@git.zone/tspublish

publish multiple, concise and small packages from monorepos

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

readme.md for @git.zone/tspublish

> **Effortlessly publish multiple TypeScript packages from your monorepo**

npm version License: MIT

What is tspublish?

`@git.zone/tspublish` is a powerful CLI tool and library for managing and publishing multiple TypeScript packages from a monorepo. It automates the tedious parts of package publishing — discovery, dependency resolution, building, version validation, and multi-registry publishing — while giving you full control over the process. Whether you're maintaining a suite of microservices, a component library, or any collection of related packages, tspublish makes your life dramatically easier.

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.

Key Features

- **Automatic Package Discovery** — Scans your monorepo for publishable `ts*` directories
- **Beautiful CLI Output** — Color-coded logging with progress bars and status indicators
- **Version Collision Detection** — Prevents accidental overwrites by checking the registry first

- □□ **Build Integration** — Automatically compiles TypeScript before publishing via `@git.zone/tsbuild`
- □□ **Smart Dependency Management** — Inherits dependency versions from your monorepo's `package.json`
- □□ **Multi-Registry Support** — Publish to npm, GitHub Packages, Gitea, or private registries
- □□ **Base Registry Inheritance** — Use `useBase` / `extendBase` to inherit registries from `.smartconfig.json`
- ⚡ **CLI Binary Support** — Automatically generates `cli.js` wrappers for publishable CLI packages
- □□ **Clean Builds** — Creates isolated `dist_publish_*` directories for each package

□□ Installation

```
# Using pnpm (recommended)
pnpm add -D @git.zone/tspublish

# Global installation for CLI usage
pnpm add -g @git.zone/tspublish
```

□□ Quick Start

1□ Structure Your Monorepo

Organize your packages using directories that start with `ts`:

```
my-awesome-monorepo/
├── package.json            # Main monorepo package.json (version is inherited)
├── tsconfig.json           # Shared TypeScript config
├── .smartconfig.json       # Optional: base registry configuration
├── ts_core/                # Core package
│   ├── index.ts            # Entry point
│   ├── readme.md           # Package-specific documentation
│   └── tspublish.json      # Publishing configuration
├── ts_utils/               # Utilities package
│   ├── index.ts
│   ├── readme.md
```

```
│   └── tspublish.json
└── ts_cli/                # CLI package
    ├── index.ts
    ├── readme.md
    └── tspublish.json
```

2️⃣ Configure Each Package

Create a `tspublish.json` in each publishable directory:

```
{
  "name": "@myorg/core",
  "order": 1,
  "dependencies": [
    "@push.rocks/smartpromise",
    "@push.rocks/smartfile"
  ],
  "registries": [
    "registry.npmjs.org:public"
  ],
  "bin": []
}
```

Configuration Options

Field	Type	Description
`name`	`string`	The published package name (e.g., `@myorg/core`)
`order`	`number`	Build order for interdependent packages (lower builds first)
`dependencies`	`string[]`	Dependencies to include — versions are resolved from the monorepo's `package.json`
`registries`	`string[]`	Target registries with access level (format: `"url:accessLevel"`)
`bin`	`string[]`	CLI executable names (generates a `cli.js` wrapper automatically)

3️⃣ Publish

```
# From your monorepo root
npx tspublish
```

That's it! tspublish will discover all `ts*` directories containing `tspublish.json`, build them in order, validate versions against the registry, and publish.

□□ Advanced Usage

Registry Configuration

TSPublish offers three approaches for configuring target registries:

1. Explicit Registries

Define specific registries directly in `tspublish.json`:

```
{
  "registries": [
    "registry.npmjs.org:public",
    "npm.pkg.github.com:private"
  ]
}
```

The format is `"registryUrl:accessLevel"` where `accessLevel` is `public` or `private`.

2. Use Base Configuration (`useBase`)

Inherit registries from your project's `.smartconfig.json` (managed by `@git.zone/cli`):

```
{
  "registries": ["useBase"]
}
```

This reads from `.smartconfig.json` at the key `@git.zone/cli.release.registries`.

3. Extend Base Configuration (`extendBase`)

Start with base registries and add or remove specific ones:

```
{
  "registries": [
    "extendBase",
    "custom-registry.example.com:public",
    "-https://registry.npmjs.org"
  ]
}
```

The `-` prefix excludes a registry from the base configuration. All other entries (besides `extendBase`) are added.

Empty Registries

If `registries` is an empty array `[]`, the package will be built but **not published** — useful for internal-only packages that other packages depend on.

Build Order Management

When packages depend on each other, use the `order` field to control build sequence:

```
// ts_core/tspublish.json — builds first
{
  "name": "@myorg/core",
  "order": 1,
  "dependencies": [],
  "registries": ["useBase"],
  "bin": []
}

// ts_utils/tspublish.json — builds second, depends on core
{
  "name": "@myorg/utils",
  "order": 2,
  "dependencies": ["@myorg/core"],
  "registries": ["useBase"],
  "bin": []
}
```

CLI Binary Packages

For packages that ship CLI tools, specify the binary names in the `bin` array:

```
{
  "name": "@myorg/cli",
  "order": 3,
  "dependencies": ["commander", "@myorg/core"],
  "registries": ["registry.npmjs.org:public"],
  "bin": ["my-cli", "my-tool"]
}
```

TSPublish will:

1. Fetch the standard `cli.js` template from the `@git.zone/cli` assets repository
2. Adjust the import path to point to the correct `dist_*` folder
3. Configure the `bin` field in the generated `package.json`

Programmatic Usage

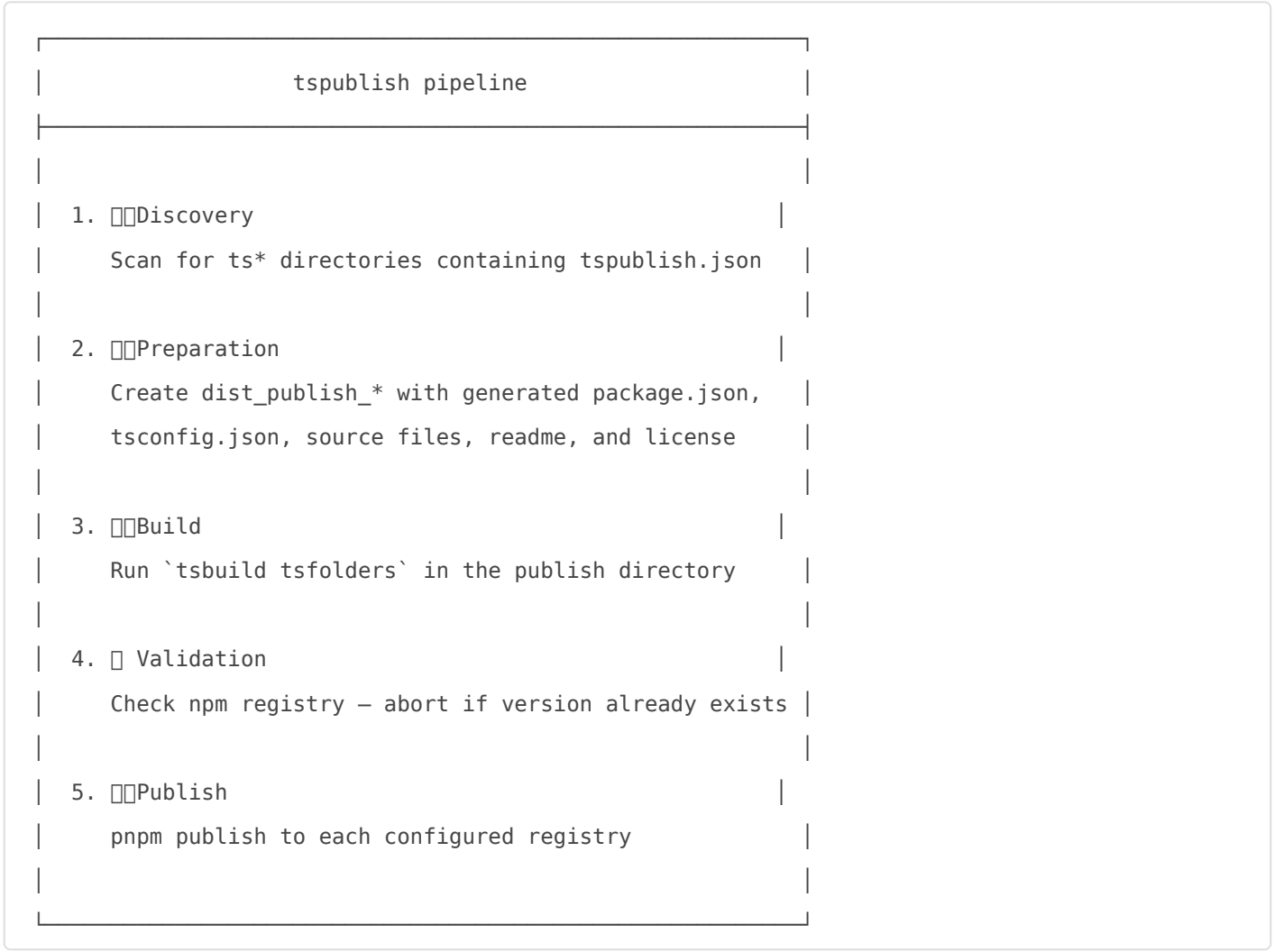
```
import { TsPublish } from '@git.zone/tspublish';

const publisher = new TsPublish();
await publisher.publish(process.cwd());
```

Custom Publishing Pipeline

```
import { TsPublish, PublishModule } from '@git.zone/tspublish';

const publisher = new TsPublish();
const modules = await publisher.getModuleSubDirs('./my-monorepo');

for (const [name, config] of Object.entries(modules)) {
  const mod = new PublishModule(publisher, {
    monoRepoDir: './my-monorepo',
    packageSubFolder: name,
  });

  await mod.init();                    // Resolve deps, validate version
  await mod.createPublishModuleDir(); // Scaffold dist_publish_* directory
  await mod.build();                   // Compile TypeScript
```

```
  await mod.publish();               // Publish to registries
}
```

□□ How It Works



For each discovered module, tspublish:

1. **Discovers** all directories starting with `ts` that contain a `tspublish.json`
2. **Resolves dependencies** from the monorepo's `package.json`, using the monorepo version for packages not found in `dependencies`
3. **Creates an isolated publish directory** (`dist_publish_<folder>`) with a generated `package.json`, `tsconfig.json`, source code copy, readme, and license
4. **Builds** the package using `pnpm run build` (which calls `tsbuild tsfolders`)
5. **Validates** against the target registry — if the name+version already exists, it exits with an error
6. **Publishes** to each configured registry via `pnpm publish`

□□ API Reference

TsPublish

```
class TsPublish {
  /** Publish all discovered modules in a monorepo directory */
  async publish(monorepoDirPath: string): Promise<void>;

  /** Discover and return all publishable modules with their tspublish.json configs */
  async getModuleSubDirs(dirPath: string): Promise<Record<string, ITsPublishJson>>;
}
```

PublishModule

```
class PublishModule {
  /** Initialize: resolve dependencies, validate version against registry */
  async init(): Promise<void>;

  /** Create the dist_publish_* directory with all necessary files */
  async createPublishModuleDir(): Promise<void>;

  /** Build the TypeScript package */
  async build(): Promise<void>;

  /** Publish to all configured registries */
  async publish(): Promise<void>;
}
```

ITsPublishJson

```
interface ITsPublishJson {
  name: string;           // Published package name
  order: number;          // Build sequence (lower = earlier)
  dependencies: string[]; // Dependencies to include from monorepo
  registries: string[];   // Target registries ("url:accessLevel", "useBase", or "extendBase")
```

```
  bin: string[];          // CLI binary names
}
```

GiteaAssets

```
class GiteaAssets {
  constructor(options: { giteaBaseUrl: string; token?: string });

  /** Fetch files from a Gitea repository directory */
  async getFiles(owner: string, repo: string, directory: string, branch?: string):
Promise<IRepoFile[]>;

  /** Get the standard cli.js entry file template */
  async getBinCliEntryFile(): Promise<IRepoFile>;
}
```

□□ Troubleshooting

Problem	Solution
"Package X already exists with version Y"	Bump the version in your monorepo's `package.json`
No publish modules found	Ensure directories start with `ts` and contain a valid `tspublish.json`
Build failures	Check TypeScript errors — tspublish runs `tsbuild tsfolders` in the generated directory
useBase/extendBase error	Ensure `.smartconfig.json` has registries at `@git.zone/cli.release.registries`
Missing dependency versions	Add the dependency to your monorepo's `package.json` `dependencies` field

License and Legal Information

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/tspublish

2026-03-24 - 1.11.4 - fix(config)

rename npmextra configuration to .smartconfig.json and align publish packaging with updated config handling

- replaces packaged config file references from npmextra.json to .smartconfig.json
- updates publish module typings and null checks for tsPublishJson, package info, and file paths
- bumps build and runtime dependencies to newer compatible versions
- simplifies tsconfig.json by removing unused baseUrl and paths settings
- refreshes README content to document current CLI, registry, and package publishing behavior

2026-03-24 - 1.11.3 - fix(config)

replace npmextra with smartconfig for base registry resolution

- Switch the base registry source from npmextra.json to smartconfig.json for publish configuration.
- Update plugin imports and runtime config loading to use @push.rocks/smartconfig.
- Adjust packaged files and error messages to reference smartconfig.json consistently.

2026-03-05 - 1.11.2 - fix(deps)

bump @push.rocks/smartshell dependency to ^3.3.7 and update package version to 1.11.1

- package.json: version 1.11.0 -> 1.11.1
- package.json: @push.rocks/smartshell ^3.3.0 -> ^3.3.7
- ts/00_commitinfo_data.ts: version updated to 1.11.1
- changelog.md: added entry documenting the dependency bump and patch release

2026-03-05 - 1.11.1 - fix(deps)

bump @push.rocks/smartshell dependency to ^3.3.7

- package.json: @push.rocks/smartshell ^3.3.0 -> ^3.3.7
- Only dependency version updated (patch-level)

2025-12-16 - 1.11.0 - feat(publish)

add registry resolution (useBase/extendBase) and migrate filesystem operations to async SmartFs; improve publish flow and docs

- Add resolveRegistries supporting 'useBase' and 'extendBase' and explicit registries; reads base registries from npmextra.json at @git.zone/cli.release
- Migrate sync smartfile APIs to async @push.rocks/smartfs and expose smartfs and npmextra via plugins
- Ensure publish directory is recreated cleanly and use async file copy/read/write for package, tsconfig, readme and license
- Handle empty registries by skipping publish with a warning and throw a clear error if useBase/extendBase is used but no base registries configured
- Publish now normalizes registry URLs and supports accessLevel per-registry when running pnpm publish
- Update README: registry configuration docs, issue reporting/security section and various wording/formatting improvements; bump several dependencies/devDependencies

2025-12-15 - 1.10.4 - fix(.serena)

cleanup: remove .serena assistant memories, cache and project config

- Removed multiple .serena/memories markdown files (code_style_conventions.md, logging_improvements_2025.md, project_overview.md, smartrequest_api_update_2025.md, suggested_commands.md, task_completion_checklist.md) — these were assistant metadata/notes
- Removed .serena/project.yml (project configuration for the assistant)
- Removed .serena/cache/typescript/document_symbols_cache_v23-06-25.pkl (generated symbol cache)

2025-08-18 - 1.10.3 - fix(devDependencies)

Bump development dependencies and add local Claude settings

- Bumped @git.zone/tsbuild from ^2.6.4 to ^2.6.6 in package.json
- Bumped @push.rocks/smartnpm from ^2.0.4 to ^2.0.6 in package.json
- Added .claude/settings.local.json for local Claude permissions/configuration

2025-08-18 - 1.10.2 - fix(dependencies)

Update dependency versions and add local Claude settings

- Bump devDependency @git.zone/tstest from ^2.3.2 to ^2.3.4
- Bump dependency @push.rocks/smartfile from ^11.2.5 to ^11.2.7
- Bump dependency @push.rocks/smartrequest from ^4.2.1 to ^4.2.2
- Bump dependency @push.rocks/smartshell from ^3.2.3 to ^3.3.0
- Add .claude/settings.local.json (local Claude permissions/config)

2025-08-08 - 1.10.1 - fix(core)

Refactor smartrequest usage, update dependency versions, and improve documentation and tests

- Refactored getFiles method in classes.giteaassets to use SmartRequest builder and handle branch query parameters.
- Updated package.json dependency versions for smartfile, smartlog, tsbuild, tsbundle, and tstest.
- Added pnpm-workspace.yaml configuration for onlyBuiltDependencies.
- Enhanced readme with detailed usage instructions, CI/CD integration examples, and advanced feature descriptions.
- Updated test files to import tapbundle from @git.zone/tstest instead of @push.rocks/tapbundle.
- Added .claude/settings.local.json for managing local permissions.

2025-08-08 - 1.10.0 - feat(logging)

Enhance logging and module publishing with color-coded output, progress tracking, and improved CLI startup

- Refactored logging to introduce color-coded symbols and helper functions (logInfo, logWarn, logSuccess, logBuild, logPublish, etc.)
- Added visual components such as headers, separators, and progress indicators for better operational visibility
- Integrated enhanced logging into module publishing and CLI startup, replacing generic console logs
- Updated various configuration and documentation files to reflect new code style conventions and dependency updates

2025-01-02 - 1.9.1 - fix(publishmodule)

Fix incorrect CLI script path during publish module creation

- Updated the `createBinCliSetup` method to correctly adjust the CLI script path.
- Replaced path in base64-decoded CLI file content from './dist_ts/index.js' to './dist_<packageSubFolder>/index.js'.

2025-01-02 - 1.9.0 - feat(core)

Refactor gitea asset handling and module initialization

- Introduced GiteaAssets class to handle gitea asset fetching.
- Updated TsPublish and PublishModule classes to use GiteaAssets.
- Fixed queryParams in getFiles method of GiteaAssets.

2025-01-01 - 1.8.0 - feat(core)

Added GiteaAssets class for managing files in Gitea repositories

- Introduced GiteaAssets class to handle file retrieval from Gitea repositories.
- Added tests for GiteaAssets implementation.
- Updated plugins module to include smartrequest for HTTP requests.

2024-11-05 - 1.7.7 - fix(core)

Fix dependency resolution in package initialization

- Corrected the resolution of dependencies from tspublish.json against monorepo's package.json.
- Ensures unlisted dependencies in monorepo's package.json default to its version.

2024-11-05 - 1.7.6 - fix(tspublish)

Fix the logging of the number of found publish modules

- Corrected the way the number of publish modules is logged by using Object.keys(publishModules).length instead of publishModules.length.

2024-11-05 - 1.7.5 - fix(core)

Fix issue with tspublish.json name validation in TsPublish class

- Resolved incorrect JSON parsing and validation for 'name' property in tspublish.json in the TsPublish.publish method.
- Removed redundant JSON parse from plugin.smartfile.fs.toStringSync in publish method.

2024-11-05 - 1.7.4 - fix(classes.tspublish)

Refactor getModuleSubDirs method to streamline name validation for publish modules

- Moved the check for the presence of the 'name' field in tspublish.json from getModuleSubDirs to the publish method.
- Added log warning and continue flow if 'name' is not found during the publish process.

2024-11-05 - 1.7.3 - fix(TsPublish)

Add validation for tspublish.json name field

- Ensure that the tspublish.json file contains a valid name field before processing.
- Log a warning message if the name is not found in tspublish.json.

2024-11-05 - 1.7.2 - fix(project)

Fixed minor formatting issues and improved code consistency.

- Added missing semicolons for consistency
- Improved indentation in various files for better readability
- Corrected usage of newlines and whitespace across the codebase

2024-11-05 - 1.7.1 - fix(core)

Implement error handling for missing publish module directories

- Improved logging for package publish steps
- Enhanced CLI feedback messages during the publishing process
- Restructured package.json to ensure proper dependencies are published

2024-11-05 - 1.7.0 - feat(core)

Enhanced tspublish with ordered compilation and updated dependencies

- Added 'order' property to ITsPublishJson interface to ensure project compilation order.
- Updated development dependencies: @git.zone/tsbuild, @git.zone/tsbundle, @git.zone/tsrun, and @types/node.

2024-10-28 - 1.6.0 - feat(classes.publishmodule)

Added copying of readme and license files to publish directory

- Enhanced the createPublishModuleDir method in PublishModule class to copy the 'readme.md' and 'license' files to the publish directory.

2024-10-28 - 1.5.5 - fix(core)

Handled non-existent package in publish module to avoid errors

- Added error handling in TsPublish for packages not yet existing in the registry.

2024-10-28 - 1.5.4 - fix(core)

Fix issues with path keys in tsconfig and logger setup in logging.ts.

- Corrected the iteration over paths in the createTsconfigJson method of PublishModule.
- Fixed logger setup by ensuring console is enabled in logging.ts.

2024-10-28 - 1.5.3 - fix(core)

Fix incorrect logging and directory preparation

- Corrected logging to accurately report the number of detected publish modules.
- Ensured the publish directory is emptied before creating package.json.

2024-10-28 - 1.5.2 - fix(core)

Add logging for found publish modules

- Added console logging to display the count and list of discovered publish modules during the publish process.
- Included a startup log message indicating the beginning of the tspublish process.

2024-10-28 - 1.5.1 - fix(core)

Fixes handling of undefined paths in tsconfig generation.

- Added a null check for `paths` in the original tsconfig before modifying it.
- Enhanced testing by adding a test case for creating a TsPublish instance.

2024-10-28 - 1.5.0 - feat(classes.publishmodule)

Add method to create and write tsconfig.json during publish module setup

- Introduced createTsconfigJson method in PublishModule class to generate a tsconfig.json for each publishable module.
- Modified createPublishModuleDir method to include writing of tsconfig.json file.

2024-10-26 - 1.4.0 - feat(core)

Refactor directory reading and module discovery for publishing process

- Renamed 'readDirectory' method to 'getModuleSubDirs' for clarity in describing function purpose.
- Enhanced 'getModuleSubDirs' to return module information including parsed 'tspublish.json' data for each module.
- Introduced new 'interfaces' directory to define TypeScript interfaces like 'ITsPublishJson'.

2024-10-23 - 1.3.3 - fix(core)

Fix logging mechanism on existing package version check

- Changed the error handling mechanism when a package with the same version already exists to use logger and process exit instead of throwing an error.

2024-10-23 - 1.3.2 - fix(core)

Corrected file patterns in dynamically created package.json files.

- Fixed incorrect file pattern from 'ts_web//' ***to*** **'*ts_*//*' in package.json creation process to include all subdirectories starting with 'ts'.

2024-10-23 - 1.3.1 - fix(classes.publishmodule)

Fix template string in createPackageJson method for export path

- Corrected the syntax for template string in the exports path of created package.json

2024-10-21 - 1.3.0 - feat(core)

Add support for multiple registries in the publish process

- Updated the PublishModule class to handle multiple registries for publishing.
- Refactored the handling of tspublish.json by incorporating it into the PublishModule options.
- Implemented logic to parse registry access level and apply it during publication.

2024-10-21 - 1.2.4 - fix(publishmodule)

Fix syntax errors and improve formatting in classes.publishmodule.ts

- Fixed missing semicolons in multiple locations for improved syntax correctness.
- Improved the formatting for better code readability.
- Added --no-git-checks flag to the pnpm publish command.

2024-10-21 - 1.2.3 - fix(logs)

Improve logging mechanism with structured logs for publish process

- Enhanced log messages to provide more clarity during module publishing.

- Ensured logging captures steps of publish and init process in TsPublish and PublishModule classes respectively.

2024-10-21 - 1.2.3 - fix(classes.publishmodule)

Add missing 'type: module' to dynamically generated package.json

- Ensure that the 'type: module' field is included in each dynamically generated package.json file for consistent module handling.

2024-10-21 - 1.2.3 - fix(classes.publishmodule)

Add missing 'type: module' to dynamically generated package.json

- Ensure that the 'type: module' field is included in each dynamically generated package.json file for consistent module handling.

2024-10-21 - 1.2.2 - fix(publishmodule)

Fix bug in package.json creation for publish module

- Fixed an issue where `package.json` was not being written to the publish module directory.

2024-10-21 - 1.2.1 - fix(package.json)

Ensure bin field is properly restructured

- Fixed the structure of the package.json to ensure the bin field is accurately set.

2024-10-21 - 1.2.0 - feat(core)

Enhance package publication workflow with dependency handling and CLI improvements.

- Updated package description and keywords in package.json and npmextra.json.
- Integrated dependency extraction from root package.json into sub-package tspublish.json during initialization.
- Added build and publish script executions for each submodule.
- Improved CLI documentation and usage guidance in readme.md.

2024-10-21 - 1.1.0 - feat(core)

Add runCli function to execute TsPublish process

- Introduced runCli function to start publish workflow using TsPublish class.
- Enhanced the process to read directory and create publish module directories.
- Improved logging for better tracking of found publish modules.

2024-10-21 - 1.0.1 - Initial Release

Initial release of the project.

- First version of the codebase