



# DIDKit

Docker 19.03.x

Rust v1.51.0

ssi v0.1

License Apache-2.0

22

## What is DIDKit?

DIDKit provides Verifiable Credential and Decentralized Identifier functionality across different platforms. DIDKit's core libraries are written in Rust due to Rust's expressive type system, memory safety, simple dependency web, and suitability across different platforms including embedded systems, but the comprehensive DIDKit SDK includes many libraries and interfaces for using it almost everywhere.

## Key Features

DIDKit supports the following key capabilities:

- It can sign and verify [W3C Verifiable Credentials](#) almost anywhere you can install it.
  - Where is anywhere, you ask? See the "DIDKit Interfaces" section to the left for a

growing list of language-specific libraries, many available via package manager, and foreign function interfaces.

- **Credible** is also anywhere-- DIDKit powers our SDK for web-wallets and mobile app development, which spans a whole additional range of anywheres you might need to handle DIDs and VCs.
- If you need a server, not a library, DIDKit also powers a dockerized, ready-to-go HTTP/HTTPS server that can be called using the **VC-HTTP-API** standard or customized to use any other API interface.
- It can juggle and translate between the two major signing systems and proof formats used in Verifiable Credentials today: Linked Data Proofs and the JOSE family of tokens and envelopes, abstracting out all the complexity of both.
- It can handle, authenticate, validate, register, and even deterministically generate many kinds of **W3C Decentralized Identifiers**, aka the titular "DIDs": full-featured "on-chain" DIDs, implicit or "off-chain" DIDs, disposable, short-lived DIDs, pseudo-DIDs generated by key material borrowed from other systems
  - This includes "GitHub keys", HSM keys, any blockchain addresses representable as CAIP codes... the **list** keeps growing!
- It can also issue and consume authorization tokens based on the Object Capabilities model, also known as "ZCaps", which drive the

security model of our powerful Kepler storage system, as well as many other next-generation resource management systems.

## Quickstart

You can build DIDKit's command-line interface and HTTP server in just a few minutes from your command line.

Prerequisites:

- Any major GNU/Linux distribution, including MacOS or Microsoft's [WSL2](#)
- [Stable Rust](#)

Building `didkit` :

```
$ git clone
https://github.com/spruceid/ssi --
recurse-submodules
$ git clone
https://github.com/spruceid/didkit
$ cd didkit/
$ cargo build
```

That's it-- you're now ready to use `didkit`'s CLI. For comprehensive documentation of CLI commands, see [Github](#), and for a more skimmable overview, see the [CLI page](#)) here. For example, these basic commands should confirm the installation was succesful:

```
$ ./target/debug/didkit -h
$ ./target/debug/didkit generate-
ed25519-key > key.jwk
```

You're also ready to spin up a `didkit` -powered HTTP server for internal or external use, depending on your context. For comprehensive documentation of the HTTP commands, see [Github](#) , and for a more skimmable overview, see the [HTTP page](#) here. The HTTP server can be spun up with a single command if passed a key and some flags, and will respond with the port on which it will listen for valid calls:

```
$ ./target/debug/didkit-http -k
key.jwk
Listening on http://127.0.0.1:51467/
```


More detailed installation instructions and variants, including Docker instructions, can be found on our [installation page](#).

## Roadmap

The following tools and features are high priority for subsequent releases:

1. Exposing interfaces for JWT-based Verifiable Credential workflows
2. JSON-LD context editor and hosting/publication tool

3. Registration of several new LD signature suites and support for new cryptography
4. DIDComm support
5. Aries interoperability profile support

 [Edit this page](#)

# Installation

DIDKit can be installed as a package via [cargo](#), [manually](#) from source, or in [containerized](#) form.

## Cargo install

1. To install the DIDKit command line program on GNU/Linux, MacOS, or Windows+WSL, first install [cargo](#).
2. Install `build-essential` or equivalent tools if they aren't already installed.
3. Then simply run `cargo install` for the given target package: `didkit-cli` | `didkit-http`.

For example, for DIDKit CLI, run:

```
cargo install didkit-cli
```

This will add the binary `didkit` to your Cargo installation (typically `~/.cargo/bin`), which can be added to your system's PATH for ease of use.

## Manual

DIDKit is written in [Rust](#). To get Rust, you can use [Rustup](#).

We do not depend on any Rust nightly features, so our installation instructions assume `stable` versions; be sure to [switch the installation defaults](#) to `nightly` if the calling application or forked source-code does depend on them.

Spruce's `ssi` library must be cloned alongside the `didkit` repository in a parallel directory between downloading didkit and building it.

```
$ mkdir didkit
$ git clone
https://github.com/spruceid/didkit &&
cd didkit
$ git clone
https://github.com/spruceid/ssi
../ssi --recurse-submodules
```

Build DIDKit using [Cargo](#), from root directory of DIDKit project:

```
$ cargo build
```

This will give you the DIDKit CLI and HTTP server executables located at `target/debug/didkit` and `target/debug/didkit-http`, respectively. You can also build and install DIDKit's components separately. Building the FFI libraries will require additional dependencies. See the corresponding readmes linked below for more info.

# Container

Both the CLI and HTTP server are containerised and available under `ghcr.io/spruceid/didkit-(cli|http)`.

The image is private for now, so a [Personal Access Token](#) is required. Once created you can login like so:

```
$ docker login ghcr.io -u USERNAME --password-stdin
```

You can use the images like CLIs:

```
$ docker run ghcr.io/spruceid/didkit-cli:latest --help
$ docker run --init -p 8080 ghcr.io/spruceid/didkit-http:latest -port 8080
```

Note: You can pass JWKs either by sharing a volume with `docker run --volume`, or by passing the JWK directly with `docker run -e JWK=$MY_JWK` or `docker run didkit-http --jwk $MY_JWK`.

## Build Images

The Dockerfiles rely on having `ssi` in the root of `didkit` (a symbolic link will not work,



unfortunately).

Then the images can be built with:


```
$ docker build -f Dockerfile-cli . -t didkit-cli
$ docker build -f Dockerfile-http . -t didkit-http
```

And to use them, replace

```
ghcr.io/spruceid/didkit-
(cli|http):latest with didkit-(cli|http) .
```

## Building Interfaces

While many of the DIDKit interfaces can be installed as libraries via each language's dedicated package manager, they can also be built manually. For instructions, see the "Installation" section of each interface's dedicated page in the section to the left.

 [Edit this page](#)

spruceid / didkit Public

A cross-platform toolkit for decentralized identity.

spruceid.dev/docs/didkit

Apache-2.0 License

103 stars 21 forks

Star

Notifications

Code

Issues 26

Pull requests 8

Actions

Projects

Wiki

Security

Insights

main

Go to file



theosirian and clehner Add ResolutionInputMetadata; Change Module Name ...

4 days ago

241

View code

README.md



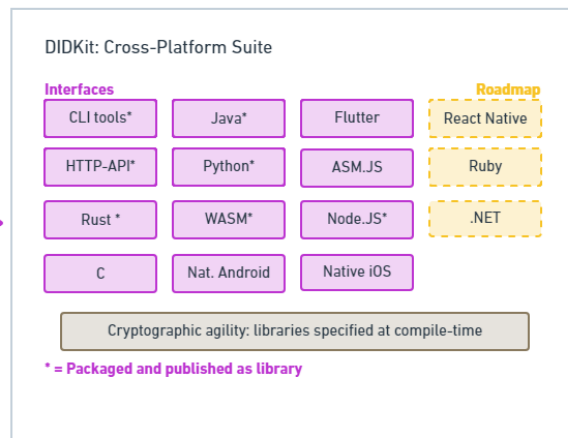
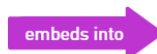
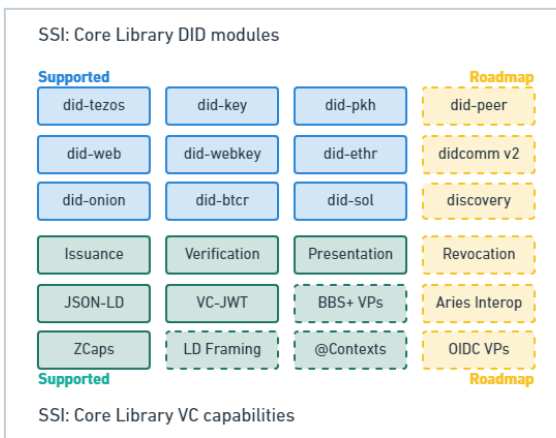
# DIDKit

build passing Docker 19.03.x Rust v1.51.0 ssi v0.1 License Apache-2.0 Follow 5.5k

Check out the DIDKit documentation [here](#).

## DIDKit

DIDKit provides Verifiable Credential and Decentralized Identifier functionality across different platforms. It was written primarily in Rust due to Rust's expressive type system, memory safety, simple dependency web, and suitability across different platforms including embedded systems. DIDKit embeds the [ssi](#) library, which contains the core functionality.



## Maturity Disclaimer

---

In the v0.1 release on January 27th, 2021, DIDKit has not yet undergone a formal security audit and to desired levels of confidence for suitable use in production systems. This implementation is currently suitable for exploratory work and experimentation only. We welcome feedback on the usability, architecture, and security of this implementation and are committed to a conducting a formal audit with a reputable security firm before the v1.0 release.

We are setting up a process to accept contributions. Please feel free to open issues or PRs in the interim, but we cannot merge external changes until this process is in place.

We are also in the process of creating crates.io entries for the DIDKit and SSI packages.

## Install

---

### Manual

DIDKit is written in [Rust](#). To get Rust, you can use [Rustup](#).

Spruce's [ssi](#) library must be cloned alongside the `didkit` repository:

```
$ git clone https://github.com/spruceid/ssi ../ssi --recurse-submodules
```

Build DIDKit using [Cargo](#):

```
$ cargo build
```

That will give you the DIDKit CLI and HTTP server executables located at `target/debug/didkit` and `target/debug/didkit-http`, respectively. You can also build and install DIDKit's components separately. Building the FFI libraries will require additional dependencies. See the corresponding readmes linked below for more info.

### Container

Both the CLI and HTTP server are containerised and available under `ghcr.io/spruceid/didkit-(cli|http)`.

You can use the images like CLIs:

```
$ docker run ghcr.io/spruceid/didkit-cli:latest --help
$ docker run --init -p 8080 ghcr.io/spruceid/didkit-http:latest --port 8080
```

You can pass JWKs either by sharing a volume with `docker run --volume`, or by passing the JWK directly with `docker run -e JWK=$MY_JWK` or `docker run didkit-http --jwk $MY_JWK`.

### Build Images

The Dockerfiles rely on having `ssi` in the root of `didkit` (a symbolic link will not work unfortunately).

Then the images can be built with:

```
$ docker build -f Dockerfile-cli . -t didkit-cli
$ docker build -f Dockerfile-http . -t didkit-http
```

And to use them, replace `ghcr.io/spruceid/didkit-(cli|http):latest` with `didkit-(cli|http)`.

## Usage

---

DIDKit can be used in any of the following ways:

- [CLI](#) - didkit command-line program
- [HTTP](#) - HTTP server (Rust library and CLI program)
- [FFI](#) - libraries for C, Java, Android, and Dart/Flutter

## Releases

8 tags

## Packages 3

- didkit-http
- didkit-cli
- com.spruceid.didkit

## Contributors 13



+ 2 contributors

## Languages

