Goal: Describe the user's steps to submit a transaction candidate to Anoma using the Cairo backend

.

(These are notes taken from Juvix ↔ Anoma session at HHH on 13-Sep-24 with@xuyang @camofu and led by @Lukasz).

This process represents the first transition of theAnoma Node Transaction Code Lifecycle:

[

Untitled-2024-09-04-1524 (1)

1920×678 58.4 KB

](https://europe1.discourse-cdn.com/flex013/uploads/anoma1/original/1X/8b7b41bd9a54e0707157184ddde82e0f13b2327a.jpeg)

1. Compile Juvix to Cairo

bytecode.

1. Generate the proofs

for the resource logics using the Anoma API (e.g. CLI) in a local environment. * Inputs

: * Resource logic Cairo bytecode (from step 1)

- Resource logic inputs in a .json file
- Resource logic Cairo bytecode (from step 1)
- Resource logic inputs in a .json file
- Outputs

: * Program proof binary

- Program proof binary
- Inputs

: * Resource logic Cairo bytecode (from step 1)

- Resource logic inputs in a .json file
- Resource logic Cairo bytecode (from step 1)
- Resource logic inputs in a .json file
- Outputs

: * Program proof binary

1. Program proof binary
2. Generate compliance proof
3. Inputs

: * compliance.json file, where the "logic" field points to the path to the proof binary field. The Juvix API computes the hash of the file under the hood. This reduces the cognitive burden from the user.

- compliance.json file, where the "logic" field points to the path to the proof binary field. The Juvix API computes the hash of the file under the hood. This reduces the cognitive burden from the user.
- Inputs

: * compliance.json file, where the "logic" field points to the path to the proof binary field. The Juvix API computes the hash of the file under the hood. This reduces the cognitive burden from the user.

1. compliance.json file, where the "logic" field points to the path to the proof binary field. The Juvix API computes the hash of the file under the hood. This reduces the cognitive burden from the user.
2. Submit transaction

to Anoma * Create partial transaction(s) * Inputs

: * Some number of resource proof binaries

(generated by step 2)

- Some number of compliance proof binaries

(generated by step 3)

- Some number of resource proof binaries

(generated by step 2)

- Some number of compliance proof binaries

(generated by step 3)

- Inputs

: * Some number of resource proof binaries

(generated by step 2)

- Some number of compliance proof binaries

(generated by step 3)

- Some number of resource proof binaries

(generated by step 2)

- Some number of compliance proof binaries

(generated by step 3)

- Create final transaction
- Inputs

: * Partial transactions

- Delta: List of “rcv” fields in the compliance_input.json files corresponding to the provided compliance proofs

- Partial transactions

- Delta: List of “rcv” fields in the compliance_input.json files corresponding to the provided compliance proofs

- Inputs

: * Partial transactions

- Delta: List of “rcv” fields in the compliance_input.json files corresponding to the provided compliance proofs

- Partial transactions

- Delta: List of “rcv” fields in the compliance_input.json files corresponding to the provided compliance proofs

- Encode into Nock Transaction Candidate

- Create partial transaction(s)

- Inputs

: * Some number of resource proof binaries

(generated by step 2)

- Some number of compliance proof binaries

(generated by step 3)

- Some number of resource proof binaries

(generated by step 2)

- Some number of compliance proof binaries

(generated by step 3)

1. Inputs

: * Some number of resource proof binaries

(generated by step 2)

- Some number of compliance proof binaries

(generated by step 3)

1. Some number of resource proof binaries

(generated by step 2)

1. Some number of compliance proof binaries

(generated by step 3)

1. Create final transaction
2. Inputs

: * Partial transactions

- Delta: List of “rcv” fields in the compliance_input.json files corresponding to the provided compliance proofs

- Partial transactions

- Delta: List of “rcv” fields in the compliance_input.json files corresponding to the provided compliance proofs

- Inputs

: * Partial transactions

- Delta: List of “rcv” fields in the compliance_input.json files corresponding to the provided compliance proofs

- Partial transactions

- Delta: List of “rcv” fields in the compliance_input.json files corresponding to the provided compliance proofs

- Encode into Nock Transaction Candidate