



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

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

☐☐ API Reference

SmartDeno

Constructor

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

Methods

Method	Description
<code>start(options?)</code>	Initialize SmartDeno (downloads Deno if needed)
<code>stop()</code>	Stop SmartDeno instance
<code>isRunning()</code>	Check if SmartDeno is currently running
<code>executeScript(script, options?)</code>	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.

Revision #3

Created 2026-03-28 11:10:32 UTC by foss.global Team

Updated 2026-03-28 12:17:18 UTC by foss.global Team