@push.rocks/smartdeno

A module to run Deno scripts from Node.js, including functionalities for downloading Deno and executing Deno scripts.

- readme.md for @push.rocks/smartdeno
- changelog.md for @push.rocks/smartdeno

readme.md for @push.rocks/smartdeno

Run Deno scripts from Node.js with automatic Deno management, permission control, and ephemeral script execution. □□➡□□

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

```
npm install @push.rocks/smartdeno
# or
pnpm add @push.rocks/smartdeno
```

Usage

□□ Basic Example

```
import { SmartDeno } from '@push.rocks/smartdeno';

const smartDeno = new SmartDeno();

// Start SmartDeno (downloads Deno if needed)
```

```
await smartDeno.start();

// Execute a Deno script
const result = await smartDeno.executeScript(`console.log("Hello from Deno!")`);
console.log(result.stdout); // "Hello from Deno!"
console.log(result.exitCode); // 0

// Clean up when done
await smartDeno.stop();
```

⚙ Configuration Options

Start Options

```
await smartDeno.start({
  // Force download a local copy of Deno even if it's in PATH
  forceLocalDeno: true,
});
```

□□ Deno Permissions

Control what your Deno scripts can access using the permissions option:

```
// Allow network access
const result = await smartDeno.executeScript(
  `const response = await fetch("https://api.example.com/data");
   console.log(await response.text());`,
  { permissions: ['net'] }
);

// Allow environment variable access
const envResult = await smartDeno.executeScript(
  `console.log(Deno.env.get("HOME"))`,
  { permissions: ['env'] }
);

// Allow all permissions (use with caution! ⚠□)
const fullResult = await smartDeno.executeScript(
```

```
  `// Script with full access`,
  { permissions: ['all'] }
);
```

Available Permissions

Permission	Description
`all`	Grant all permissions (`-A` flag)
`env`	Environment variable access
`ffi`	Foreign function interface
`hrtime`	High-resolution time measurement
`net`	Network access
`read`	File system read access
`run`	Subprocess execution
`sys`	System information access
`write`	File system write access

□□ Execution Results

The `executeScript` method returns detailed execution information:

```
const result = await smartDeno.executeScript(`console.log("test")`);

console.log(result.exitCode); // 0 for success, non-zero for errors
console.log(result.stdout);   // Standard output
console.log(result.stderr);   // Standard error output
```

□□ Error Handling

```
// Scripts that throw errors return non-zero exit codes
const result = await smartDeno.executeScript(`throw new Error("Something went wrong")`);

if (result.exitCode !== 0) {
  console.error('Script failed:', result.stderr);
}
```

```
// Attempting to execute before starting throws an error
try {
  const unstarted = new SmartDeno();
  await unstarted.executeScript(`console.log("test")`);
} catch (error) {
  console.error(error.message); // "SmartDeno is not started. Call start() first."
}
```

□□ Lifecycle Management

```
const smartDeno = new SmartDeno();

// Check if running
console.log(smartDeno.isRunning()); // false

await smartDeno.start();
console.log(smartDeno.isRunning()); // true

// Execute scripts...

await smartDeno.stop();
console.log(smartDeno.isRunning()); // false

// Safe to call stop() multiple times
await smartDeno.stop(); // No error
```

□□ Integration Example

Using SmartDeno in an Express server:

```
import express from 'express';
import { SmartDeno } from '@push.rocks/smartdeno';

const app = express();
const smartDeno = new SmartDeno();

app.use(express.json());
```

```
// Initialize on startup
await smartDeno.start();

app.post('/execute', async (req, res) => {
  const { script, permissions } = req.body;

  try {
    const result = await smartDeno.executeScript(script, { permissions });
    res.json(result);
  } catch (error) {
    res.status(500).json({ error: error.message });
  }
});

// Cleanup on shutdown
process.on('SIGTERM', async () => {
  await smartDeno.stop();
  process.exit(0);
});

app.listen(3000);
```

□□ How It Works

SmartDeno works by:

1. □□ **Deno Management** — Automatically downloads the latest Deno binary for your platform if not available or if `forceLocalDeno` is set
2. □□ **In-Memory Execution** — Scripts < 2MB are executed via stdin (`deno run -`), staying fully in-memory with no disk I/O
3. □□ **Large Script Support** — Scripts >= 2MB automatically use a temp file (cleaned up after execution)
4. □□ **Permission Control** — Translates permission options to Deno's security flags

□□ API Reference

`SmartDeno`

Constructor

```
const smartDeno = new SmartDeno();
```

Methods

Method	Description
`start(options?)`	Initialize SmartDeno (downloads Deno if needed)
`stop()`	Stop SmartDeno instance
`isRunning()`	Check if SmartDeno is currently running
`executeScript(script, options?)`	Execute a Deno script

Types

```
interface ISmartDenoOptions {
  forceLocalDeno?: boolean;
}

interface IExecuteScriptOptions {
  permissions?: TDenoPermission[];
}

type TDenoPermission =
  | 'all'
  | 'env'
  | 'ffi'
  | 'hrtime'
  | 'net'
  | 'read'
  | 'run'
  | 'sys'
  | 'write';
```

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 @push.rocks/smartdeno

2025-12-02 - 1.2.0 - feat(smartdeno)

Run small scripts in-memory via stdin, fall back to temp files for large scripts; remove internal HTTP script server and simplify plugin dependencies; update API and docs.

- Execute scripts < 2MB via stdin (deno run -) to support fully in-memory execution with no disk I/O.
- Automatically write scripts >= 2MB to a temp file under .nogit and clean up after execution.
- Removed internal HTTP script server implementation and related types; start() no longer starts a script server.
- Dropped plugin dependencies and exports related to @api.global/typedserver and @push.rocks/lik; plugins.ts simplified to only include necessary push.rocks modules and node path.
- Updated package.json to remove unused dependencies and adjust keywords to reflect in-memory execution.
- Updated README to document new start() behavior, in-memory execution, temp-file fallback, and simplified API signatures (start/stop/isRunning/executeScript).
- ts index now only exports SmartDeno (removed direct type export of TDenoPermission from separate file).

2025-12-02 - 1.1.0 - feat(core)

Add permission-controlled Deno execution, configurable script server port, improved downloader, dependency bumps and test updates

- Add TDenoPermission type and support passing permissions to executeScript (translates to Deno flags)

- Introduce IDenoExecutionOptions and make DenoExecution build and use permission flags; execution now served via ScriptServer URL and cleaned from executionMap after run
- Make ScriptServer configurable with IScriptServerOptions (port), expose getPort(), return 404 for unknown execution ids, properly stop server and wipe execution map
- SmartDeno: add ISmartDenoOptions (forceLocalDeno, port), guard executeScript to require start(), pass denoBinaryPath and permissions to DenoExecution
- DenoDownloader: track denoBinaryPath, use @push.rocks/smartfs, extract archive, set executable permissions on Unix, and clean up zip file
- Update plugins to export smartfs and smartshell (fix typos) and re-export path/typedserver namespaces
- Export TDenoPermission from package entry (ts/index.ts)
- Update tests to new tapbundle import, add lifecycle and error handling tests, and adapt to API changes
- Bump various dependencies and devDependencies in package.json and add packageManager lock info
- Add project memory/docs files (.serena) and project config updates

2024-05-29 - 1.0.3 - maintenance

Consolidated metadata and build configuration updates for 1.0.3.

- Updated package description (2024-05-29).
- Updated TypeScript configuration (tsconfig) (2024-04-14).
- Updated npmextra.json githost entries (multiple commits on 2024-03-30 and 2024-04-01).
- Duplicate/no-op version commits are summarized below.

2024-03-17 - 1.0.2 - fix(core)

Core fix included in the 1.0.2 release.

- fix(core): update (2024-03-17).

2024-03-16 - 1.0.1 - fix(core)

Core fix included in the 1.0.1 release.

- fix(core): update (2024-03-16).

2024-03-16 - 2024-03-17 - 1.0.2..1.0.3 - housekeeping

Version-only commits (no substantive code changes) recorded for completeness.

- Commits with messages equal to the version number (e.g., "1.0.2", "1.0.3") were made on 2024-03-16–2024-03-17; no additional changes to summarize.