@git.zone/tstest

Documentation for @git.zone/tstest

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

readme.md for @git.zone/tstest

◫ **A powerful, modern test runner for TypeScript** — beautiful output, multi-runtime support, and a batteries-included test framework that makes testing actually enjoyable.

Availability and Links

- npmjs.org (npm package)
- code.foss.global (source)

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.

Why tstest?

Most TypeScript test runners feel like an afterthought — clunky configuration, ugly output, and poor TypeScript support. **tstest** was built from the ground up for TypeScript developers who want:

- ◫ **Zero config** — Point it at your test directory and go
- ◫ **Multi-runtime** — Run the same tests on Node.js, Deno, Bun, and Chromium
- ◫ **Beautiful output** — Color-coded results with emojis, progress bars, and visual diffs
- ⚡ **Built-in everything** — Assertions, snapshots, fixtures, retries, timeouts, parallel execution
- ◫ **Server-side tooling** — Free ports, HTTPS certs, ephemeral databases, S3 storage — all out of the box

Installation

```
pnpm install --save-dev @git.zone/tstest
```

Module Exports

tstest ships as four modules, each optimized for a different use case:

Export Path	Environment	Purpose
`@git.zone/tstest`	CLI	Test runner — discovers and executes test files
`@git.zone/tstest/tapbundle`	Browser + Node	Core test framework — `tap`, `expect`, lifecycle hooks
`@git.zone/tstest/tapbundle_serverside`	Node.js only	Server-side utilities — ports, certs, databases, shell
`@git.zone/tstest/tapbundle_protocol`	Isomorphic	TAP Protocol V2 — structured metadata, events, diffs

Quick Start

1. Write a test

```
// test/test.math.ts
import { tap, expect } from '@git.zone/tstest/tapbundle';

tap.test('should add numbers', async () => {
  expect(2 + 2).toEqual(4);
});

tap.test('should handle async operations', async (tools) => {
  await tools.delayFor(100);
  const result = await fetchData();
  expect(result).toBeTruthy();
});
```

```
export default tap.start();
```

2. Run it

```
# Run all tests
tstest test/

# Run a specific file
tstest test/test.math.ts

# Use glob patterns
tstest "test/**/*.ts"

# Verbose mode (shows console output)
tstest test/ --verbose

# Watch mode
tstest test/ --watch
```

3. See beautiful output

```
□□Test Discovery
   Mode: directory
   Pattern: test
   Found: 4 test file(s)

▶□  test/test.math.ts (1/4)
   Runtime: Node.js
   □ should add numbers (2ms)
   □ should handle async operations (105ms)
   Summary: 2/2 PASSED in 1.2s

□□Test Summary
┌──────────────────────────────┐
│ Total Files:                 4 │
│ Total Tests:                 8 │
```

```
│ Passed:                      8 │
│ Failed:                      0 │
│ Duration:                 2.4s │
└──────────────────────────────┘


ALL TESTS PASSED! □□
```

Multi-Runtime Architecture

tstest supports running your tests across **four JavaScript runtimes**, letting you verify cross-platform compatibility with zero extra effort.

Test File Naming Convention

Name your test files with runtime specifiers to control where they run:

Pattern	Runtimes	Example
`*.ts`	Node.js (default)	`test.api.ts`
`*.node.ts`	Node.js only	`test.server.node.ts`
`*.chromium.ts`	Chromium browser	`test.dom.chromium.ts`
`*.deno.ts`	Deno	`test.http.deno.ts`
`*.bun.ts`	Bun	`test.fast.bun.ts`
`*.all.ts`	All runtimes	`test.universal.all.ts`
`*.node+chromium.ts`	Node.js + Chromium	`test.isomorphic.node+chromium.ts`
`*.node+deno.ts`	Node.js + Deno	`test.cross.node+deno.ts`
`*.chromium.nonci.ts`	Chromium, skip in CI	`test.visual.chromium.nonci.ts`

Runtime Execution Order

When multiple runtimes are specified, tests execute in this order: **Node.js → Bun → Deno → Chromium**

Migration from Legacy Naming

```
# Dry run — see what would change
tstest migrate --dry-run

# Apply migrations (uses git mv to preserve history)
tstest migrate --write
```

Legacy Pattern	Modern Equivalent
`*.browser.ts`	`*.chromium.ts`
`*.both.ts`	`*.node+chromium.ts`

CLI Options

Option	Description
`--quiet`, `-q`	Minimal output — perfect for CI
`--verbose`, `-v`	Show all console output from tests
`--no-color`	Disable colored output
`--json`	Output results as JSON (CI/CD pipelines)
`--logfile`	Save detailed logs with error/diff tracking
`--tags <tags>`	Run only tests with specific tags
`--timeout <seconds>`	Timeout test files after N seconds
`--startFrom <n>`	Start from test file number N
`--stopAt <n>`	Stop at test file number N
`--watch`, `-w`	Re-run tests on file changes
`--watch-ignore <patterns>`	Ignore patterns in watch mode
`--only`	Run only tests marked with `.only`

Writing Tests with tapbundle

Basic Syntax

```
import { tap, expect } from '@git.zone/tstest/tapbundle';
```

```
tap.test('basic test', async () => {
  expect(2 + 2).toEqual(4);
});

tap.test('with tools', async (tools) => {
  await tools.delayFor(100);
  tools.timeout(5000);
  expect(true).toBeTrue();
});

export default tap.start();
```

Test Modifiers

```
// Skip
tap.skip.test('not ready yet', async () => { /* skipped */ });

// Only (exclusive)
tap.only.test('focus on this', async () => { /* only this runs */ });

// Todo
tap.todo.test('implement later', async () => { /* marked as todo */ });

// Fluent chaining
tap.timeout(5000)
   .retry(3)
   .tags('api', 'integration')
   .test('complex test', async (tools) => { /* configured */ });
```

Test Organization with describe()

```
tap.describe('User Management', () => {
  tap.beforeEach(async () => {
    // setup before each test
  });

  tap.afterEach(async () => {
```

```
    // cleanup after each test
  });

  tap.test('should create user', async () => { /* ... */ });
  tap.test('should delete user', async () => { /* ... */ });

  tap.describe('Permissions', () => {
    tap.test('should set admin role', async () => { /* ... */ });
  });
});
```

Pre-Tasks and Post-Tasks

```
tap.preTask('setup database', async () => {
  await initializeDatabase();
});

tap.test('uses the database', async () => { /* ... */ });

tap.postTask('cleanup database', async () => {
  await cleanupDatabase();
});
```

Test Tools

Every test function receives a `tools` parameter packed with utilities:

```
tap.test('tools demo', async (tools) => {
  // □□ Delays
  await tools.delayFor(1000);
  await tools.delayForRandom(100, 500);

  // □□ Skip
  tools.skipIf(process.env.CI === 'true', 'Skipping in CI');
  tools.skip('reason');

  // □□Retry & timeout
  tools.retry(3);
```

```
  tools.timeout(10000);

  // □□Context sharing between tests
  tools.context.set('userId', 12345);
  const userId = tools.context.get('userId');

  // □□Deferred promises
  const deferred = tools.defer();
  setTimeout(() => deferred.resolve('done'), 100);
  await deferred.promise;

  // □□Error capture
  const error = await tools.returnError(async () => {
    throw new Error('Expected error');
  });
  expect(error).toBeInstanceOf(Error);

  // □ Allow failure (test won't fail the suite)
  tools.allowFailure();
});
```

Snapshot Testing

```
tap.test('snapshot test', async (tools) => {
  const output = generateComplexOutput();
  await tools.matchSnapshot(output);
  await tools.matchSnapshot(output.header, 'header');
});

// Update snapshots: UPDATE_SNAPSHOTS=true tstest test/
```

Test Fixtures

```
tap.defineFixture('testUser', async (data) => ({
  id: Date.now(),
  name: data?.name || 'Test User',
  email: data?.email || 'test@example.com',
```

```
}));

tap.test('fixture test', async (tools) => {
  const user = await tools.fixture('testUser', { name: 'John' });
  expect(user.name).toEqual('John');

  // Factory pattern for multiple instances
  const users = await tools.factory('testUser').createMany(5);
  expect(users).toHaveLength(5);
});
```

Parallel Execution

```
// Within a file
tap.parallel().test('parallel test 1', async () => { /* ... */ });
tap.parallel().test('parallel test 2', async () => { /* ... */ });

// Across files — same suffix = parallel group
// test.api.para__1.ts  ←— run together
// test.db.para__1.ts   ←— run together
// test.auth.para__2.ts ←— runs after para__1 completes
```

Assertions (expect)

tapbundle uses [@push.rocks/smartexpect](#) for assertions with automatic diff generation on failures:

```
// Equality
expect(value).toEqual(5);
expect(obj).toDeepEqual({ a: 1, b: 2 });

// Types
expect('hello').toBeTypeofString();
expect(42).toBeTypeofNumber();
expect([]).toBeArray();

// Comparisons
expect(5).toBeGreaterThan(3);
```

```
expect(0.1 + 0.2).toBeCloseTo(0.3, 10);

// Truthiness
expect(true).toBeTrue();
expect(null).toBeNull();
expect(undefined).toBeUndefined();

// Strings
expect('hello world').toStartWith('hello');
expect('hello world').toEndWith('world');
expect('hello world').toInclude('lo wo');
expect('hello world').toMatch(/^hello/);

// Arrays
expect([1, 2, 3]).toContain(2);
expect([1, 2, 3]).toContainAll([1, 3]);
expect([1, 2, 3]).toHaveLength(3);

// Objects
expect(obj).toHaveProperty('name');
expect(obj).toMatchObject({ name: 'John' });

// Functions & Promises
expect(() => { throw new Error(); }).toThrow();
await expect(Promise.resolve('val')).resolves.toEqual('val');
await expect(Promise.reject(new Error())).rejects.toThrow();

// Custom
expect(7).customAssertion(v => v % 2 === 1, 'Value is not odd');
```

Server-Side Tools (tapbundle_serverside)

For Node.js-only tests, import server-side utilities:

```
import { tapNodeTools } from '@git.zone/tstest/tapbundle_serverside';
import { tap, expect } from '@git.zone/tstest/tapbundle';
```

🌐 Network Utilities

Find free local ports for test servers — no more port conflicts:

```
tap.test('should start server on free port', async () => {
  // Single free port (random in range 3000–60000)
  const port = await tapNodeTools.findFreePort();

  // Custom range
  const port2 = await tapNodeTools.findFreePort({ startPort: 8000, endPort: 9000 });

  // With exclusions
  const port3 = await tapNodeTools.findFreePort({ exclude: [8080, 8443] });
});

tap.test('should allocate multiple ports', async () => {
  // Multiple distinct ports
  const [httpPort, wsPort, adminPort] = await tapNodeTools.findFreePorts(3);

  // Consecutive port range (e.g., 4000, 4001, 4002)
  const portRange = await tapNodeTools.findFreePortRange(3, {
    startPort: 20000,
    endPort: 30000,
  });
});
```

🔒 HTTPS Certificates

Generate self-signed certs for testing secure connections:

```
tap.test('should serve over HTTPS', async () => {
  const { key, cert } = await tapNodeTools.createHttpsCert('localhost');
  const server = https.createServer({ key, cert }, handler);
  server.listen(port);
});
```

Shell Commands

```
const result = await tapNodeTools.runCommand('ls -la');
console.log(result.exitCode); // 0
```

Environment Variables

```
const apiKey = await tapNodeTools.getEnvVarOnDemand('GITHUB_API_KEY');
// Prompts if not set, stores in .nogit/.env for future use
```

Ephemeral MongoDB

```
const mongo = await tapNodeTools.createSmartmongo();
// ... run database tests ...
await mongo.stop();
```

Local S3 Storage

```
const s3 = await tapNodeTools.createSmarts3();
// ... run object storage tests ...
await s3.stop();
```

Test File Directives

Control runtime behavior directly from your test files using special comment directives at the top of the file. Directives must appear before any `import` statements.

Deno Permissions

By default, Deno tests run with `--allow-read`, `--allow-env`, `--allow-net`, `--allow-write`, `--allow-sys`, and `--allow-import`. Add directives to request additional permissions:

```
// tstest:deno:allowAll
import { tap, expect } from '@git.zone/tstest/tapbundle';
```

```
tap.test('test with full Deno permissions', async () => {
  // Runs with --allow-all (e.g., for FFI, subprocess spawning, etc.)
});

export default tap.start();
```

Available Directives

Directive	Effect
`// tstest:deno:allowAll`	Grants all Deno permissions (`--allow-all`)
`// tstest:deno:allowRun`	Adds `--allow-run` for subprocess spawning
`// tstest:deno:allowFfi`	Adds `--allow-ffi` for native library calls
`// tstest:deno:allowHrtime`	Adds `--allow-hrtime` for high-res timers
`// tstest:deno:flag:--unstable-ffi`	Passes any arbitrary Deno flag
`// tstest:node:flag:--max-old-space-size=4096`	Passes flags to Node.js
`// tstest:bun:flag:--smol`	Passes flags to Bun

Multiple Directives

Combine as many directives as needed:

```
// tstest:deno:allowRun
// tstest:deno:allowFfi
// tstest:deno:flag:--unstable-ffi
import { tap, expect } from '@git.zone/tstest/tapbundle';

tap.test('test with Rust FFI', async () => {
  // Has --allow-run, --allow-ffi, and --unstable-ffi in addition to defaults
});

export default tap.start();
```

Shared Directives via 00init.ts

Directives in a `00init.ts` file apply to all test files in that directory. Test file directives are merged with (and extend) init file directives.

```
// test/00init.ts
// tstest:deno:allowRun
```

```
// test/test.mytest.deno.ts
// tstest:deno:allowFfi
// Both --allow-run (from 00init.ts) and --allow-ffi are active
import { tap, expect } from '@git.zone/tstest/tapbundle';
```

Advanced Features

Watch Mode

```
tstest test/ --watch
tstest test/ --watch --watch-ignore "dist/**,coverage/**"
```

- Shows which files triggered the re-run
- 300ms debouncing to batch rapid changes
- Clears console between runs

Visual Diffs

When assertions fail, you get beautiful diffs:

```
 should return correct user data


   Object Diff:
   {
     name: "John",
   - age: 30,
   + age: 31,
     email: "john@example.com"
   }
```

Enhanced Logging

```
tstest test/ --logfile
```

Folder	Contents
`.nogit/testlogs/`	Current run logs
`.nogit/testlogs/previous/`	Previous run logs
`.nogit/testlogs/00err/`	Failed test logs
`.nogit/testlogs/00diff/`	Changed output diffs

JSON Output (CI/CD)

```
tstest test/ --json > test-results.json
```

```
{"event":"discovery","count":4,"pattern":"test","executionMode":"directory"}
{"event":"testResult","testName":"prepare test","passed":true,"duration":1}
{"event":"summary","summary":{"totalFiles":4,"totalTests":4,"totalPassed":4,"totalFailed":0}}
```

Tag Filtering

```
tap.tags('unit', 'api').test('api unit test', async () => { /* ... */ });
```

```
tstest test/ --tags unit,api
```

Test File Range

```
tstest test/ --startFrom 5 --stopAt 10  # Run files 5-10 only
```

Browser Testing with webhelpers

```
import { tap, webhelpers } from '@git.zone/tstest/tapbundle';


tap.test('DOM test', async () => {
```

```
  const element = await webhelpers.fixture(webhelpers.html`
    <div class="container">
      <h1>Hello</h1>
    </div>
  `);
  expect(element.querySelector('h1').textContent).toEqual('Hello');
});
```

TapWrap (Global Lifecycle)

```
import { TapWrap } from '@git.zone/tstest/tapbundle';

const tapWrap = new TapWrap({
  before: async () => { await globalSetup(); },
  after: async () => { await globalCleanup(); },
});
```

tapbundle Protocol V2

tstest includes an enhanced TAP protocol that extends TAP 13 with structured metadata while staying backwards compatible. Protocol markers (`⟦TSTEST:...⟧`) are invisible to standard TAP parsers.

```
import { ProtocolEmitter, ProtocolParser } from '@git.zone/tstest/tapbundle_protocol';

// Emit
const emitter = new ProtocolEmitter();
console.log(emitter.emitProtocolHeader());  // ⟦TSTEST:PROTOCOL:2.0.0⟧
console.log(emitter.emitTest({
  ok: true, testNumber: 1, description: 'test',
  metadata: { time: 42, tags: ['unit'] }
}).join('\n'));

// Parse
const parser = new ProtocolParser();
const messages = parser.parseLine('ok 1 - test ⟦TSTEST:time:42⟧');
```

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/tstest

2026-03-27 - 3.6.3 - fix(tapbundle_serverside)

add TapNodeTools cleanup hooks to stop SmartNetwork processes after tests

- replace the exported tapNodeTools singleton with a TapNodeTools class that registers cleanup on the tap instance
- run registered cleanup functions at the end of tap.start() to prevent lingering background processes from keeping Node.js alive
- stop SmartNetwork after selecting Chromium runtime ports and update smartnetwork to ^4.5.2

2026-03-27 - 3.6.2 - fix(cli,chromium-runtime)

clean up long-lived test runner resources after runs to prevent hanging processes

- call TsTest.cleanup() after non-watch CLI runs so smartexit handlers are deregistered
- terminate active WebSocket clients before closing the Chromium runtime server to release open connections cleanly

2026-03-26 - 3.6.1 - fix(chromium runtime)

encode Chromium test bundle names correctly for nested file paths

- Use replaceAll() so bundle cache filenames consistently handle test paths with multiple directory separators.
- URL-encode the bundleName query parameter before opening the Chromium test page to avoid lookup issues with generated bundle filenames.

2026-03-24 - 3.6.0 - feat(config)

migrate project metadata to .smartconfig and refresh build configuration

- replace npmextra.json with .smartconfig.json and update packaged files accordingly
- add module-level README documentation under ts/readme.md
- bump build and runtime dependencies including tsbuild, tsbundle, smartstorage, smartwatch, smartserve, and ws
- relax TypeScript compiler strictness and include Node types in tsconfig
- update README license links to point to license.md

2026-03-23 - 3.5.1 - fix(runtime)

handle expected exitCode rejection after terminating timed out test processes

- swallow the child process exitCode rejection triggered by timeout termination
- reduce unhandled rejection noise when test files exceed the configured timeout

2026-03-19 - 3.5.0 - feat(tstest)

add support for package.json before scripts during test execution

- load test:before or test:before:once once per test run from package.json scripts
- run test:before:testfile before each test file execution and pass TSTEST_FILE and TSTEST_RUNTIME environment variables
- log before-script lifecycle events and abort or skip execution when setup scripts fail

2026-03-18 - 3.4.0 - feat(tapbundle,deno)

replace smarts3 test tooling with smartstorage and pre-resolve Deno test dependencies

- switch TapNodeTools storage helper from @push.rocks/smarts3 to @push.rocks/smartstorage and rename createSmarts3() to createSmartStorage()
- update node tests and tapbundle server-side documentation to use the new smartstorage helper
- run `deno install --entrypoint` before executing Deno tests to resolve dependencies up front
- bump supporting development dependencies including @types/node and @push.rocks/smartshell

2026-03-09 - 3.3.2 - fix(deps)

bump dependency versions and reorder smartserve in package.json

- bump @push.rocks/smartbrowser from ^2.0.10 to ^2.0.11
- bump @push.rocks/smartfs from ^1.4.0 to ^1.5.0
- move @push.rocks/smartserve to later position in dependencies (version unchanged: ^2.0.1)

2026-03-09 - 3.3.1 - fix(serve)

migrate test HTTP server to @push.rocks/smartserve and update related dependencies

- Replace @api.global/typedserver with @push.rocks/smartserve and FileServer; use SmartServe.setHandler to serve static assets and a custom /test response.
- Export smartserve from ts/tstest.plugins.ts and remove typedserver import/export.
- Update package.json dependencies: add @push.rocks/smartserve@^2.0.1 and bump @push.rocks/smartbrowser to ^2.0.10.

2026-03-06 - 3.3.0 - feat(testfile-directives)

Add per-test file directives to control runtime permissions and flags (Deno, Node, Bun, Chromium)

- Introduce test file directive parser (ts/tstest.classes.testfile.directives.ts) to parse comments like // tstest:deno:allowAll and map them to runtime options.

- Add DENO_DEFAULT_PERMISSIONS constant and centralize Deno default flags (ts/tstest.classes.runtime.deno.ts) to avoid repeating the list.
- Integrate directives into the test runner (ts/tstest.classes.tstest.ts): read directives from test files and optional 00init.ts, merge them, and pass runtime-specific options to adapters.
- Documentation: add a "Test File Directives" section to readme.md with examples and available directives.
- Add automated tests for directives behavior (test/test.directives.node.ts).
- Bump package metadata and minor dependency updates; update package description and npmextra.json to reflect new functionality.

2026-03-03 - 3.2.0 - feat(tapbundle_serverside)

add network port discovery utilities and migrate file I/O to smartfs; refactor runtimes to use Node fs and SmartFs, update server APIs and bump dependencies

- Add tapNodeTools.findFreePort, findFreePorts and findFreePortRange to provide network port discovery and ranges for tests
- Integrate @push.rocks/smartfs (smartfsInstance) and replace many @push.rocks/smartfile.fs usages with smartfsInstance file/directory APIs
- Switch several internals to Node's fs (exported via plugins) and introduce SmartFileFactory.nodeFs() for file handling
- Replace smartchok with smartwatch for file watching and update watch/start/stop flows
- Update server instantiation to TypedServer and change addRoute usage to the new handler signature; serve bundled test directory
- Add tests for network tools and update migration/test code to use smartfsInstance
- Bump multiple dependencies (e.g. @api.global/typedserver, @push.rocks/smartfile/smartfs/smartnetwork/smarts3/smartwatch) and @git.zone/tsbuild

2026-01-25 - 3.1.8 - fix(tapbundle)

treat tests that call tools.allowFailure() as passing and update tests to use tools parameter

- Set testResult.ok to this.failureAllowed so allowed failures are considered passing in the tap test runner implementation (ts_tapbundle/tapbundle.classes.taptest.ts).
- Updated multiple tests to accept the tools parameter and call tools.allowFailure() where failures are intended (test/tapbundle/test.performance-metrics.ts, test/tstest/test.fail.ts, test/tstest/test.failing-with-logs.ts).

- Prevents intentionally-failing tests from skewing timing/metric calculations and preserves console logs for allowed failures.

2026-01-25 - 3.1.7 - fix(tap-parser)

append newline to WebSocket tap log messages to ensure proper line-by-line processing

- Fixes handling of WebSocket console.log messages by appending a trailing newline before processing to avoid merged lines.
- Modified ts/tstest.classes.tap.parser.ts: handleTapLog now calls _processLog(tapLog + '\n').

2026-01-19 - 3.1.6 - fix(logging)

handle mid-line streaming output in test logger and add streaming tests

- Introduce isOutputMidLine flag to track when streaming output does not end with a newline
- Only prepend the visual prefix at the start of a line and append segments to the last buffered entry when mid-line
- Write consistent output to log files for both complete lines and raw streaming segments
- Add tests to exercise streaming behavior: test/tstest/test.gap-debug.ts and test/tstest/test.gap-debug2.ts

2026-01-19 - 3.1.5 - fix(tstest)

preserve streaming console output and correctly buffer incomplete TAP lines

- Reworked TapParser._processLog to buffer incomplete lines and only parse complete TAP protocol lines
- Added TapParser.lineBuffer and _looksLikeTapStart() to detect and buffer starts of TAP messages
- Added TapParser._handleConsoleOutput() to centralize console output handling and snapshot parsing; flushes buffered content on process exit
- Added TapTestResult.addLogLineRaw() to append streaming text without adding newlines
- Added TsTestLogger.testConsoleOutputStreaming() and logToTestFileRaw() to preserve streaming output formatting in both console and logfile

2025-12-30 - 3.1.4 - fix(webhelpers)

improve browser test fixture to append element and await custom element upgrade and Lit update completion; add generic return type; update npm packaging release config; remove pnpm onlyBuiltDependencies

- ts_tapbundle/webhelpers.ts: make fixture generic and return T; append created element to document; await customElements.whenDefined for custom elements and await updateComplete for Lit/async components to ensure stable rendering in tests
- npmextra.json: add @git.zone/cli module metadata and release.registries/accessLevel; add @ship.zone/szci entry
- pnpm-workspace.yaml: remove onlyBuiltDependencies entries

2025-11-21 - 3.1.3 - fix(docs)

Update package author and expand license/legal and issue-reporting information in tapbundle docs

- Update package.json author field to Task Venture Capital GmbH
- Add expanded License and Legal Information to ts_tapbundle/readme.md (clarifies MIT license scope and trademark guidance)
- Add expanded License and Legal Information to ts_tapbundle_protocol/readme.md (clarifies MIT license scope and trademark guidance)
- Add Issue Reporting and Security section to ts_tapbundle_protocol/readme.md pointing users to the community hub for bug/security reports
- Include company information and trademark usage guidance in readmes

2025-11-21 - 3.1.2 - fix(docs)

Update README: add issue reporting/security guidance and expanded changelog (3.1.1/3.1.0)

- Add 'Issue Reporting and Security' section pointing to https://community.foss.global/ for bug/security reports and contributor onboarding.
- Expand Changelog with Version 3.1.1 notes: fixed TapTools parameter passing to suite lifecycle hooks (beforeAll/afterAll) and updated @push.rocks/smarts3 dependency to ^3.0.0.

- Include Changelog entries for Version 3.1.0: postTask() API, suite beforeAll/afterAll, new parallel() fluent API, and enhanced tapbundle documentation.
- Documentation-only change (no source code modifications).

2025-11-21 - 3.1.1 - fix(tapbundle)

Pass TapTools to suite lifecycle hooks (beforeAll/afterAll) and update @push.rocks/smarts3 to ^3.0.0

- Replace usage of a Deferred promise with a TapTools instance when invoking suite.beforeAll and suite.afterAll
- Add import for TapTools in ts_tapbundle/tapbundle.classes.tap.ts
- Bump dependency @push.rocks/smarts3 from ^2.2.7 to ^3.0.0 in package.json

2025-11-20 - 3.1.0 - feat(tapbundle)

Add global postTask (teardown) and suite lifecycle hooks (beforeAll/afterAll) to tapbundle

- Introduce PostTask class (ts_tapbundle/tapbundle.classes.posttask.ts) and tap.postTask() API for global teardown.
- Integrate postTask execution into Tap.start() so postTasks run after all tests and before the global afterAll hook.
- Add suite-level beforeAll and afterAll support and ensure afterAll runs after child suites and their tests (changes in ts_tapbundle/tapbundle.classes.tap.ts).
- Add lifecycle tests (test/tapbundle/test.new-lifecycle.ts) verifying execution order, including parallel tests.
- Update documentation (readme.hints.md) describing Phase 1 API improvements and usage notes.
- This is additive and backward-compatible (no breaking changes).

2025-11-20 - 3.0.1 - fix(@push.rocks/smarts3)

Bump @push.rocks/smarts3 dependency to ^2.2.7

- Update package.json: @push.rocks/smarts3 upgraded from ^2.2.6 to ^2.2.7

2025-11-19 - 3.0.0 - BREAKING CHANGE(tapbundle_serverside)

Rename Node-specific tapbundle module to tapbundle_serverside and migrate server-side utilities

- Change public export in package.json from ./tapbundle_node to ./tapbundle_serverside — consumers must update imports to @git.zone/tstest/tapbundle_serverside
- Move and re-create Node-only implementation files under ts_tapbundle_serverside (plugins, paths, classes.tapnodetools, classes.testfileprovider, index, tspublish.json) and remove legacy ts_tapbundle_node sources
- Update internal imports and tests to reference the new tapbundle_serverside path (e.g. test/tapbundle/test.node.ts updated)
- Update documentation (readme.md and readme.hints.md) to describe the new tapbundle_serverside export and its server-side utilities
- Ensure build outputs and publish metadata reflect the new module directory (tspublish.json order preserved)

2025-11-19 - 2.8.3 - fix(dependencies)

Update dependency versions

- Bump devDependency @git.zone/tsbuild to ^3.1.0
- Upgrade @git.zone/tsrun to ^2.0.0 (major)
- Upgrade @push.rocks/smartenv to ^6.0.0 (major)
- Upgrade @push.rocks/smartrequest to ^5.0.1 (major/feature in dependency)
- Patch updates: @api.global/typedserver → ^3.0.80, @git.zone/tsbundle → ^2.5.2, @push.rocks/smartmongo → ^2.0.14

2025-11-17 - 2.8.2 - fix(logging)

Include runtime identifier in per-test logfile name and sanitize runtime string

- Append a sanitized runtime identifier to the per-test log filename (format: <safeFilename>__<safeRuntime>.log) so runs for different runtimes don't clash
- Sanitize runtime names by lowercasing and removing non-alphanumeric characters to produce filesystem-safe filenames

2025-11-17 - 2.8.1 - fix(config)

Remove Bun config file and set deno.json useDefineForClassFields to false for compatibility

- Removed bunfig.toml (Bun-specific TypeScript decorator configuration) — stops shipping a project-local Bun transpiler config.
- Updated deno.json: set compilerOptions.useDefineForClassFields = false to keep legacy class field semantics and avoid runtime/emit incompatibilities in Deno.

2025-11-17 - 2.8.0 - feat(runtime-adapters)

Enable TypeScript decorator support for Deno and Bun runtimes and add decorator tests

- Add bunfig.toml to enable experimentalDecorators for Bun runtime
- Add deno.json to enable experimentalDecorators and set target/lib for Deno
- Update Bun runtime adapter to note bunfig.toml discovery so Bun runs with decorator support
- Update Deno runtime adapter to auto-detect deno.json / deno.jsonc and pass configPath in default options
- Add integration tests for decorators (test/decorator.all.ts) to verify decorator support across runtimes

2025-10-26 - 2.7.0 - feat(tapbundle_protocol)

Add package export for tapbundle_protocol to expose protocol utilities

- Add './tapbundle_protocol' export in package.json pointing to './dist_ts_tapbundle_protocol/index.js'.

- Allows consumers to import protocol utilities (ProtocolEmitter, ProtocolParser, types) via '@git.zone/tstest/tapbundle_protocol'.
- Non-breaking: only extends package exports surface.

2025-10-17 - 2.6.2 - fix(@push.rocks/smartrequest)

Bump @push.rocks/smartrequest from ^4.3.1 to ^4.3.2

- Update dependency @push.rocks/smartrequest from ^4.3.1 to ^4.3.2

2025-10-17 - 2.6.1 - fix(runtime-adapters)

Silence shell version checks for Bun and Deno; add local Claude settings

- Replace smartshell.exec with execSilent in ts/tstest.classes.runtime.bun.ts to suppress output when checking Bun availability
- Replace smartshell.exec with execSilent in ts/tstest.classes.runtime.deno.ts to suppress output when checking Deno availability
- Add .claude/settings.local.json to record local Claude agent permissions/config used for development

2025-10-17 - 2.6.0 - feat(runtime-adapters)

Add runtime environment availability check and logger output; normalize runtime version strings

- Introduce checkEnvironment() in TsTest and invoke it at the start of run() to detect available runtimes before executing tests.
- Add environmentCheck(availability) to TsTestLogger to print a human-friendly environment summary (with JSON and quiet-mode handling).

- Normalize reported runtime version strings from adapters: prefix Deno and Bun versions with 'v' and simplify Chromium version text.
- Display runtime availability information to the user before moving previous logs or running tests.
- Includes addition of local .claude/settings.local.json (local dev/tooling settings).

2025-10-17 - 2.5.2 - fix(runtime.node)

Improve Node runtime adapter to use tsrun.spawnPath, strengthen tsrun detection, and improve process lifecycle and loader handling; update tsrun dependency.

- Use tsrun.spawnPath to spawn Node test processes and pass structured spawn options (cwd, env, args, stdio).
- Detect tsrun availability via plugins.tsrun and require spawnPath; provide a clearer error message when tsrun is missing or outdated.
- Pass --web via spawn args and set TSTEST_FILTER_TAGS on the spawned process env instead of mutating the parent process.env.
- When a 00init.ts exists, create a temporary loader that imports both 00init.ts and the test file, run the loader via tsrun.spawnPath, and clean up the loader after execution.
- Use tsrunProcess.terminate()/kill for timeouts to ensure proper process termination and improve cleanup handling.
- Export tsrun from ts/tstest.plugins.ts so runtime code can access tsrun APIs via the plugins object.
- Bump dependency @git.zone/tsrun from ^1.3.4 to ^1.6.2 in package.json.

2025-10-16 - 2.5.1 - fix(deps)

Bump dependencies and add local tooling settings

- Bumped @api.global/typedserver from ^3.0.78 to ^3.0.79
- Bumped @git.zone/tsrun from ^1.3.3 to ^1.3.4
- Bumped @push.rocks/smartjson from ^5.0.20 to ^5.2.0
- Bumped @push.rocks/smartlog from ^3.1.9 to ^3.1.10
- Add local settings configuration file for developer tooling

2025-10-12 - 2.5.0 - feat(tstest.classes.runtime.parser)

Add support for "all" runtime token and update docs/tests; regenerate lockfile and add local settings

- Add support for the `all` runtime token (expands to node, chromium, deno, bun) in tstest filename parser (tstest.classes.runtime.parser)
- Handle `all` with modifiers (e.g. `*.all.nonci.ts`) and mixed tokens (e.g. `node+all`) so it expands to the full runtime set
- Add unit tests covering `all` cases in test/test.runtime.parser.node.ts
- Update README (examples and tables) to document `.all.ts` and `.all.nonci.ts` usage and include a universal example
- Update ts files' parser comments and constants to include ALL_RUNTIMES
- Add deno.lock (dependency lockfile) and a local .claude/settings.local.json for project permissions / local settings

2025-10-11 - 2.4.3 - fix(docs)

Update documentation: expand README with multi-runtime architecture, add module READMEs, and add local dev settings

- Expanded project README: fixed typos, clarified availability header, and added a detailed Multi-Runtime Architecture section (runtimes, naming conventions, migration tool, examples, and runtime-specific notes).
- Inserted additional example output and adjusted JSON/example sections to reflect multi-runtime flows and updated totals/durations in examples.
- Added dedicated README files for ts_tapbundle, ts_tapbundle_node, and ts_tapbundle_protocol modules with API overviews and usage guides.
- Added .claude/settings.local.json to provide local development permissions/settings used by the project tooling.
- Minor formatting and documentation cleanup (whitespace, headings, and changelog entries).

2025-10-10 - 2.4.2 - fix(deno)

Enable additional Deno permissions for runtime adapters and add local dev settings

- Add --allow-sys, --allow-import and --node-modules-dir to the default Deno permission set used by the Deno runtime adapter
- Include the new permission flags in the fallback permissions array when constructing Deno command args
- Add .claude/settings.local.json to capture local development permissions and helper commands

2025-10-10 - 2.4.1 - fix(runtime/deno)

Enable Deno runtime tests by adding required permissions and local settings

- ts/tstest.classes.runtime.deno.ts: expanded default Deno permissions to include --allow-net, --allow-write and --sloppy-imports to allow network access, file writes and permissive JS/TS imports
- ts/tstest.classes.runtime.deno.ts: updated fallback permissions used when building the Deno command to match the new default set
- Added .claude/settings.local.json with a set of allowed local commands/permissions used for local development/CI tooling

2025-10-10 - 2.4.0 - feat(runtime)

Add runtime adapters, filename runtime parser and migration tool; integrate runtime selection into TsTest and add tests

- Introduce RuntimeAdapter abstraction and RuntimeAdapterRegistry to manage multiple runtimes
- Add runtime adapters: NodeRuntimeAdapter, ChromiumRuntimeAdapter, DenoRuntimeAdapter and BunRuntimeAdapter
- Add filename runtime parser utilities: parseTestFilename, isLegacyFilename and getLegacyMigrationTarget
- Add Migration class to detect and (dry-run) migrate legacy test filenames to the new naming convention
- Integrate runtime registry into TsTest and choose execution adapters based on parsed runtimes; show deprecation warnings for legacy naming
- Add tests covering runtime parsing and migration: test/test.runtime.parser.node.ts and test/test.migration.node.ts

2025-09-12 - 2.3.8 - fix(tstest)

Improve free port selection for Chrome runner and bump smartnetwork dependency

- Use randomized port selection when finding free HTTP and WebSocket ports to reduce collision probability in concurrent runs
- Ensure WebSocket port search excludes the chosen HTTP port so the two ports will not conflict
- Simplify failure handling: throw early if a free WebSocket port cannot be found instead of retrying with a less robust fallback
- Bump @push.rocks/smartnetwork dependency from ^4.2.0 to ^4.4.0 to pick up new findFreePort options

2025-09-12 - 2.3.7 - fix(tests)

Remove flaky dynamic-ports browser test and add local dev tool settings

- Removed test/tapbundle/test.dynamicports.ts — deletes a browser test that relied on injected dynamic WebSocket ports (reduces flaky CI/browser runs).
- Added .claude/settings.local.json — local development settings for the CLAUDE helper (grants allowed dev/automation commands and webfetch permissions).

2025-09-03 - 2.3.6 - fix(tstest)

Update deps, fix chrome server route for static bundles, add local tool settings and CI ignore

- Bump devDependency @git.zone/tsbuild to ^2.6.8
- Bump dependencies: @api.global/typedserver to ^3.0.78, @push.rocks/smartlog to ^3.1.9, @push.rocks/smartrequest to ^4.3.1
- Fix test server static route in ts/tstest.classes.tstest.ts: replace '(.*)' with '/*splat' so bundled test files are served correctly in Chromium runs
- Add .claude/settings.local.json with local permissions for development tasks
- Add .serena/.gitignore to ignore /cache

2025-08-18 - 2.3.5 - fix(core)

Use SmartRequest with Buffer for binary downloads, tighten static route handling, bump dependencies and add workspace/config files

- ts_tapbundle_node/classes.testfileprovider.ts: switch to SmartRequest.create().url(...).get() and convert response to a Buffer before writing to disk to fix binary download handling for the Docker Alpine image.
- ts/tstest.classes.tstest.ts: change server.addRoute from '' *to* '*(.*)' so the typedserver static handler uses a proper regex route.
- package.json: bump several dependencies (e.g. @api.global/typedserver, @git.zone/tsbuild, @push.rocks/smartfile, @push.rocks/smartpath, @push.rocks/smartrequest, @push.rocks/smartshell) to newer patch/minor versions.
- pnpm-workspace.yaml: add onlyBuiltDependencies list (esbuild, mongodb-memory-server, puppeteer).
- Remove registry setting from .npmrc (cleanup).
- Add project/agent config files: .serena/project.yml and .claude/settings.local.json for local tooling/agent configuration.

2025-08-16 - 2.3.4 - fix(ci)

Add local Claude settings to allow required WebFetch and Bash permissions for local tooling and tests

- Add .claude/settings.local.json to configure allowed permissions for local assistant/automation
- Grants WebFetch access for code.foss.global and www.npmjs.com
- Allows various Bash commands used by local tasks and test runs (mkdir, tsbuild, pnpm, node, tsx, tstest, ls, rm, grep, cat)
- No runtime/library code changes — configuration only

2025-08-16 - 2.3.3 - fix(dependencies)

Bump dependency versions and add local Claude settings

- Bumped devDependency @git.zone/tsbuild ^2.6.3 → ^2.6.4
- Updated @git.zone/tsbundle ^2.2.5 → ^2.5.1
- Updated @push.rocks/consolecolor ^2.0.2 → ^2.0.3
- Updated @push.rocks/qenv ^6.1.0 → ^6.1.3
- Updated @push.rocks/smartchok ^1.0.34 → ^1.1.1
- Updated @push.rocks/smartenv ^5.0.12 → ^5.0.13
- Updated @push.rocks/smartfile ^11.2.3 → ^11.2.5
- Updated @push.rocks/smarts3 ^2.2.5 → ^2.2.6

- Updated @push.rocks/smartshell ^3.2.3 → ^3.2.4
- Updated ws ^8.18.2 → ^8.18.3
- Added .claude/settings.local.json for local Claude permissions and tooling (local-only configuration)

2025-07-24 - 2.3.2 - fix(tapbundle)

Fix TypeScript IDE warning about tapTools parameter possibly being undefined

- Changed ITestFunction from interface with optional parameter to union type
- Updated test runner to handle both function signatures (with and without tapTools)
- Resolves IDE warnings while maintaining backward compatibility

2025-05-26 - 2.3.1 - fix(tapParser/logger)

Fix test duration reporting and summary formatting in TAP parser and logger

- Introduce startTime in TapParser to capture the overall test duration
- Pass computed duration to logger methods in evaluateFinalResult for accurate timing
- Update summary output to format duration in a human-readable way (ms vs. s)
- Add local permission settings configuration to .claude/settings.local.json

2025-05-26 - 2.3.0 - feat(cli)

Add '--version' option and warn against global tstest usage in the tstest project

- Introduced a new '--version' CLI flag that prints the version from package.json
- Added logic in ts/index.ts to detect if tstest is run globally within its own project and issue a warning
- Added .claude/settings.local.json to configure allowed permissions for various commands

2025-05-26 - 2.2.6 - fix(tstest)

Improve timeout warning timer management and summary output formatting in the test runner.

- Removed the global timeoutWarningTimer and replaced it with local warning timers in runInNode and runInChrome methods.
- Added warnings when test files run for over one minute if no timeout is specified.
- Ensured proper clearing of warning timers on successful completion or timeout.
- Enhanced quiet mode summary output to clearly display passed and failed test counts.

2025-05-26 - 2.2.5 - fix(protocol)

Fix inline timing metadata parsing and enhance test coverage for performance metrics and timing edge cases

- Updated the protocol parser to correctly parse inline key:value pairs while excluding prefixed formats (META:, SKIP:, TODO:, EVENT:)
- Added new tests for performance metrics, timing edge cases, and protocol timing to verify accurate timing capture and retry handling
- Expanded documentation in readme.hints.md to detail the updated timing implementation and parser fixes

2025-05-26 - 2.2.4 - fix(logging)

Improve performance metrics reporting and add local permissions configuration

- Add .claude/settings.local.json to configure allowed permissions for various commands
- Update tstest logging: compute average test duration from actual durations and adjust slowest test display formatting

2025-05-26 - 2.2.3 - fix(readme/ts/tstest.plugins)

Update npm package scope and documentation to use '@git.zone' instead of '@gitzone', and add local settings configuration.

- Changed npm package links and source repository URLs in readme from '@gitzone/tstest' to '@git.zone/tstest'.
- Updated comments in ts/tstest.plugins.ts to reflect the correct '@git.zone' scope.
- Added .claude/settings.local.json file with local permission settings.

2025-05-26 - 2.2.2 - fix(config)

Cleanup project configuration by adding local CLAUDE settings and removing redundant license files

- Added .claude/settings.local.json with updated permissions for CLI and build tasks
- Removed license and license.md files to streamline repository content

2025-05-26 - 2.2.1 - fix(repo configuration)

Update repository metadata to use 'git.zone' naming and add local permission settings

- Changed githost from 'gitlab.com' to 'code.foss.global' and gitscope from 'gitzone' to 'git.zone' in npmextra.json
- Updated npm package name from '@gitzone/tstest' to '@git.zone/tstest' in npmextra.json and readme.md
- Added .claude/settings.local.json with new permission configuration

2025-05-26 - 2.2.0 - feat(watch mode)

Add watch mode support with CLI options and enhanced documentation

- Introduce '--watch' (or '-w') and '--watch-ignore' CLI flags for automatic test re-runs
- Integrate @push.rocks/smartchok for file watching with 300ms debouncing
- Update readme.md and readme.hints.md with detailed instructions and examples for watch mode
- Add a demo test file (test/watch-demo/test.demo.ts) to illustrate the new feature
- Add smartchok dependency in package.json

2025-05-26 - 2.1.0 - feat(core)

Implement Protocol V2 with enhanced settings and lifecycle hooks

- Migrated to Protocol V2 using Unicode markers and structured metadata with new ts_tapbundle_protocol module
- Refactored TAP parser/emitter to support improved protocol parsing and error reporting
- Integrated global settings via tap.settings() and lifecycle hooks (beforeAll/afterAll, beforeEach/afterEach)
- Enhanced expect wrapper with diff generation for clearer assertion failures
- Updated test loader to automatically run 00init.ts for proper test configuration
- Revised documentation (readme.hints.md, readme.plan.md) to reflect current implementation status and remaining work

2025-05-25 - 2.0.0 - BREAKING CHANGE(protocol)

Introduce protocol v2 implementation and update build configuration with revised build order, new tspublish files, and enhanced documentation

- Added ts_tapbundle_protocol directory with isomorphic implementation for protocol v2
- Updated readme.hints.md and readme.plan.md to explain the complete replacement of the v1 protocol and new build process
- Revised build order in tspublish.json files across ts, ts_tapbundle, ts_tapbundle_node, and ts_tapbundle_protocol
- Introduced .claude/settings.local.json with updated permission settings for CLI and build tools

2025-05-24 - 1.11.5 - fix(tstest)

Fix timeout handling to correctly evaluate TAP results after killing the test process.

- Added call to evaluateFinalResult() after killing the process in runInNode to ensure final TAP output is processed.

2025-05-24 - 1.11.4 - fix(logging)

Improve warning logging and add permission settings file

- Replace multiple logger.error calls with logger.warning for tests running over 1 minute
- Add warning method in tstest logger to display warning messages consistently

- Introduce .claude/settings.local.json to configure allowed permissions

2025-05-24 - 1.11.3 - fix(tstest)

Add timeout warning for long-running tests and introduce local settings configuration

- Add .claude/settings.local.json with permission configuration for local development
- Implement a timeout warning timer that notifies when tests run longer than 1 minute without an explicit timeout
- Clear the timeout warning timer upon test completion
- Remove unused import of logPrefixes in tstest.classes.tstest.ts

2025-05-24 - 1.11.2 - fix(tstest)

Improve timeout and error handling in test execution along with TAP parser timeout logic improvements.

- In the TAP parser, ensure that expected tests are properly set when no tests are defined to avoid false negatives on timeout.
- Use smartshell's terminate method and fallback kill to properly stop the entire process tree on timeout.
- Clean up browser, server, and WebSocket instances reliably even when a timeout occurs.
- Minor improvements in log file filtering and error logging for better clarity.

2025-05-24 - 1.11.1 - fix(tstest)

Clear timeout identifiers after successful test execution and add local CLAUDE settings

- Ensure timeout IDs are cleared when tests complete to prevent lingering timeouts
- Add .claude/settings.local.json with updated permission settings for CLI commands

2025-05-24 - 1.11.0 - feat(cli)

Add new timeout and file range options with enhanced logfile diff logging

- Introduce --timeout option to safeguard tests from running too long
- Add --startFrom and --stopAt options to control the range of test files executed

- Enhance logfile organization by automatically moving previous logs and generating diff reports for failed or changed test outputs
- Update CLI argument parsing and internal timeout handling for both Node.js and browser tests

2025-05-24 - 1.10.2 - fix(tstest-logging)

Improve log file handling with log rotation and diff reporting

- Add .claude/settings.local.json to configure allowed shell and web operations
- Introduce movePreviousLogFiles function to archive previous log files when --logfile is used
- Enhance logging to generate error copies and diff reports between current and previous logs
- Add type annotations for console overrides in browser evaluations for improved stability

2025-05-23 - 1.10.1 - fix(tstest)

Improve file range filtering and summary logging by skipping test files outside the specified range and reporting them in the final summary.

- Introduce runSingleTestOrSkip to check file index against startFrom/stopAt values.
- Log skipped files with appropriate messages and add them to the summary.
- Update the logger to include total skipped files in the test summary.
- Add permission settings in .claude/settings.local.json to support new operations.

2025-05-23 - 1.10.0 - feat(cli)

Add --startFrom and --stopAt options to filter test files by range

- Introduced CLI options --startFrom and --stopAt in ts/index.ts for selective test execution
- Added validation to ensure provided range values are positive and startFrom is not greater than stopAt
- Propagated file range filtering into test grouping in tstest.classes.tstest.ts, applying the range filter across serial and parallel groups
- Updated usage messages to include the new options

2025-05-23 - 1.9.4 - fix(docs)

Update documentation and configuration for legal notices and CI permissions. This commit adds a new local settings file for tool permissions, refines the legal and trademark sections in the readme, and improves glob test files with clearer log messages.

- Added .claude/settings.local.json to configure permissions for various CLI commands
- Revised legal and trademark documentation in the readme to clarify company ownership and usage guidelines
- Updated glob test files with improved console log messages for better clarity during test discovery

2025-05-23 - 1.9.3 - fix(tstest)

Fix test timing display issue and update TAP protocol documentation

- Changed TAP parser regex to non-greedy pattern to correctly separate test timing metadata
- Enhanced readme.hints.md with detailed explanation of test timing fix and planned protocol upgrades
- Updated readme.md with improved usage examples for tapbundle and comprehensive test framework documentation
- Added new protocol design document (readme.protocol.md) and improvement plan (readme.plan.md) outlining future changes
- Introduced .claude/settings.local.json update for npm and CLI permissions
- Exported protocol utilities and added tapbundle protocol implementation for future enhancements

2025-05-23 - 1.9.2 - fix(logging)

Fix log file naming to prevent collisions and update logging system documentation.

- Enhance safe filename generation in tstest logging to preserve directory structure using double underscores.
- Update readme.hints.md to include detailed logging system documentation and behavior.
- Add .claude/settings.local.json with updated permissions for build tools.

2025-05-23 - 1.9.1 - fix(dependencies)

Update dependency versions and add local configuration files

- Bump @git.zone/tsbuild from ^2.5.1 to ^2.6.3
- Bump @types/node from ^22.15.18 to ^22.15.21
- Bump @push.rocks/smartexpect from ^2.4.2 to ^2.5.0
- Bump @push.rocks/smartfile from ^11.2.0 to ^11.2.3
- Bump @push.rocks/smartlog from ^3.1.1 to ^3.1.8
- Add .npmrc with npm registry configuration
- Add .claude/settings.local.json for local permissions

2025-05-16 - 1.9.0 - feat(docs)

Update documentation to embed tapbundle and clarify module exports for browser compatibility; also add CI permission settings.

- Embed tapbundle directly into tstest to simplify usage and ensure browser support.
- Update import paths in examples from '@push.rocks/tapbundle' to '@git.zone/tstest/tapbundle'.
- Revise the changelog to reflect version 1.8.0 improvements including enhanced test lifecycle hooks and parallel execution fixes.
- Add .claude/settings.local.json to configure CI-related permissions and tool operations.

2025-05-16 - 1.8.0 - feat(documentation)

Enhance README with detailed test features and update local settings for build permissions.

- Expanded the documentation to include tag filtering, parallel test execution groups, lifecycle hooks, snapshot testing, timeout control, retry logic, and test fixtures
- Updated .claude/settings.local.json to allow additional permissions for various build and test commands

2025-05-16 - 1.7.0 - feat(tstest)

Enhance tstest with fluent API, suite grouping, tag filtering, fixture & snapshot testing, and parallel execution improvements

- Updated npm scripts to run tests in verbose mode and support glob patterns with quotes
- Introduced tag filtering support (--tags) in the CLI to run tests by specified tags
- Implemented fluent syntax methods (tags, priority, retry, timeout) for defining tests and applying settings
- Added test suite grouping with describe(), along with beforeEach and afterEach lifecycle hooks
- Integrated a fixture system and snapshot testing via TapTools with base64 snapshot communication
- Enhanced TAP parser regex, error collection, and snapshot handling for improved debugging
- Improved parallel test execution by grouping files with a 'para__' pattern and running them concurrently

2025-05-15 - 1.6.0 - feat(package)

Revamp package exports and update permissions with an extensive improvement plan for test runner enhancements.

- Replaced 'main' and 'typings' in package.json with explicit exports for improved module resolution.
- Added .claude/settings.local.json to configure permissions for bash commands and web fetches.
- Updated readme.plan.md with a comprehensive roadmap covering enhanced error reporting, rich test metadata, nested test suites, and advanced test features.

2025-05-15 - 1.5.0 - feat(cli)

Improve test runner configuration: update test scripts, reorganize test directories, update dependencies and add local settings for command permissions.

- Updated package.json scripts to use pnpm and separate commands for tapbundle and tstest.
- Reorganized tests into dedicated directories (test/tapbundle and test/tstest) and removed deprecated test files.

- Refactored import paths and bumped dependency versions in tapbundle, tstest, and associated node utilities.
- Added .claude/settings.local.json to configure local permissions for bash and web fetch commands.
- Introduced ts/tspublish.json to define publish order.

2025-05-15 - 1.4.0 - feat(logging)

Display failed test console logs in default mode

- Introduce log buffering in TsTestLogger to capture console output for failed tests
- Enhance TapParser to collect and display error details when tests fail
- Update README and project plan to document log improvements for debugging

2025-05-15 - 1.3.1 - fix(settings)

Add local permissions configuration and remove obsolete test output log

- Added .claude/settings.local.json to configure allowed permissions for web fetch and bash commands
- Removed test-output.log to eliminate accidental commit of test artifacts

2025-05-15 - 1.3.0 - feat(logger)

Improve logging output and add --logfile support for persistent logs

- Add new .claude/settings.local.json with logging permissions configuration
- Remove obsolete readme.plan.md
- Introduce test/test.console.ts to capture and display console outputs during tests
- Update CLI in ts/index.ts to replace '--log-file' with '--logfile' flag
- Enhance TsTestLogger to support file logging, clean ANSI sequences, and improved JSON output
- Forward TAP protocol logs to testConsoleOutput in TapParser for better console distinction

2025-05-15 - 1.2.0 - feat(logging)

Improve logging output, CLI option parsing, and test report formatting.

- Added a centralized TsTestLogger with support for multiple verbosity levels, JSON output, and file logging (TODO).
- Integrated new logger into CLI parsing, TapParser, TapCombinator, and TsTest classes to ensure consistent and structured output.
- Introduced new CLI options (--quiet, --verbose, --no-color, --json, --log-file) for enhanced user control.
- Enhanced visual design with progress indicators, detailed error aggregation, and performance summaries.
- Updated documentation and logging code to align with improved CI/CD behavior, including skipping non-CI tests.

2025-05-15 - 1.1.0 - feat(cli)

Enhance test discovery with support for single file and glob pattern execution using improved CLI argument detection

- Detect execution mode (file, glob, directory) based on CLI input in ts/index.ts
- Refactor TestDirectory to load test files using SmartFile for single file and glob patterns
- Update TsTest to pass execution mode and adjust test discovery accordingly
- Bump dependency versions for typedserver, tsbundle, tapbundle, and others
- Add .claude/settings.local.json for updated permissions configuration

2025-01-23 - 1.0.96 - fix(TsTest)

Fixed improper type-check for promise-like testModule defaults

- Corrected the type-check for promise-like default exports in test modules
- Removed unnecessary setTimeout used for async execution

2025-01-23 - 1.0.95 - fix(core)

Fix delay handling in Chrome test execution

- Replaced smartdelay.delayFor with native Promise-based delay mechanism in runInChrome method.

2025-01-23 - 1.0.94 - fix(TsTest)

Fix test module execution by ensuring promise resolution delay

- Added a delay to ensure promise resolution when dynamically importing test modules in the runInChrome method.

2025-01-23 - 1.0.93 - fix(tstest)

Handle globalThis.tapPromise in browser runtime evaluation

- Added support for using globalThis.tapPromise in the browser evaluation logic.
- Added log messages to indicate the usage of globalThis.tapPromise.

2025-01-23 - 1.0.92 - fix(core)

Improve error logging for test modules without default promise

- Added logging to display the exported test module content when it does not export a default promise.

2025-01-23 - 1.0.91 - fix(core)

Refactored tstest class to enhance promise handling for test modules.

- Removed .gitlab-ci.yml configuration file.
- Updated package.json dependency versions.
- Added a condition to handle promiselike objects in tests.

2024-04-18 - 1.0.89 to 1.0.90 - Enhancements and Bug Fixes

Multiple updates and fixes have been made.

- Updated core components to enhance stability and performance.

2024-03-07 - 1.0.86 to 1.0.88 - Core Updates

Continued improvements and updates in the core module.

- Applied critical fixes to enhance core stability.

2024-01-19 - 1.0.85 to 1.0.89 - Bug Fixes

Series of core updates have been implemented.

- Addressed known bugs and improved overall system functionality.

2023-11-09 - 1.0.81 to 1.0.84 - Maintenance Updates

Maintenance updates focusing on core reliability.

- Improved core module through systematic updates.
- Strengthened system robustness.

2023-08-26 - 1.0.77 to 1.0.80 - Critical Fixes

Critical fixes implemented in core functionality.

- Enhanced core processing to fix existing issues.

2023-07-13 - 1.0.75 to 1.0.76 - Stability Improvements

Stability enhancements and minor improvements.

- Focused on ensuring a stable operational core.

2022-11-08 - 1.0.73 to 1.0.74 - Routine Fixes

Routine core fixes to address reported issues.

- Addressed minor issues in the core module.

2022-08-03 - 1.0.71 to 1.0.72 - Core Enhancements

Enhancements applied to core systems.

- Tweaked core components for enhanced reliability.

2022-05-04 - 1.0.69 to 1.0.70 - System Reliability Fixes

Fixes targeting the reliability of the core systems.

- Improved system reliability through targeted core updates.

2022-03-17 - 1.0.65 to 1.0.68 - Major Core Updates

Major updates and bug fixes delivered for core components.

- Enhanced central operations through key updates.

2022-02-15 - 1.0.60 to 1.0.64 - Core Stability Improvements

Focused updates on core stability and performance.

- Reinforced stability through systematic core changes.

2021-11-07 - 1.0.54 to 1.0.59 - Core Fixes and Improvements

Multiple core updates aimed at fixing and improving the system.

- Addressed outstanding bugs and improved performance in the core.

2021-08-20 - 1.0.50 to 1.0.53 - Core Functionality Updates

Continued updates to improve core functionality and user experience.

- Implemented essential core fixes to enhance user experience.

2020-10-01 - 1.0.44 to 1.0.49 - Core System Enhancements

Critical enhancements to core systems.

- Improved core operations and tackled existing issues.

2020-09-29 - 1.0.40 to 1.0.43 - Essential Fixes

Series of essential fixes for the core system.

- Rectified known issues and bolstered core functionalities.

2020-07-10 - 1.0.35 to 1.0.39 - Core Function Fixes

Focused improvements and fixes for critical components.

- Addressed critical core functions to boost system performance.

2020-06-01 - 1.0.31 to 1.0.34 - Core Updates

Updates to maintain core functionality efficacy.

- Fixed inefficiencies and updated essential components.

2019-10-02 - 1.0.26 to 1.0.29 - Core Maintenance

Regular maintenance and updates for core reliability.

- Addressed multiple core issues and enhanced system stability.

2019-05-28 - 1.0.20 to 1.0.25 - Core Improvements

General improvements targeting core functionalities.

- Made systematic improvements to core processes.

2019-04-08 - 1.0.16 to 1.0.19 - Bug Squashing

Resolved numerous issues within core operations.

- Fixed and optimized core functionalities for better performance.

2018-12-06 - 1.0.15 - Dependency Updates

Updates aimed at improving dependency management.

- Ensured dependencies are up-to-date for optimal performance.

2018-08-14 - 1.0.14 - Test Improvement

Major improvements in testing mechanisms and logging.

- Improved test results handling for accuracy and reliability.
- Enhanced logging features for increased clarity.

2018-08-04 - 1.0.1 to 1.0.13 - Initial Implementation and Fixes

Initial release and critical updates focusing on core stability and functionality.

- Implemented core components and established initial system structure.
- Addressed key bugs and enhanced initial functionality.