@git.zone/tspm

no fuzz process manager.

- readme.md for @git.zone/tspm
- changelog.md for @git.zone/tspm

readme.md for @git.zone/tspm

TypeScript Process Manager — A robust, no-fuss process manager built for the modern TypeScript and Node.js ecosystem. Production-ready process management without the bloat.

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.

□□ What is TSPM?

TSPM is your production-ready process manager that handles the hard parts of running Node.js applications. Think PM2, but built from scratch for the TypeScript-first ecosystem with better memory management, intelligent logging, a clean daemon architecture, and native ESM support.

□ Key Features

- □□ **Smart Memory Management** — Tracks memory across entire process trees (parent + children), enforces limits, and auto-restarts on OOM
- □□ **Persistent Log Storage** — 10MB in-memory ring buffer per process, auto-persists to disk on stop/restart/crash
- □□ **Intelligent Auto-Restart** — Crashed processes restart with incremental backoff (1s → 10s), auto-stop after 10 consecutive failures
- □□ **File Watching** — Auto-restart on file changes for seamless development workflows
- □□ **Process Tree Tracking** — Monitors parent and all child processes as a unit — no orphans, ever
- □□ **Daemon Architecture** — Persistent background service that survives terminal sessions via Unix socket IPC

- □□ **Beautiful CLI** — Clean, informative output with table views, real-time log streaming, and search
- ⚡ **Zero Config** — Works out of the box; customize only when you need to
- □□ **Systemd Integration** — Run as a system service for production deployments with `tspm enable`
- □□ **Crash Log Reports** — Detailed crash reports with metadata, memory snapshots, and log history

□□ Installation

```
# Global install (recommended)
pnpm add -g @git.zone/tspm

# Or with npm
npm install -g @git.zone/tspm

# Or as a project dependency
pnpm add --save-dev @git.zone/tspm
```

□□ Quick Start

```
# Start the daemon
tspm daemon start

# Add a process
tspm add "node server.js" --name my-server --memory 1GB

# Start it
tspm start name:my-server

# See what's running
tspm list

# View logs
tspm logs name:my-server
```

```
# Stop it
tspm stop name:my-server
```

□□ CLI Reference

Targeting Processes

Most commands accept flexible process targeting:

Format	Example	Description
Numeric ID	`tspm start 1`	Direct ID reference
`id:N`	`tspm start id:1`	Explicit ID prefix
`name:LABEL`	`tspm start name:api`	Target by name

Use `tspm search <query>` to find processes by name or ID substring.

Process Management

`tspm add <command> [options]`

Register a new process configuration (without starting it).

Option	Description	Default
`--name <name>`	Process name	command string
`--memory <size>`	Memory limit (e.g. `512MB`, `2GB`)	`512MB`
`--cwd <path>`	Working directory	current directory
`--watch`	Enable file watching	`false`
`--watch-paths <paths>`	Comma-separated watch paths	—
`--autorestart`	Auto-restart on crash	`true`
`-i, --interactive`	Enter interactive edit after adding	—

```
# Simple Node.js app
tspm add "node server.js" --name api-server

# TypeScript with 2GB memory limit
```

```
tspm add "tsx src/index.ts" --name production-api --memory 2GB

# Dev mode with file watching
tspm add "tsx watch src/index.ts" --name dev-server --watch --watch-paths "src,config"

# One-shot worker (no auto-restart)
tspm add "node worker.js" --name batch-job --autorestart false

# Add + interactive edit
tspm add "node server.js" --name api -i
```

`tspm start <target>`

Start a registered process.

```
tspm start name:my-server
tspm start id:1
tspm start 1          # bare numeric id also works
```

`tspm stop <target>`

Gracefully stop a process (SIGTERM → 5s grace → SIGKILL).

```
tspm stop name:my-server
```

`tspm restart <target>`

Stop and restart a process, preserving its configuration.

```
tspm restart name:my-server
```

`tspm delete <target>`

Stop, remove from management, and delete persisted logs.

```
tspm delete name:old-server
```

`tspm edit <target>`

Interactively modify a process configuration (name, command, memory, etc.).

```
tspm edit name:my-server
```

`tspm search <query>`

Search processes by name or ID substring.

```
tspm search api
# Matches for "api":
#   id:3    name:api-server
```

Monitoring

`tspm list`

Display all managed processes in a table.

```
┌─────┬──────────────┬──────────┬───────┬────────────┬────────────┐
│ ID  │ Name         │ Status   │ PID   │ Memory     │ Restarts   │
├─────┼──────────────┼──────────┼───────┼────────────┼────────────┤
│ 1   │ my-app       │ online   │ 45123 │ 245.3 MB   │ 0          │
│ 2   │ worker       │ online   │ 45456 │ 128.7 MB   │ 2          │
│ 3   │ api-server   │ stopped  │ -     │ 0 B        │ 5          │
└─────┴──────────────┴──────────┴───────┴────────────┴────────────┘
```

`tspm describe <target>`

Detailed information about a specific process.

```
tspm describe name:my-server


# Process Details: my-server
# ──────────────────────────────────────────
# Status:       online
# PID:          45123
# Memory:       245.3 MB
# Uptime:       3600s
# Restarts:     0
#
# Configuration:
# ──────────────────────────────────────────
# Command:      node server.js
# Directory:    /home/user/project
```

```
# Memory Limit: 2 GB
# Auto-restart: true
# Watch:        disabled
```

`tspm logs <target> [options]`

View and stream process logs.

Option	Description	Default
`--lines <n>`	Number of lines	`50`
`--since <dur>`	Time filter (`10m`, `2h`, `1d`)	—
`--stderr-only`	Only stderr	—
`--stdout-only`	Only stdout	—
`--ndjson`	Output as newline-delimited JSON	—
`--follow`	Real-time streaming (like `tail -f`)	—

```
# View last 50 lines
tspm logs name:my-server

# Last 100 lines of stderr only
tspm logs name:my-server --lines 100 --stderr-only

# Stream logs in real time
tspm logs name:my-server --follow

# NDJSON output since 10 minutes ago
tspm logs name:my-server --since 10m --ndjson
```

Batch Operations

```
tspm start-all      # Start all saved processes
tspm stop-all       # Stop all running processes
tspm restart-all    # Restart all running processes
tspm reset          # ⚠ Stop all + clear all configs (prompts for confirmation)
```

Daemon Management

The daemon is a persistent background service that manages all processes. It starts automatically when needed.

```
tspm daemon start     # Start the daemon
tspm daemon stop      # Stop daemon + all managed processes
tspm daemon restart   # Restart daemon (preserves processes)
tspm daemon status    # Check daemon health + stats
```

System Service (systemd)

```
sudo tspm enable      # Install + enable as systemd service (auto-start on boot)
sudo tspm disable     # Remove systemd service
```

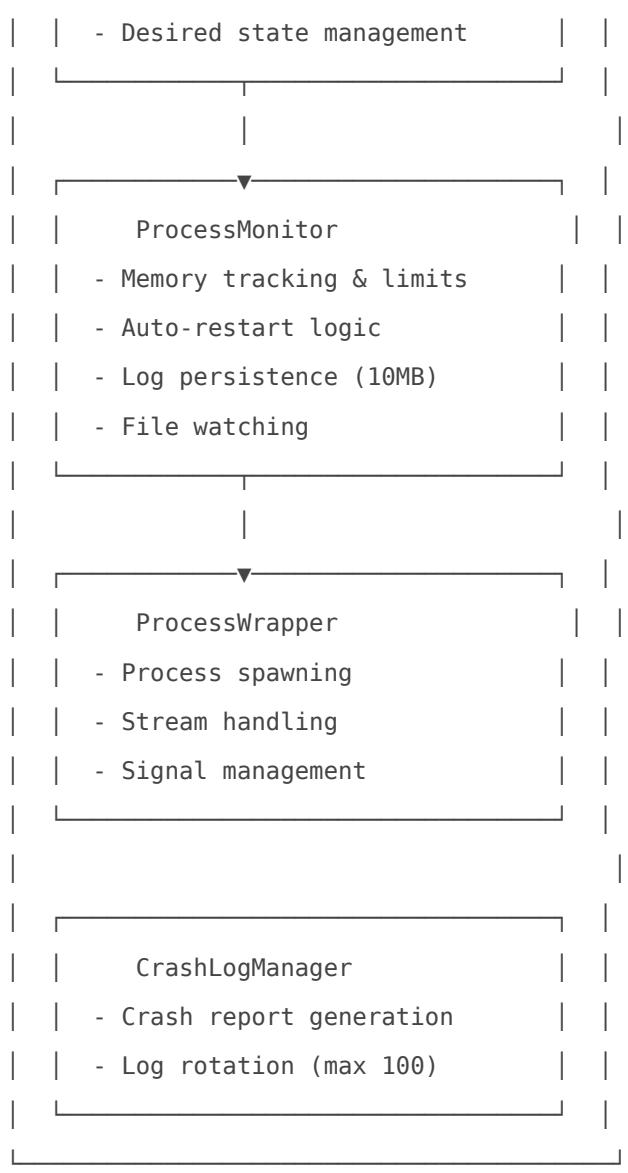
Version Check

```
tspm -v
# tspm CLI: 5.x.y
# Daemon: running v5.x.z (pid 1234)
# Version mismatch detected → optionally refresh the systemd service
```

□□ Architecture

TSPM uses a clean three-tier architecture:

```
┌──────────────────────────────────────────┐
│            CLI Interface                 │
│         (tspm commands)                  │
└──────────────────┬───────────────────────┘
                   │ Unix Socket IPC
┌──────────────────▼───────────────────────┐
│          TSPM Daemon                     │
│     (Background Service)                 │
│  ┌──────────────────────────────────┐    │
│  │     ProcessManager               │    │
│  │  - Configuration persistence     │    │
│  │  - Process lifecycle             │    │
```



Key Components

Component	Role
CLI	Lightweight client that sends commands to daemon via IPC
Daemon	Persistent background service managing all processes
ProcessManager	High-level orchestration, config persistence, state management
ProcessMonitor	Memory limits, auto-restart with backoff, log persistence, file watching
ProcessWrapper	Low-level process lifecycle, stream handling, signal management
CrashLogManager	Detailed crash reports with metadata and log history

□□ Programmatic API

TSPM exposes a typed IPC client for programmatic use:

```
import { TspmIpcClient } from '@git.zone/tspm/client';

const client = new TspmIpcClient();
await client.connect();

// Add a process configuration
const { id } = await client.request('add', {
  config: {
    command: 'node worker.js',
    name: 'background-worker',
    projectDir: process.cwd(),
    memoryLimitBytes: 512 * 1024 * 1024,
    autorestart: true,
  },
});

// Start it
await client.request('startById', { id });

// Get process info
const { processInfo } = await client.request('describe', { id });
console.log(`Status: ${processInfo.status}, Memory: ${processInfo.memory} bytes`);

// Get logs
const { logs } = await client.request('getLogs', { id, limit: 100 });
logs.forEach(log => console.log(`[${log.timestamp}] [${log.type}] ${log.message}`));

// Stop and remove
await client.request('stop', { id });
await client.request('delete', { id });
await client.disconnect();
```

Module Exports

Export Path	Purpose
`@git.zone/tspm`	Main entry point (re-exports client + daemon)
`@git.zone/tspm/client`	IPC client (`TspmIpcClient`, `TspmServiceManager`)
`@git.zone/tspm/daemon`	Daemon entry point (`startDaemon`)
`@git.zone/tspm/protocol`	IPC type definitions

□□ Advanced Features

Restart Backoff & Failure Handling

TSPM handles crashed processes with intelligent backoff:

- **Incremental delay**: Grows linearly from 1s up to 10s for consecutive restarts
- **Failure threshold**: After 10 consecutive failures, the process is marked `errored` and auto-restart stops
- **Auto-reset**: The retry counter resets if no failure occurs for 1 hour
- **Manual recovery**: `tspm restart id:1` always works, even on errored processes

Memory Management

Full process tree memory tracking:

- Discovers all child processes via `ps-tree`
- Calculates combined memory usage across the entire tree
- Gracefully restarts when limit is exceeded (SIGTERM → SIGKILL)
- Prevents memory leaks from taking down production systems

Log Persistence

Smart in-memory log management:

- 10MB ring buffer per process with automatic trimming
- Flushes to disk on stop, restart, or crash
- Reloads persisted logs when process restarts
- Crash logs stored separately with full metadata (exit code, signal, memory, timestamps)

Graceful Shutdown

Multi-stage shutdown for reliability:

1. Send **SIGTERM** for graceful shutdown
2. Wait **5 seconds** for process cleanup
3. Send **SIGKILL** if still alive
4. Clean up **all child processes** in the tree

File Watching

Development-friendly auto-restart:

- Watch specific directories or files
- `node_modules` ignored by default
- Debounced restart on file changes
- Configurable via `--watch-paths`

□□ Debugging

```
# Check daemon status
tspm daemon status

# View process logs
tspm logs name:my-app --lines 200

# Check daemon stderr
tail -f /tmp/daemon-stderr.log

# Force daemon restart
tspm daemon restart
```

Common issues:

Problem	Solution
"Daemon not running"	`tspm daemon start` or `sudo tspm enable`
"Permission denied"	Check socket permissions in `~/.tspm/`
Process won't start	Check logs with `tspm logs <target>`
Memory limit exceeded	Increase with `tspm edit <target>`

□□ Why TSPM?

Feature	TSPM	PM2
TypeScript Native	□ Built in TypeScript	□ JavaScript
Memory Tracking	□ Full process tree	⚠ Main process only
Log Management	□ Smart 10MB buffer	⚠ Can grow unbounded
Architecture	□ Clean 3-tier daemon	□ Monolithic
Dependencies	□ Minimal	□ Heavy
ESM Support	□ Native	⚠ Partial
Crash Reports	□ Detailed with metadata	□ Basic

Perfect For

- □□ **Production Node.js apps** — Reliable process management with memory guards
- □□ **Microservices** — Manage multiple services from a single tool
- □□□ **Development** — File watching and instant auto-restart
- □□ **Workers & Jobs** — Queue workers, cron jobs, background tasks
- □□ **Resource-constrained environments** — Memory limits prevent OOM kills

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.

changelog.md for @git.zone/tspm

2026-03-24 - 5.10.4 - fix(crash-logging)

migrate filesystem persistence to smartfs and stabilize crash log tests

- replace smartfile usage with smartfs in crash log and log persistence modules
- update crash log tests to use tap assertions and current CLI command output
- move project config from npmextra.json to .smartconfig.json and refresh build dependencies

2026-03-24 - 5.10.3 - fix(config)

replace npmextra with smartconfig for daemon key-value storage and release settings

- swap the configuration storage dependency from @push.rocks/npmextra to @push.rocks/smartconfig
- update daemon config accessors to use the smartconfig KeyValueStore implementation
- add @git.zone/cli release registry and access configuration to npmextra.json

2025-09-03 - 5.10.2 - fix(processmonitor)

Bump smartdaemon and stop aggressive pidusage cache clearing in ProcessMonitor

- Update dependency @push.rocks/smartdaemon from ^2.0.9 to ^2.1.0 in package.json.

- Remove per-PID pidusage.clear calls in ts/daemon/processmonitor.ts (getProcessGroupStats) to avoid potential errors or unexpected behavior from manually clearing pidusage cache.

2025-09-03 - 5.10.1 - fix(processmonitor)

Skip null pidusage entries when aggregating process-group memory/CPU to avoid errors

- Add defensive check for null/undefined entries returned by pidusage before accessing memory/cpu fields
- Log a debug message when an individual process stat is null (process may have exited)
- Improve robustness of ProcessMonitor.getProcessGroupStats to prevent runtime exceptions during aggregation

2025-09-01 - 5.10.0 - feat(daemon)

Add crash log manager with rotation and integrate crash logging; improve IPC & process listener cleanup

- Introduce CrashLogManager to create formatted crash reports, persist them to disk and rotate old logs (max 100)
- Persist recent process logs, include metadata (exit code, signal, restart attempts, memory) and human-readable sizes in crash reports
- Integrate crash logging into ProcessMonitor: save crash logs on non-zero exits and errors, and persist/rotate logs
- Improve ProcessMonitor and ProcessWrapper by tracking and removing event listeners to avoid memory leaks
- Clear pidusage cache more aggressively to prevent stale entries
- Enhance TspmIpcClient to store/remove lifecycle event handlers on disconnect to avoid dangling listeners
- Add tests and utilities: test/test.crashlog.direct.ts, test/test.crashlog.manual.ts and test/test.crashlog.ts to validate crash log creation and rotation

2025-08-31 - 5.9.0 - feat(cli)

Add interactive edit flow to CLI and improve UX

- Add -i / --interactive flag to tspm add to open an interactive editor immediately after adding a process
- Implement interactiveEditProcess helper (smartinteract-based) to provide interactive editing for process configs
- Enable tspm edit to launch the interactive editor (replaces prior placeholder flow)
- Improve user-facing message when no processes are configured in tspm list
- Lower verbosity for missing saved configs on daemon startup (changed logger.info → logger.debug)

2025-08-31 - 5.8.0 - feat(core)

Add core TypeScript TSPM implementation: CLI, daemon, client, process management and tests

- Add CLI entrypoint and command set (start/stop/add/list/logs/daemon/service/stats/reset and batch ops)
- Add daemon implementation with ProcessManager, ProcessMonitor, ProcessWrapper, LogPersistence and config storage
- Add IPC client (tspmIpcClient) and TspmServiceManager for systemd integration using smartipc/smartdaemon
- Introduce shared protocol types, process ID helpers and standardized error codes for stable IPC
- Include tests and test assets for daemon, integration and IPC client scenarios
- Add README and package metadata (package.json, npmextra.json, commitinfo)

2025-08-31 - 5.7.0 - feat(cli)

Add 'stats' CLI command and daemon stats aggregation; fix process manager & wrapper state handling

- Add new 'stats' CLI command to show daemon + process statistics (memory, CPU, uptime, logs in memory, paths, configs) and include it in the default help output
- Implement daemon-side aggregation for logs-in-memory, per-process log counts/bytes, and expose tspmDir/socket/pidFile and config counts in daemon:status
- Enhance startById handler to detect already-running monitors and return current status/pid instead of attempting to restart
- Improve ProcessManager start/restart/stop behavior: if an existing monitor exists but is not running, restart it; ensure PID and status are updated consistently (clear PID on stop)
- Fix ProcessWrapper lifecycle handling: clear internal process reference on exit, improve isRunning() and getPid() semantics to reflect actual runtime state

- Update IPC types to include optional metadata fields (paths, configs, logsInMemory) in DaemonStatusResponse

2025-08-31 - 5.6.2 - fix(processmanager)

Improve process lifecycle handling and cleanup in daemon, monitors and wrappers

- StartAll: when a monitor exists but is not running, restart it instead of skipping — ensures saved processes are reliably brought online.
- ProcessMonitor.stop: cancel any pending restart timers to prevent stray restarts after explicit stop.
- ProcessWrapper: add killProcessTree helper and use it for graceful (SIGTERM) and force (SIGKILL) shutdowns to reliably signal child processes.
- Daemon stopAll: yield briefly after stopping processes and inspect monitors (not only processInfo) to accurately report stopped vs failed processes.

2025-08-31 - 5.6.1 - fix(daemon)

Ensure robust process shutdown and improve logs/subscriber diagnostics

- Make ProcessWrapper.stop asynchronous and awaitable to avoid race conditions when stopping processes
- Signal entire process groups on POSIX (kill by negative PID) and fall back to per-PID signalling; escalate to SIGKILL after a timeout
- Await processWrapper.stop() from ProcessMonitor when enforcing memory limits or handling exits/errors to ensure child processes are cleaned up
- Add logs:subscribers IPC endpoint and corresponding types to inspect current subscribers for a process log topic
- Add optional CLI debug output in logs command (enabled via TSPM_DEBUG=true) to print subscriber counts and details
- Support passing request.lines to getLogs handler in daemon to limit returned log entries

2025-08-30 - 5.6.0 - feat(processmonitor)

Add CPU monitoring and display CPU in process list

- CLI: show a CPU column in the `tspm list` output (adds formatting and placeholder name display)
- Daemon: ProcessMonitor now collects CPU usage for the process group in addition to memory
- Daemon: ProcessMonitor exposes getLastCpuUsage() and ProcessManager syncs CPU values into IProcessInfo
- Non-breaking: UI and internal stats enriched to surface CPU metrics for processes

2025-08-30 - 5.5.0 - feat(logs)

Improve logs streaming and backlog delivery; add CLI filters and ndjson output

- CLI: add new logs options: --since, --stderr-only, --stdout-only and --ndjson; enhance streaming output and gap detection
- CLI: fetch backlog conditionally (honoring --since) and print filtered results before live streaming
- Client: add TspmIpcClient.requestLogsBacklogStream, onStream and onBacklogTopic helpers to receive backlog chunks and streams
- Daemon: add logs:subscribe IPC handler to stream backlog entries to requesting client in small batches
- Protocol: extend IPC types with LogsSubscribeRequest/Response and register 'logs:subscribe' method
- Dependency: bump @push.rocks/smartipc to ^2.3.0 to support the streaming/IPC changes

2025-08-30 - 5.4.2 - fix(cli/process/logs)

Reset log sequence on process restart to avoid false log gap warnings

- Track process runId when streaming logs and initialize lastRunId from fetched logs
- When a new runId is detected, reset lastSeq so that subsequent streamed logs are accepted (prevents spurious gap warnings)
- Emit an informational message when a restart/runId change is detected to aid debugging of log streams

2025-08-30 - 5.4.1 - fix(processmonitor)

Bump tsbuild devDependency and relax ps-tree callback typing in ProcessMonitor

- Update devDependency @git.zone/tsbuild from ^2.6.7 to ^2.6.8
- Change psTree callback types in ts/daemon/processmonitor.ts to accept any error and ReadonlyArray for children to improve type compatibility

2025-08-30 - 5.4.0 - feat(daemon)

Add CLI systemd service refresh on version mismatch and fix daemon memory leak; update dependencies

- CLI: when client and daemon versions differ, prompt to refresh the systemd service and optionally disable/enable the service automatically
- Daemon: clear pidusage state for PIDs on process exit/stop to prevent memory leaks in long-running monitors
- Client: expose smartdaemon in client plugin exports and fix import path for tspm.servicemanager
- Package: tighten dependency ranges (set specific versions) and add @types for pidusage and ps-tree
- Misc: ensure IPC disconnects and PID/socket handling improvements were integrated alongside the above changes

2025-08-30 - 5.3.2 - fix(daemon)

Improve daemon log delivery and process monitor memory accounting; gate debug output and update tests to numeric ProcessId

- Deliver process logs only to subscribed clients instead of broadcasting to all connections (reduce unnecessary IPC traffic and noise)
- Implement incremental log memory accounting in ProcessMonitor using an estimateLogSize helper and WeakMap to avoid repeated JSON.stringify and reduce CPU/memory overhead
- Seed the incremental size map when loading persisted logs so memory accounting is accurate after restart

- Trim logs incrementally by subtracting estimated sizes of removed entries (avoids O(n) recalculation)
- Gate verbose console/debug output behind TSPM_DEBUG to prevent spamming in normal runs (applies to ProcessWrapper and ProcessMonitor)
- Improve process wrapper stdout/stderr debug logging to be conditional on debug mode
- Update tests to use numeric ProcessId via toProcessId(...) for consistency with typed IDs

2025-08-30 - 5.3.1 - fix(client(tspmIpcClient))

Use bare topic names for IPC client subscribe/unsubscribe to fix log subscription issues

- Updated ts/client/tspm.ipcclient.ts to call ipcClient.subscribe/unsubscribe with the bare topic (e.g. 'logs.') instead of prefixed 'topic:<...>'.
- Added comments clarifying that the IpcClient registers the 'topic:' prefix internally.
- Fixes incorrect topic registration that could prevent log streaming handlers from receiving messages.

2025-08-30 - 5.3.0 - feat(cli/daemon/processmonitor)

Add flexible target resolution and search command; improve restart/backoff and error handling

- Add new cli command `search` to find processes by id or name fragment.
- Allow flexible process targets in CLI commands (accepts numeric id, id:, or name:) for start/stop/restart/delete/describe/logs/edit commands.
- Introduce a new daemon IPC method `resolveTarget` to normalize user-provided targets to ProcessId (supports id:, name:, or bare numeric id).
- Keep `remove` as a CLI alias but daemon exposes `delete` only; CLI resolves targets and always calls daemon `delete`.
- Implement scheduled restart/backoff in ProcessMonitor with incremental debounce, max retries, and a 1-hour reset window.
- Emit a `failed` event from ProcessMonitor when max restart attempts are exceeded; ProcessManager listens and marks processes as `errored` and clears pid.
- Ensure desired state is set to `stopped` before deleting a process to avoid race conditions.
- Improve cli output messages to include resolved names alongside numeric ids where available.

2025-08-30 - 5.2.0 - feat(cli)

Preserve CLI environment when adding processes, simplify edit flow, and refresh README docs

- CLI: When adding a process, capture and persist essential environment variables from the CLI (PATH, HOME, USER, SHELL, LANG, LC_ALL, NODE_ENV, NODE_PATH, npm_config_prefix and any TSPM_* variables). Undefined values are removed before storing.
- CLI: Interactive edit flow temporarily disabled. The edit command now displays the current configuration and updates stored environment variables to match the current CLI environment.
- Docs: Major README refresh — reorganized sections, clarified add vs start semantics, expanded examples, added daemon/service usage and programmatic API examples, and improved command reference and output examples.

2025-08-30 - 5.1.0 - feat(cli)

Add interactive edit command and update support for process configurations

- Add 'tspm edit' interactive CLI command to modify saved process configurations (prompts for name, command, args, cwd, memory, autorestart, watch, watch paths) with an option to replace stored PATH.
- Implement ProcessManager.update(id, updates) to merge updates, persist them, and return the updated configuration.
- Add 'update' IPC method and daemon handler to allow remote/configurations updates via IPC.
- Persist the current CLI PATH when adding a process so managed processes inherit the same PATH environment.
- Merge provided env with the runtime process.env when spawning processes to avoid fully overriding the runtime environment.

2025-08-30 - 5.0.0 - BREAKING CHANGE(daemon)

Introduce persistent log storage, numeric ProcessId type, and improved process monitoring / IPC handling

- Add LogPersistence: persistent on-disk storage for process logs (save/load/delete/cleanup).
- Persist logs on process exit/error/stop and trim in-memory buffers to avoid excessive memory usage.
- Introduce a branded numeric ProcessId type and toProcessId helpers; migrate IPC types and internal maps from string ids to ProcessId.
- ProcessManager refactor: typed maps for processes/configs/info/logs, async start/stop/restart flows, improved PID/uptime/restart tracking, and desired state persistence handling.
- ProcessMonitor refactor: async lifecycle (start/stop), load persisted logs on startup, flush logs to disk on exit/error/stop, log memory capping, and improved event emissions.
- ProcessWrapper improvements: buffer stdout/stderr remainders, flush partial lines on stream end, clearer debug logging.
- IPC client/server changes: handlers now normalize ids with toProcessId, subscribe/unsubscribe accept numeric/string ids, getLogs/start/stop/restart/delete use typed ids.
- CLI tweaks: format process id output safely with String() to avoid formatting issues.
- Add dependency and plugin export for @push.rocks/smartfile and update package.json accordingly.

2025-08-29 - 4.4.2 - fix(daemon)

Fix daemon IPC id handling, reload configs on demand and correct CLI daemon start path

- Normalize process IDs in daemon IPC handlers (trim strings) to avoid lookup mismatches
- Attempt to reload saved process configurations when a startById request cannot find a config (handles races/stale state)
- Use normalized IDs in responses and messages for stop/restart/delete/remove/describe handlers
- Fix CLI daemon start path to point at dist_ts/daemon/tspm.daemon.js when launching the background daemon
- Ensure the IPC client disconnects after showing CLI version/status to avoid leaked connections

2025-08-29 - 4.4.1 - fix(cli)

Use server-side start-by-id flow for starting processes

- CLI: 'tspm start ' now calls a new 'startById' IPC method instead of fetching the full config via 'describe' and submitting it back to 'start'.

- Daemon: Added server-side handler for 'startById' which resolves the stored process config and starts the process on the daemon.
- Protocol: Added StartByIdRequest/StartByIdResponse types and registered 'startById' in the IPC method map.

2025-08-29 - 4.4.0 - feat(daemon)

Persist desired process states and add daemon restart command

- Persist desired process states: ProcessManager now stores desiredStates to user storage (desiredStates key) and reloads them on startup.
- Start/stop operations update desired state: IPC handlers in the daemon now set desired state when processes are started, stopped, restarted or when batch start/stop is invoked.
- Resume desired state on daemon start: Daemon loads desired states and calls startDesired() to bring processes to their desired 'online' state after startup.
- Remove desired state on deletion/reset: Deleting a process or resetting clears its desired state; reset clears all desired states as well.
- CLI: Added 'tspm daemon restart' — stops the daemon (gracefully) and restarts it in the foreground for the current session, with checks and informative output.

2025-08-29 - 4.3.1 - fix(daemon)

Fix daemon describe handler to return correct process info and config; bump @push.rocks/smartipc to ^2.2.2

- Corrected the 'describe' IPC handler in the daemon to use ProcessManager.describe(...) result and return { processInfo, config } — this fixes a mismatch between the handler and the ProcessManager.describe() return shape.
- Bumped dependency @push.rocks/smartipc to ^2.2.2 in package.json.

2025-08-29 - 4.3.0 - feat(cli)

Correct CLI plugin imports and add reset command/IPC to stop processes and clear persisted configs

- Fixed relative plugin imports in many CLI command modules to use the local CLI plugin wrapper (reduces startup surface and fixes import paths).
- Added a lightweight ts/cli/plugins.ts that exposes only the minimal plugin set used by the CLI.

- Implemented ProcessManager.reset(): stops running processes, collects per-id stop errors, clears in-memory maps and removes persisted configurations (with fallback to write an empty list on delete failure).
- Daemon now exposes a 'reset' IPC handler that delegates to ProcessManager.reset() so CLI can perform a single RPC to reset TSPM state.
- Updated shared IPC protocol types to include ResetRequest and ResetResponse.
- Refactored the CLI reset command to call the new 'reset' RPC (replaces previous stopAll + per-config removal logic).

2025-08-29 - 4.2.0 - feat(cli)

Add 'reset' CLI command to stop all processes and clear saved configurations; integrate interactive confirmation and client plugin updates

- Add new CLI command 'reset' (ts/cli/commands/reset.ts) which stops all processes and removes saved process configurations after an interactive confirmation.
- Use @push.rocks/smartinteract for a confirmation prompt before destructive action.
- Register the new reset command in the CLI bootstrap (ts/cli/index.ts).
- Expose smartinteract from ts/plugins.ts and add @push.rocks/smartinteract to package.json dependencies.
- Introduce a lightweight client plugin shim (ts/client/plugins.ts) and switch tspm.ipcclient to import client plugins from ./plugins.js.

2025-08-29 - 4.1.1 - fix(daemon)

Bump @push.rocks/smartdaemon to ^2.0.9

- Update @push.rocks/smartdaemon from ^2.0.8 to ^2.0.9 (dependency version bump)

2025-08-29 - 4.1.0 - feat(cli)

Add support for restarting all processes from CLI; improve usage message and reporting

- CLI 'restart' command now accepts 'all' to restart all processes via the daemon (tspm restart all).
- Improved usage/help output when no process id is provided.
- CLI now prints summaries of restarted process IDs and failed restarts and sets a non-zero exit code when any restarts failed.

2025-08-29 - 4.0.0 - BREAKING CHANGE(cli)

Add persistent process registration (tspm add), alias remove, and change start to use saved process IDs (breaking CLI behavior)

- Add a new CLI command `tspm add` that registers a process configuration without starting it; daemon assigns a sequential numeric ID and returns the stored config.
- Change `tspm start` to accept a process ID and start the saved configuration instead of accepting ad-hoc commands/files. This is a breaking change to the CLI contract.
- Add `remove` as an alias for the existing `delete` command; both CLI and daemon now support `remove` which stops and deletes the stored process.
- Daemon and IPC protocol updated to support `add` and `remove` methods; shared IPC types extended accordingly.
- ProcessManager: implemented add() and getNextSequentialId() to persist configs and produce numeric IDs.
- CLI registration updated (registerIpcCommand) to accept multiple command names, enabling aliases for commands.

2025-08-29 - 3.1.3 - fix(client)

Improve IPC client robustness and daemon debug logging; update tests and package metadata

- IPC client: generate unique clientId for each CLI session, increase register timeout, mark client disconnected on lifecycle events and socket errors, and surface a clearer connection error message
- Daemon: add debug hooks to log client connect/disconnect and server errors to help troubleshoot IPC issues
- Tests: update imports to new client/daemon locations, add helpers to start the daemon and retry connections, relax timing assertions, and improve test reliability
- Package: add exports map and typings entry, update test script to run with verbose logging and longer timeout, and bump @push.rocks/smartipc to ^2.2.1

2025-08-28 - 3.1.2 - fix(daemon)

Reorganize project into daemon/client/shared layout, update imports and protocol, rename Tspm → ProcessManager, and bump smartipc to ^2.1.3

- Reorganized source tree: moved files into ts/daemon, ts/client and ts/shared with updated index/barrel exports.
- Renamed core class Tspm → ProcessManager and updated all references.
- Consolidated IPC types under ts/shared/protocol/ipc.types.ts and added protocol.version + standardized error codes.
- Updated CLI to use the new client API (tspmIpcClient) and adjusted command registration/registration helpers.
- Bumped dependency @push.rocks/smartipc from ^2.1.2 to ^2.1.3 to address daemon connectivity; updated daemon heartbeat behavior (heartbeatThrowOnTimeout=false).
- Updated readme.plan.md to reflect completed refactor tasks and testing status.
- Minor fixes and stabilization across daemon, process manager/monitor/wrapper, and client service manager implementations.

2025-08-28 - 3.1.1 - fix(cli)

Fix internal imports, centralize IPC types and improve daemon entry/start behavior

- Corrected import paths in CLI commands and utilities to use client/tspm.ipcclient and shared/common/utils.errorhandler
- Centralized process/IPC type definitions into ts/shared/protocol/ipc.types.ts and updated references across daemon and client code
- Refactored ts/daemon/index.ts to export startDaemon and only auto-start the daemon when the module is executed directly
- Adjusted ts/index.ts exports to expose client API, shared protocol types, and daemon start entrypoint

2025-08-28 - 3.1.0 - feat(daemon)

Reorganize and refactor core into client/daemon/shared modules; add IPC protocol and tests

- Reorganized core code: split daemon and client logic into ts/daemon and ts/client directories
- Moved process management into ProcessManager, ProcessMonitor and ProcessWrapper under ts/daemon
- Added a dedicated IPC client and service manager under ts/client (tspm.ipcclient, tspm.servicemanager)
- Introduced shared protocol and error handling: ts/shared/protocol/ipc.types.ts, protocol.version.ts and ts/shared/common/utils.errorhandler.ts
- Updated CLI to import Logger from shared/common utils and updated related helpers
- Added daemon entrypoint at ts/daemon/index.ts and reorganized daemon startup/shutdown/heartbeat handling

- Added test assets (test/testassets/simple-test.ts, simple-script2.ts) and expanded test files under test/
- Removed legacy top-level class files (classes.*) in favor of the new structured layout

2025-08-28 - 3.0.2 - fix(daemon)

Ensure TSPM runtime dir exists and improve daemon startup/debug output

- Create ~/.tspm directory before starting the daemon to avoid missing-directory errors
- Start daemon child process with stdio inherited when TSPM_DEBUG=true to surface startup errors during debugging
- Add warning and troubleshooting guidance when daemon process starts but does not respond (suggest checking socket file and using TSPM_DEBUG)
- Bump package version to 3.0.1

2025-08-28 - 3.0.0 - BREAKING CHANGE(daemon)

Refactor daemon and service management: remove IPC auto-spawn, add TspmServiceManager, tighten IPC/client/CLI behavior and tests

- Remove automatic daemon spawn from the IPC client — clients now error with guidance and require the daemon to be started manually or enabled as a system service
- Add TspmServiceManager to manage the daemon as a systemd service (enable/disable/reload/status)
- Update IPC server/client to use SmartIpc.createServer/createClient with heartbeat defaults and explicit onMessage handlers
- Daemon publishes per-process logs to topics (logs.<processId>) and re-emits ProcessMonitor logs for pub/sub
- CLI updated: add enable/disable service commands, adjust daemon start/stop/status workflows and improve user hints when daemon is not running
- Add/adjust integration and unit tests to cover daemon lifecycle, IPC client behavior, log streaming, heartbeat and resource reporting
- Documentation expanded (README, readme.plan.md, changelog) to reflect the refactor and migration notes
- Various code cleanups, formatting fixes and defensive checks across modules

2025-08-28 - 2.0.0 - BREAKING CHANGE(daemon)

Refactor daemon lifecycle and service management: remove IPC auto-spawn, add TspmServiceManager and CLI enable/disable

- Do not auto-spawn the daemon from the IPC client anymore — attempts to connect will now error with instructions to start the daemon manually or enable the system service (breaking change).
- Add TspmServiceManager to manage the daemon as a systemd service via smartdaemon (enable/disable/reload/status helpers).
- CLI: add 'enable' and 'disable' commands to install/uninstall the daemon as a system service and add 'daemon start-service' entrypoint used by systemd.
- CLI: improve error handling and user hints when the daemon is not running (suggests `tspm daemon start` or `tspm enable`).
- IPC client: removed startDaemon() and related auto-reconnect/start logic; request() no longer auto-reconnects or implicitly start the daemon.
- Export TspmServiceManager from the package index so service management is part of the public API.
- Updated development plan/readme (readme.plan.md) to reflect the refactor toward proper SmartDaemon integration and migration notes.

2025-08-26 - 1.8.0 - feat(daemon)

Add real-time log streaming and pub/sub: daemon publishes per-process logs, IPC client subscribe/unsubscribe, CLI --follow streaming, and sequencing for logs

- Upgrade @push.rocks/smartipc dependency to ^2.1.2
- Daemon: initialize SmartIpc server with heartbeat and publish process logs to topic `logs.<processId>`; write PID file and start heartbeat monitoring
- Tspm: re-emit monitor log events as 'process:log' so daemon can broadcast logs
- ProcessWrapper: include seq and runId on IProcessLog entries and maintain nextSeq/runId (adds sequencing to logs); default log buffer size applied
- TspmIpcClient: improved connect options (retries, timeouts, heartbeat handling), add subscribe/unsubscribe for real-time logs, and use SmartIpc.waitForServer when starting daemon
- CLI: add --follow flag to `logs` command to stream live logs, detect sequence gaps/duplicates, and handle graceful cleanup on Ctrl+C
- ProcessMonitor: now extends EventEmitter and re-emits process logs for upstream consumption

- Standardized heartbeat and IPC timing defaults (heartbeatInterval: 5000ms, heartbeatTimeout: 20000ms, heartbeatInitialGracePeriodMs: 10000ms)

2025-08-25 - 1.7.0 - feat(readme)

Add comprehensive README with detailed usage, command reference, daemon management, architecture and development instructions

- Expanded README from a short placeholder to a full documentation covering: Quick Start, Installation, Command Reference, Daemon Management, Monitoring & Information, Batch Operations, Architecture, Programmatic Usage, Advanced Features, Development, Debugging, Performance, and Legal information
- Included usage examples and CLI command reference for start/stop/restart/delete/list/describe/logs and batch/daemon commands
- Added human-friendly memory formatting and examples, process and daemon status outputs, and programmatic TypeScript usage snippet
- Improved onboarding instructions: cloning, installing, testing, building, and running the project

2025-08-25 - 1.6.1 - fix(daemon)

Fix smartipc integration and add daemon/ipc integration tests

- Replace direct smartipc server/client construction with SmartIpc.createServer/createClient and set heartbeat: false
- Switch IPC handler registration to use onMessage and add explicit Request/Response typing for handlers
- Update IPC client to use SmartIpc.createClient and improve daemon start/connect logic
- Add comprehensive tests: unit tests for TspmDaemon and TspmIpcClient and full integration tests for daemon lifecycle, process management, error handling, heartbeat and resource reporting

2025-08-25 - 1.6.0 - feat(daemon)

Add central TSPM daemon and IPC client; refactor CLI to use daemon and improve monitoring/error handling

- Add central daemon implementation (ts/classes.daemon.ts) to manage all processes via a single background service and Unix socket.

- Introduce IPC client and typed IPC contracts (ts/classes.ipcclient.ts, ts/ipc.types.ts) so CLI communicates with the daemon.
- Refactor CLI to use the daemon for commands (ts/cli.ts): start/stop/restart/delete/list/describe/logs/start-all/stop-all/restart-all and new daemon start/stop/status commands.
- Enhance process monitoring and wrapping: ProcessMonitor and ProcessWrapper improvements (ts/classes.processmonitor.ts, ts/classes.processwrapper.ts) with better logging, memory checks, and restart behavior.
- Improve centralized error handling and Logger behavior (ts/utils.errorhandler.ts).
- Persist and load process configurations via TspmConfig and config storage changes (ts/classes.config.ts, ts/classes.tspm.ts).
- Bump dependency and devDependency versions and add packageManager entry in package.json.
- Add ts/daemon entrypoint and export daemon/ipc types from ts/index.ts; add paths for tspm runtime dir (ts/paths.ts).
- Update tests and test tooling imports (test/test.ts) and adjust commitinfo and readme hints.

2025-03-10 - 1.5.1 - fix(core)

Improve error handling, logging, and test suite; update dependency versions

- Updated devDependencies versions in package.json (@git.zone/tsbuild, @push.rocks/tapbundle, and @push.rocks/smartdaemon)
- Refactored error handling and enhanced logging in ProcessMonitor and ProcessWrapper modules
- Improved test structure by adding clear module import tests and usage examples in test files

2025-03-04 - 1.5.0 - feat(cli)

Enhance CLI with new process management commands

- Added comprehensive CLI commands for process management including start, stop, restart, list, describe and logs.
- Implemented memory string parsing for process memory limits.
- Enhanced CLI output with formatted table listings for active processes.

2025-03-03 - 1.4.0 - feat(core)

Introduced process management features using ProcessWrapper and enhanced configuration.

- Added ProcessWrapper for wrapping and managing child processes.
- Refactored process monitoring logic using ProcessWrapper.
- Introduced TspmConfig for configuration handling.
- Enhanced CLI to support new process management commands like 'startAsDaemon'.

2025-03-01 - 1.3.1 - fix(test)

Update test script to fix type references and remove private method call

- Corrected type references in test script for IMonitorConfig.
- Fixed test script to use console.log instead of private method monitor.log.

2025-03-01 - 1.3.0 - feat(cli)

Add CLI support with command parsing and version display

- Added a basic CLI interface using smartcli.
- Implemented command parsing with a 'restart' command.
- Integrated project version display in the CLI.

2025-03-01 - 1.2.0 - feat(core)

Introduce ProcessMonitor with memory management and spawning features

- Added ProcessMonitor class with functionality to manage process execution and memory usage.
- Implemented process spawning with ability to handle command arguments and directories.
- Added periodic memory monitoring and automatic restarts when memory thresholds are exceeded.
- ProcessMonitor now logs its actions with optional configuration name for better identification.
- Updated test file to include example usage of ProcessMonitor.

2025-03-01 - 1.1.1 - fix(package)

Update dependencies and pnpm configuration

- Updated @types/node to 22.13.8
- Updated pnpm configuration to include onlyBuiltDependencies with esbuild, mongodb-memory-server, and puppeteer

2025-03-01 - 1.1.0 - feat(core)

Introduce ProcessMonitor class and integrate native and external plugins

- Added a new ProcessMonitor class to manage and monitor child processes with memory constraints.
- Integrated native 'path' and external '@push.rocks/smartpath' packages in a unified plugins file.
- Adjusted index and related files for improved modular structure.

2025-02-24 - 1.0.3 - fix(core)

Corrected description in package.json and readme.md from 'task manager' to 'process manager'.

- Updated the project description in package.json.
- Aligned the description in readme.md with package.json.

2025-02-24 - 1.0.2 - fix(core)

Internal changes with no functional impact.

2025-02-24 - 1.0.1 - initial release

Initial release with baseline functionality.