TxPipe Docs

NodeJS Backend

To integrate with NodeJS you need to make sure that your *trix* project has the `typescript` bindings generation enabled in the config:

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

With the binding generation enabled, run the following command from your CLI to trigger the code generator:

```
trix bindgen
```

If things went well, you should see the following new files:

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

That new protocol.ts file has the required types and functions to call your protocol from a Typescript code. It works on the backend and also in the browser (with the corresponding bundling).

The `package.json` and `tsconfig.json` files are standard files for Typescript projects.

Now, we can use the generated bindings to build a transaction using our protocol.

First access to the `gen/typescript` folder and install the dependencies:

```
cd gen/typescript && npm install
```

NOTE: You can use yarn, pnpm or any other package manager you prefer.

We need to open `protocol.ts` and update the TRP endpoint to point to the correct URL (unless you have it configured on trix.toml). The default value is `http://localhost:3000/trp`, but if you're using different port or endpoint, make sure to update it.

For the following example, we will use the main.tx3 file generated by trix.

Now, we can create a new file called `test.ts` in the same folder (or any other folder) and add the following code:

```
import { protocol, type TransferParams } from
'./gen/typescript/protocol';

const transferParams: TransferParams ={
  sender:
'addr_test1vpgcjapuwl7gfnzhzg6svtj0ph3gxu8kyuadudmf0kzsksqrfugfc',
  receiver:
'addr_test1vpry6n9s987fpnmjqcqt9un35t2rx5t66v4x06k9awdm5hqpma4pp',
  quantity: 100000000,
};

async function main() {
  try {
    let cbor = await protocol.transferTx(transferParams);
    console.log(cbor);
  } catch (error) {
    console.error('Error:', error);
  }
}

main();
```

Finally, we can run the test file to build a transaction using our protocol:

```
npx tsx test.ts
```

If everything went well, you should see a message like this:

```
{
  tx:
'84a400d90102818258207e5d956d318264043baf8031e250c14b8703b69947ab2d3212d67
}
```