMS2O]/local/do[1]/status
okay (active)

/pdu[AAABMS2O]/local/do[2]/status
okay (active)
```

8 Element management

The **element tree** represents the internal structure of the PDU. Each device is organized hierarchically, with the **PDU itself** at the top and various components (inlets, outlets, sensors, phases, MCBs, GPIOs, etc.) listed as child elements.

- Running

```
cli element list
```

Displays this hierarchy in a **tree-like structure**, where each element type (e.g., Inlet, Outlet, Phase, MCB) is shown with its **label** and **description**.

- Sub-elements are indented under their parent. For example, an **Inlet** may contain **Phases**, and each **Phase** may contain **MCBs**.
- This structure makes it easy to navigate through the PDU's internal components and identify where specific measurements, alarms, or controls belong.

The `cli element` command supports the following subcommands:

- **list** – Explore the hierarchy and view measurements/alarms.
- **alarm** – Configure alarms for elements (these can be connected to signal chains or traps).
- **edit** – Change labels or descriptions to make identification easier.
- **remove** – Delete logical elements if supported.

8.1 element list

The `cli element list` command allows you to explore the **hierarchical structure** of the PDU and its sub-elements. Depending on the flags used, you can display **measurements, alarms, controls, indices, or JSON output**.

General Syntax:

```
cli element list [OPTIONS]
```

Options:

- -m → Show measurements
- -a → Show alarms
- -c → Show controls
- -i → Show indices
- -j → Show JSON output

If no option is given, the structure of the device is displayed with element types, labels, and descriptions.

- ***Listing Measurements (-m)***

  Shows all measurement points (e.g., current, power, voltage, frequency).
- ***Listing Alarms (-a)***

  The -a option does not show currently active alarms.

  Instead, it lists all measurements that are alarm-capable, meaning alarms can be configured on them.
  - If a measurement is listed here, the user may later assign thresholds or conditions with cli element alarm.
  - If it is not listed, no alarm can be defined for that measurement.

  This option therefore serves as a reference for which parts of the signal tree support alarm configuration.
- ***Listing Controls (-c)***

  Shows control points (e.g., switching digital outputs, triggering RCM self-test).

  Useful for direct interaction with hardware components.
- ***Listing Indices (-i)***

  Index of the element (relevant for access via Modbus and SNMP)

- ***JSON Output (-j)***

  Provides the full **device structure in JSON format**, including measurements, alarms, and controls with their attributes.

  This output is designed for **machine processing** or integration with external tools and monitoring systems.

  It is the most detailed format, containing metadata.

8.2 element alarm

The `cli element alarm` command allows configuration and inspection of **alarm settings** on logical elements.
It works **only** for measurements that are alarm-capable, as shown in the cli element list -a output.

Usage:

```
cli element alarm [OPTIONS] COMMAND [ARGS]...
```

Commands:

- get → Retrieve alarm settings for an element
- set → Configure alarm settings for an element

➢ element alarm set

The `cli element alarm set -h` gives the correct syntax to be used when setting a threshold:

Options:

```
  --warning-low FLOAT         Set warning low
  --alarm-low FLOAT           Set alarm low
  --warning-high FLOAT        Set warning high
  --alarm-high FLOAT          Set alarm high
  --hysteresis FLOAT          Set hysteresis
  --warning-threshold FLOAT   Set warning threshold (RCM only)
```

```
  --alarm-threshold FLOAT    Set alarm threshold (RCM only)
  --warning-gradient FLOAT   Set warning gradient (RCM only)
  --alarm-gradient FLOAT     Set alarm gradient (RCM only)
```

The set subcommand configures thresholds for alarms and warnings.

```
BN-PRO-7A93CFE2:~$ cli element alarm set /pdu[AAABMS2O]/core/in-
let[1]/phase[3]/voltage --warning-low 70
Alarm settings changed successfully
```

- element alarm get

```
BN-PRO-7A93CFE2:~$ cli element alarm get /pdu[AAABMS2O]/core/in-
let[1]/phase[3]/voltage
Key           Value
------------  ----------
alarm_type    "standard"
hysteresis    5.0
alarm_high    270.0
alarm_low     50.0
warning_high  270.0
warning_low   70.0
```

8.3 element edit

The `cli element edit` command allows renaming **(label)** or/and **add a description** to logical elements in the hierarchy.
This is especially useful for easier identification in environments with many elements, since the default names are generic (e.g., *Phase 1*, *MCB 2*).

Note: Labels and descriptions do **not** affect the functionality of the element. They are purely for identification and documentation purposes.

- Usage: `cli element edit [OPTIONS] ADDRESS`

- See `cli element edit -h` for more information

- **Example:**

```
BN-PRO-7A93CFE2:~$ cli element edit /pdu[AAABMS2O]/core/inlet[1]/phase[3] -
l "room 2" -d "24 hours observation"

Label and/or description changed successfully
```

The label and description changes are visible in subsequent cli element list outputs and in the web frontend, making it easier to track elements according to your operational context.

8.4 element remove

The `cli element remove` command is used to **delete a logical element** from the hierarchy.
This is typically relevant for **removable or external elements** (e.g., external sensors) that are physically disconnected.

Important behavior notes for sensors:

- If a sensor goes **offline** (e.g., there are 11 sensors connected, the 11th takes the place of the 10th, and the 10th shows *offline*), the PDU global state turns **red** due to the offline state.
  → In this case, the user can use cli element remove to clean up the offline sensor from the element list.
- If a sensor is **connected and green (healthy)**, even if the user runs cli element remove and the CLI responds with *"removed successfully"*, the sensor will **not actually be removed**.

→ As long as the sensor remains physically connected and online, it will reappear in the element list.

- Usage: `cli element remove [OPTIONS] ADDRESS`

- **Example:**

```
cli element remove /pdu[AAABMS2O]/ext/sensor[5600000012bd3ed0]

Element successfully removed
```

If the sensor was offline, it will be cleared from the list.

If it was online/green, it will remain present.

This command helps keep the element tree clean by ensuring that disconnected devices do not remain in the configuration.

8.5 Interaction with Signal Chians (link to Subchapter 9.1)

Alarm configuration on elements is not limited to defining thresholds (e.g., --warning-low, --alarm-high), alarms are also linked to **signal chains** that determine the actions to be taken once the condition is met.

- **Thresholds** define the conditions for the alarm (e.g., voltage below 210 V).
- **Signal chains** define the actions triggered by the alarm (e.g., GUI alarm, SNMP trap, or SMTP email).

When a signal chain is attached to an element:

- The alarm is not only visible in the **local GUI(on the display or a Beeper is triggered)** but can also be forwarded using communication protocols such as **SNMP** (for traps) or **SMTP** (for email alerts).

- These protocols must be properly configured on the system, but the element → signal chain link ensures that the **specific condition on a device results in an actionable notification**.

This means that configuring alarms on elements is only one part of the process. To ensure operators are informed, the alarm should be **connected to a signal chain** that activates the desired notification path.

For more details about configuring and managing signal chains, see **Chapter 9 – Signal Chains**.

9 Signal Chains

Signal chains define automated reactions to events inside the PDU system. They connect measurement values, alarms, and status changes to specific actions. These actions can include triggering local GUI notifications, generating SNMP traps, or sending emails through SMTP (assuming the protocols are configured).

By using signal chains, the user can build a flexible automation layer: instead of only setting thresholds for alarms (see Chapter 7), signal chains allow combining those alarms with actions that propagate outside the device. This is what makes them the "glue" between monitoring, alarming, and external notification systems.

Signal chains are managed entirely through the CLI. The available subcommands cover the full lifecycle: creating, attaching, detaching, editing, listing, and deleting signal chains.

9.1 Relation to alarms and elements (link to Subchapter 8.5)

Signal chains do not operate in isolation — they are tightly related to **elements and alarms**

- An **element** (for example, a phase, inlet, or sensor) can have thresholds configured through the cli element alarm commands.
- When those thresholds are reached, an **alarm event** is raised locally on the PDU.
- A **signal chain** can then be attached to that alarm address. This attachment ensures that when the alarm is triggered, the signal chain executes the associated action (e.g., activating a trap, sending an email, or showing a local GUI alarm).

This connection means that alarms define the **conditions**, while signal chains define the **reactions**.
Together, they form the complete monitoring and notification workflow.

9.2 signalchain add

The `cli signalchain add` command is used to create a new signal chain. Each signal chain is uniquely identified by a **UUID**, which is automatically generated at creation. This UUID (Universally Unique Identifier) is essential because it will later be required when attaching the signal chain to alarms, status and startup.

Note: Even if two signal chains look identical, the system still treats them as different because of their unique UUIDs.

A signal chain can be created in two ways:

- **Minimal creation**:
  Running `cli signalchain add` without arguments creates a signal chain that receives:
  - **Automatic name** in the format Signalchain YYYY-MM-DD HH:MM:SS (creation timestamp).
  - **Active = True** (by default).
  - **No description, no email receivers, no local GUI or beeper triggers, no SNMP traps, and no Outlets

- **Custom creation**:
  Options allow the user to assign a name, description, state (active/inactive), and define the actions triggered by the signal chain, such as:
  - Sending an email (requires SMTP configuration)
  - Triggering the local GUI alarm indication
  - Triggering the beeper
  - Sending SNMP traps (requires SNMP configuration)
  - Switching outlets or digital outputs

- **Important:**
  - **UUID is always unique**, and it is the *only* identifier that distinguishes signal chains.
  - Users can create multiple signal chains with the **same name** and even with the **same configuration**. In this case, the UUID will be the factor that differentiates them.
  - Therefore, when referencing a signal chain (for attachment, editing, or deletion), the UUID must always be used.

Example – simple creation with defaults:

```
BN-PRO-7A93CFE2:~$ cli signalchain add

Successfully created new signal chain with UUID=2acfb0ec-374c-4918-adc5-170865d76bb6
```

Example – creation with multiple actions:

```
BN-PRO-7A93CFE2:~$ cli signalchain add --name SigChainCli \
> --active \
> --email user@name.de \
> --enable-localgui \
> --enable-beeper \
> --digital-output /pdu[AAABMS2O]/local/do[1] on

Warning: Missing SMTP configuration

Successfully created new signal chain with UUID=5f9661fa-c911-4332-84be-b5d50e7a1f57
```

Important about SMTP and SNMP:

- If the user specifies --email, the CLI may warn about missing SMTP configuration. This means the signal chain will still be created successfully, but the **email action will not work** until the SMTP server settings are configured.
- Similarly, if the user adds a trap receiver with --trap, the signal chain will reference it, but SNMP traps can only be sent if SNMP configuration is completed.

9.3 signalchain attach/detach

9.3.1 signalchain attach

After a signal chain is created (see 9.2 signalchain add), it does not yet influence the system. To make it active in relation to a measurement or alarm, it needs to be **attached** to the corresponding alarm address.

The syntax is: `cli signalchain attach [ALARM] [UUID]`

- **ALARM** = the full address of the measurement/alarm (for example, /pdu[AAABMS2O]/core/inlet[1]/phase[2]/current)
- **UUID** = the unique identifier of the signal chain (see **8.2**)

Example:

```
cli signalchain attach /pdu[AAABMS2O]/core/inlet[1]/phase[2]/current
b2dacca3-9491-47ad-8244-35b8b98b415f

Successfully linked signal chain(s) to alarm address
/pdu[AAABMS2O]/core/inlet[1]/phase[2]/current
```

It is also possible to attach **multiple signal chains** to the same alarm address by listing more than one UUID in the command:

```
cli signalchain attach /pdu[AAABMS2O]/core/inlet[1]/phase[2]/current
b2dacca3-9491-47ad-8244-35b8b98b415f 1c440a76-bb2f-45bf-b5fb-59428d9c8da0
5f9661fa-c911-4332-84be-b5d50e7a1f57

Successfully linked signal chain(s) to alarm address
/pdu[AAABMS2O]/core/inlet[1]/phase[2]/current
```

9.3.2 signalchain detach

If a signal chain should no longer be triggered by an alarm, it can be removed with the

```
cli signalchain detach [ALARM] [UUID]
```

detach command:

Example:

```
cli signalchain detach /pdu[AAABMS2O]/core/inlet[1]/phase[2]/current
b2dacca3-9491-47ad-8244-35b8b98b415f

Successfully detached signal chain(s) from alarm address
/pdu[AAABMS2O]/core/inlet[1]/phase[2]/current
```

Important points:

- The UUID must be known before attaching or detaching. The easiest way to retrieve it is by running cli signalchain list (see **9.6** signalchain list).
- An alarm can have more than one signal chain attached, and a single signal chain can also be attached to multiple alarms.
- Detaching a signal chain does not delete it; it only removes its connection to that alarm.

9.4 signalchain attach-startup/detach-startup

In addition to linking signal chains to measurement alarms, it is also possible to link them to **PDU startup.** This is useful when the PDU reboots or powers on again, for example after a power failure or maintenance.

When a signal chain is attached to the **status address** of a PDU, it will be triggered each time the PDU starts up. Typical use cases include:

- Sending an **email notification** that the PDU is back online.
- Activating **the beeper** or display GUI notification.

9.4.1 signalchain attach-startup

The syntax for attaching is:

```
cli signalchain attach-startup [ADDRESS] [UUID]
```

- **ADDRESS** = the PDU status address (e.g., /pdu[AAABMS2O]/status)
- **UUID** = the UUID of the signal chain

Example:

```
cli signalchain attach-startup /pdu[AAABMS2O]/status b2dacca3-9491-47ad-
8244-35b8b98b415f

Successfully linked signal chain(s) to /pdu[AAABMS2O]/status startup
```

9.4.2 signalchain detach-startup

To remove this link, use the **detach-startup** command:

```
cli signalchain detach-startup [ADDRESS] [UUID]
```

Example:

```
cli signalchain detach-startup /pdu[AAABMS2O]/status b2dacca3-9491-47ad-8244-35b8b98b415f

Successfully detached signal chain(s) from /pdu[AAABMS2O]/status startup
```

Important points:

- This feature is especially useful for **monitoring availability**: you can be informed immediately if the PDU comes back online in case it rebooted itself or something else happened that it goes offline.
- The **UUID** must belong to an existing signal chain
- Attaching here does not depend on thresholds or alarms — the trigger is strictly the **startup event** of the PDU.

9.5 signalchain attach-status/detach-status

Signal chains can also be linked to **status changes** of specific elements, namely:

- **MCBs (Miniature Circuit Breakers)** → possible trigger events: on or tripped
- **Outlets** → possible trigger events: on or off

This mechanism allows an immediate reaction when, for example:

- An **MCB trips** due to overload or short circuit.
- An **outlet is switched off**, either manually or because of a configuration.
- An **outlet is switched on**, which could trigger notifications or external controls.

9.5.1 signalchain attach-status

The syntax for attaching is:

```
cli signalchain attach-status [STATUS_ADDRESS] [TRIGGER_EVENT] [UUID]
```

- **STATUS_ADDRESS** = the address of the MCB or outlet status (e.g., /pdu[AAABMS2O]/core/inlet[1]/phase[1]/outlet[1]/status)
- **TRIGGER_EVENT** = either on/off for outlets or on/tripped for MCBs
- **UUID** = the UUID of the signal chain

Examples:

```
cli signalchain attach-status /pdu[AAABMS2O]/core/inlet[1]/phase[1]/out-
let[1]/status on 1c440a76-bb2f-45bf-b5fb-59428d9c8da0

Successfully linked signal chain(s) to status address
/pdu[AAABMS2O]/core/inlet[1]/phase[1]/outlet[1]/status
```

```
cli signalchain attach-status /pdu[AAABMS2O]/core/in-
let[1]/phase[1]/mcb[1]/status tripped 1c440a76-bb2f-45bf-b5fb-59428d9c8da0

Successfully linked signal chain(s) to status address

/pdu[AAABMS2O] /core/inlet[1]/phase[1]/mcb[1]/status
```

9.5.2 signalchain detach-status

To remove this binding, use:

```
cli signalchain detach-status [STATUS_ADDRESS] [TRIGGER_EVENT] [UUID]
```

Example:

```
cli signalchain detach-status /pdu[AAABMS2O]/core/inlet[1]/phase[1]/out-
let[1]/status on 1c440a76-bb2f-45bf-b5fb-59428d9c8da0

Successfully detached signal chain(s) from status address
/pdu[AAABMS2O]/core/inlet[1]/phase[1]/outlet[1]/status
```

Important notes:

- This is **event-driven**, not threshold-based. It reacts to discrete state changes of **MCBs and outlets**.
- The **UUID** must be copied from an existing signal chain .
- Multiple signal chains can be attached to the same status event.
- This is especially useful for **safety monitoring** (e.g., MCB trip) or **power management automation** (e.g., notify when outlet is switched off).

9.6 signalchain list

The `cli signalchain list` command displays all existing signal chains on the PDU together with their configuration details. This is especially useful for:

- **Identifying signal chains by UUID** (required for attach/detach operations).
- **Checking if a signal chain is active or inactive**.
- **Reviewing configured actions**, such as email notifications, local GUI triggers, beeper alerts, SNMP traps, or digital outputs.
- **Verifying the current setup** before editing or deleting a signal chain:

```
BN-PRO-7A93CFE2:~$ cli signalchain list

UUID:                e24c9576-2960-4859-a061-548f1bb610fb

-----------------------------------------------------

Name:                SigChainCli
```

```
Active:              True
Description:         -
E-Mail receivers:
    user@name.de
Trigger local GUI:   True
Trigger beeper:      True
SNMP trap receiver UUIDs: 06ea15da-8b68-4efe-a38c-bbe3480061b7
Digital Outputs:
    /pdu[AAABMS2O]/local/do[1] on
Outlets:


UUID:                6e5a6ef6-07a2-4677-82e3-6d2613346685
------------------------------------------------------------
Name:                Signalchain 2025-09-11 11:57:37
Active:              True
Description:         -
E-Mail receivers:
Trigger local GUI:   False
Trigger beeper:      False
SNMP trap receiver UUIDs: -
Digital Outputs:
Outlets:
```

Key fields explained:

- **UUID** → The unique identifier for each signal chain (used when attaching, detaching, editing, or deleting).

- **Name** → The given name or automatically generated one (based on creation date/time if not specified).
- **Active** → Indicates whether the signal chain is enabled (True) or disabled (False).
    - Per default at creation it is active, but the user can use the "–-inactive" option to make it inactive
- **E-Mail receivers** → List of email addresses that will receive notifications (requires SMTP configuration for a full functionality).
- **Trigger local GUI** → Whether the local GUI is notified.
- **Trigger beeper** → Whether the beeper is activated.
- **SNMP trap receiver UUIDs** → Linked SNMP trap receivers.
- **Digital Outputs / Outlets** → Shows which outputs or outlets are affected when the signal chain is triggered.

Notes:

- From here, the user can copy the **UUIDs** for use in attach, detach, edit, or delete.
- Signal chains can have **identical names**, but the **UUID is always unique** and is the only reliable way to reference them.
- If the SMTP service is not configured, email addresses may still be displayed, but emails will **not be delivered**.

9.7 signalchain edit

The `cli signalchain edit` command allows users to modify an **existing signal chain** using its **UUID**. This is the primary way to refine configurations after initial creation, especially when a chain was created with default or incomplete parameters.

General behavior

- The **UUID** of the signal chain must be specified.
- Parameters provided with edit will **overwrite existing values** (e.g., email addresses, name, description).
- Removal flags always take priority over add flags.
    - **Example:** --remove-digital-output ... --add-digital-output ... results in the digital output being removed, not added.

Example usage:

1. **Update properties:**

```
cli signalchain edit 6e5a6ef6-07a2-4677-82e3-6d2613346685 \
--email user@example.com \
--enable-localgui \
--enable-beeper
Successfully edited signal chain
```

Overwrites the signal chain to send email notifications, trigger the local GUI, and activate the beeper.

2. **Modify description:**

Adds or updates the description field.

```
cli signalchain edit 6e5a6ef6-07a2-4677-82e3-6d2613346685 -d "signalchain
for outlets only"
Successfully edited signal chainSuccessfully
```

3. **Remove and add digital outputs:**

```
cli signalchain edit 6e5a6ef6-07a2-4677-82e3-6d2613346685 \
--remove-digital-output /pdu[AAABMS2O]/local/do[1] \
--add-digital-output /pdu[AAABMS2O]/local/do[2] on
Successfully edited signal chain
```

Digital output 1 is removed and digital output 2 is added with state on.

4. **Clear SNMP traps:**

```
cli signalchain edit e24c9576-2960-4859-a061-548f1bb610fb --clear-traps
Successfully edited signal chain
```

Removes all SNMP trap receivers associated with the signal chain.

9.8 signalchain delete

The `cli signalchain delete` command removes an existing signal chain from the system, identified by its **UUID**.

Important behavior

- A signal chain **cannot be deleted** if it is still attached to any alarm address, startup event, or status event.
- The user must first **detach** the signal chain from all linked elements before deletion is possible.
- If deletion is attempted while the signal chain is still linked, an error will be returned.

Example usage:

1. **Attempt to delete a linked signal chain:**

```
cli signalchain delete e24c9576-2960-4859-a061-548f1bb610fb

Error: HTTP status code 400 (Bad Request)

    Signal chain e24c9576-2960-4859-a061-548f1bb610fb is still linked to
alarm address /alarm/phys/pdu[AAABMS2O]/core/inlet[1]/current.
```

This means the signal chain is still actively linked to an alarm and must be detached first.

2. **Successful deletion:**

```
cli signalchain delete e24c9576-2960-4859-a061-548f1bb610fb

Successfully deleted signal chain e24c9576-2960-4859-a061-548f1bb610fb.
```

10 Outlet Groups

Outlet groups provide a way to **bundle multiple outlets together** and control them as a single unit. This is useful in scenarios where several devices connected to different outlets must be managed simultaneously — for example, powering down a set of servers in a rack, or rebooting multiple network devices at once.

Each outlet group consists of:

- A **label** – a human-readable name that makes it easier to identify the group (e.g., "Server Rack A").
- An optional **description** – for documentation or clarification.
- A list of **outlets** – the specific outlets that belong to this group.
- A **UUID** – a unique identifier automatically assigned to the group at creation.

Important:

- The **UUID** is the true reference of the group. All CLI operations such as edit, delete, or list rely on the UUID, not on the label.
- Labels may be duplicated, but UUIDs are always unique.
- While the **Web interface** allows creating outlet groups without a label, in the **CLI a label is mandatory**.

Once created, outlet groups simplify the management of outlets, especially when applying actions like switching power on/off across multiple outlets at once.

10.1 outlet-group add

The `cli outlet-group add` command is used to create a new outlet group. Outlet groups allow users to manage multiple outlets collectively (e.g., switching them together).

General usage:

```
cli outlet-group add [OPTIONS] LABEL
```

- **LABEL (required):** A human-readable name for the group.
  Unlike in the Web interface, the CLI requires a label to be provided. This ensures that groups are easier to distinguish when listed in text output.
- **UUID generation:** Each outlet group receives a unique UUID upon creation, which is the true reference used in further operations (edit, delete, attach, etc.). The label is only for readability.

Example:

```
BN-PRO-7A93CFE2:~$ cli outlet-group add "serverRoom" \
> --add-outlet /pdu[AAABMS2O]/core/inlet[1]/phase[3]/outlet[4] \
> --add-outlet /pdu[AAABMS2O]/core/inlet[1]/phase[1]/outlet[4] \
Successfully created new outlet group with UUID=ab11c40f-2260-4433-aeb3-2defe5a2b0b5
```

In this case:

- The **label** is " serverRoom ".
- The group contains **two outlets**.
- The group is uniquely identified by its **UUID** ab11c40f-2260-4433-aeb3-2defe5a2b0b5.

10.2 outlet-group list

The `cli outlet-group list` command displays all existing outlet groups, including their **UUID**, **label**, **description**, and the list of assigned outlets.
This is useful for identifying the UUID of a group, which is required for further operations (edit, attach, delete, etc.).

Command with output:

```
BN-PRO-7A93CFE2:~$ cli outlet-group list
UUID:                ab11c40f-2260-4433-aeb3-2defe5a2b0b5
------------------------------------------------------------
Label:               serverRoom
Description:
Outlets:
    /pdu[AAABMS2O]/core/inlet[1]/phase[1]/outlet[4]
    /pdu[AAABMS2O]/core/inlet[1]/phase[3]/outlet[4]

UUID:                38fc627f-bcc6-4079-8a32-97886c01d8ff
------------------------------------------------------------
Label:               Group 1
Description:
Outlets:
    /pdu[AAABMS2O]/core/inlet[1]/phase[3]/outlet[3]
    /pdu[AAABMS2O]/core/inlet[1]/phase[3]/outlet[4]
```

Key points:

- **UUID**: The unique identifier of the outlet group (used for referencing in other commands).
- **Label**: The human-readable name of the group (optional during creation but displayed if provided).
- **Description**: Optional field, empty by default unless set.
- **Outlets**: Lists all the outlets that belong to this group.

Even if two groups have the same **label**, their **UUIDs** will always differ, making UUID the reliable identifier for further operations.

10.3 Signal Chain attach/detach to/from an Outlet Group (Current)

Outlet groups can be linked with **signal chains** to trigger actions (e.g., alarms, notifications, switching).
At the **outlet group level**, a signal chain can **only** be attached to the **current measurement**.

Step 1 – Identify the Outlet Group UUID and measurement path

```
BN-PRO-7A93CFE2:~$ cli get -a - | grep group

/group[38fc627f-bcc6-4079-8a32-97886c01d8ff]/current
0.0  A       okay

/group[38fc627f-bcc6-4079-8a32-97886c01d8ff]/current_max
0.0  A

/group[38fc627f-bcc6-4079-8a32-97886c01d8ff]/energy_active
0.0  kWh

/group[38fc627f-bcc6-4079-8a32-97886c01d8ff]/energy_active2
0.0  kWh

/group[38fc627f-bcc6-4079-8a32-97886c01d8ff]/energy_apparent
0.0  kVAh
```

```
/group[38fc627f-bcc6-4079-8a32-97886c01d8ff]/energy_reactive
0.0  kvarh

/group[38fc627f-bcc6-4079-8a32-97886c01d8ff]/power_active
0  W

/group[38fc627f-bcc6-4079-8a32-97886c01d8ff]/power_active_max
0  W

/group[38fc627f-bcc6-4079-8a32-97886c01d8ff]/power_apparent
0  VA

/group[38fc627f-bcc6-4079-8a32-97886c01d8ff]/power_reactive
0  var
```

Notice that only /group[UUID]/current can be used for signal chain attachment.

Step 2 – Identify the Signal Chain UUID

List available signal chains:

```
BN-PRO-7A93CFE2:~$ cli signalchain list

UUID:                6e5a6ef6-07a2-4677-82e3-6d2613346685

------------------------------------------------------------

Name:                Signalchain 2025-09-11 11:57:37

Active:              False

Description:         signalchain for outlets only

E-Mail receivers:

Trigger local GUI:   False

Trigger beeper:      False

SNMP trap receiver UUIDs: -

Digital Outputs:

Outlets:
```

Step 3 – Attach Signal Chain to Outlet Group Current

Now link the chosen signal chain UUID to the outlet group's current measurement:

```
cli signalchain attach /group[38fc627f-bcc6-4079-8a32-97886c01d8ff]/current
6e5a6ef6-07a2-4677-82e3-6d2613346685

Successfully linked signal chain(s) to alarm address /group[38fc627f-bcc6-
4079-8a32-97886c01d8ff]/current
```

Step 4 – Detach Signal Chain

To remove the link between the signal chain and the outlet group:

```
cli signalchain detach /group[38fc627f-bcc6-4079-8a32-97886c01d8ff]/current
6e5a6ef6-07a2-4677-82e3-6d2613346685

Successfully detached signal chain(s) from alarm address /group[38fc627f-
bcc6-4079-8a32-97886c01d8ff]/current
```

10.4 outlet-group edit

The `cli outlet-group edit` command allows modifying an existing outlet group.
Edits can include:

- Adding outlets
- Removing outlets
- Updating the label
- Updating the description
- See more accessing the command "cli outlet-group edit -h"

Command Syntax:

```
cli outlet-group edit [OPTIONS] UUID
```

Examples:

1. Remove an outlet from a group and add a new one

```
cli outlet-group edit 38fc627f-bcc6-4079-8a32-97886c01d8ff --remove-outlet
/pdu[AAABMS2O]/core/inlet[1]/phase[3]/outlet[4] --add-outlet
/pdu[AAABMS2O]/core/inlet[1]/phase[3]/outlet[1]

Successfully edited outlet group
```

2. Edit description

```
cli outlet-group edit 38fc627f-bcc6-4079-8a32-97886c01d8ff --description
"for exteral rooms"

Successfully edited outlet group
```

10.5 outlet group delete

The `cli outlet-group delete` command is used to remove an existing outlet group.

Deletion is based on the **UUID** of the outlet group.

Command Syntax:

```
cli outlet-group delete [OPTIONS] UUID
```

Example:

Delete outlet group by UUID

```
cli outlet-group delete 38fc627f-bcc6-4079-8a32-97886c01d8ff

Successfully deleted outlet group: 38fc627f-bcc6-4079-8a32-97886c01d8ff
```

10.6 Control operation on Outlet Groups

Besides creating, editing, and deleting outlet groups, it is also possible to **control all outlets within a group simultaneously** using the cli set command.

This allows batch operations such as switching ON/OFF or activating the identification mode for the entire outlet group.

To see which outlet groups were created use:

```
cli get -a - | grep group
```

Command Syntax:

```
cli set /group[UUID] CONTROL VALUE
```

Where:

- **UUID** = Unique identifier of the outlet group
- **CONTROL** = switch or identify
- **VALUE** = on or off

Examples:

Enable identification for all outlets in a group

```
cli set /group[f9eaea37-7a06-4a5b-b515-d24018a9133f] identify on

Control set successfully
```

Switch off all outlets in a group

```
cli set /group[f9eaea37-7a06-4a5b-b515-d24018a9133f] switch off

Control set successfully
```

Note: The operation affects **all outlets belonging to the group**

10.7 outlet-sequence set and get

The CLI provides a way to configure the **switching sequence** of outlets.

This is useful when powering up multiple outlets in a controlled order with specific time delays to prevent overloads or to ensure equipment starts in the right sequence.

See more information using the commands: `cli outlet-sequence set -h` **and** `cli outlet-sequence get -h`

Command Syntax for set:

```
cli outlet-sequence set FILE
```

Where:

- **FILE** is a text file containing the outlet sequence definition (order + delay).

Workflow:

1. **Export the current sequence** to a file:

```
cli outlet-sequence get > sequence.txt
```

2. **Edit the file** with your preferred text editor (e.g., vi, nano).
    - The file format lists each outlet and its associated **delay in milliseconds (ms)**.

3. **Re-upload the modified sequence**:

```
cli outlet-sequence set sequence.txt
```

Example of a Sequence File:

```
Element                                                Delay [ms]
/pdu[a0000022]/core/inlet[1]/phase[1]/mcb[1]/outlet[1]: 100
/pdu[a0000022]/core/inlet[1]/phase[1]/mcb[1]/outlet[2]: 200
/pdu[a0000022]/core/inlet[1]/phase[1]/mcb[1]/outlet[3]: 300
/pdu[a0000022]/core/inlet[1]/phase[1]/mcb[1]/outlet[4]: 400
...
/pdu[a0000022]/core/inlet[1]/phase[1]/mcb[2]/outlet[18]: 3600
```

11 Logs

The `cli log` command provides access to the **event log**, where important system events are stored, including authentication attempts, alarms, and warnings.

11.1 log list

To view the event log:

```
BN-PRO-7A93CFE2:~$ cli log list
Date                        Level    Type            Message
--------------------------  -------  --------------  ----------------------
-------------------------------------------------------------
2025-09-11 11:52:59.884048  INFO     Alarming        Starting alarm ob-
server
2025-09-11 11:57:11.568902  INFO     Authentication  Authentication suc-
cessful for user: 'admin'
2025-09-11 11:57:20.447052  INFO     Authentication  Authentication suc-
cessful for user: 'admin'
2025-09-11 11:57:29.021401  INFO     Authentication  Authentication suc-
cessful for user: 'admin'
2025-09-11 12:40:31.216012  INFO     Authentication  Authentication suc-
cessful for user: 'admin'
2025-09-11 13:27:08.897287  INFO     Authentication  Authentication suc-
cessful for user: 'admin'
2025-09-11 13:31:52.480094  INFO     Authentication  Authentication suc-
cessful for user: 'admin'
2025-09-12 05:54:47.021526  INFO     Authentication  Authentication suc-
cessful for user: 'admin'
2025-09-12 05:54:55.991683  INFO     Authentication  Authentication suc-
cessful for user: 'admin'
2025-09-12 05:58:55.637599  INFO     Authentication  Authentication suc-
cessful for user: 'admin'
```

```
2025-09-12 06:11:02.457833  INFO     Authentication  Authentication suc-
cessful for user: 'admin'

2025-09-12 07:00:26.276550  INFO     Authentication  Authentication suc-
cessful for user: 'admin'

2025-09-12 07:00:36.505776  WARNING  Status          New alarm state:
/pdu[AAABMS2O]/core/inlet[1]/phase[1]/outlet[1]/current = warninglow
```

Log fields explained:

- **Date** → Timestamp of the event.
- **Level** → Severity level (INFO, WARNING, ERROR).
- **Type** → Category (Authentication, Status, Alarming, etc.).
- **Message** → Detailed event message.

11.2 log remove

To clear the entire event log:

```
BN-PRO-7A93CFE2:~$ cli log remove

Removed event log successfully
```

12 RCM selftest

The `cli rcm-selftest` command provides access to the **Residual Current Monitoring (RCM) self-test** functionality of the system.
This feature is essential for ensuring **electrical safety compliance**, as it verifies the proper functioning of the residual current detection mechanism that helps protect users and equipment from leakage currents.

Command Syntax:

```
cli rcm-selftest [OPTIONS] COMMAND [ARGS]...
```

Available Commands:

- log → Print the RCM selftest log.
- result → Show the most recent selftest results.

trigger → Start/trigger a new selftest procedure.

12.1 rcm-seltest trigger

To start a selftest, use:

```
BN-PRO-7A93CE28:~$ cli rcm-selftest trigger
RCM selftest triggered ...
```

This initiates the selftest procedure. If no rcm is pecified the selftest will be done on all RCMs available. The results can then be checked in the log or via the result command.

12.2 rcm-selftest log

```
BN-PRO-7A93CE28:~$ cli rcm-selftest log
RCM:                /pdu[AABBCCDQ]/core/inlet[1]/rcm[1]
-------------------------------------------------------
Timestamp:          2025-09-12 12:43:08
Status:             okay (active)
Peak RCM AC:        0.0288
Peak RCM DC:        0.0148
RCM:                /pdu[AABBCCDQ]/core/inlet[1]/rcm[1]
-------------------------------------------------------
Timestamp:          2025-09-12 12:43:28
Status:             okay (active)
Peak RCM AC:        0.0262
```

```
Peak RCM DC:          0.0125
RCM:                  /pdu[AABBCCDQ]/core/inlet[1]/rcm[1]
-------------------------------------------------------
Timestamp:            2025-09-12 12:47:14
Status:               okay (active)
Peak RCM AC:          0.0292
Peak RCM DC:          0.0148
```

The log command prints all available RCM selftest results in chronological order.

Each entry includes:

- **Timestamp** – when the selftest was performed
- **Status** – result of the selftest
- **Peak RCM AC/DC** – measured leakage current values during the test

12.3 rcm-selftest result

The result command prints only the most recent selftest result.

```
BN-PRO-7A93CE28:~$ cli rcm-selftest result
RCM:                  /pdu[AABBCCDQ]/core/inlet[1]/rcm[1]
-------------------------------------------------------
Timestamp:            2025-09-12 12:47:14
Status:               okay (active)
Peak RCM AC:          0.0292
Peak RCM DC:          0.0148
```

This is effectively the **last entry from the log**, making it a quick way to verify the outcome of the latest triggered selftest.

13 Monitor measurements

The `cli monitor` command continuously observes selected measurements and outputs them to the console (stdout).
This is especially useful for **real-time monitoring**, troubleshooting, or validating behavior during load changes.

When no interval is specified, cli monitor defaults to **1 second**. The output shows a timestamp followed by the requested measurements.

Syntax:

```
cli monitor [OPTIONS] PREFIX [MEASUREMENT]
```

Examples:

1. **Monitor all available measurements:**

```
cli monitor -
```

2. **Monitor all current measurements of a phase:**

```
BN-PRO-7A93CE28:~$ cli monitor /pdu[AABBCCDQ]/core/inlet[1]/phase[2] cur-
rent

timestamp (hh:mm:ss:ms);/pdu[AABBCCDQ]/core/inlet[1]/phase[2]/cur-
rent;/pdu[AABBCCDQ]/core/inlet[1]/phase[2]/mcb[1]/cur-
rent;/pdu[AABBCCDQ]/core/inlet[1]/phase[2]/mcb[2]/current

13:01:45:743;0.0;0.0;0.0

13:01:46:744;0.0;0.0;0.0

13:01:47:746;0.0;0.0;0.0

13:01:48:746;0.0;0.0;0.0

13:01:49:747;0.0;0.0;0.0
```

- The **first row** is the header with the timestamp and measurement paths.
- Each **subsequent row** contains values at one-second intervals.
- The output continues indefinitely until stopped with Ctrl + C.

3. **Monitor all current measurements in the system:**

```
cli monitor - current
```

4. **Monitor with a custom interval (e.g. every 5 seconds):**

```
cli monitor – voltage -i 5
```

Tip:

Unlike cli get, which prints the current state once, cli monitor keeps running and printing updates until it is stopped (e.g., with Ctrl + C).

14 Product Information

The `cli product-info` command provides detailed information about the PDU and its components.
It displays build version, hardware details, serial numbers, software versions, Ethernet interfaces, and connected CANopen nodes.

Command syntax:

```
cli product-info [OPTIONS]
```

When the command “cli product-info” is entered a default format will be printed in the output. The user has the option to print the output in a JSON format too.

Key Information Provided

- **build_id** → Software build version of the CLI and PDU.
- **model_string** → Model identification string.
- **operating_hours** → Total hours of device operation.
- **order_no** → Order number (manufacturing reference).
- **serial** → Device serial number.
- **sw_version** → Firmware/software version of the PDU.
- **eth1 / eth2** → Network interface details (MAC, IPv4, IPv6).
- **canopen_nodes** → Internal module details (node ID, product ID, serial, HW/SW version).

15 Licenses

The system provides access to information about installed licenses and corresponding license files. This is useful for compliance and auditing purposes.

- **Listing All Licenses**

  The command cli license-info licenses prints a complete list of all licenses associated with installed packages.
  This includes every package dependency.

```
BN-PRO-7A93CE28:~$ cli license-info licenses
<very long list of licenses and package names>
```

- **Viewing a license File**

  From the list above, you can select a package name and display its license file using:

```
BN-PRO-7A93CE28:~$ cli license-info license-file util-linux-wdctl
<content of the license file is displayed here>
```

  This provides the full text of the license used by the specified package.

16 Cluster

A **PDU Cluster** combines up to **20 PDUs (one Main, 19 Links)** into one *“virtual PDU”*. In every cluster, one PDU is configured as the **Main PDU**, while all others act as **Link PDUs**.

The user will interact mainly with the Main PDU:

- It provides the management Interface
- It collects and synchronizes measurements and events
- It distributes configuration and control to all link PDUs

Cluster communication is handled automatically:

- The **Main and Link PDUs** are physically connected via **LINK IN / LINK OUT** ports.
- For network management, the **Main PDU** must be connected to the Ethernet switch (either *eth1* or *eth2*).
- The cluster supports both **bus (line)** and **ring** topologies, with ring offering better resilience in case of a cable fault.

In the following chapters, we’ll look at:

- How to set up **physical connections**
- The roles of **Main vs Link PDUs**
- Supported **cluster topologies**
- **Cluster behavior** during faults or maintenance
- **Best practices and pitfalls** to avoid

16.1 Physical Connections

- Each PDU has **LINK IN** and **LINK OUT** ports.
- These ports are used to build the cluster by daisy-chaining devices together.
- The **Main PDU** is the only one that requires an Ethernet connection to the network switch. All other PDUs (called **Link PDUs**) communicate through the cluster cabling.
- The order of connecting **LINK IN** ↔ **LINK OUT** does **not** matter if every device is properly linked into the chain.

16.2 Main vs Link PDUs

When building a PDU Cluster, one PDU takes the role of the **Main**, while all others act as **Links**.

Main PDU

- **The Main is the central controller of the cluster.**
- **It provides services such as:**
  - **Web interface and CLI control for the entire cluster,**
  - **DNS, NTP, and cluster management functions,**
  - **Cluster-wide alarm and measurement aggregation.**
- **Additional configuration options are available on the Main, including:**
  - **Setting its IP address,**
  - **Assigning its role as Main,**
  - **Accepting new Link PDUs into the cluster.**
- **For network access, the Main can be connected to either eth1 or eth2**

Link PDU

- A **Link** is part of the cluster.
- Links always forward their data and status to the Main.
- In the **local GUI**, a Link still shows:
    - Its own measurements,
    - Its own alarms.
- However, Links do not provide cluster-level settings:
    - They cannot accept new Links,
    - They cannot be configured as Main directly from their own GUI.
- Higher-level services (e.g. Modbus, SNMP, full alarm observer) are disabled on a Link.
- Links continuously check for connectivity to the Main
- The Link PDUs can also be connected to the Switch, they will get an IP address, but in the Link Connected state, they are not accessible for login (SSH/serial)
- In the “Cluster Modes” section these special behaviors will be explained:
    - If the Main fails, a Link enters Maintenance Mode
    - In this mode, the Link regains login access over SSH, or serial and ca be promoted to Main mode
    - Also, how the transfer-main/promote-to-main commands work when doing a planned switch.

Replica Behavior

- Each Link maintains a **replica** of the Main's configuration and cluster state.
- This allows any Link to be promoted to become the new Main if the current Main fails.
- Replication happens automatically in the background.

Failover Concept

- If the Main becomes unreachable for more than 60 seconds, Links switch to a **maintenance mode**.
- In this state (maintenance), the Link can be promoted to be the Main — restoring full functionality to the cluster.
- This provides resilience: even if the Main is lost, the cluster can continue operating.

16.3 Topologies

Two types of connection topologies are possible:

- **Ring Topology**
  - PDUs are connected in a closed loop (LINK-IN/LINK-OUT cables).
  - Advantage: high redundancy. If any single link cable is damaged, communication is automatically rerouted.
  - Cluster operation is unaffected if there is no second failure.

- **Bus (Line) Topology**
    - PDUs are connected in a straight line, without closing the loop.
    - Simpler cabling, but less resilient.
    - If the cable between the **Main PDU and the first Link PDU** fails, the **entire cluster** goes into *Link Disconnected* state for about **60 seconds**, followed by *Link Maintenance* state.
    - If a cable further down the line fails (e.g., between Link PDU 2 and Link PDU 3), then only the PDUs **beyond the break** are lost from the cluster.
    - Communication recovery is automatic once the broken cable is replaced with a functional one

17 Cluster CLI Reference

The `cli cluster` command provides all subcommands necessary to configure, monitor, and manage a PDU Cluster. It serves as the main entry point for interacting with the cluster from the CLI.

Usage.

```
cli cluster [OPTIONS] COMMAND [ARGS]...
```

Description:

- The cluster CLI allows users to manage **Link PDUs**, monitor the cluster state, configure cluster modes, and manipulate cluster topology.
- Subcommands include operations for **candidate acceptance**, **mode configuration**, **listing cluster PDUs**, **swapping positions**, and **removing PDUs from the cluster**.

Notes:

- It is recommended to understand **physical connections** (Main vs Link) and **topologies** before executing cluster CLI commands.
- Changes to cluster configuration may temporarily disrupt connections to Link PDUs, especially during firmware updates.
- All operations should follow the recommended sequence described in the **Cluster Setup Workflow** (Chapter 18) to avoid misconfiguration.

17.1 Cluster Mode

17.1.1 get

Purpose: Show the current cluster mode of the PDU.

Example:

```
BN-PRO-7AD4B490:~$ cli cluster mode get
Cluster Mode: main
```

17.1.2 promote-to-main

Purpose: Promote a Link PDU that is currently in **Maintenance Mode** to become the new Main PDU.

Example:

```
BN-PRO-7AD4B490:~$ cli cluster mode promote-to-main
Successfully promoted to Main PDU
```

Notes:

- Must be executed on a Link PDU that is in **Maintenance Mode**.
- After promotion, the PDU takes over the Main role, and services are adjusted accordingly.

17.1.3 set

Purpose: Set the cluster mode of the PDU.

Example:

```
BN-PRO-7AD4B490:~$ cli cluster mode set scan-main
Cluster mode changed successfully
BN-PRO-7AD4B490:~$ cli cluster mode set main
Cluster mode changed successfully
```

```
BN-PRO-7AD4B490:~$ cli cluster mode set bridged
Cluster mode changed successfully
BN-PRO-7AD4B490:~$ cli cluster mode set power-only
Cluster mode changed successfully
```

17.1.4 transfer-main

Purpose:

Transfer the Main PDU role from the current Main PDU to a designated Link PDU. This command is executed **from the current Main PDU**.

Example:

```
BN-PRO-7AD4B490:~$ cli cluster mode transfer-main BAMASTUT
Successfully transferred Main PDU to BAMASTUT, rebooting now ...
BN-PRO-7AD4B490:~$ Connection to 10.150.37.13 closed by remote host.
Connection to 10.150.37.13 closed.
```

Note:

- Both the previous Main PDU and the new Main PDU reboot during the transfer.

17.2 Cluster State

Purpose:

Show the **current state of the cluster** and individual PDUs. It is useful to check if a cluster is formed, if the Main PDU is available, or if a Link PDU is connected.

Example:

```
BN-PRO-7AD4B490:~$ cli cluster state get
Cluster State: Single Power-Only
BN-PRO-7AD4B490:~$ cli cluster state get
```

```
Cluster State: Single
```

17.3 Cluster Candidate

Purpose:

Manage and accept **Link PDUs** that can join the cluster. This is used to:

- see available candidates.
- accept them into the cluster.
- replace a defect PDU with one that is functional

Example:

1. see available candidates

```
BN-PRO-7AD4B490:~$ cli cluster candidate list

Available Link-Candidates

-------------------------

AABBCCDQ    pin: ABF2
```

2. accept the candidates into the cluster using the PIN

```
BN-PRO-7AD4B490:~$ cli cluster candidate accept --validate-pin ABF2
AABBCCDQ

Accepted given Link-PDU (AABBCCDQ) into the cluster
```

3. accept the candidates into the cluster without using the PIN

```
BN-PRO-7AD4B490:~$ cli cluster candidate accept -- a0000017

Accepted given Link-PDU (a0000017) into the cluster
```

4. replace a defective PDU with one that is functional.

```
BN-PRO-7AD4B490:~$ cli cluster candidate list

Available Link-Candidates

-------------------------

AABC00AD    pin: 1789

BN-PRO-7AD4B490:~$ cli cluster candidate accept --replace-pdu-id AABBCCDQ --  AABC00AD

The replacement option removes the replaced PDU from the Cluster. Are you sure you want to continue? [y/N]: y

Accepted given Link-PDU (AABC00AD) into the cluster
```

This PDU must be connected to the main PDU. Only in this case it will be seen in the candidate list then accepted as a replacement for an already linked PDU that is defective.

17.4 Cluster List

List all PDUs in the cluster with index and mode.

Example:

```
BN-PRO-7AD4B490:~$ cli cluster list

  Index   PDU-ID

-------    --------

      1      AAABMS0O

      2      AABC00AD

BN-PRO-7AD4B490:~$ cli cluster list -m

  Index    PDU-ID    Mode

-------    --------          ------

      1   AAABMS0O   main

      2   AABC00AD     link
```

```
BN-PRO-7AD4B490:~$ cli cluster list -j

["AAABMS0O", "AABC00AD", null, null, null, null, null, null, null, null,
null, null, null, null, null, null, null, null, null, null]
```

17.5 Cluster Swap

Purpose:

Swap the position of two PDUs in the cluster list.

The swap can also exchange configuration data between the two PDUs.

There are 2 options:

1.Swap PDU position only:

```
BN-PRO-7AD4B490:~$ cli cluster swap 2 1

Successfully swapped
```

This command only swaps the positions of the two PDUs in the cluster list.

2.Swap position and configuration

When the --swap-config option is used, configuration details such as labels, descriptions, alarm configuration are also exchanged.

Intended Use Case for --swap-config

cli cluster swap --swap-config is designed for **PDU replacement scenarios**.

If a PDU in the cluster is partially defective — for example:

- no proper measurement reporting
- alarms or thresholds malfunction
- communication unstable
- hardware aging or damaged

... but the device is still **visible in the cluster list**, even if unhealthy, the user may replace it

Workflow example:

1. The faulty PDU is still listed in the cluster (e.g. slot 3).
2. A new physical PDU is installed and detected as a link-candidate.
3. The user accepts it into the cluster (joins empty slot 14, for example).
4. Then the user runs:

```
BN-PRO-7AD4B490:~$ cli cluster swap --swap-config 14 3
Successfully swapped
```

Notes:

- The main use case of --swap-config is replacing a broken PDU in the cluster.
- The user does **not** need to re-configure the replacement PDU manually if this command is used

17.6 Cluster Remove

Purpose: Remove a link PDU from the cluster

There are 2 options:

1. Removing a link PDU that is in the state Link (connected).

```
BN-PRO-7AD4B490:~$ cli cluster remove AABC00AD
Successfully removed given Link-PDU (AABC00AD) from the cluster
```

2. Forcing a removal of a PDU

```
BN-PRO-7AD4B490:~$ cli cluster remove -f a0000017
```

```
Successfully removed given Link-PDU (a0000017) from the cluster
```

18 Cluster Setup Workflow

This chapter provides step-by-step examples on how to create and manage a **PDU Cluster** in different operating scenarios.
While the previous section described the CLI commands in detail, this chapter focuses on how these commands are used together in real workflows.

A cluster can be formed in several ways depending on the **network configuration** and **intended purpose** of the PDUs.
The most common scenario is the **Main–Link connection**, where one PDU acts as the *Main* and the others operate as *Link PDUs*.
Other configurations, such as **Bridged Mode** or **Power Only Mode**, are also supported for specialized use cases.

Each workflow below explains:

- The **physical setup** (power, Ethernet, Link connections)
- The **required CLI steps**
- The **expected states and behavior**
- Important **notes and recommendations** for stable operation

These examples are intended to guide administrators through the full cluster creation and management process — from first connection to final verification.

18.1 Creating a New Cluster from Scratch

Overview
In this case, a user forms a cluster by connecting multiple PDUs through their LINK IN / LINK OUT ports.
One PDU, connected to the Ethernet switch, will act as the Main PDU.
All other PDUs will become Link PDUs, connected sequentially or in a ring topology.

18.1.1 Power on and Basic Setup

When PDUs are delivered from the factory, **each device starts in the state Single**.
This means that the unit operates independently — it is not yet part of any cluster and behaves like a standalone PDU.

Before forming a cluster, the user should check that:

- All PDUs are properly connected to power.
- The intended *Main PDU* is also connected to the **Ethernet switch**, since it will manage communication within the cluster.
- Each *Link PDU* remains powered on

Once powered, the user can verify the state, see chapter 17.2 Cluster State

Notes:

2. The link PDUs can be physically linked to the main PDU through LINK IN-LINK OUT ports before or after the main PDU is chosen. Either way the functionality should work properly in the cluster.
3. The link PDUs can also be connected to the Ethernet switch. This case will help later if:
    - Main PDU is defected, and the user wants from the links PDU to choose one to make it the new main, *see chapter 17.1.2* promote-to-main
    - No PDU is defected, but the user wants to choose another PDU to be the new main PDU see *Chapter 17.1.4* transfer-main

18.1.2 Set the Main PDU and accept the candidates

After all PDUs are connected to power and are in Single state, the next step is to select which one will act as the Main PDU.

The Main PDU will coordinate the entire cluster and manage communication with all Link PDUs.

1. Choose the Main PDU: From any powered PDU, the user should decide which device will become the *Main PDU*. It is recommended that this unit be **connected to the Ethernet switch**, as it will handle network communication for the cluster.
2. Set the mode to main, see *Chapter 17.1.3* set
3. Verify the new mode of the PDU, see *Chapter 17.1.1* get
4. Connect the Link PDUs: Once the Main PDU is configured, physically connect the **Link PDUs** to it using the **LINK IN / LINK OUT** ports. These connections are required for the devices to communicate and for the Main-PDU to detect available Link candidates.

   Important:

   The *Link In / Link Out* connections are mandatory for the PDUs to appear in the candidate list. Without this physical link, the Main PDU cannot discover or communicate with the others.
5. View available candidates and accept them in the cluster: After the physical connections are established, the Main PDU will automatically detect the other units and can accept them in the cluster one by one, see *Chapter 17.3* Cluster Candidate
6. Check the candidate list is empty. This can be verified by accessing the command:

   ```
   BN-PRO-7AD4B490:~$ cli cluster candidate list
   Available Link-Candidates
   -------------------------
   ```

   If there are no available links the list will be empty.
7. Check the cluster list

The user can verify if all chosen link Candidates are now part of the cluster and listed as link PDUs, see *Chapter 17.4* Cluster List

Note:

If the Link PDUs do not have the same software version as the Main PDU, they will be automatically updated before joining the cluster. This update involves a reboot of each Link device. The user should wait until all PDUs have completed their updates and appear in the candidate list.

18.2 Software Update in an Existing Cluster

When a new software version becomes available, it is sufficient to update only the **Main PDU**. The Main PDU will automatically coordinate and propagate the update to all connected Link PDUs.

1. Update the Main PDU

Perform the firmware update on the Main PDU using the procedure described in **Chapter 4.4** Firmware update**.** Once the update is completed, the Main PDU will automatically reboot and start operating with the new software version.

2. Automatic Link Updates

After the Main PDU is updated and running, all connected Link PDUs will automatically detect the version difference.
They will begin updating sequentially to match the Main PDU's software version.

Important Notes:

- The update of all Link PDUs can take some time, depending on the number of devices in the cluster.
- During this process, each Link PDU will reboot after its update is complete.

- Cluster communication might be temporarily interrupted while updates are ongoing.

3. Verification

Once all Link PDUs have completed their updates:

- The cluster will stabilize automatically.
- The user can confirm that all PDUs have the same version by running the command:

```
cli product-info
```

For each PDU the “build_id” parameter should be the same

4. System Ready

After all Link PDUs finish updating, the cluster is fully synchronized and operational again.
No further user action is required.

18.3 Power Only Mode

Overview

Power-Only Mode is an operating mode in which the PDU's **Link-In/Link-Out** ports remain powered, but **all data communication across the link interface is disabled**.
No cluster membership is formed and no PDU exchanges state or control information through the Link ports.
In this mode, each PDU behaves as a **fully independent device**, even if physically chained to other PDUs.

Purpose of Power-Only Mode

Power-Only Mode is intended for installations where multiple PDUs may be linked physically but must remain logically separate and individually managed.

This mode should be used when:

- The user requires **independent management access per PDU**
- A logical cluster **must not** or **should not** be formed
- Installations require **large-scale deployments** without cluster-size limitations
- Link cabling is required only as a physical link path, not for data transport

Allowed Transitions to Power-Only Mode

A PDU may enter Power-Only Mode when currently operating in one of the following states:

Current operating state	Switch to Power-Only allowed
Main (no linked PDUs)	✓
Single-PDU operation	✓
Bridged mode	✓
Active cluster with linked PDUs	✗ *Not allowed — link endpoints must be removed first*

➢ The CLI Command to activate Power-Only Mode

```
BN-PRO-7A93CE24:~$ cli cluster mode set power-only

Cluster mode changed successfully
```

Once applied, cluster services are immediately disabled, and link communication is terminated.

Notes:

Power-Only Mode is used to keep PDUs logically isolated even if physically chained.
Communication via Link-In/Link-Out is disabled, and each device operates as an independent unit.
If PDUs are already physically linked and one is connected to Ethernet,
switching that PDU to Power Only Mode will automatically switch all linked PDUs as well.

18.4 Bridged Mode

Overview

Bridged Mode allows multiple PDUs to share one network connection through the Link-In/Link-Out ports.
The PDUs can be chained using link cables, connecting **only one** of them to the LAN, and **all PDUs will receive an IP address** through that single connection.
Every PDU still behaves as an individual device with its **own IP**, but they all reach the network through the first connected unit.

How Bridged Mode Works

- PDUs are connected in a chain or ring using **Link-In → Link-Out** ports.
- Inside the PDUs, these link ports are automatically bridged together.
- This bridge behaves like a built-in network switch.
- Only **one PDU has to be connected to the Ethernet network**.
- All other PDUs "hop" through the Link chain and receive an IP over the same uplink.

Example:

LAN Router —(Ethernet)— PDU1 —(Link cable)— PDU2 —(Link cable)— PDU3

Even though only **PDU1** is directly connected to the network:

- PDU2 and PDU3 can still reach the router
- DHCP requests pass through PDU1
- Each PDU receives its own IP address

No external switch is required — the PDUs create their own virtual network automatically.

Switching to Bridged Mode

A PDU can enter Bridged Mode when it is in:

Current state	Allowed to switch to bridged
Main (no links)	✓
Single-PDU mode	✓
Power-Only Mode	✓
Active cluster mode with multiple linked units	✗ links must be cleared first

Note:

If PDUs are already physically linked and one is connected to Ethernet,
switching that PDU to Bridged Mode will automatically switch all linked PDUs as well.
(Same behavior as Power-Only Mode.)

- The CLI Command to activate Bridged Mode

```
BN-PRO-7A93CE24:~$ cli cluster mode set bridged

Cluster mode changed successfully
```

Once activated, the link ports form a software bridge and network forwarding becomes active.

19 Cluster – Best Practices and Common Pitfalls

This chapter provides guidance on typical situations that may occur during cluster operation.
It describes common mistakes, how to prevent them, and possible recovery actions if an issue arises.

Examples include scenarios such as a factory reset of the Main PDU, cable disconnections, or Link PDUs becoming unresponsive.
Following these recommendations helps ensure stable operation and simplifies troubleshooting in the field.

19.1 Access and Connectivity Overview

Maintaining stable access and connectivity is essential for ensuring cluster integrity and enabling recovery operations when issues occur.

Access Possibilities by PDU State

The CLI can be accessed both locally (via USB-C) and remotely (via the Ethernet interface, typically over SSH) depending on the PDU's operational state.
The following table summarizes in which cluster states user login is possible:

PDU State	CLI Access (Local)	CLI Access (Remote)	Notes
Main	✅ Yes	✅ Yes	Fully operational; manages the cluster.
Single	✅ Yes	✅ Yes	Independent operation; no cluster connection.
Single Link-Candidate	✅ Yes	✅ Yes	Ready to join a cluster; can be accepted by a Main PDU.

PDU State	CLI Access (Local)	CLI Access (Remote)	Notes
Link (Maintenance)	✅ Yes	✅ Yes	Limited CLI functionality; allows promotion to Main.
Single Bridged	✅ Yes	✅ Yes	
Single Power Only	✅ Yes	✅ Yes	
Link (connected)	❌ No	❌ No	
Link (Disconnected)	❌ No	❌ No	Temporary state (up to ~60 seconds) after communication loss with Main PDU. CLI access unavailable during this period.

Important Connectivity Guideline!!!!

Do not disconnect physically the PDUs from the network or from each other (LINK IN LINK OUT) during a cluster reformation or recovery process.

19.2 Factory Reset on a Main PDU that has link PDUs connected

Performing a factory reset on the **Main PDU** in an active cluster will cause the cluster connection to be lost.
After the reset, the Main PDU reverts to its factory defaults and returns to the **Single** state.
The Link PDUs, which were previously connected to the Main, will react as follows:

Cluster Reaction

- **First 60 seconds:**

  All Link PDUs enter the state Link (Disconnected) – attempting to re-establish communication with the former Main PDU.

- **After 60 seconds:**
  They automatically transition to Link (Maintenance) – a mode that allows user intervention (for example, promotion to Main).
  See Chapter 17.1.2 promote-to-main

Access during this state is possible if:

- **The Link PDU is connected to the Ethernet switch, or**
- **The user connects locally via the USB-C port (see Chapter 1.1** Accessing the CLI (Local and Remote)**).**

Remote Access Options

If the Link PDUs are also connected to the Ethernet switch, users might attempt to connect to them remotely (e.g., via SSH).
However, executing a factory reset remotely on a Link PDU will **not** work in this state.
Example output:

```
BN-PRO-7A93CF72:~$ cli do fw-reset
Reset all settings to factory defaults? [y/N]: y
Error: HTTP status code 404 (Not Found)
    Not Found
```

This error appears because Link PDUs in maintenance mode have only a limited set of CLI functions available.

Possible Recovery Options – Promoting a Link PDU to Main

When the original Main PDU becomes unavailable (for example, after a factory reset or hardware failure), any connected Link PDU that enters the Link (Maintenance) state can be promoted to become the new Main PDU.

In this state, the device can be accessed via SSH, and its current cluster state can be verified:

```
BN-PRO-7AD4B490:~$ cli cluster state get
Cluster State: Link Maintenance
```

To recover cluster functionality, the Link PDU in maintenance mode can be promoted to Main using:

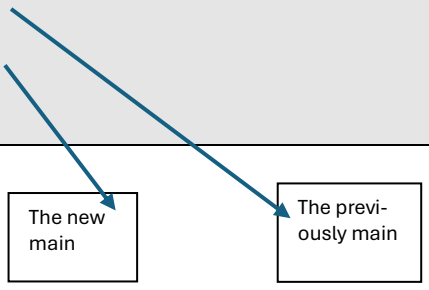
```
BN-PRO-7AD4B490:~$ cli cluster mode promote-to-main
Successfully promoted to Main PDU
```

After promotion, the PDU changes its state:

When the cluster list is checked afterwards, all previously known PDUs are still displayed:

```
BN-PRO-7AD4B490:~$ cli cluster state get
Cluster State: Main
```

```
BN-PRO-7AD4B490:~$ cli cluster list -m
  Index   PDU-ID     Mode
-------     --------     ------
      1     AAABMS0S   link
      2     AAABMS0R   main
      3     a0000017     link
```



19.2.1 How to bring back the former main PDU into the cluster as link

Even though the PDU that was previously the Main device, has been factory-reset and is no longer active in the cluster, it still appears in the list as a Link.

To fully clean up the configuration, the non-functional (previous Main) PDU can be removed from the cluster list using the **remove force** command in this case:

```
BN-PRO-7AD4B490:~$ cli cluster remove --force AAABMS0S

Successfully removed given Link-PDU (AAABMS0S) from the cluster
```

Note:

The presence of this inactive entry does not affect normal cluster operation.

However, removing it is recommended to maintain a clean and accurate cluster configuration.

After removal, the previously Main PDU (which has been factory reset) appears again as a **new Link Candidate**.

```
cli cluster candidate list

Available Link-Candidates

-------------------------

AAABMS0S
```

This means it can be re-added to the cluster using the standard acceptance process.

If the user knows the IP address of the reset PDU or can connect locally via the USB-C connection (see Chapter 1.1 Accessing the CLI (Local and Remote)), the cluster can be restored easily.

This method ensures that the **cluster is not lost completely**—only the previously Main PDU is reset and temporarily removed.

Once it reappears as a candidate, it can be accepted back into the cluster to restore full functionality.

Note:

After removal, the cluster configuration is refreshed on all remaining PDUs.

Only the reset PDU reverts to factory defaults, while the rest of the cluster remains operational and synchronized.

19.3 Connection Lost Status on a Link PDU

A connection lost status indicates that one of the Link PDUs is no longer communicating with the Main PDU.

This can be detected when running the following command:

```
cli get -a -
```

Here one of the statuses will show:

```
/pdu[a0000017]/status                                    connection lost
```

If one of the PDUs shows the status **connection lost** for a long time in the measurement list, it means that the device is currently not functional or unreachable within the cluster.

Recommended Action

When a Link PDU remains in the *Connection Lost* state for an extended period, the user should first attempt to remove it **normally** from the cluster:

```
BN-PRO-7A93CF72:~$ cli cluster list -m
  Index     PDU-ID          Mode
-------     --------       ------
    1       AAABMS0R         main
    2       a0000017         link
    3       AAABMS0S         link
```

1. Attempt a standard removal:

```
BN-PRO-7A93CF72:~$ cli cluster remove a0000017
Error: HTTP status code 504 (Gateway Timeout)
    Link didn't confirm CMD in time (timeout)
```

This usually indicates that:

- The **physical connection (Ethernet cable (LINK IN/LINK OUT)) is damaged or disconnected**, or
- The **Link PDU is powered off**.

2. Forced Removal
   If the normal removal fails with a timeout or similar error, the device can be removed by force:

19.4 Restoring Cluster Backups – Risks, Expected Behavior, and Limitations

This section describes what happens when a backup is restored onto a system that no longer matches the original cluster structure. This scenario is relevant when a user exports a cluster configuration, removes link PDUs, and later restores the old configuration while the physical topology was not changed.

Observed scenario:

Step Performed	**Result**
1. A full cluster backup was created	Backup stored successfully
2. All links were removed via CLI	Main stands alone, but link hardware is still physically connected
3. Backup was uploaded to the Main PDU	Configuration restored
4. Physically connected Link PDUs reappear automatically	Both as **links** and as **cluster candidates**

Example output:

```
BN-PRO-7AD4B474:~$ cli cluster list -m
```

```
  Index   PDU-ID    Mode
-------   --------  ------
      1   BUBALUBA  main
      2   -5308498  link
      3   -6453291  link
...
     14   -8123170  link

BN-PRO-7AD4B474:~$ cli cluster candidate list
Available Link-Candidates
-------------------------
AAAAAAAF    pin: AFA2
AABBCCDQ    pin: 1756
AABC00AD    pin: 047C
MABARORO    pin: 0027
...
AAABMSSS    pin: 703C
```

After restoring, two states overlap:

Shown as existing links	Seen as new candidates
Old links from the backup	All physically connected PDUs

Critical Risk – Double Presence

When the user accepts candidates **without removing the existing links first**, the cluster can reach or exceed its 20-PDU limit:

If this happens, some PDUs may enter a reboot loop and **cannot be recovered** through the cluster interface anymore.

Important Warning

A cluster backup must **never** be restored onto a PDU system that no longer matches the original cluster structure.

PDUs that still exist physically will be detected again as new link candidates, resulting in duplicate entries.
This may cause reboot loops and permanent device loss.

Recommended Safe Practice

Safe Procedure	**Unsafe Procedure**
Restore a backup **only** on a system that has the same cluster structure it was captured from	Restore a backup while previous cluster links are still physically connected
Remove link PDUs **physically**, not only logically	Remove links via CLI only, then restore backup

19.5 CLI behavior during Firmware Updates in a Cluster

When the Main PDU is updated via:

```
cli do fw-update <file>
```

Expected behavior:

Action	**Effect**
Main updates first	Links remain in old version temporarily
Links auto-update	Order unrelated to physical link order
No user action required	Accepting them again is **NOT** expected

! ! ! Do not accept or remove Link PDUs during update propagation.

19.6 Issue when upgrading to Firmware v1.2.x or newer while older Versions are still active on the PDU

When a PDU still contains older firmware bundles (e.g. **v1.0.3** or **v1.1.1-1**) and the user installs a software version **newer than v1.2.x**, the device may enter an intermediate state (**scan-main**) in which it cannot be promoted to **main** directly.

Observed Behavior:

If the user attempts to set this PDU as **main using** `cli cluster mode set main`, the system rejects the request:

CLI response:

```
Error: HTTP status code 400 (Bad Request)

PDU (BAND0023) is already part of the cluster
```

Root Cause

This condition occurs only when:

- Firmware versions (< v1.2.x) are still stored on the device, and
- A newer firmware is installed above them

This combination blocks direct promotion to main mode.

Recovery Procedure

To successfully switch the PDU to main, the mode must first be changed:

Step 1 – Change to Bridged Mode OR Power-Only Mode

```
cli cluster mode set bridged
```

OR

```
cli cluster mode set power-only
```

Step 2 – Promote to Main

```
cli cluster mode set main
```

Only after switching through one of these transition states will the device accept **main mode**.

If you have any questions or concerns regarding the PDUs or the documentation, please do not hesitate to contact us.

Bachmann GmbH
Ernsthaldenstraße 33
70565 Stuttgart Deutschland
Telefon: +49 711 86602-888
E-Mail: bluenet.support@bachmann.com
Internet: www.bachmann.com

We power your life.