readme.md for @push.rocks/smartsign

sign documents

Install

To install @push.rocks/smartsign, run the following command in your terminal:

```
npm install @push.rocks/smartsign --save
```

This will add @push.rocks/smartsign to your project's dependencies and enable you to start using its functionality to sign PDF documents.

Usage

@push.rocks/smartsign provides a powerful yet simple API designed for signing documents, especially PDFs, in Node.js applications. The package seamlessly integrates with TypeScript, offering comprehensive type support for better development experience. Let's dive into how to use @push.rocks/smartsign in your TypeScript projects.

Prerequisites

Make sure you have a PDF document prepared for signing and a P12 certificate. In a real-world scenario, the P12 certificate should be obtained from a certified authority and securely stored.

Setting Up

First, you'll need to import the `SmartSign` class from the package:

```
import { SmartSign } from '@push.rocks/smartsign';
import * as fs from 'fs';
```

Prepare the P12 certificate and the PDF document you intend to sign:

```
// Load your P12 certificate as a Buffer
const p12CertificatePath = './path/to/your/certificate.p12';
const p12CertificateBuffer = fs.readFileSync(p12CertificatePath);

// Specify the path to the PDF you want to sign
const pdfFilePath = './path/to/your/document.pdf';
```

Initialize and Use SmartSign

Create an instance of `SmartSign` by passing the P12 certificate buffer to its constructor. Then, start the SmartSign service to prepare it for signing operations:

```
const smartSignInstance = new SmartSign(p12CertificateBuffer);

// Start the smartsign instance
await smartSignInstance.start();
```

Once the service is started, you can create a signing envelope for your document. The envelope wraps the document and prepares it for the signing process:

```
import { SmartPdf, IPdf } from '@push.rocks/smartpdf';

// Assume you have loaded your PDF into a SmartPdf instance
const mySmartPdf: SmartPdf = new SmartPdf();
await mySmartPdf.loadPdf(pdfFilePath);
// Create an envelope for the document
const signingEnvelope = await smartSignInstance.createEnvelopeFromPdf(mySmartPdf);
```

At this point, the `signingEnvelope` object can be used to apply various signing operations defined by your business logic.

Signing the Document

With the signing envelope prepared, you can add a signature to the document. Here is a simplified example of how to apply an invisible signature:

```
// The detailed implementation of applying a signature will depend on the document's
requirements
```

```
// and the level of customization provided by the @push.rocks/smartsign API.

await signingEnvelope.signPdf({ /* signature options */ });
```

After signing the document, you can export the signed PDF:

```
const signedPdfBuffer = await signingEnvelope.exportSignedPdf();

// Save the signed PDF to a file
fs.writeFileSync('./path/to/signed_document.pdf', signedPdfBuffer);
```

Finalizing

Don't forget to stop the SmartSign instance once all operations are completed to release any resources:

```
await smartSignInstance.stop();
```

Conclusion

The @push.rocks/smartsign package offers a streamlined, TypeScript-friendly way to sign PDF documents in your Node.js applications. By leveraging TypeScript, it provides a rich development experience with autocomplete and type checking, ensuring that you can integrate document signing capabilities into your projects efficiently and reliably.

This introduction covered the basic usage of @push.rocks/smartsign. Depending on your specific use case, you might explore more advanced features and customization options provided by the library, such as signing with visible signatures, timestamping, and handling multiple signatures.

For further information, consult the official documentation.

License and Legal Information

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 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, and any usage must be approved in writing by Task Venture Capital GmbH.

Company Information

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

For any legal inquiries or if you require 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 11:12:33 UTC by foss.global Team
Updated 2026-03-28 12:19:18 UTC by foss.global Team