readme.md for @git.zone/tsdoc

AI-Powered Documentation & Commit Intelligence for TypeScript Projects □□

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.

Install

```
# Global installation (recommended for CLI usage)
pnpm add -g @git.zone/tsdoc

# Or use with npx (no install needed)
npx @git.zone/tsdoc

# Or install locally as a project dependency
pnpm add @git.zone/tsdoc
```

Usage

`@git.zone/tsdoc` is a TypeScript documentation powerhouse that combines traditional TypeDoc API docs with AI-powered documentation workflows. It uses OpenAI models via `@push.rocks/smartai` and autonomous agents via `@push.rocks/smartagent` to generate READMEs, project descriptions, keywords, and semantic commit messages — all by intelligently exploring your project's codebase with scoped filesystem tools.

CLI Commands

Command	Description
`tsdoc`	□□ Auto-detects project type and runs TypeDoc
`tsdoc aidoc`	□□ Generates AI-powered README + description/keywords
`tsdoc readme`	□□ Generates AI-powered README only
`tsdoc description`	□□ Generates AI-powered description and keywords only
`tsdoc commit`	□□ Generates a semantic commit message from uncommitted changes
`tsdoc typedoc`	□□ Generates traditional TypeDoc API documentation

□□ AI-Powered Documentation (`aidoc`)

The `aidoc` command is the all-in-one workflow that combines README generation and description/keyword generation:

```
# In your project root
tsdoc aidoc
```

This will:

1. Spin up an AI agent with read-only filesystem access scoped to your project
2. The agent autonomously explores your project structure, reads source files, and understands the API
3. Generate a comprehensive `readme.md` with install instructions, usage examples, and architecture overview
4. Update `package.json` and `.smartconfig.json` with an AI-generated description and keywords

You can also run these steps individually:

```
# Generate only the README
tsdoc readme

# Generate only the description and keywords
tsdoc description
```

□□ Smart Commit Messages (`commit`)

The `commit` command analyzes your uncommitted changes and produces a structured commit object following Conventional Commits:

```
tsdoc commit
```

Output is a JSON object:

```
{
  "recommendedNextVersionLevel": "feat",
  "recommendedNextVersionScope": "core",
  "recommendedNextVersionMessage": "add smart diff processing for large changesets",
  "recommendedNextVersionDetails": [
    "implemented intelligent diff sampling with head/tail extraction",
    "added file prioritization by importance score"
  ],
  "recommendedNextVersion": "1.13.0",
  "changelog": "# Changelog\n\n## 2026-03-24 - 1.13.0 - core\n..."
}
```

Under the hood, the commit flow:

- **Excludes noise**: Lock files, build artifacts (`dist/`, `dist_*/`), IDE directories, caches, and source maps are filtered out before processing
- **Prioritizes what matters**: Source files rank higher than test files, which rank higher than config, docs, and build artifacts
- **Handles large diffs gracefully**: The `DiffProcessor` categorizes files by size — small files (< 300 lines) are included in full, medium files (< 800 lines) get head/tail sampling, and large files are metadata-only
- **Respects token budgets**: Dynamically calculates available tokens based on the model's context limit minus overhead
- **Auto-generates changelogs**: If no `changelog.md` exists, one is created from the full git history

□□ TypeDoc Generation (`typedoc`)

For traditional API documentation:

```
# Generate to default ./public directory
tsdoc typedoc

# Generate to a specific subdirectory
```

```
tsdoc typedoc --publicSubdir docs
```

TypeDoc generation auto-detects your source directories (`ts/` and `ts_web/`) and creates a temporary tsconfig for compilation.

□□ Monorepo Support

When generating READMEs, tsdoc automatically detects monorepo submodules via `@git.zone/tspublish` conventions. Each submodule directory containing a `tspublish.json` gets its own generated README with the legal section appended.

Programmatic API

You can use tsdoc programmatically in your own tools:

```
import { AiDoc } from '@git.zone/tsdoc';

const aidoc = new AiDoc();

// Initialize — prompts for OpenAI token on first run, then persists it
await aidoc.start();

// Generate a comprehensive README for a project
const readmeContent = await aidoc.buildReadme('/path/to/project');

// Generate description and keywords, updating package.json and .smartconfig.json
await aidoc.buildDescription('/path/to/project');

// Generate a structured commit message object from uncommitted changes
const commitObj = await aidoc.buildNextCommitObject('/path/to/project');
console.log(commitObj.recommendedNextVersionLevel); // 'fix' | 'feat' | 'BREAKING CHANGE'
console.log(commitObj.recommendedNextVersionMessage);
console.log(commitObj.changelog);

// Get gathered project files (package.json, source files, tests, config)
const context = await aidoc.getProjectContext('/path/to/project');

// Get token count for a project's context
const tokenCount = await aidoc.getProjectContextTokenCount('/path/to/project');
```

```
// Estimate tokens in arbitrary text
const tokens = aidoc.countTokens('some text here');

await aidoc.stop();
```

You can also pass the OpenAI token directly via the constructor:

```
const aidoc = new AiDoc({ OPENAI_TOKEN: 'sk-...' });
await aidoc.start();
```

Configuration

OpenAI Token

An OpenAI API key is required for all AI features. It can be provided in three ways (checked in order):

1. **Environment variable**: Set `OPENAI_TOKEN` in your environment or `.env` file
2. **Constructor argument**: Pass `{ OPENAI_TOKEN: 'sk-...' }` to `new AiDoc()`
3. **Interactive prompt**: On first run, tsdoc will prompt for the token and persist it

The token is persisted at `~/.smartconfig/kv/@git.zone/tsdoc.json` for subsequent runs.

.smartconfig.json

tsdoc uses `.smartconfig.json` for project metadata. The `tsdoc` key holds legal information that gets appended to generated READMEs:

```
{
  "tsdoc": {
    "legal": "\n## License and Legal Information\n\n..."
  },
  "gitzone": {
    "module": {
      "githost": "gitlab.com",
      "gitscope": "gitzone",
      "gitrepo": "tsdoc",
```

```
      "npmPackagename": "@git.zone/tsdoc",
      "description": "...",
      "keywords": ["..."]
    }
  }
}
```

The `description` command writes updated description/keywords to both `gitzone.module` in `.smartconfig.json` and to `package.json`.

Architecture

Core Components

```
@git.zone/tsdoc
├── AiDoc                # Main orchestrator — manages AI model, delegates to task classes
├── TypeDoc              # Traditional TypeDoc API documentation generation
├── ProjectContext       # Gathers project files (package.json, source, tests, config)
├── DiffProcessor        # Intelligent git diff processing with prioritization & sampling
├── Readme               # AI agent-driven README generation with filesystem tools
├── Commit               # AI agent-driven commit message generation with diff analysis
├── Description          # AI agent-driven description and keyword generation
└── CLI                  # Command-line interface built on @push.rocks/smartcli
```

□□ AI Agent Architecture

Each documentation task (readme, commit, description) runs an autonomous AI agent via `@push.rocks/smartagent`'s `runAgent()`:

1. **System prompt** defines the agent's role, constraints, and output format
2. **Filesystem tools** give the agent scoped, read-only access to the project directory
3. **Autonomous exploration** — the agent decides which files to read, in what order
4. **Structured output** — README markdown, commit JSON, or description JSON

The agents use `@push.rocks/smartai`'s `getModel()` to create a language model instance backed by OpenAI.

⚡ Diff Processing Pipeline

The `DiffProcessor` handles large git diffs without blowing up token budgets:

File Category	Threshold	Treatment
Small	< 300 lines changed	Included in full
Medium	< 800 lines changed	Head (75 lines) + tail (75 lines) sampling
Large	≥ 800 lines changed	Metadata only (filepath + stats)

Files are scored by importance:

- **100** — Source files (`src/`, `lib/`, `app/`, `components/`, `pages/`, `api/`)
- **80** — Test files (`test/`, `*.test.ts`, `*.spec.ts`)
- **70** — Interface/type files, entry points (`index.ts`, `mod.ts`)
- **60** — Configuration files (`.json`, `.yaml`, `.config.ts`)
- **40** — Documentation (`.md`, `.txt`)
- **10** — Build artifacts (`dist/`, `build/`, `.next/`)

Token budget is calculated dynamically: `context_limit - safety_margin - overhead - prompt_size`.

Requirements

- **Node.js** >= 18
- **TypeScript** project with a `ts/` source directory
- **OpenAI API key** for AI features

License and Legal Information

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

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

Trademarks

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

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

Company Information

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

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

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

Revision #5
Created 2026-03-28 10:49:27 UTC by foss.global Team
Updated 2026-03-28 12:15:24 UTC by foss.global Team