TxPipe Docs

Trix

Trix is a command-line interface (CLI) tool designed to help developers manage projects written in the Tx3 language. It provides commands for initializing new projects, checking code for errors, generating code bindings, and interacting with development networks.

This documentation will guide you through installing Trix, setting up your first project, and using the available commands to build and manage your Tx3 projects.

Installation Requirements and Initial Setup

There are two main ways to install Trix:

1. Using `tx3up` (Recommended):

   `tx3up` is a tool installer that installs every tool for the Tx3 language, including Trix. It is the recommended way to install Trix as it ensures you have compatible versions of all necessary Tx3 tools. Follow the instructions on the tx3up to install `tx3up`, and then run:

```
tx3up
```

2. Downloading from GitHub Releases:

   If you only need the Trix executable and prefer not to use `tx3up`, you can download the appropriate binary for your operating system from the latest releases page on GitHub.

   After downloading, you may need to make the binary executable and add it to your system's PATH.

You can verify that Trix is installed correctly by running:

```
trix --help
```

To start a new Tx3 project, navigate to the directory where you want to create your project and run the `init` command:

```
trix init
```

The `init` command will interactively ask you for some details about your project, such as the name, owner scope, description, and version. It will also ask which language bindings you want to generate. Based on your input, it will create a `trix.toml` configuration file and a basic `main.tx3` file in your current directory.

Overview Commands

Trix provides the following main commands:

- `trix init`: Initialize a new Tx3 project.
- `trix check`: Check a Tx3 package and its dependencies for errors.
- `trix bindgen`: Generate code bindings for smart contracts.
- `trix devnet`: Start a local development network.
- `trix explore`: Explore a network using CShell.
- `trix invoke`: Invoke a transaction using CShell.

Command Details

`trix init`

Initialize a new Tx3 project.

```
trix init
```

`trix check`

Check a Tx3 package and all of its dependencies for errors.

```
trix check
```

`trix bindgen`

Generate code bindings for smart contracts.

```
trix bindgen
```

trix devnet

Start a local development network (powered by Dolos). Note that running `tx3up` is required to prepare the environment before starting the devnet. For more detailed information on using the development network, please refer to the Devnet Guide.

```
trix devnet [OPTIONS]
```

Available Options:

- `-b, --background`: Run devnet as a background process.

trix explore

Explore a network (powered by CShell).

```
trix explore
```

trix invoke

Invoke a transaction (powered by CShell).

```
trix invoke
```

Common Workflows

Here are some common workflows you might follow when developing with trix:

1. Starting a New Project:

```
mkdir my_tx3_project
cd my_tx3_project
trix init
```

Follow the prompts to set up your project.

2. Writing and Checking Code:

Edit the `main.tx3` file and any other `.tx3` files in your project. Regularly check for errors:

```
trix check
```

3. Generating Bindings:

After writing your Tx3 code, generate language bindings to use your protocols in other applications:

```
trix bindgen
```

4 Developing and Testing on a Devnet:

Start a local development network:

```
trix devnet
```

In a separate terminal, invoke a transaction:

```
trix invoke
```

Then, in another terminal, explore the devnet to monitor the transaction:

```
trix explore
```