TxPipe Docs

Bindgen Basics

Declaring the interface for blockchain protocol is not very useful unless we have a way to interact programmatically with it.

Bindgen is the process of generating the bindings of your Tx3 protocol in a particular general-purpose language.

Bindings are just glue code that allows you to execute the transaction building logic by calling language-specific functions that represent the templates in your protocol.

Supported Languages

Tx3 bindgen feature can currently generate code in the following target languages:

- Typescript
- Rust
- Go
- Python

We plan on adding more languages as we move forward.

> **Tip**
>
> If you're brave enough, you can peek into the different SDK repos that available in our Github org, reverse engineer how they work and build your own. Please share back if you do!

Configuration

The bindgen configuration lives inside the `trix.toml` file. You need to provide an entry for each language target. Here's an example of the config that targets both Typescript and Rust:

```
[[bindings]]
plugin = "typescript"
output_dir = "./gen/typescript"
options = { standalone = true }

[[bindings]]
plugin = "rust"
output_dir = "./gen/rust"
```

Each `bindings` entry has the following schema:

- `plugin`: keyword indicating the target language. Valid values are `typescript`, `rust`, `go` & `python`.
- `output_dir`: root location where the generated code artifacts will be output to.
- `options`: target-specific options that allows you to customize the output.

> ⚠ **Caution**
>
> Generated code is **not** meant to be manually modified. Any manual changes will be overridden when the bindgen process is executed again.

Code generation procedure

To run the code generation process, execute the following `trix` command from your CLI:

```
trix bindgen
```

If things went well, you should have new code artifacts in the corresponding output locations as defined in your `trix.toml` file.

The output of the previous configuration example would yield the following new files:

Typescript Rust

```
gen/
└── typescript/
    ├── my-protocol.ts
    ├── test.ts
    ├── package.json
    └── tsconfig.json
```

We won't get into details of how to use the *ts* code, you can read more about it on the NodeJS bindings reference.