readme.md for @serve.zone/nupst

⚡ NUPST — Network UPS Shutdown Tool

Keep your systems safe when the power goes out. NUPST is a lightweight, battle-tested CLI tool that monitors UPS devices via SNMP or NUT (UPSD) and orchestrates graceful shutdowns during power emergencies — including Proxmox VMs, LXC containers, and the host itself.

Distributed as **self-contained binaries** with zero runtime dependencies. No Node.js, no Python, no package managers. Just download and run.

Issue Reporting and Security

For reporting bugs, issues, or security vulnerabilities, please visit community.foss.global/. This is the central community hub for all issue reporting. Developers who sign and comply with our contribution agreement and go through identification can also get a code.foss.global/ account to submit Pull Requests directly.

□ Features

- □□ **Multi-UPS Support** — Monitor multiple UPS devices from a single daemon
- □□ **Dual Protocol Support** — SNMP (v1/v2c/v3) for network UPS + UPSD/NIS for USB-connected UPS via NUT
- □□ **Proxmox Integration** — Gracefully shut down QEMU VMs and LXC containers before host shutdown

- □□ **Group Management** — Organize UPS devices into groups with flexible operating modes
  - **Redundant Mode** — Only trigger actions when ALL UPS devices in a group are critical
  - **Non-Redundant Mode** — Trigger actions when ANY UPS device is critical
- ⚙ **Action System** — Define custom responses with flexible trigger conditions
  - Battery & runtime threshold triggers
  - Power status change triggers
  - Webhook notifications (POST/GET)
  - Custom shell scripts
  - Proxmox VM/LXC shutdown
  - Configurable shutdown delays
- □□ **Multiple UPS Brands** — CyberPower, APC, Eaton, TrippLite, Liebert/Vertiv, and custom OID configurations
- □□ **HTTP API** — Optional JSON status endpoint with token authentication
- □ **Pause/Resume** — Temporarily suppress actions during maintenance windows
- □□ **Network Loss Detection** — Detects unreachable UPS devices and prevents false shutdowns
- □□ **Power Metrics** — Monitor output load, power (watts), voltage, and current
- □□ **Single Binary** — Zero runtime dependencies. Download, `chmod +x`, run.
- □□ **Cross-Platform** — Linux (x64, ARM64), macOS (Intel, Apple Silicon), Windows

□□ Quick Start

One-Line Installation

```
curl -sSL https://code.foss.global/serve.zone/nupst/raw/branch/main/install.sh | sudo bash
```

Initial Setup

```
# 1. Add your first UPS device (interactive wizard)
sudo nupst ups add

# 2. Test the connection
nupst ups test

# 3. Enable and start monitoring
sudo nupst service enable
sudo nupst service start
```

```
# 4. Check status
nupst service status
```

That's it! Your system is now protected. □□

□□ Installation

Automated Installer (Recommended)

```
curl -sSL https://code.foss.global/serve.zone/nupst/raw/branch/main/install.sh | sudo bash
```

What it does:

1. Detects your platform (OS + architecture)
2. Downloads the latest pre-compiled binary
3. Installs to `/opt/nupst/nupst`
4. Creates symlink at `/usr/local/bin/nupst`
5. Preserves existing configuration

Options:

```
# Install specific version
curl -sSL https://code.foss.global/serve.zone/nupst/raw/branch/main/install.sh | \
  sudo bash -s -- --version v5.0.0

# Custom installation directory
curl -sSL https://code.foss.global/serve.zone/nupst/raw/branch/main/install.sh | \
  sudo bash -s -- --install-dir /usr/local/nupst

# Show help
curl -sSL https://code.foss.global/serve.zone/nupst/raw/branch/main/install.sh | bash -s -- --
help
```

Manual Installation

Download the binary for your platform from releases:

Platform	Binary
Linux x64	`nupst-linux-x64`
Linux ARM64	`nupst-linux-arm64`
macOS Intel	`nupst-macos-x64`
macOS Apple Silicon	`nupst-macos-arm64`
Windows x64	`nupst-windows-x64.exe`

```
# Download (replace with your platform)
curl -sSL https://code.foss.global/serve.zone/nupst/releases/download/v5.0.0/nupst-linux-x64 -o nupst
chmod +x nupst
sudo mv nupst /usr/local/bin/nupst
```

Via npm

```
npm install -g @serve.zone/nupst
```

> This downloads the appropriate pre-compiled binary for your platform during installation.

Verify Installation

```
nupst --version
nupst help
```

□□ CLI Reference

Command Structure

```
nupst <command> [subcommand] [options]
```

Global Options

Flag	Description
`--version`, `-v`	Show version
`--help`, `-h`	Show help
`--debug`, `-d`	Enable debug mode (verbose SNMP/UPSD logging)

Service Management

```
nupst service enable          # Install and enable systemd service
nupst service disable         # Stop and disable systemd service
nupst service start           # Start the service
nupst service stop            # Stop the service
nupst service restart         # Restart the service
nupst service status          # Show service and UPS status
nupst service logs            # Tail live service logs (Ctrl+C to exit)
```

UPS Device Management

```
nupst ups add                 # Add a new UPS device (interactive wizard)
nupst ups edit [id]           # Edit a UPS device
nupst ups remove <id>         # Remove a UPS device
nupst ups list                # List all UPS devices
nupst ups test                # Test all UPS connections
```

During `nupst ups add`, you'll choose a communication protocol:

- **SNMP** — For network-attached UPS with an SNMP agent (default)
- **UPSD/NIS** — For USB-connected UPS managed by a local NUT server

Group Management

```
nupst group add               # Create a new UPS group
nupst group edit <id>         # Edit a group
nupst group remove <id>       # Remove a group
nupst group list              # List all groups
```

Action Management

```
nupst action add <target-id>          # Add action to a UPS or group
nupst action remove <target-id> <idx> # Remove an action by index
nupst action list [target-id]         # List actions (optionally for a target)
```

Pause/Resume

Temporarily suppress actions during maintenance (UPS polling continues):

```
nupst pause                   # Pause indefinitely
nupst pause --duration 30m    # Pause for 30 minutes (auto-resume)
nupst pause --duration 2h     # Pause for 2 hours
nupst pause --duration 1d     # Pause for 1 day (max: 24h)
nupst resume                  # Resume immediately
```

When paused:

- UPS polling continues (status is still visible)
- All actions are suppressed (no shutdowns, webhooks, scripts)
- The HTTP API response includes `"paused": true`
- Status display shows a `[PAUSED]` indicator

Feature Management

```
nupst feature httpServer      # Configure HTTP JSON status API
```

Other Commands

```
nupst config show             # Display current configuration
nupst upgrade                 # Upgrade to latest version (requires root)
nupst uninstall               # Completely remove NUPST (requires root)
```

⚙ Configuration

NUPST stores configuration at `/etc/nupst/config.json`. The easiest way to configure is through the interactive CLI commands, but you can also edit the JSON directly.

Example Configuration

```
{
  "version": "4.3",
  "checkInterval": 30000,
  "httpServer": {
    "enabled": true,
    "port": 8080,
    "path": "/ups-status",
    "authToken": "your-secret-token"
  },
  "upsDevices": [
    {
      "id": "ups-main",
      "name": "Main Server UPS",
      "protocol": "snmp",
      "snmp": {
        "host": "192.168.1.100",
        "port": 161,
        "community": "public",
        "version": 1,
        "timeout": 5000,
        "upsModel": "cyberpower",
        "runtimeUnit": "ticks"
      },
      "actions": [
        {
          "type": "proxmox",
          "triggerMode": "onlyThresholds",
          "thresholds": { "battery": 30, "runtime": 15 },
          "proxmoxHost": "localhost",
          "proxmoxPort": 8006,
          "proxmoxTokenId": "root@pam!nupst",
          "proxmoxTokenSecret": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx"
        },
        {
```

```
        "type": "shutdown",
        "triggerMode": "onlyThresholds",
        "thresholds": { "battery": 20, "runtime": 10 },
        "shutdownDelay": 10
      }
    ],
    "groups": ["datacenter"]
  },
  {
    "id": "ups-usb",
    "name": "Local USB UPS",
    "protocol": "upsd",
    "upsd": {
      "host": "127.0.0.1",
      "port": 3493,
      "upsName": "ups",
      "timeout": 5000
    },
    "actions": [
      {
        "type": "shutdown",
        "triggerMode": "onlyThresholds",
        "thresholds": { "battery": 15, "runtime": 5 },
        "shutdownDelay": 5
      }
    ],
    "groups": []
  }
],
"groups": [
  {
    "id": "datacenter",
    "name": "Data Center",
    "mode": "redundant",
    "description": "Redundant UPS setup",
    "actions": [
      {
        "type": "shutdown",
        "triggerMode": "onlyThresholds",
```

```
          "thresholds": { "battery": 10, "runtime": 5 },
          "shutdownDelay": 15
        }
      ]
    }
  ]
}
```

UPS Device Configuration

Protocol Selection

Each UPS device has a `protocol` field:

Protocol	Use Case	Default Port
`snmp`	Network-attached UPS with SNMP agent	161
`upsd`	USB-connected UPS via local NUT server	3493

SNMP Settings (`snmp` object)

Field	Description	Values / Default
`host`	IP address or hostname	e.g., `"192.168.1.100"`
`port`	SNMP port	Default: `161`
`version`	SNMP version	`1`, `2`, or `3`
`timeout`	Timeout in milliseconds	Default: `5000`
`upsModel`	UPS brand/model	`cyberpower`, `apc`, `eaton`, `tripplite`, `liebert`, `custom`
`runtimeUnit`	Battery runtime unit	`minutes`, `seconds`, or `ticks` (1/100s). Overrides auto-detection
`community`	Community string (v1/v2c)	Default: `"public"`

SNMPv3 fields (when `version: 3`):

Field	Description	Values
`securityLevel`	Security level	`noAuthNoPriv`, `authNoPriv`, `authPriv`
`username`	Authentication username	—

Field	Description	Values
`authProtocol`	Auth protocol	`MD5` or `SHA`
`authKey`	Auth password	—
`privProtocol`	Encryption protocol	`DES` or `AES`
`privKey`	Encryption password	—

UPSD/NIS Settings (`upsd` object)

For USB-connected UPS via NUT (Network UPS Tools):

Field	Description	Default
`host`	NUT server address	`127.0.0.1`
`port`	NUT UPSD port	`3493`
`upsName`	NUT device name	`ups`
`timeout`	Connection timeout (ms)	`5000`
`username`	Optional auth username	—
`password`	Optional auth password	—

NUT variables mapped: `ups.status`, `battery.charge`, `battery.runtime`, `ups.load`, `ups.realpower`, `output.voltage`, `output.current`

Action Configuration

Actions define automated responses to UPS conditions. They run **sequentially in array order**, so place Proxmox actions before shutdown actions.

Action Types

Type	Description
`shutdown`	Graceful system shutdown with configurable delay
`webhook`	HTTP POST/GET notification to external services
`script`	Execute custom shell scripts from `/etc/nupst/`
`proxmox`	Shut down Proxmox QEMU VMs and LXC containers via REST API

Common Fields

Field	Description	Values / Default
`type`	Action type	`shutdown`, `webhook`, `script`, `proxmox`
`thresholds`	Battery and runtime limits	`{ "battery": 0-100, "runtime": minutes }`
`triggerMode`	When to trigger	See Trigger Modes below

Trigger Modes

Mode	Description
`onlyPowerChanges`	Only when power status changes (online ↔ onBattery)
`onlyThresholds`	Only when battery or runtime thresholds are violated
`powerChangesAndThresholds`	On power changes OR threshold violations (default)
`anyChange`	On every polling cycle

Shutdown Action

```
{
  "type": "shutdown",
  "thresholds": { "battery": 20, "runtime": 10 },
  "triggerMode": "onlyThresholds",
  "shutdownDelay": 10
}
```

Field	Description	Default
`shutdownDelay`	Seconds to wait before shutdown	`5`

Webhook Action

```
{
  "type": "webhook",
  "thresholds": { "battery": 30, "runtime": 15 },
  "triggerMode": "powerChangesAndThresholds",
  "webhookUrl": "https://hooks.slack.com/services/...",
  "webhookMethod": "POST",
  "webhookTimeout": 10000
}
```

Field	Description	Default

`webhookUrl`	URL to call	Required
`webhookMethod`	HTTP method	`POST`
`webhookTimeout`	Timeout in ms	`10000`

Script Action

```
{
  "type": "script",
  "thresholds": { "battery": 25, "runtime": 10 },
  "triggerMode": "onlyThresholds",
  "scriptPath": "pre-shutdown.sh",
  "scriptTimeout": 60000
}
```

Field	Description	Default
`scriptPath`	Script filename in `/etc/nupst/`	Required
`scriptTimeout`	Execution timeout in ms	`60000`

□□ Proxmox Action

Gracefully shuts down QEMU VMs and LXC containers on a Proxmox node before the host is shut down.

```
{
  "type": "proxmox",
  "thresholds": { "battery": 30, "runtime": 15 },
  "triggerMode": "onlyThresholds",
  "proxmoxHost": "localhost",
  "proxmoxPort": 8006,
  "proxmoxTokenId": "root@pam!nupst",
  "proxmoxTokenSecret": "xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx",
  "proxmoxExcludeIds": [100, 101],
  "proxmoxStopTimeout": 120,
  "proxmoxForceStop": true,
  "proxmoxInsecure": true
}
```

Field	Description	Default
`proxmoxHost`	Proxmox API host	`localhost`

Field	Description	Default
`proxmoxPort`	Proxmox API port	`8006`
`proxmoxNode`	Proxmox node name	Auto-detect via hostname
`proxmoxTokenId`	API token ID (e.g. `root@pam!nupst`)	Required
`proxmoxTokenSecret`	API token secret (UUID)	Required
`proxmoxExcludeIds`	VM/CT IDs to skip	`[]`
`proxmoxStopTimeout`	Seconds to wait for graceful shutdown	`120`
`proxmoxForceStop`	Force-stop VMs/CTs that don't shut down	`true`
`proxmoxInsecure`	Skip TLS verification (self-signed certs)	`true`

Setting up the API token on Proxmox:

```
# Create token with full privileges (no privilege separation)
pveum user token add root@pam nupst --privsep=0
```

> ⚠ **Important:** Place the Proxmox action **before** the shutdown action in the actions array so VMs are stopped before the host shuts down.

Group Configuration

Groups coordinate actions across multiple UPS devices:

Field	Description	Values
`id`	Unique group identifier	—
`name`	Human-readable name	—
`mode`	Group operating mode	`redundant`, `nonRedundant`
`description`	Optional description	—
`actions`	Array of action configurations	—

Group Modes:

- **`redundant`** — Actions trigger only when ALL UPS devices in the group are critical. Use for setups with backup power units.
- **`nonRedundant`** — Actions trigger when ANY UPS device is critical. Use when all UPS units must be operational.

HTTP Server Configuration

```
# Interactive setup
sudo nupst feature httpServer
```

```
{
  "httpServer": {
    "enabled": true,
    "port": 8080,
    "path": "/ups-status",
    "authToken": "your-secret-token"
  }
}
```

Query the API:

```
# Bearer token
curl -H "Authorization: Bearer your-secret-token" http://localhost:8080/ups-status

# Query parameter
curl "http://localhost:8080/ups-status?token=your-secret-token"
```

Response format:

```
{
  "upsDevices": [
    {
      "id": "ups-main",
      "name": "Main Server UPS",
      "powerStatus": "online",
      "batteryCapacity": 100,
      "batteryRuntime": 45,
      "outputLoad": 23,
      "outputPower": 115,
      "outputVoltage": 230.5,
      "outputCurrent": 0.5,
      "consecutiveFailures": 0,
      "unreachableSince": 0,
      "lastStatusChange": 1729685123456,
```

```
      "lastCheckTime": 1729685153456
    }
  ],
  "paused": false
}
```

When monitoring is paused:

```
{
  "upsDevices": [...],
  "paused": true,
  "pauseState": {
    "pausedAt": 1729685123456,
    "pausedBy": "cli",
    "resumeAt": 1729686923456
  }
}
```

□□ Network Loss Detection

NUPST tracks communication failures per UPS device:

- After **3 consecutive failures**, the UPS status transitions to `unreachable`
- **Shutdown actions will NOT fire** on `unreachable` — this prevents false shutdowns from network glitches
- Webhook and script actions still fire, allowing you to send alerts
- When connectivity is restored, NUPST logs a recovery event with downtime duration
- The failure counter is capped at 100 to prevent overflow

Power status values: `online` | `onBattery` | `unknown` | `unreachable`

□□ Monitoring

Status Display

```
$ nupst service status
```

```
UPS Devices (2):
  ✓ Main Server UPS (online - 100%, 3840min)
    Host: 192.168.1.100:161 (SNMP)
    Groups: Data Center
    Action: proxmox (onlyThresholds: battery<30%, runtime<15min)
    Action: shutdown (onlyThresholds: battery<20%, runtime<10min, delay=10s)

  ✓ Local USB UPS (online - 95%, 2400min)
    Host: 127.0.0.1:3493 (UPSD)
    Action: shutdown (onlyThresholds: battery<15%, runtime<5min, delay=5s)

Groups (1):
  □ Data Center (redundant)
    UPS Devices (1): Main Server UPS
    Action: shutdown (onlyThresholds: battery<10%, runtime<5min, delay=15s)
```

Live Logs

```
nupst service logs
```

```
[2026-02-20 10:30:15] □ NUPST daemon started
[2026-02-20 10:30:15] ✓ Connected to Main Server UPS (192.168.1.100)
[2026-02-20 10:30:45] □ Status check: All systems normal
[2026-02-20 10:31:15] ⚠ Main Server UPS on battery (85%, 45min remaining)
[2026-02-20 10:35:00] ⚠ UPS Unreachable: Backup UPS (3 consecutive failures)
[2026-02-20 10:37:30] ✓ UPS Recovered: Backup UPS (downtime: 2m 30s)
```

□□ Security

Architecture

- **Single Binary** — Self-contained executable with zero runtime dependencies
- **Minimal Attack Surface** — Compiled Deno binary with only essential functionality
- **Reduced Supply Chain Risk** — Pre-compiled binaries with SHA256 checksums
- **No Telemetry** — No data sent to external servers

SNMP Security

Full SNMPv3 support with authentication and encryption:

Security Level	Description
`noAuthNoPriv`	No authentication, no encryption
`authNoPriv`	MD5/SHA authentication without encryption
`authPriv`	Authentication + DES/AES encryption □

Network Security

- Connects only to UPS devices and optionally Proxmox on local network
- HTTP API disabled by default; token-required when enabled
- No external internet connections

Verifying Downloads

```
curl -sSL https://code.foss.global/serve.zone/nupst/releases/download/v5.0.0/nupst-linux-x64 -o nupst
curl -sSL https://code.foss.global/serve.zone/nupst/releases/download/v5.0.0/SHA256SUMS.txt -o SHA256SUMS.txt
sha256sum -c SHA256SUMS.txt --ignore-missing
```

□□ Supported UPS Models

SNMP-based

Brand	Config Value	Notes
CyberPower	`cyberpower`	Full support including power metrics
APC	`apc`	Smart-UPS, Back-UPS series
Eaton	`eaton`	Eaton/Powerware UPS
TrippLite	`tripplite`	SmartPro and similar
Liebert/Vertiv	`liebert`	GXT, PSI series

Brand	Config Value	Notes
Custom	`custom`	Provide your own OID mappings

Custom OIDs example:

```
{
  "upsModel": "custom",
  "runtimeUnit": "seconds",
  "customOIDs": {
    "POWER_STATUS": "1.3.6.1.4.1.1234.1.1.0",
    "BATTERY_CAPACITY": "1.3.6.1.4.1.1234.1.2.0",
    "BATTERY_RUNTIME": "1.3.6.1.4.1.1234.1.3.0"
  }
}
```

> □□ **Tip:** If your UPS (e.g., HPE, Huawei) reports runtime in seconds instead of minutes, set `"runtimeUnit": "seconds"`. For CyberPower-style TimeTicks (1/100 second), use `"ticks"`. When omitted, NUPST auto-detects based on `upsModel`.

UPSD/NIS-based

Any UPS supported by NUT (Network UPS Tools) — this covers **hundreds of models** from virtually every manufacturer, including USB-connected devices. Check the NUT hardware compatibility list.

□□ Updating

Built-in Update

```
sudo nupst upgrade
```

Re-run Installer

```
curl -sSL https://code.foss.global/serve.zone/nupst/raw/branch/main/install.sh | sudo bash
```

The installer preserves your configuration and restarts the service if it was running.

□□ Uninstallation

```
# Interactive uninstall
sudo nupst uninstall

# Or manual removal
sudo nupst service disable
sudo rm /usr/local/bin/nupst
sudo rm -rf /opt/nupst
sudo rm -rf /etc/nupst
sudo rm /etc/systemd/system/nupst.service
sudo systemctl daemon-reload
```

□□ Troubleshooting

Binary Won't Execute

```
chmod +x /opt/nupst/nupst
uname -m  # x86_64 = x64, aarch64 = arm64
```

Service Won't Start

```
sudo systemctl status nupst
sudo journalctl -u nupst -n 50
nupst config show
nupst ups test --debug
```

Can't Connect to UPS

```
# SNMP
nupst ups test --debug
```

```
ping <ups-ip>
nc -zv <ups-ip> 161

# UPSD/NUT
nc -zv 127.0.0.1 3493
upsc ups@localhost  # if NUT CLI is installed
```

Proxmox VMs Not Shutting Down

```
# Verify API token works
curl -k -H "Authorization: PVEAPIToken=root@pam!nupst=YOUR-SECRET" \
  https://localhost:8006/api2/json/nodes/$(hostname)/qemu

# Check token permissions
pveum user token list root@pam
```

Actions Not Triggering

```
nupst action list
nupst service logs
# Check if monitoring is paused
nupst service status
```

□□ System Changes

File System

Path	Description
`/opt/nupst/nupst`	Pre-compiled binary
`/usr/local/bin/nupst`	Symlink to binary
`/etc/nupst/config.json`	Configuration file
`/etc/nupst/pause`	Pause state file (when paused)
`/etc/systemd/system/nupst.service`	Systemd service unit

Services

- Creates `nupst.service` systemd unit (when enabled)
- Runs with root permissions (required for system shutdown)

Network

- Outbound SNMP to UPS devices (port 161)
- Outbound TCP to NUT servers (port 3493)
- Outbound HTTPS to Proxmox API (port 8006, if configured)
- Optional inbound HTTP server (disabled by default)
- No external internet connections

□□ Migration from v3.x

```
curl -sSL https://code.foss.global/serve.zone/nupst/raw/branch/main/install.sh | sudo bash
```

The installer auto-detects v3.x installations, migrates the configuration, and swaps the binary. Your settings are preserved.

Aspect	v3.x	v4.x+
Runtime	Node.js + npm	Deno (self-contained)
Distribution	Git repo + npm install	Pre-compiled binaries
Runtime Dependencies	node_modules	Zero
Size	~150MB	~80MB
Commands	Flat (`nupst add`)	Subcommands (`nupst ups add`)

□□ Development

Requirements: Deno v1.x or later

```
git clone https://code.foss.global/serve.zone/nupst.git
cd nupst

# Run directly
```

```
deno run --allow-all mod.ts help

# Type check
deno task check

# Lint
deno task lint

# Run tests
deno test --allow-all test/

# Compile for current platform
deno compile --allow-all --output nupst mod.ts

# Compile for all platforms
deno task compile
```

Project Structure

```
nupst/
├── mod.ts                    # Entry point
├── deno.json                 # Deno configuration
├── ts/
│   ├── cli.ts                # CLI command routing
│   ├── nupst.ts              # Main coordinator class
│   ├── daemon.ts             # Background monitoring daemon
│   ├── systemd.ts            # Systemd service management
│   ├── http-server.ts        # Optional HTTP JSON API
│   ├── constants.ts          # Centralized constants
│   ├── snmp/                 # SNMP protocol implementation
│   ├── upsd/                 # UPSD/NIS protocol implementation (NUT)
│   ├── protocol/             # Protocol abstraction layer
│   ├── actions/              # Action system (shutdown, webhook, script, proxmox)
│   ├── migrations/           # Config version migrations
│   ├── helpers/              # Utility functions
│   ├── interfaces/           # Shared TypeScript interfaces
│   └── cli/                  # CLI command handlers
├── scripts/                  # Build and install scripts
```

```
└── test/                    # Test files
```

License and Legal Information

This repository contains open-source code licensed under the MIT License. A copy of the license can be found in the license file.

Please note: The MIT License does not grant permission to use the trade names, trademarks, service marks, or product names of the project, except as required for reasonable and customary use in describing the origin of the work and reproducing the content of the NOTICE file.

Trademarks

This project is owned and maintained by Task Venture Capital GmbH. The names and logos associated with Task Venture Capital GmbH and any related products or services are trademarks of Task Venture Capital GmbH or third parties, and are not included within the scope of the MIT license granted herein.

Use of these trademarks must comply with Task Venture Capital GmbH's Trademark Guidelines or the guidelines of the respective third-party owners, and any usage must be approved in writing. Third-party trademarks used herein are the property of their respective owners and used only in a descriptive manner, e.g. for an implementation of an API or similar.

Company Information

Task Venture Capital GmbH Registered at District Court Bremen HRB 35230 HB, Germany

For any legal inquiries or further information, please contact us via email at hello@task.vc.

By using this repository, you acknowledge that you have read this section, agree to comply with its terms, and understand that the licensing of the code does not imply endorsement by Task Venture Capital GmbH of any derivative works.

Revision #4
Created 2026-03-28 11:14:35 UTC by foss.global Team
Updated 2026-03-31 14:22:48 UTC by foss.global Team