Spending Allowance Token (SAT) Demo

By :-

Index

- Introduction to Spending Allowance Token
- Tools used in Demo
- Introduction to Metamask
- Connecting to ixxo node using Metamask
- Introduction to Remix
- How to buy SAT
- Sending tokens
  - Originated from person
    - Person to Person and Person to Smart Contract
  - Originated from Smart Contract
    - Smart contract to smart contract and Smart contract to person

Introduction to SAT

- Spending Allowance Token
- It is a stable mintable token , as it maintains and record the flow and movement of each SAT in the system.
- Can be minted by minting entity only (bank)
- It represents the any fiat currency in the system as “Stable Tokens”
- SAT is used as payment module in ixxo network for buying of service, sending to money (to entity in network), etc

Tools used

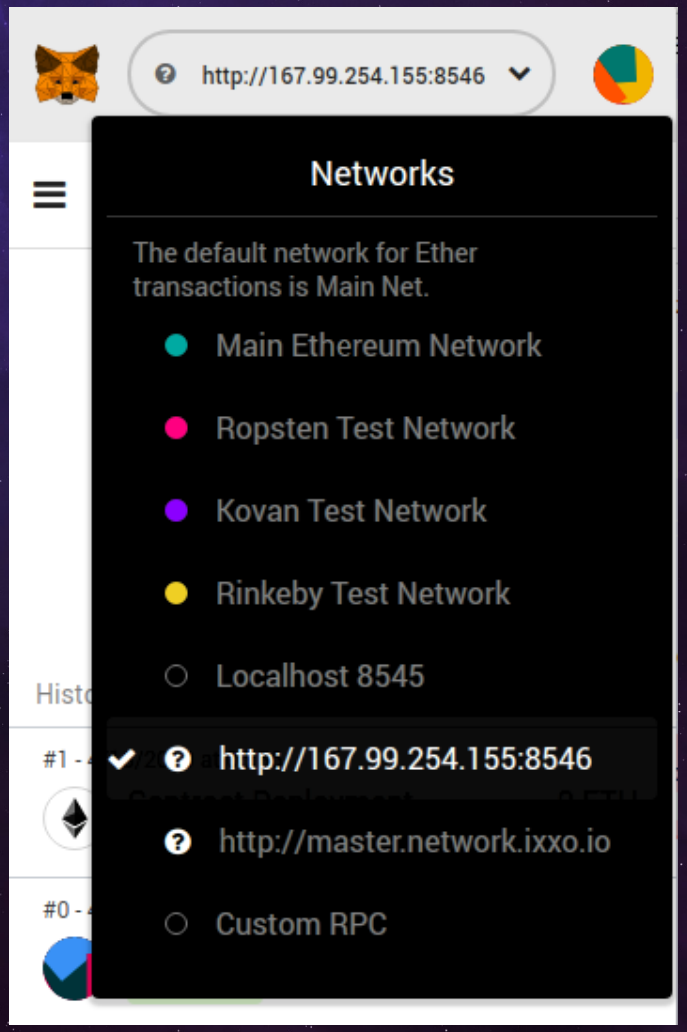
- Metamask
- Remix browser-IDE

METAMASK

remix

Connecting to ixxo node

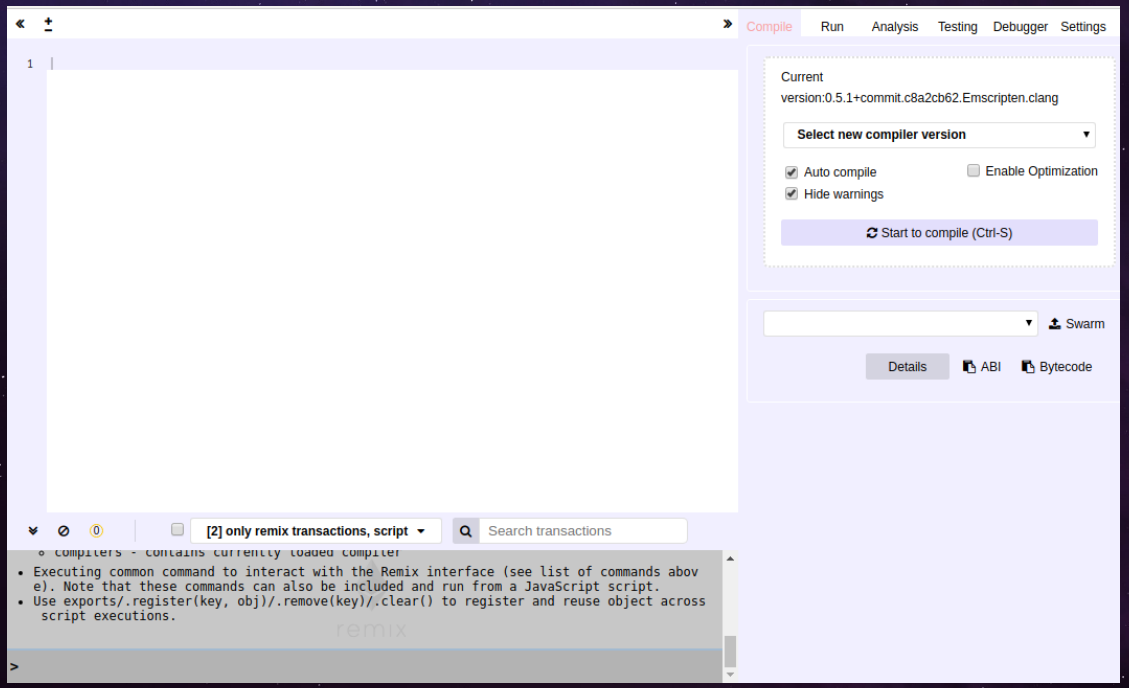
- To connect to ixxo network use Metamask
- Enter the ixxo RPC End Point in custom rpc option , and save it
- Click on options of different network ,then select ixxo RPC
- Now , we are connect to ixxo



Introduction to Remix

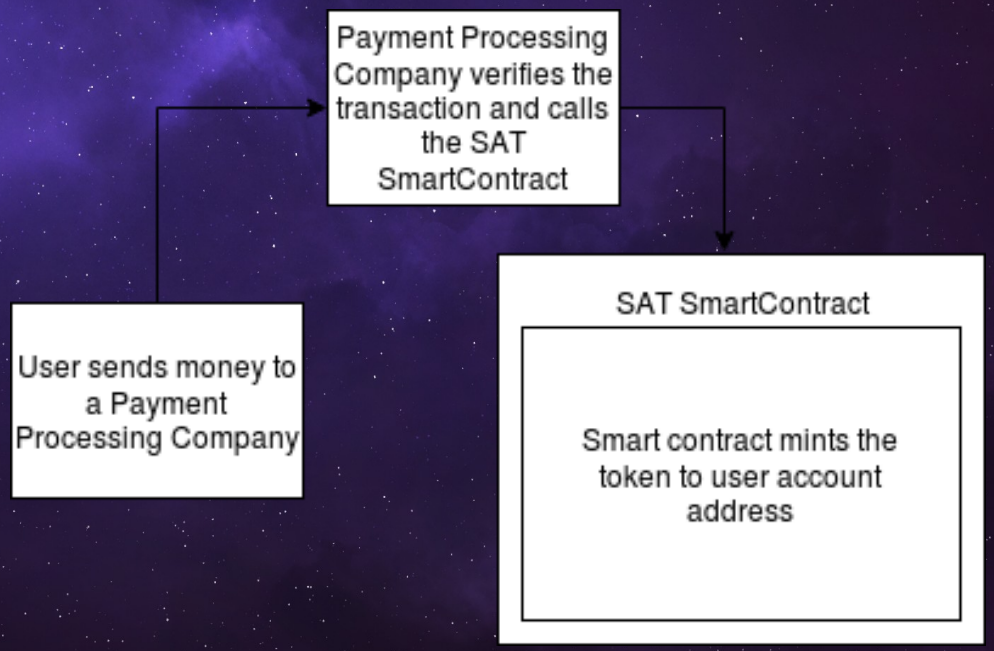
- Open browser and thrn go to remix.ethereum.org
- Remix is not just an ide , but also provide facility to interact with already deployed contract.
- We will use contract abi and contract address to fetch the SAT instance and interact with it .

- Contract address is generated when contract is deployed for the first time in the network

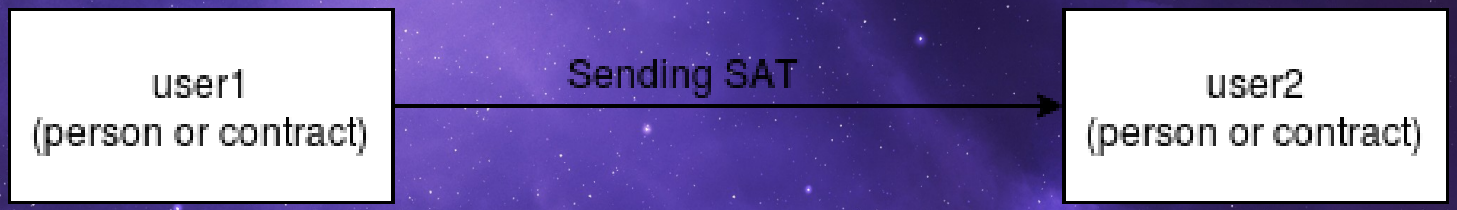


How to buy SAT

- To buy SAT an entity need to send fiat currency to the Payment Processing Company.
- Payment Processing Company mints more SAT then receiving amount, this rate is set by ixxo in compliance with Payment Processing Company
- Then Payment Processing Company mints the SAT to the entities account automatically once payment is received successfully.
- Bank calls the SAT token function “startTokenAllocation()” to mint the tokens.

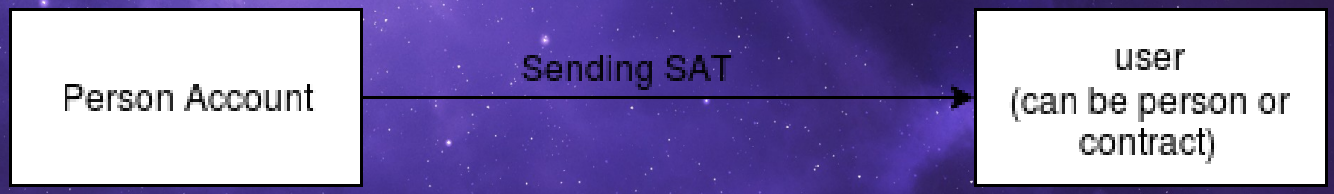


Sending Of Tokens



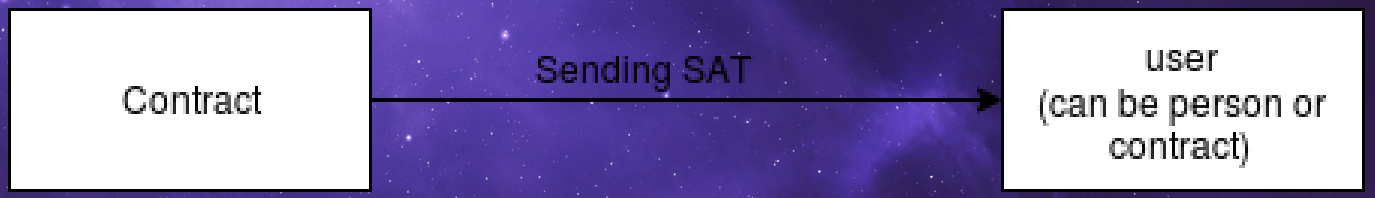
- SAT are primary spending allowances in ixxo network
- They can also be used as payment mode across the entities in the system
- If an entity sends SAT to other entity ixxo takes very minimal amount of commission to smoothly facilitate the transaction
- Sending of token can be from person account or Smart contract , on this basis , SAT are handled

Tokens originating form Person



- In this type of transaction person tries to send the SAT from persons account to other person account or Smart Contract.
- This type of transaction needs to be validated from transaction initiator.

Tokens Originating from Smart Contract



- In this type of transaction Smart Contract tries to send the SAT from Smart Contract to other Smart Contract or person account.
- This type of transaction do not need validation from transaction initiator

Test

- Now , we are going to demonstrate how Spending Allowance Tokens work.
- For this purpose , we are going to use several test cases.
- This will guide us the behavior of SAT in different scenarios

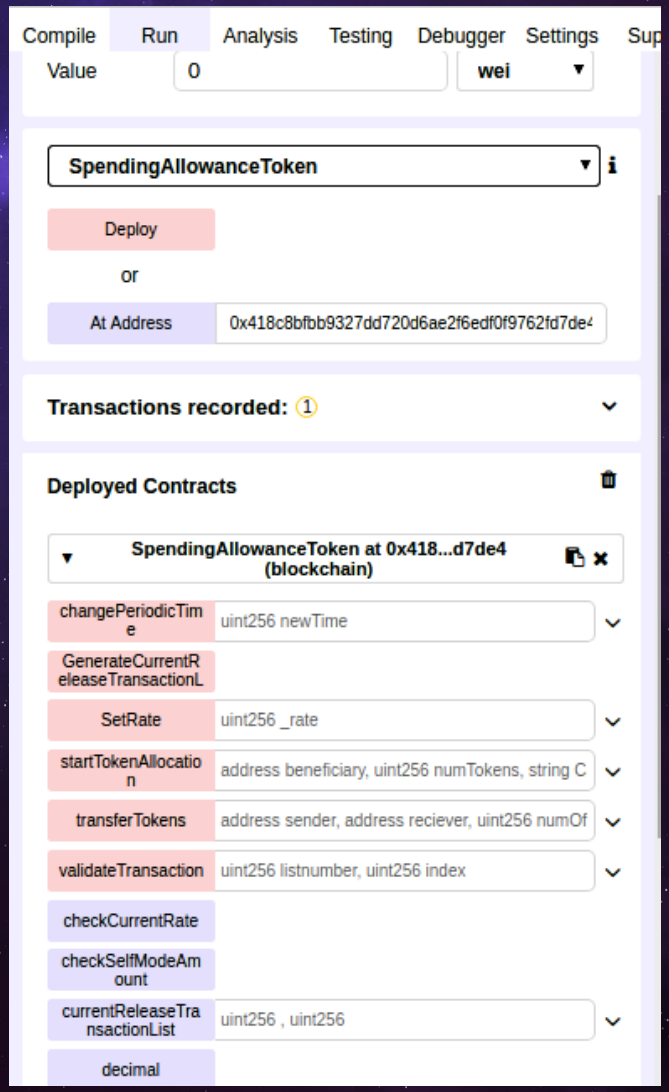
Test 1 A

- Original Transaction
- A => SmartContract1
- SmartContract1 => B (after 30 seconds)

- Expected Result
- Transaction should be generated in second period Starting

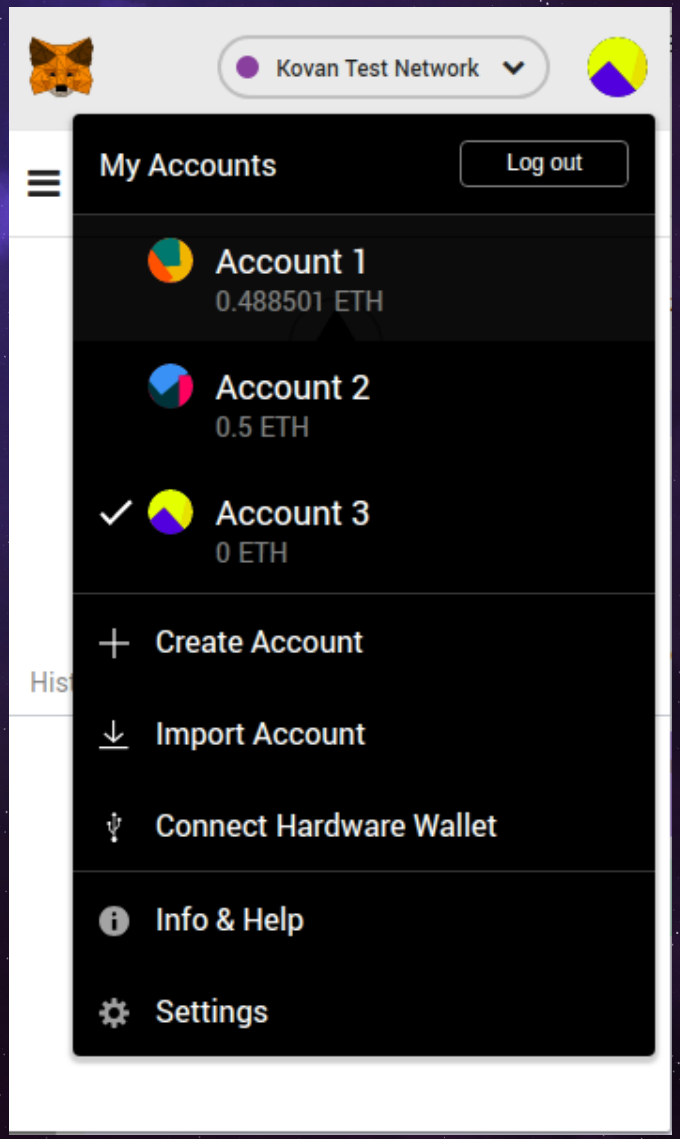
Test 1 A : step 1

- Retrieve SAT instance using SAT contract address and abi.
- Then deploying test Smart Contract ,with SAT contract address as parameter . So that test contract interact with same SAT instance.



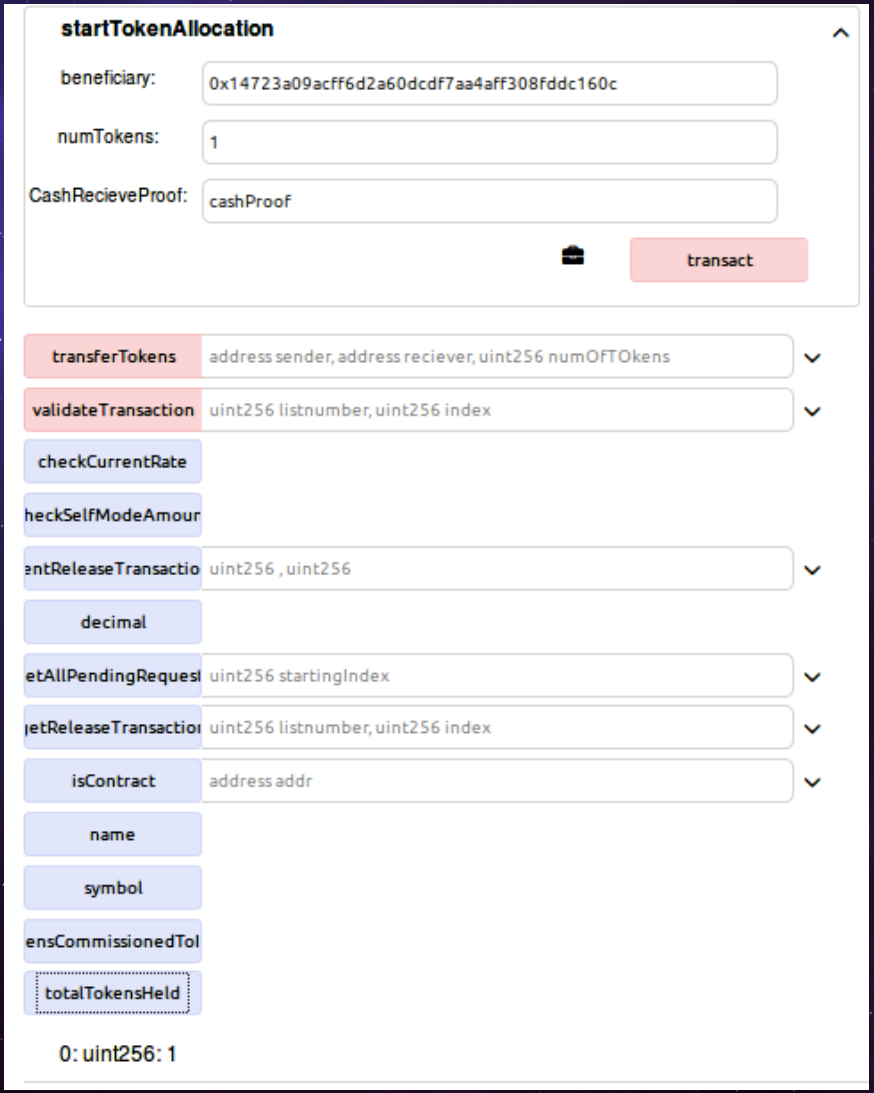
Test 1 A : Step 2

- Using Metamask switch account . Let account 2 be A .
- We can always switch accounts using Metamask . Here we are doing so , to mimic the multi-party interaction



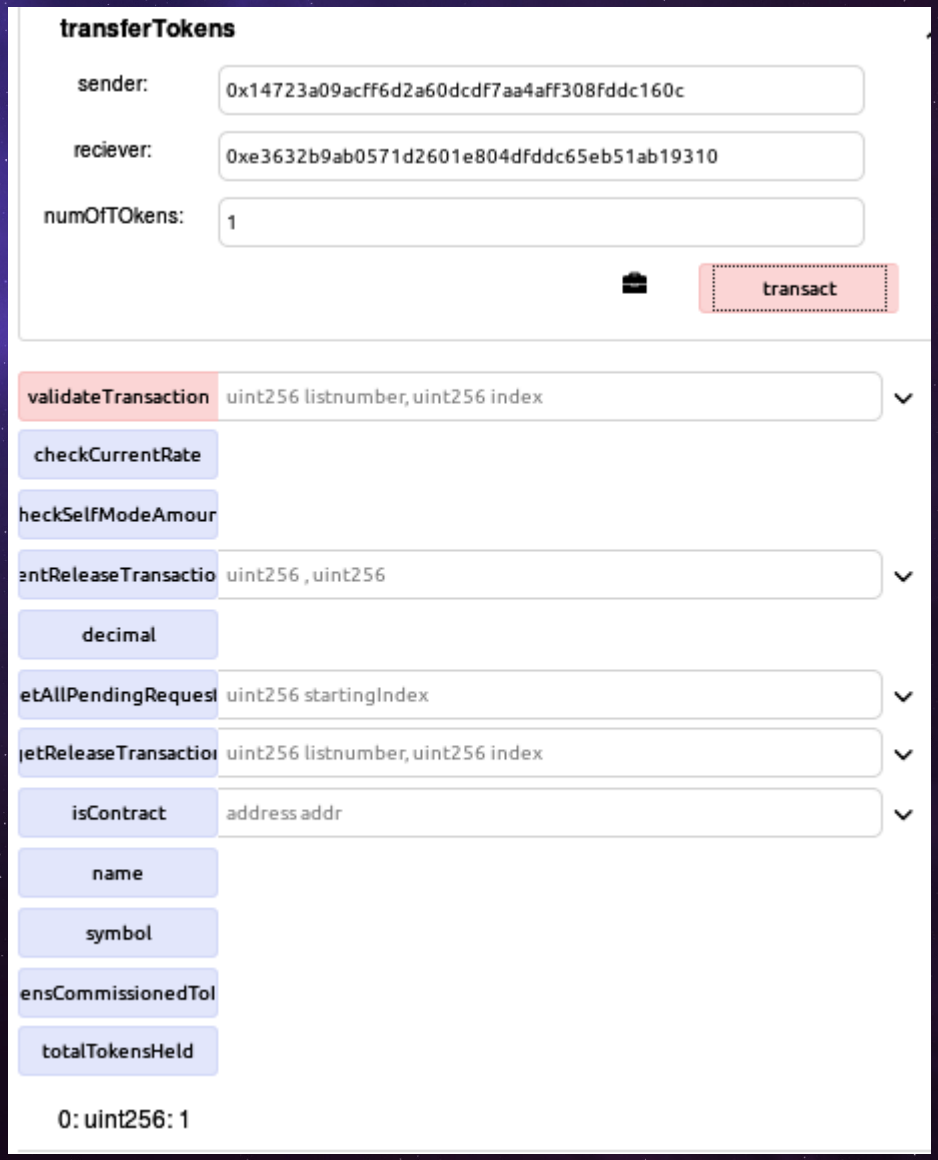
Test 1 A : Step 3

- Allocating 1 SAT to account A .
- For this startTokenAllocation() is invoked by bank , when entity A sends real money to minting entity.



Test 1 A : Step 4.1

- Sending token from account A to test contract SmartContract 1 .

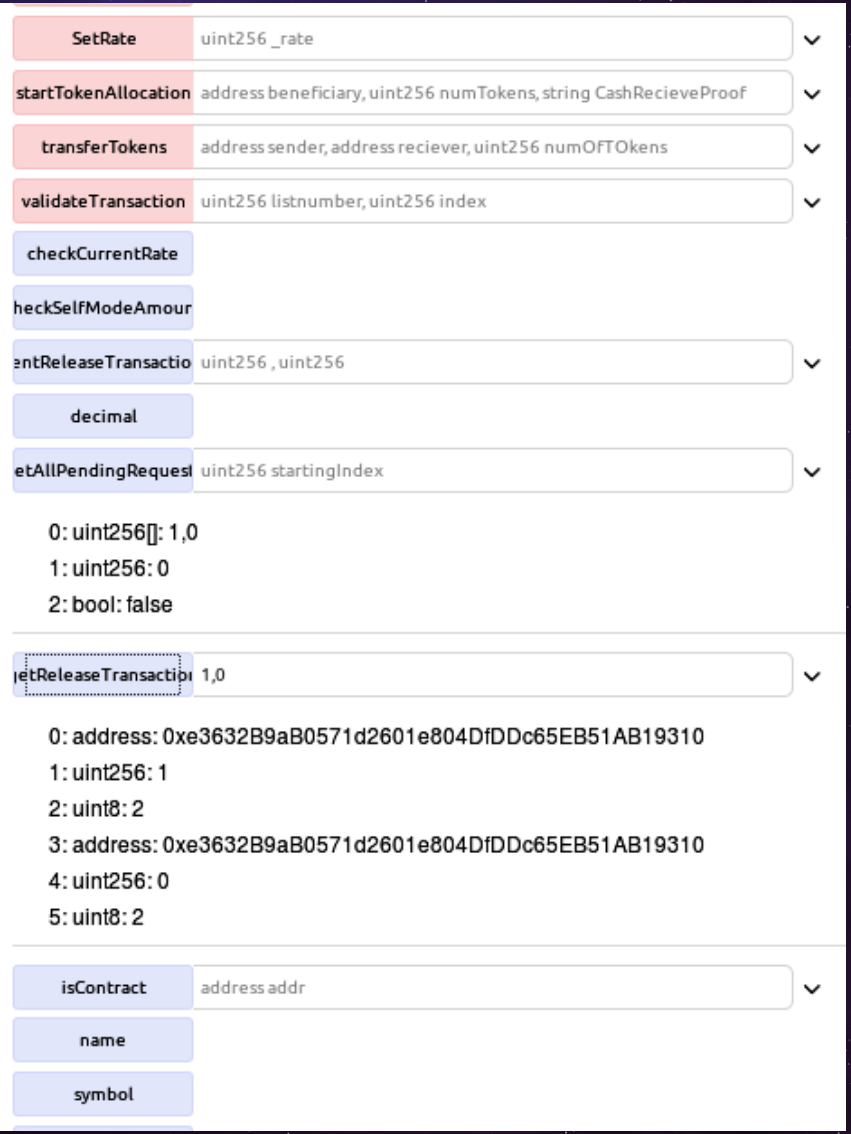


Test 1 A : Step 4.2

- We can use function “getAllPendingRequests()”, for getting all the pending request
- This function returns the period number and index of the transaction
- It returns one transaction at a time and directs if any further pending transaction is there or not .

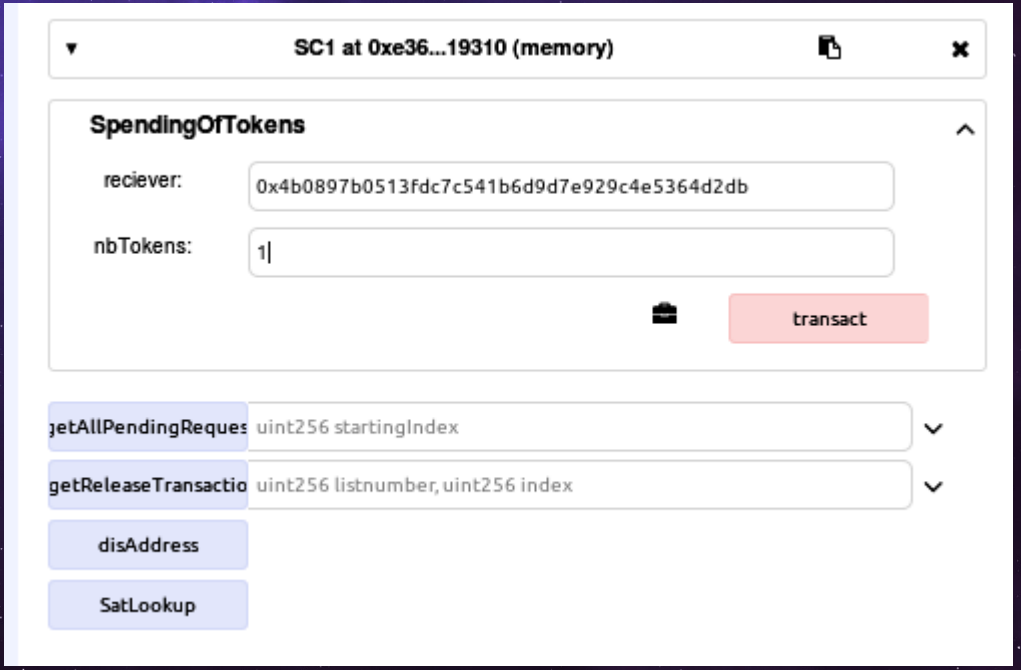
Test 1 A : Step 4.3

- Now use function “getReleaseTransaction()” with period number and index .
- This will reuturn the full information regarding that transaction .



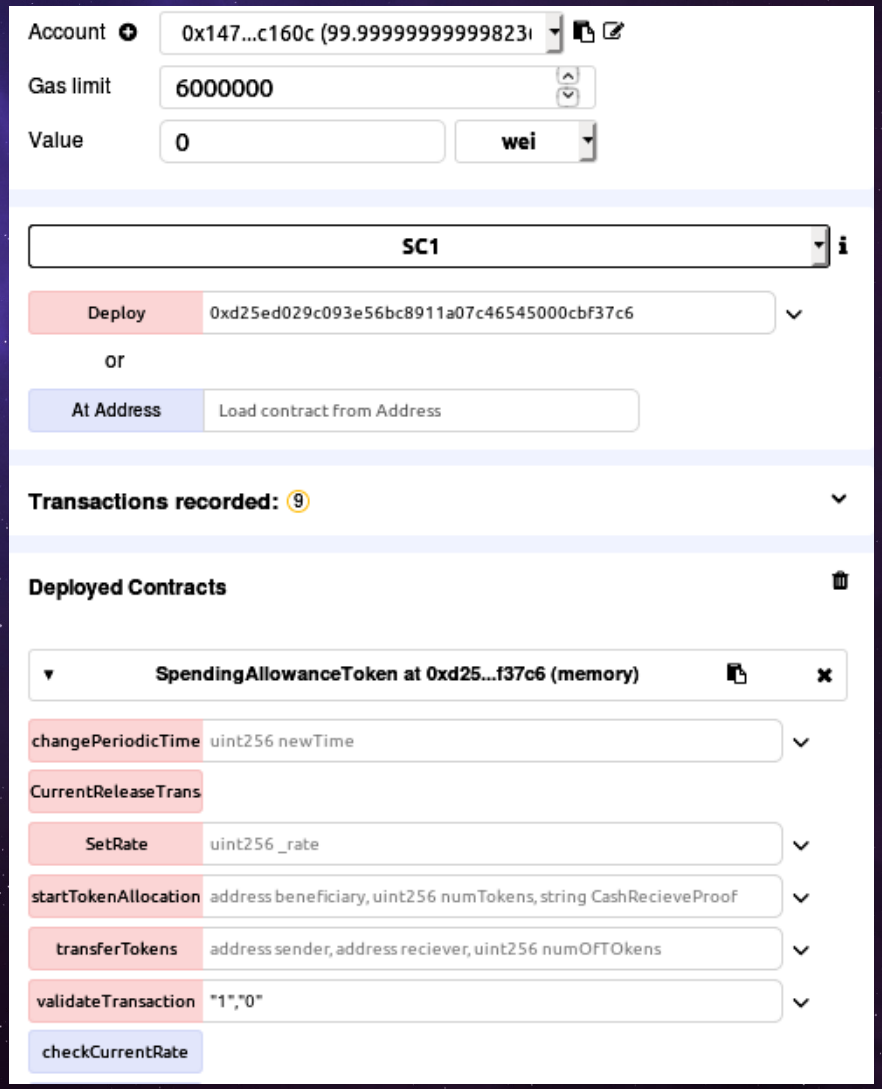
Test 1 A : Step 5

- SmartContract1 holds token for 30 second , so that first period is passed(which is of 20 seconds)
- And then send the SAT to account 3 (which we will call B)



Test 1 A : Step 6

- A validates the transaction.
- Since transaction originated from person account has to be validated in order to make transfer successful



Test 1 A : Result

- On validation all the transaction are chained and released in second period as :
- A => SmartContract1 => B , period2 (Original)
- A=>B , period2 (chained)

Test 1 B

- Original Transaction

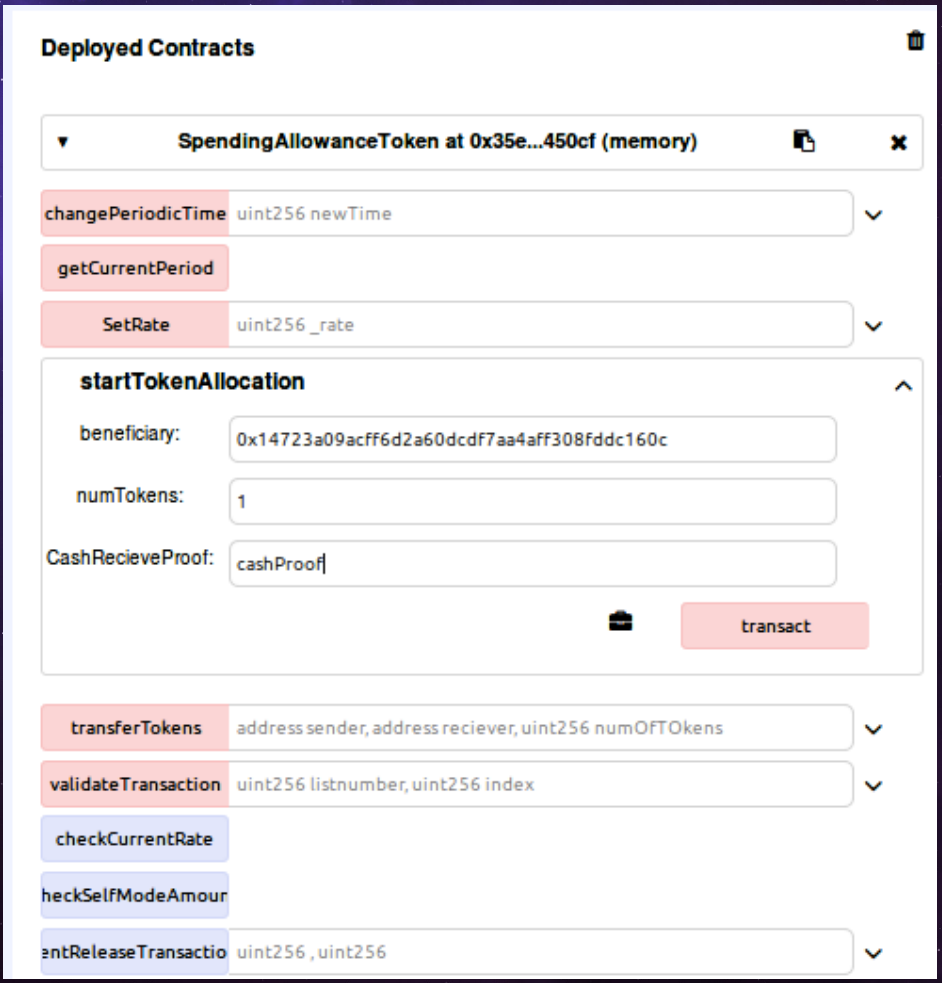
- A send 1 SAT to C
  Hour = 1
- B send 1 SAT to C
  Hour = 1
- C send 2 SAT TO D
  Hour = 1.5

- Expected Result

- At end of Period1 :
- A => C (1 SAT)
- B => C (1 SAT)

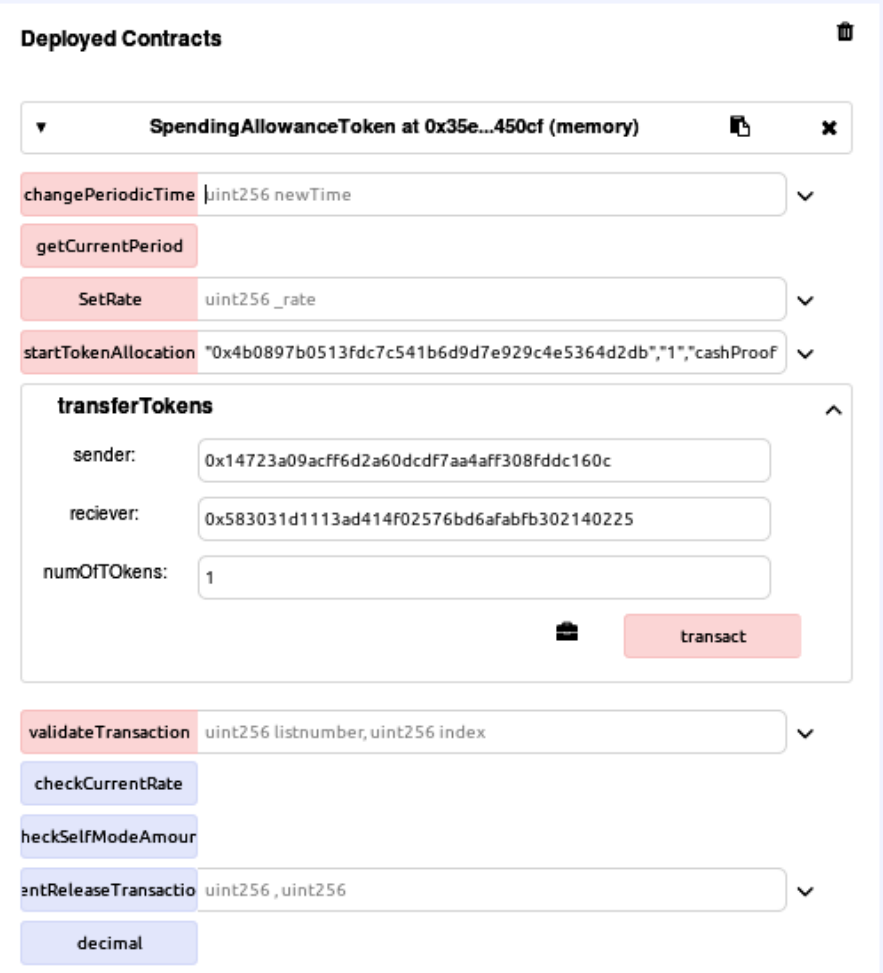
Test 1 B : Step 1

- Allocate 1 SAT to account A
- Allocate 1 Sat to account B.



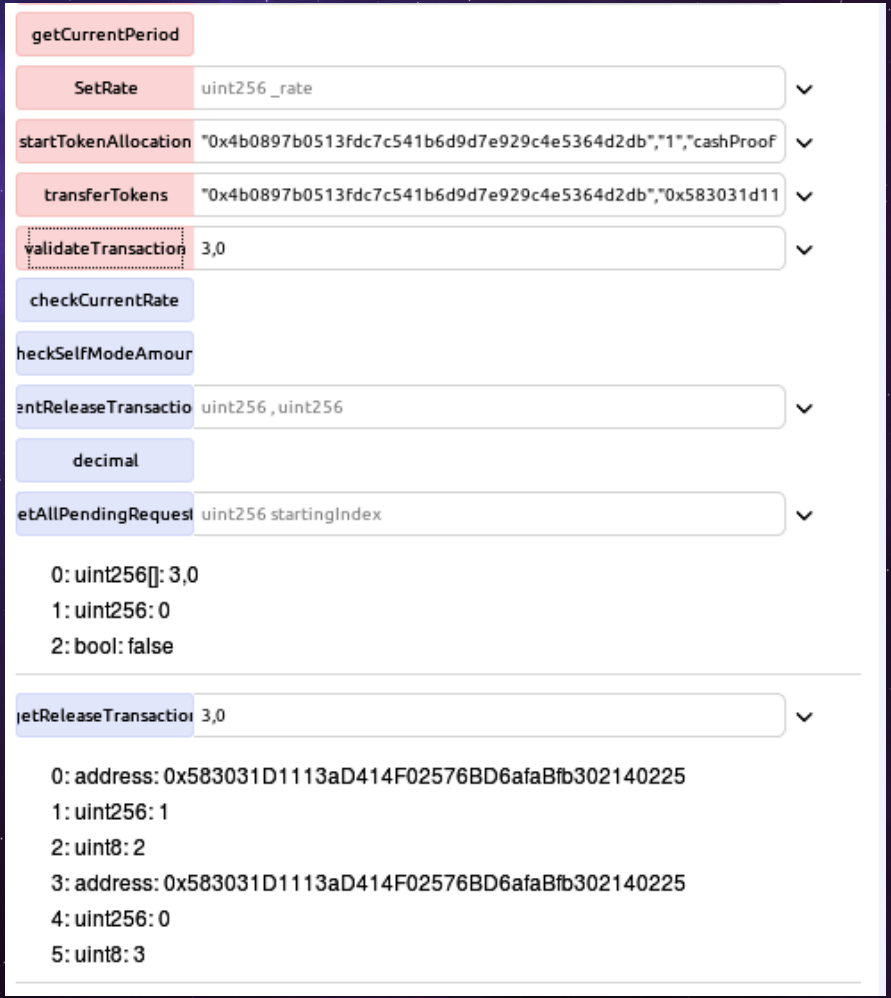
Test 1 B : Step 2

- Send 1 SAT from account A to account C
- Send 1 SAT from account B to account C.



Test 1 B : Step 3

- Validate transaction from A
- Validate Transaction from B



Test 1 B : Results

- Transactions :
  - A => C
  - B => C
- C will have 2 SAT

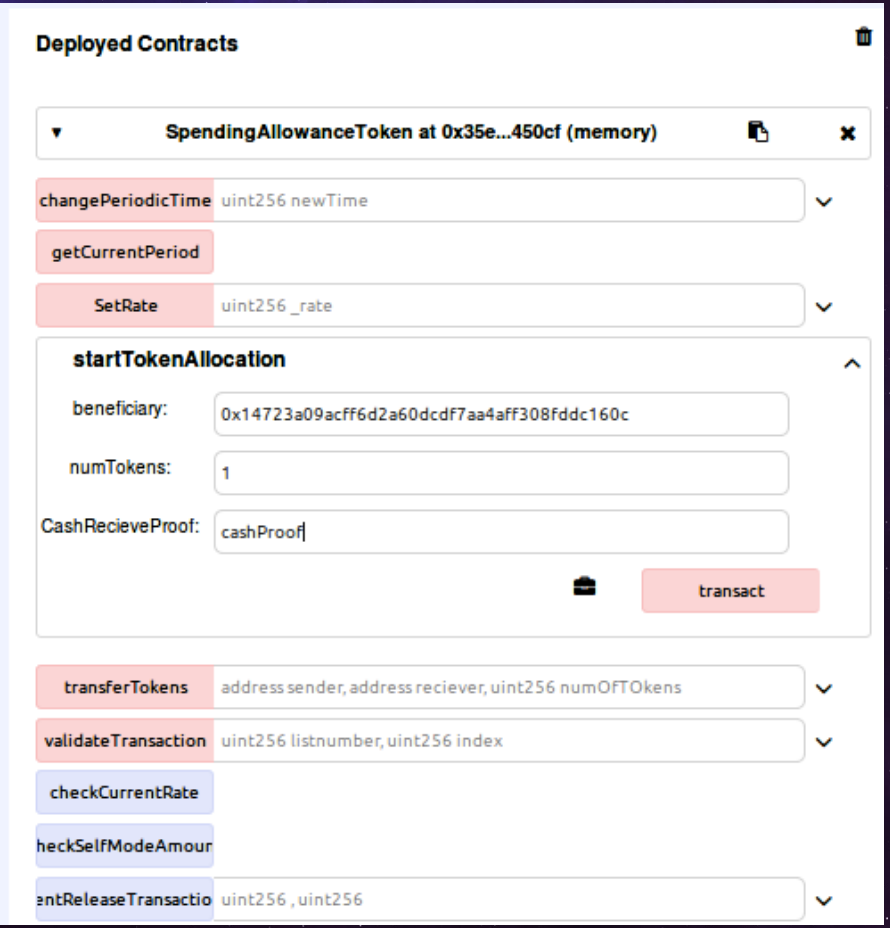
Test 1 C

- Original Transaction
- A send 1 SAT to C HOUR = 1
- B send 1 SAT to C HOUR = 1
- C send 2 SAT to smartContract SC1, who locks the money HOUR = 1.5
- SC1 => SC2 SC1 unlocks the money and sends to smart contract SC2 at HOUR = 3
- SC2 sends the money to A at HOUR = 3.5

- Expected Result
- PERIOD = 1 no payment
- PERIOD = 2 (4H). Payments generated are
- B => A (A SAT)

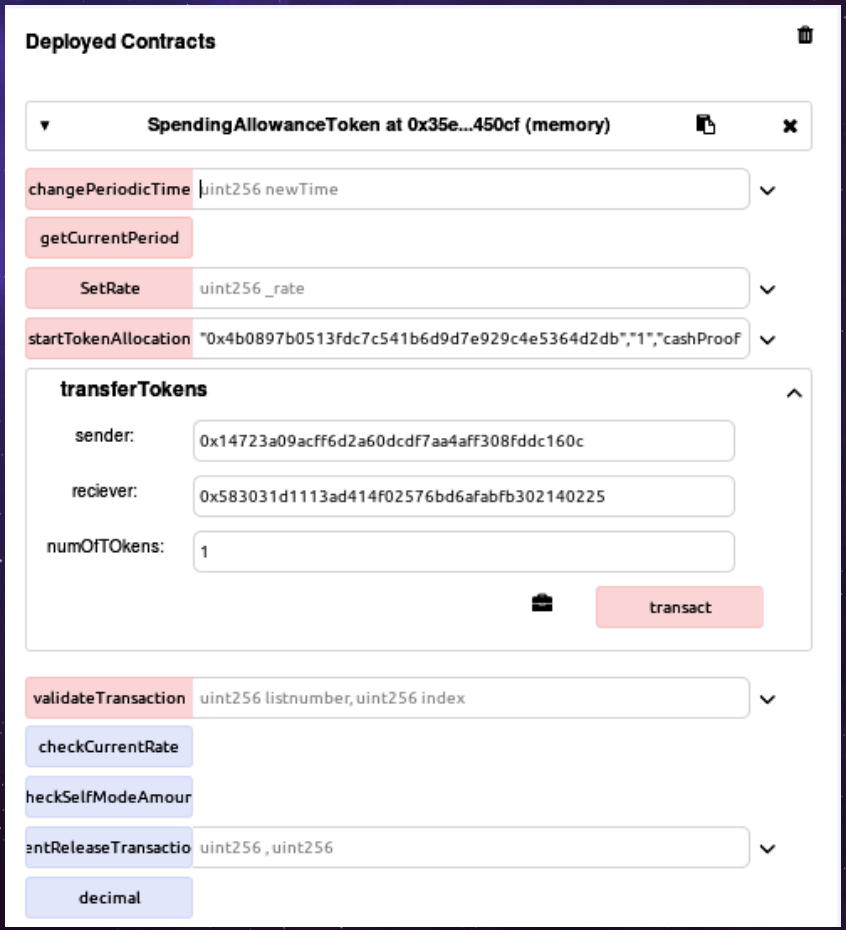
Test 1 C : Step 1

- Allocate 1 SAT to account A
- Allocate 1 SAT to account B



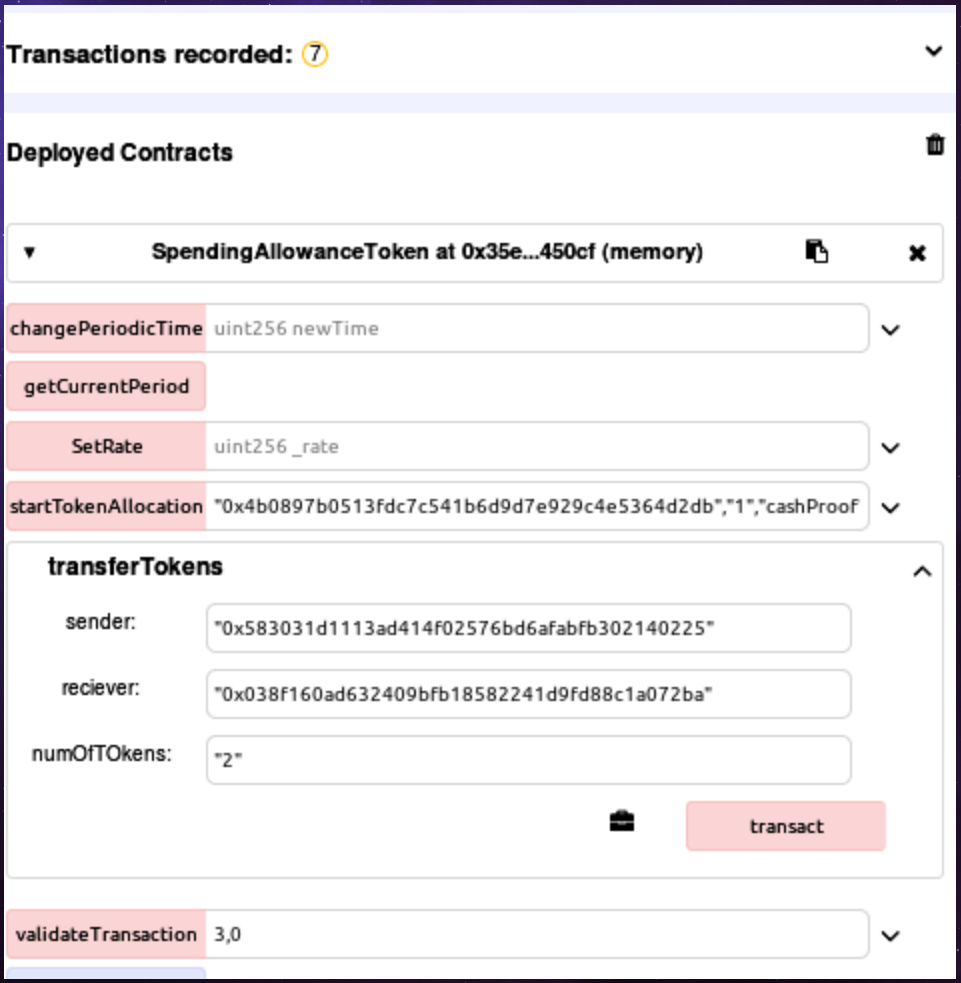
Test 1 C : Step 2

- A sends 1 SAT to C
- B sends 1 SAT to C



Test 1 C : Step 3

- C sends 2 SAT to SmartContract1

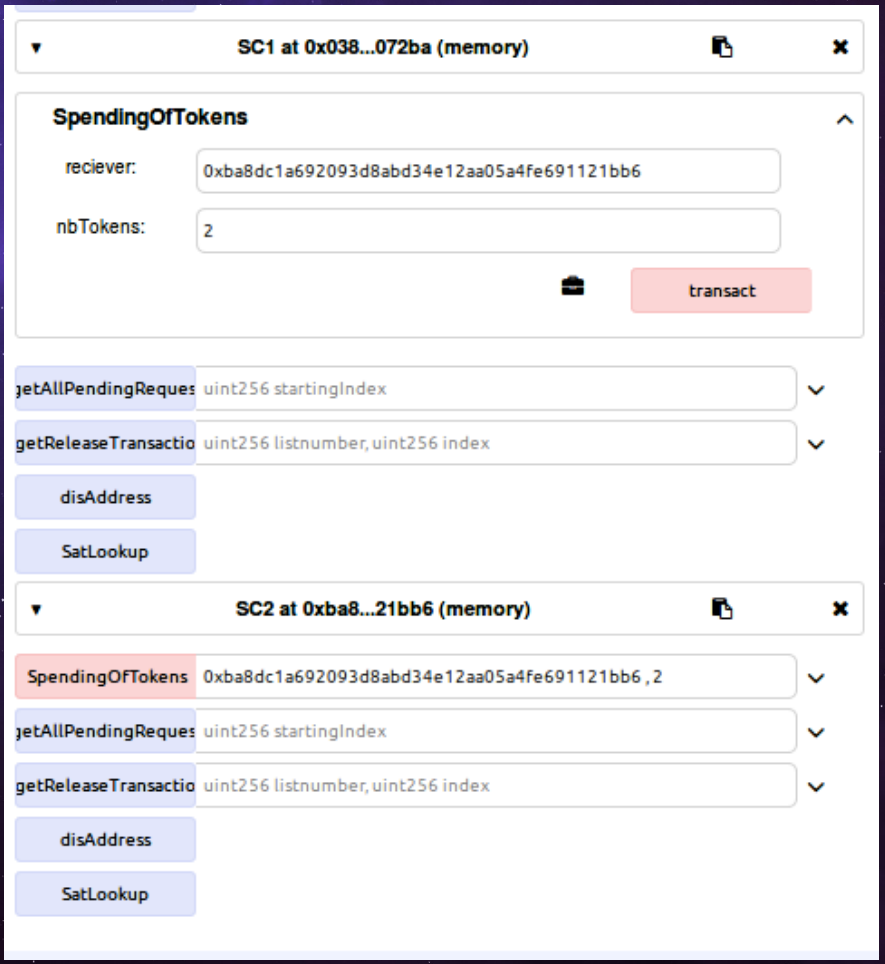


Test 1 C : Step 4

- Smart Contract1 sends 2 SAT to Smart Contract2

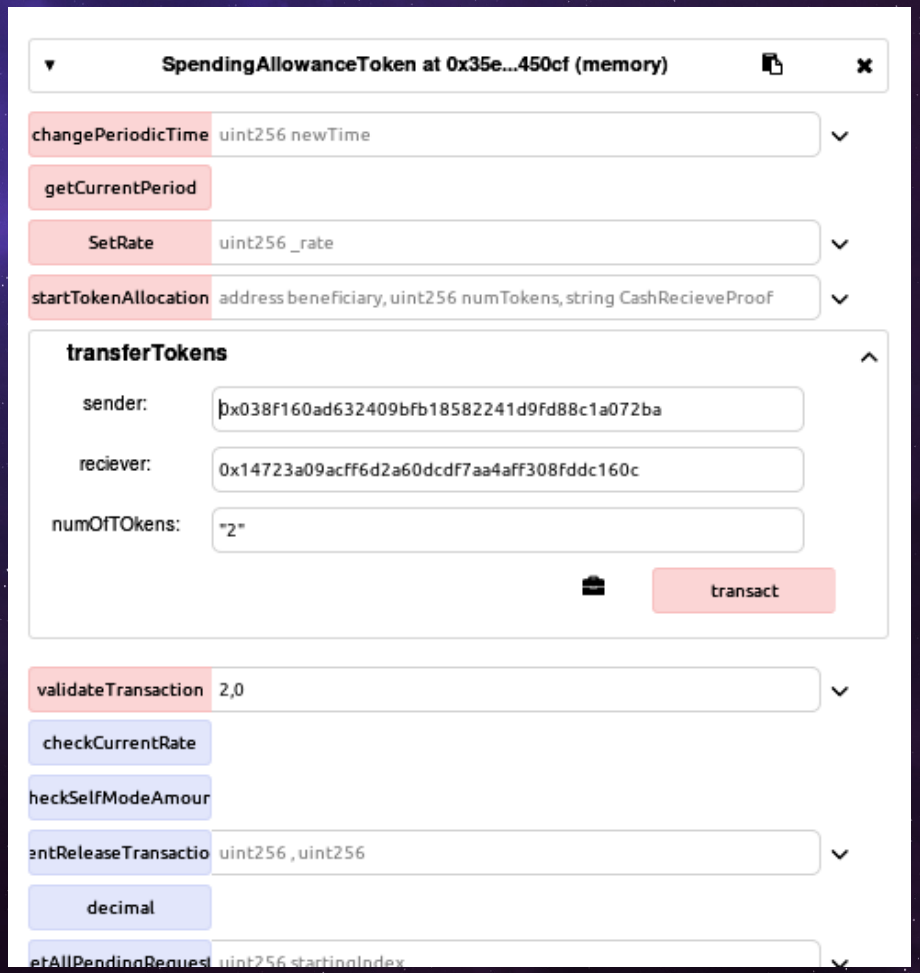
Test 1 C : Step 5

- Smart Contract2 sends 2 SAT to A



Test 1 C : Step 6

- Validating transaction from A



Test 1 C : Result

- Final Transaction
- A => A , 1 SAT (due to chaining , no actual transaction takes place)
- B => A , 1 SAT

Test 2

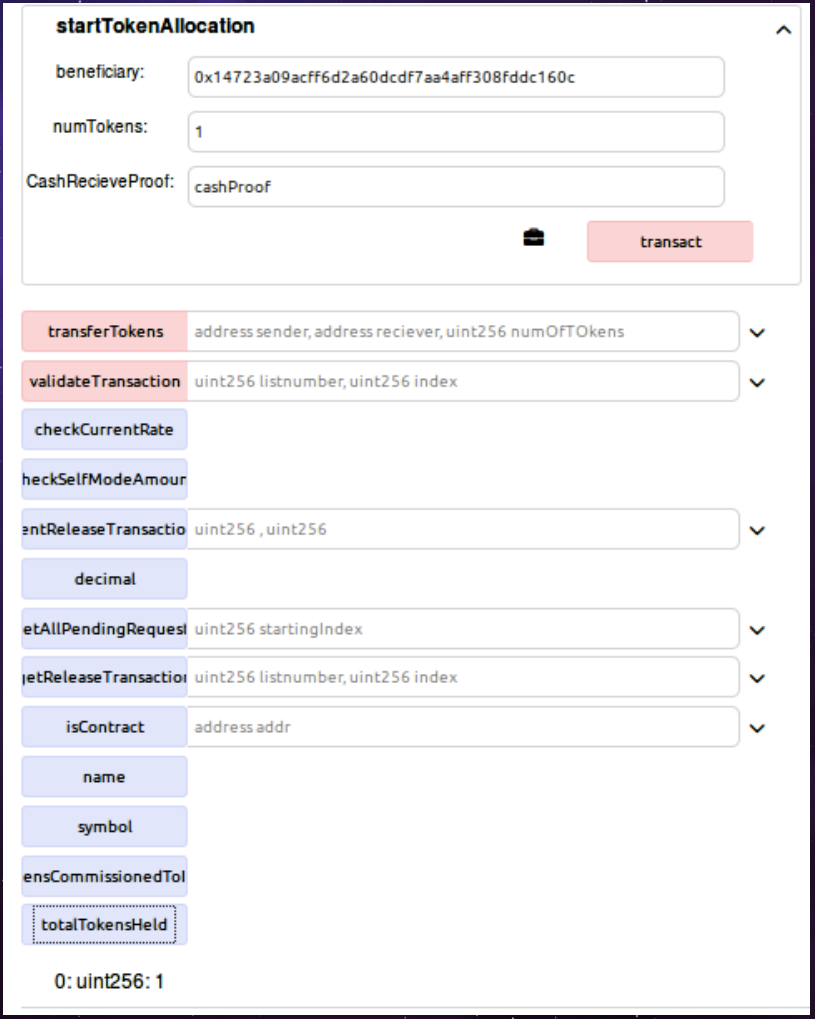
- Original Transaction
- A => B 1SAT 10 MIN
- B => C 2 SAT 20 MIN
- C => D 3 SAT 30 MIN
- D => A 4 SAT 40 MIN

- Expected Result
- AT PERIOD 1, WE SHOULD HAVE THE FOLLOWING PAYMENTS
- D => A (1 SAT)
- C => A (1 SAT)
- B => A (1 SAT)

Test 2 : Step 1

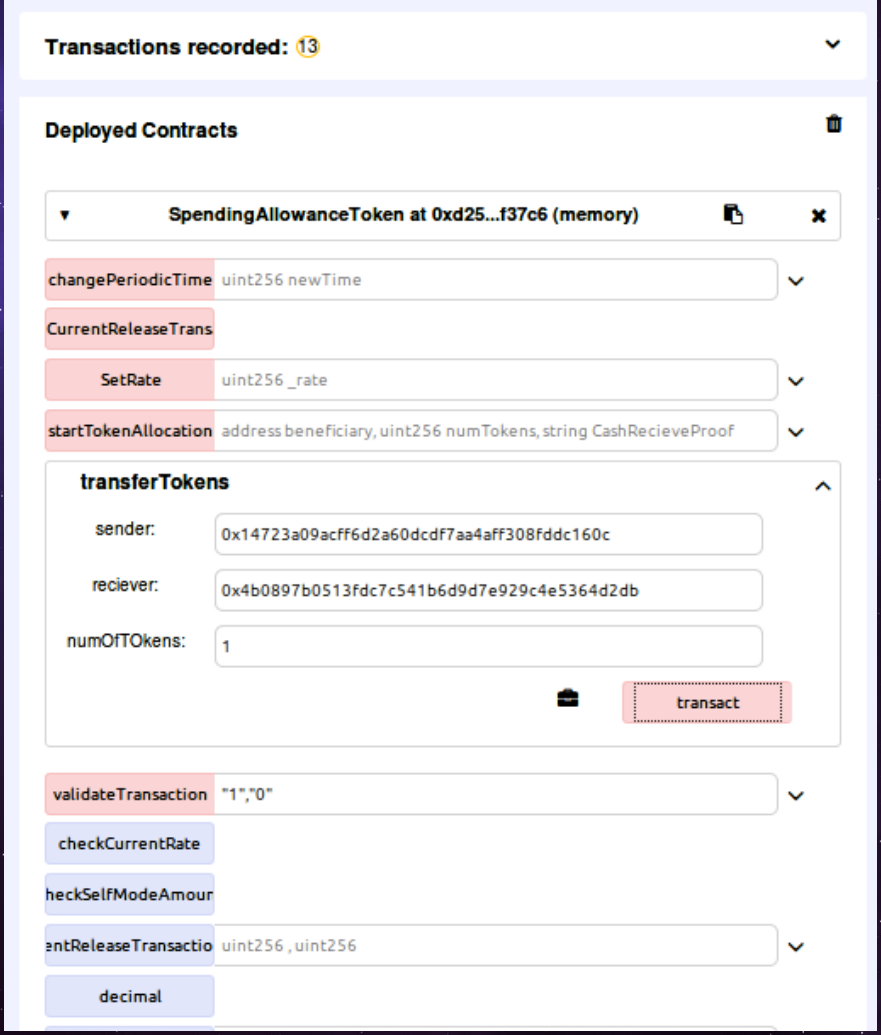
- Let account 2 be A ,
  Let account 3 be B ,
  Let account 4 be C ,

  Let account 5 be D
- Allocating 1 token to each account using function startTokenAllocation ()



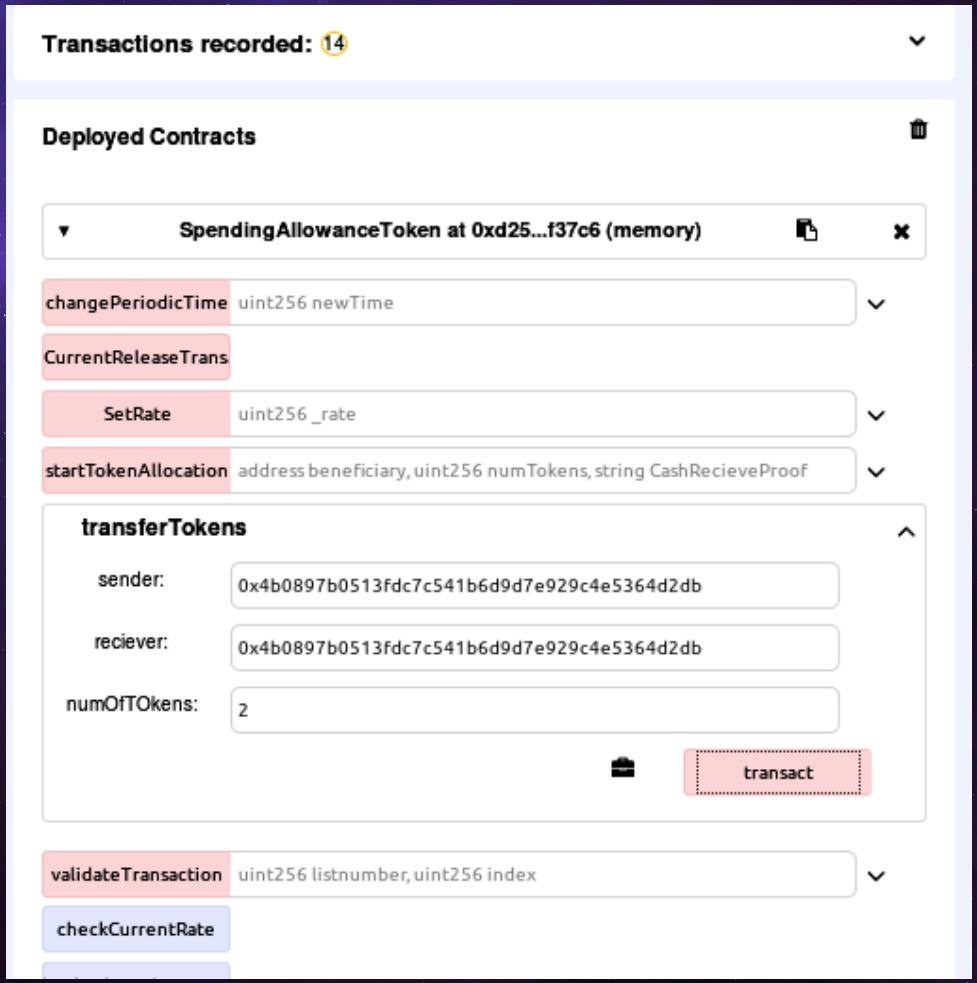
Test 2 : Step 2

- Sending 1 SAT from acount A to B
-



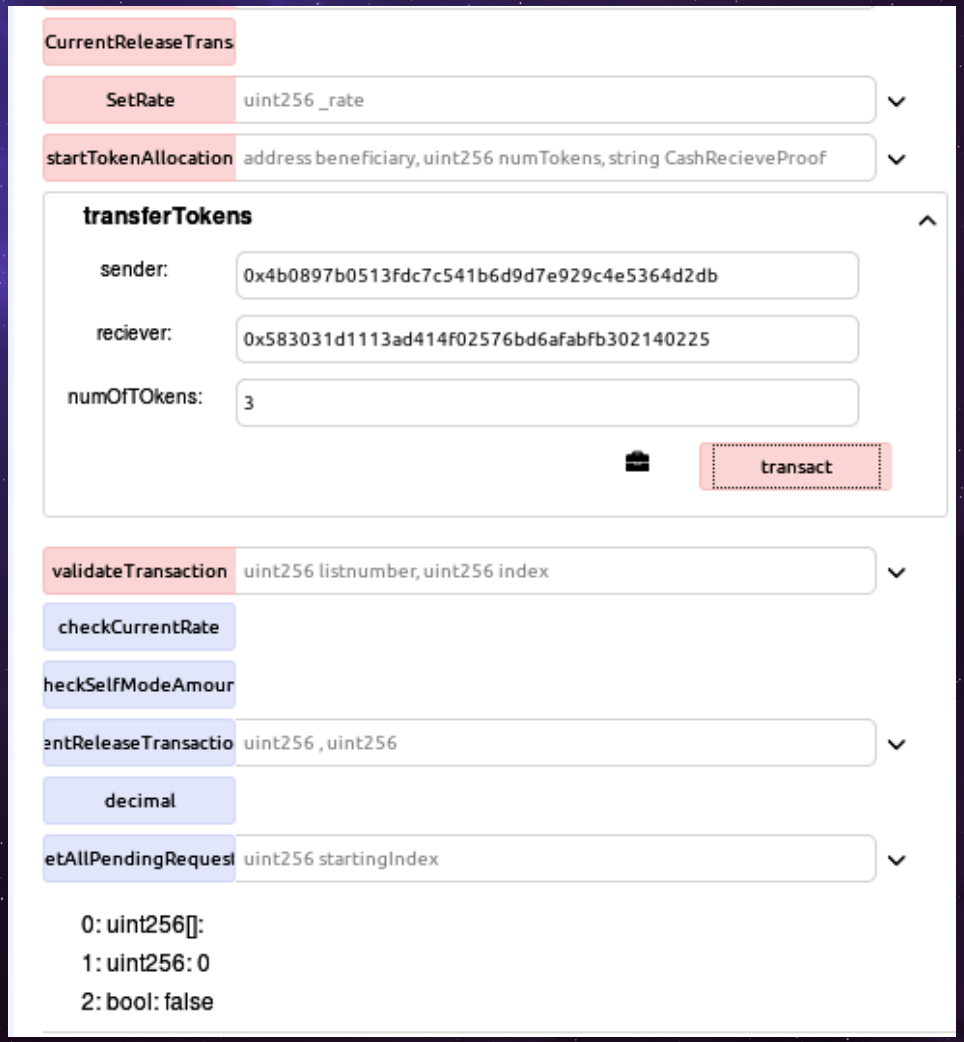
Test 2 : Step 3

- Sending 2 tokens from B to C
- Point to remember is that B only has 1 SAT but sending 2 SAT to C. This is where SAT logic comes in play by chaining transactions



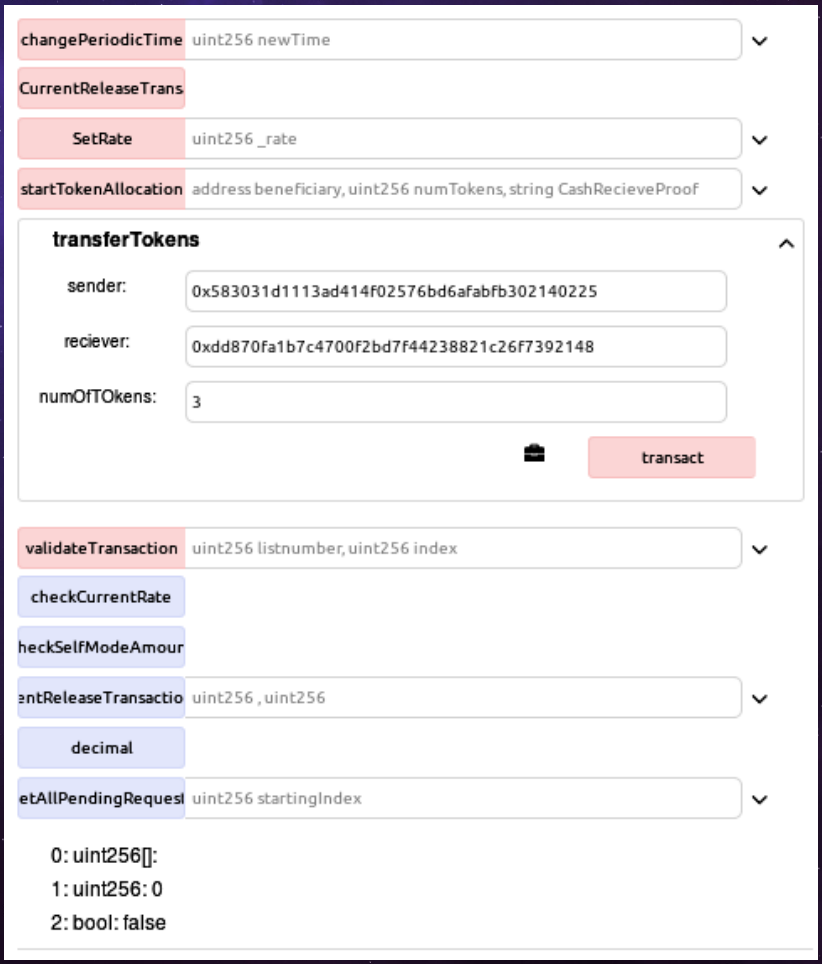
Test 2 : Step 4

- Sending 3 SAT from C to D



Test 2 : Step 5

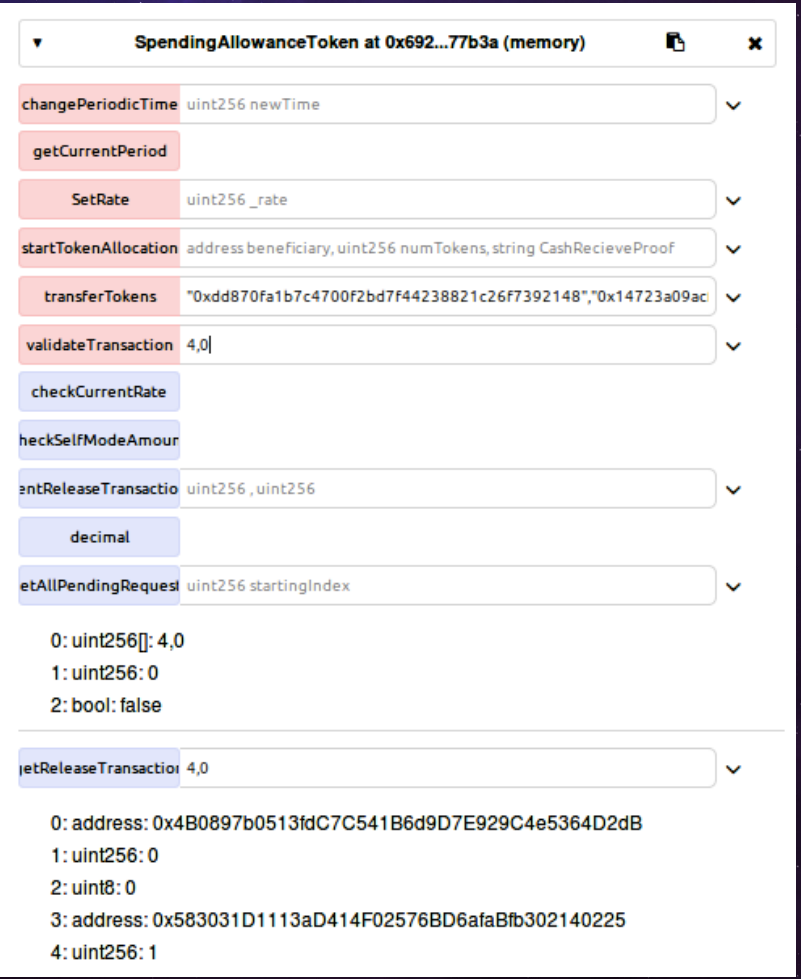
- Sending 4 SAT from D to A
- Same situation , here since D initially has 1 SAT only but tries to send 4 SAT to A



Test 2 : Step 6

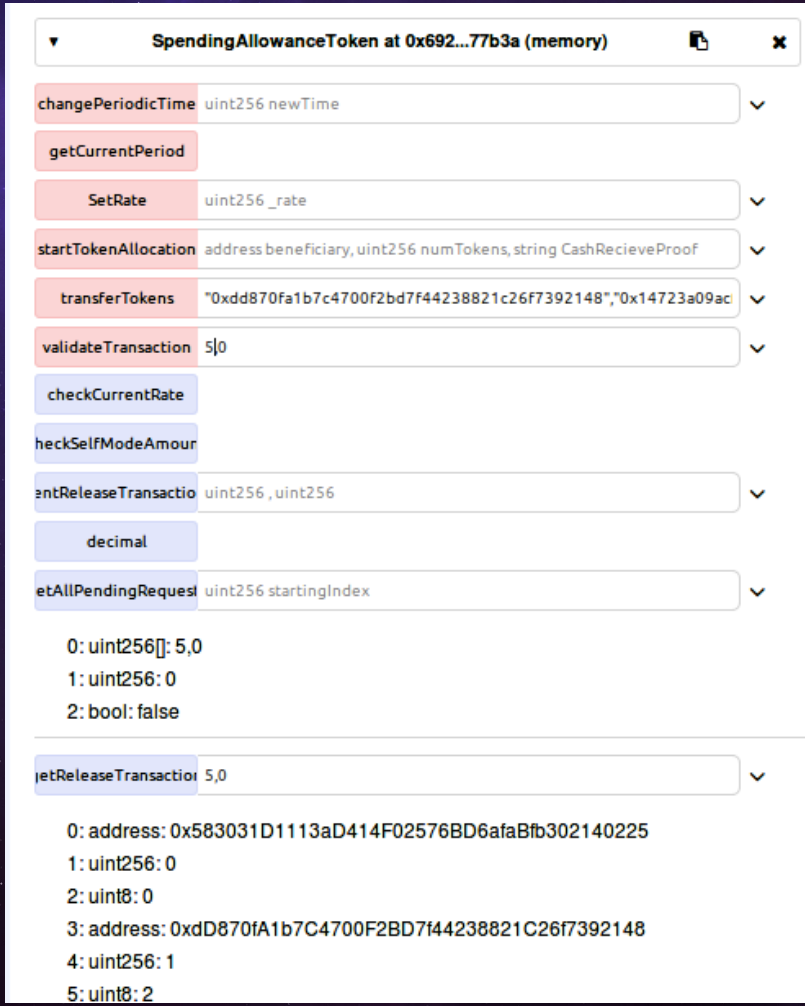
- Validating transacion from A
- Result :
- Original : A=>B (1 SAT)
- After chaining A=>B=>C=>D=>A

A=>A (1 SAT)



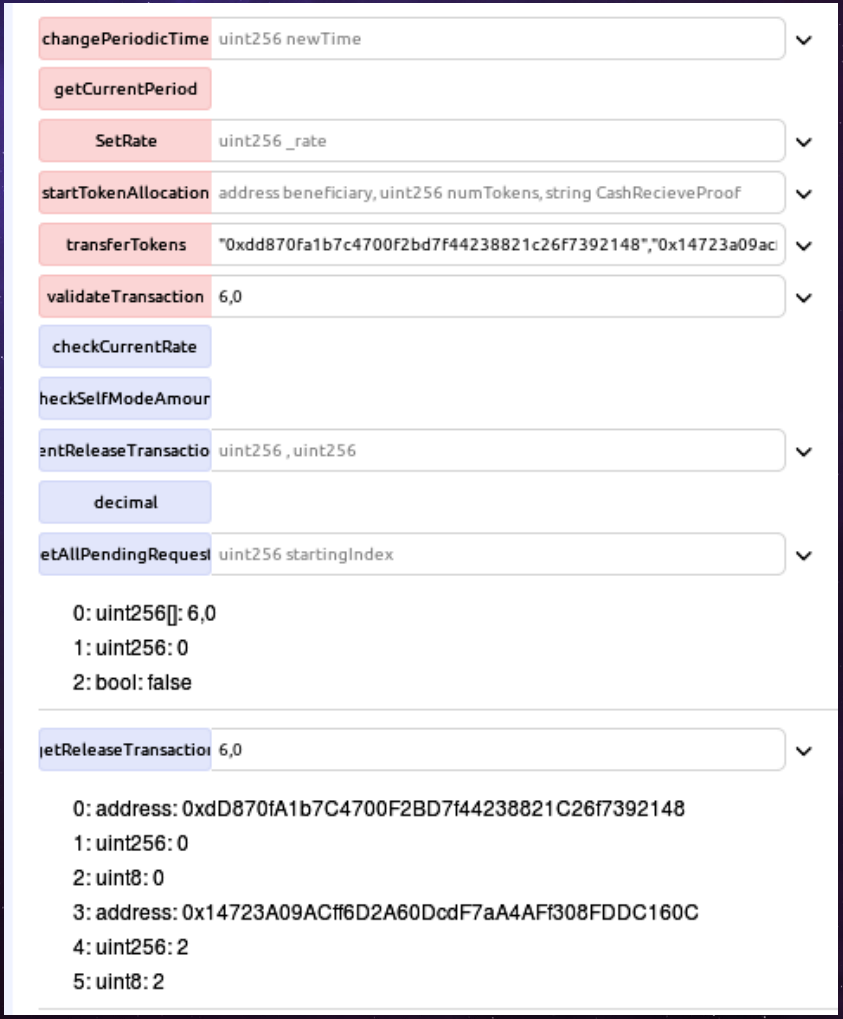
Test 2 : Step 7

- Validating transaction from B
- Result :
  Original
  B=>C (2 SAT)
- Chained
- B=>C=>D=>A
  B=>A (1 SAT)



Test 2 : Step 8

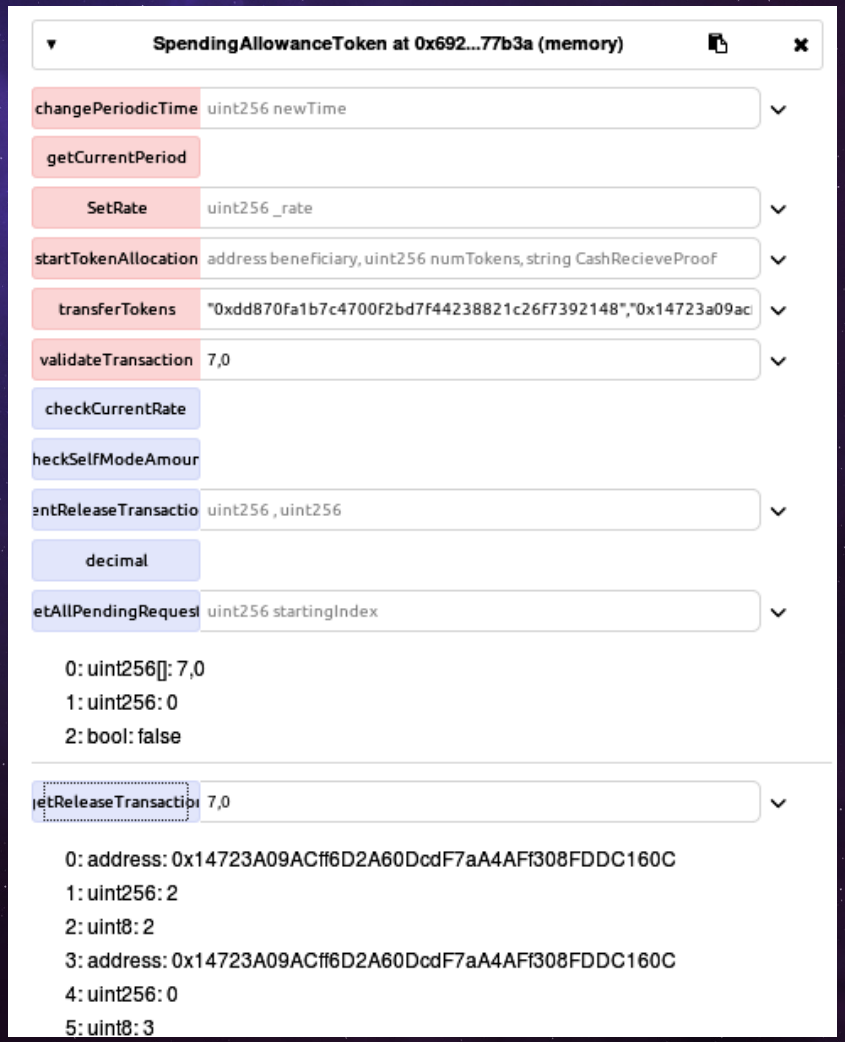
- Validating Trnasaction from C
- Original Transaction
  C => D
- Chained Transaction
  C => D => A (3 SAT)C=> A (1 SAt)



Test 2 : Step 9

- Validating Transaction from D
- Original Transaction :
  D => A
- Chained Transaction

D=>A (1 SAT)



Test 2 : Result

- Transaction which are generated from test case are :
  - A=>A (1 SAT, no actual transfer)
  - B=>A (1 SAT)
  - C=>A (1 SAT)
  - D=>A (1 SAT)

Test 3

- Original Transaction

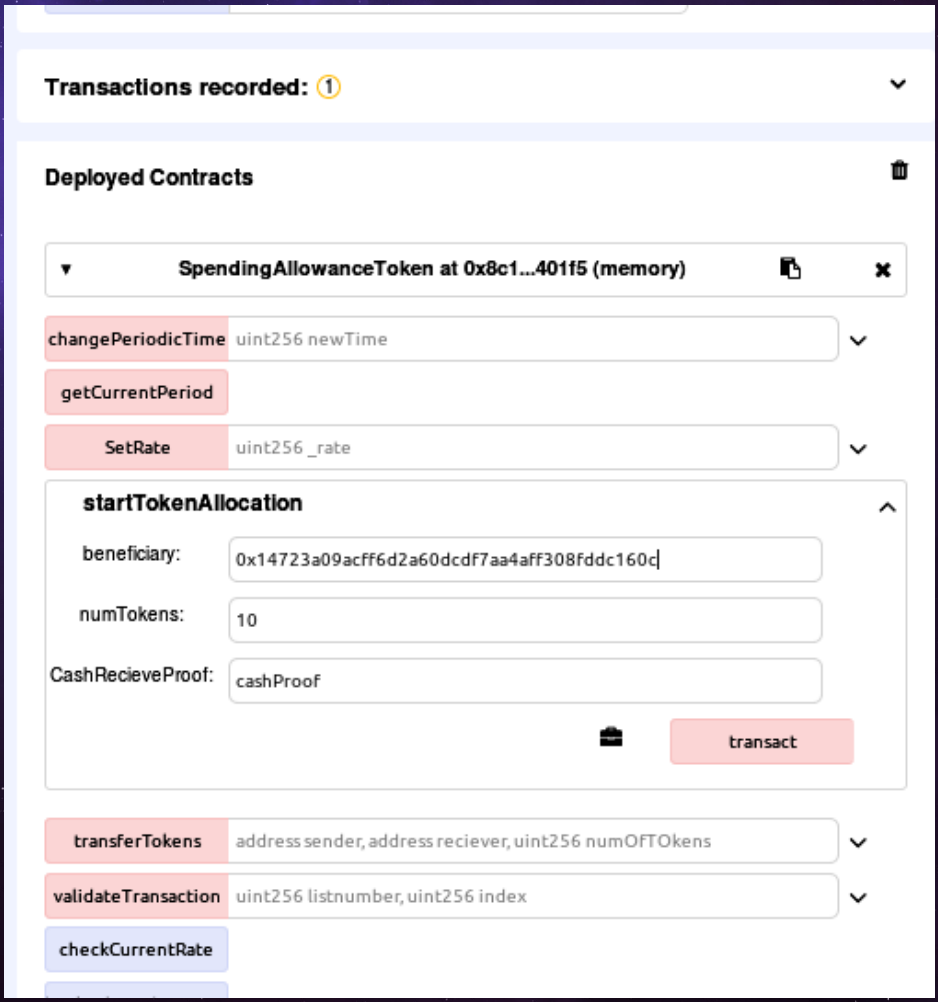
- A => B 10 SAT
- B => C 4 SAT
- B => D 4 SAT
- B => E 2 SAT
- C => F 2 SAT

- Expected Result

- AT PERIOD 1 :
  - A => C 4 SAT
  - A => D 4 SAT
  - A => E 2 SAT
  - A => F 2 SAT

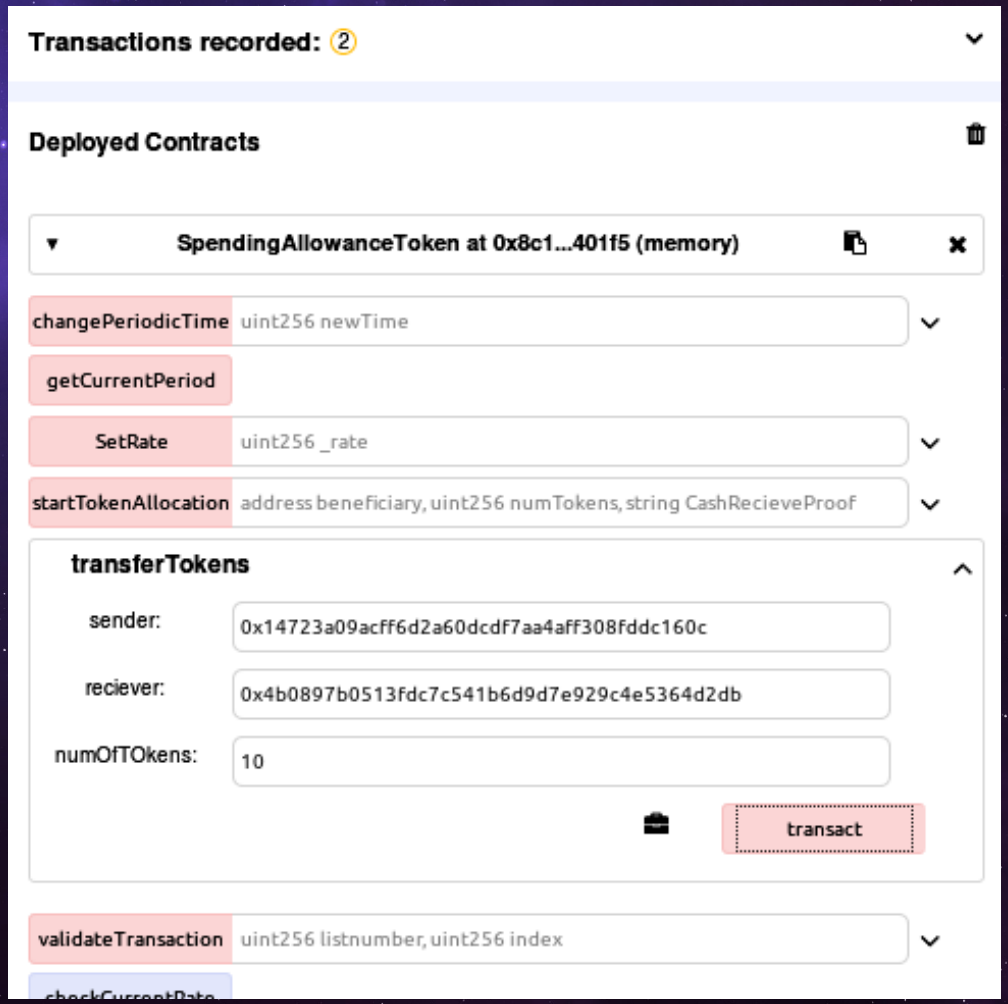
Test 3 : step 1

- Allocating 10 SAT to account A



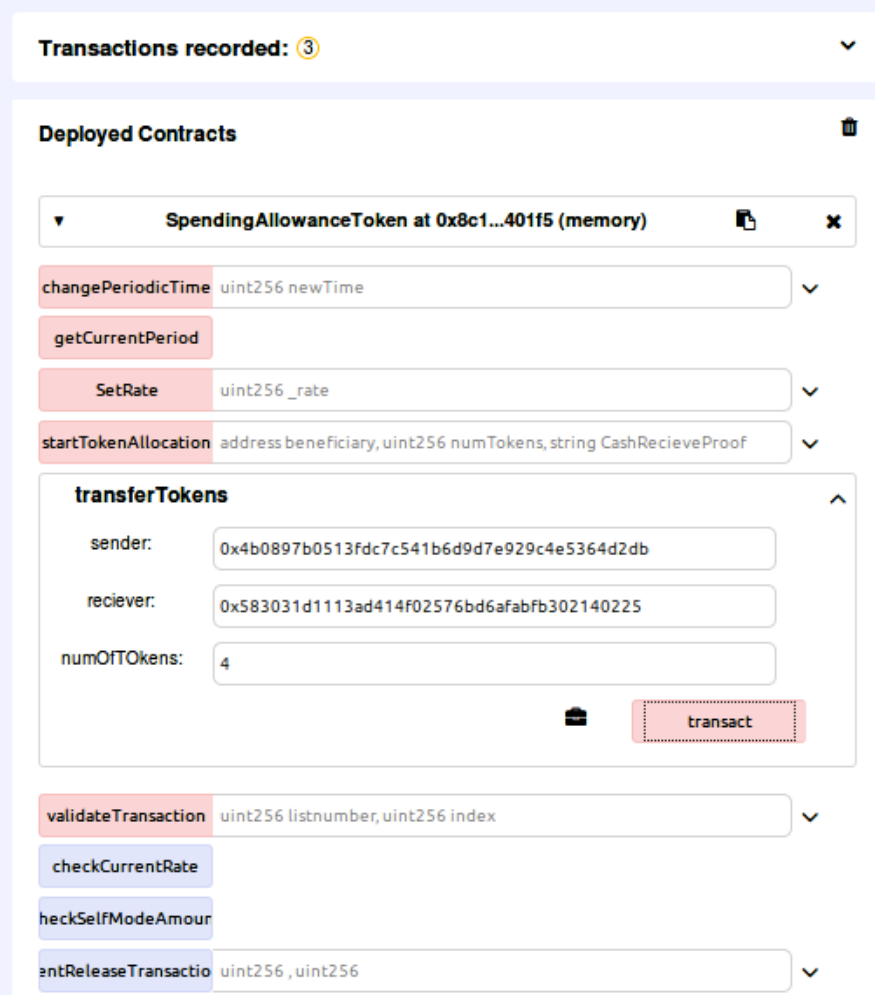
Test 3 : step 2

- Sending 10 SAT from A to B



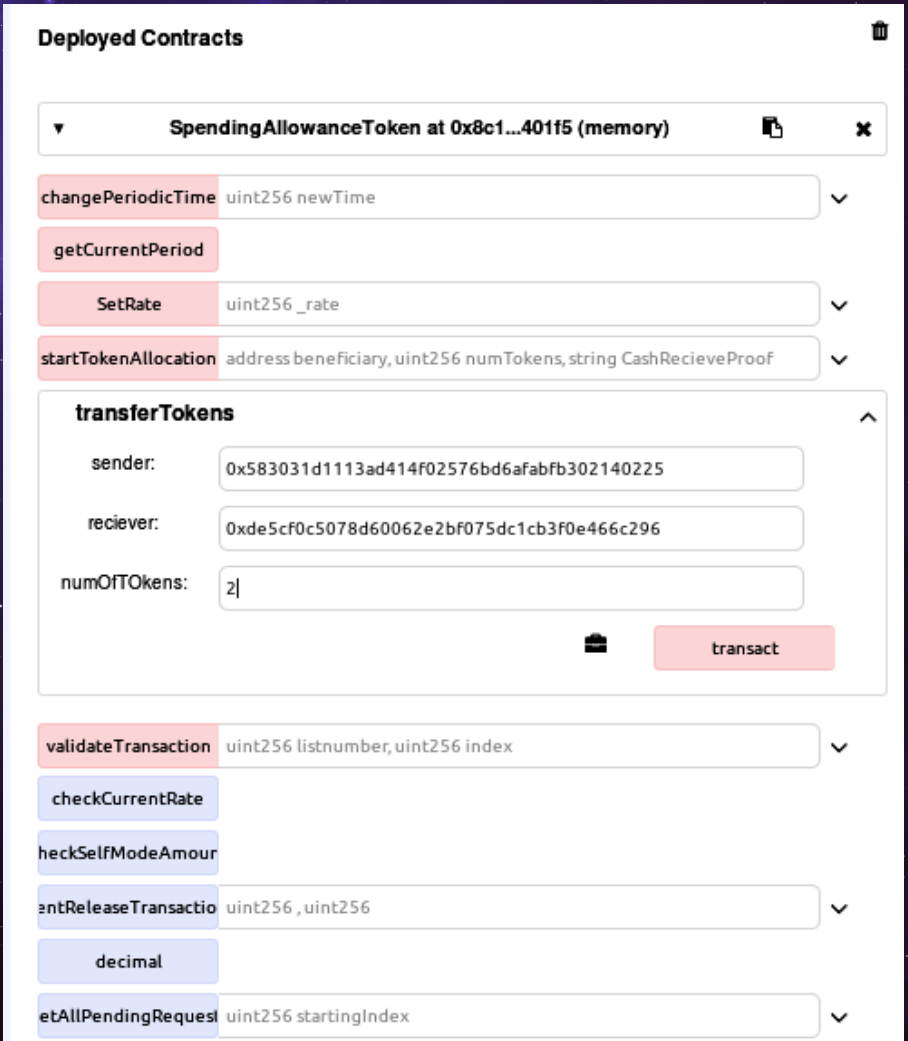
Test 3 : Step 3

- Sending SAT from B
  - B => C (4 SAT)
  - B => D (4 SAT)
  - B => E (2 SAT)



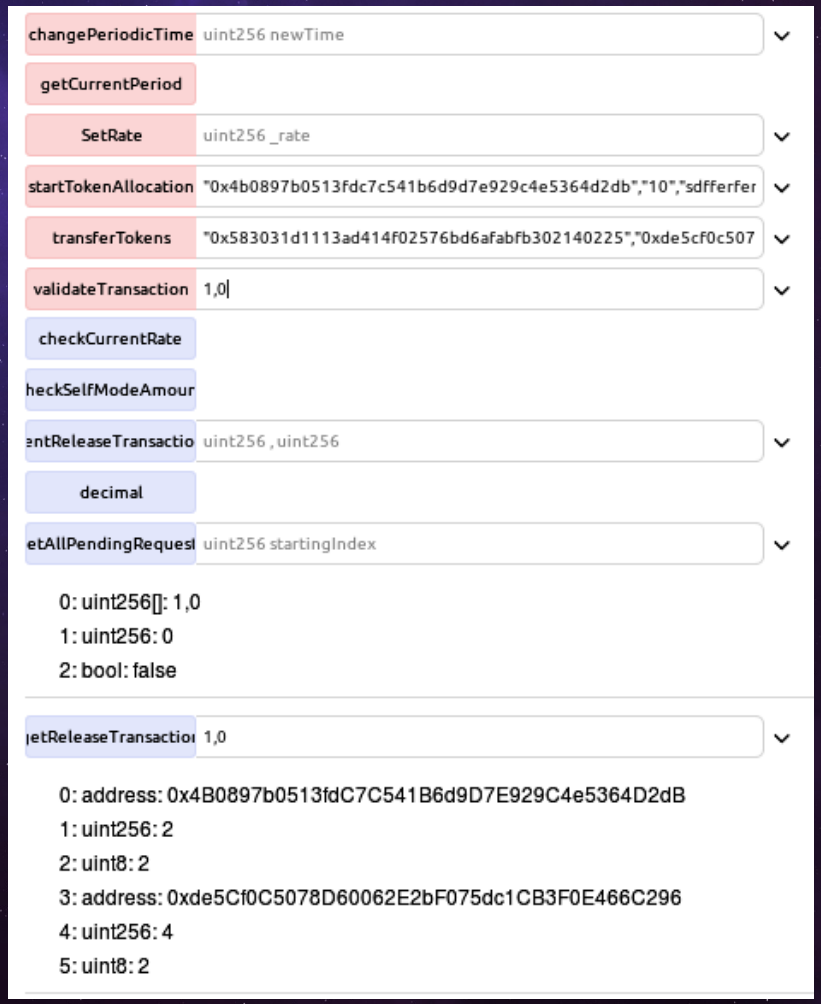
Test 3 : Step 4

- Sending 2 SAT from account C to F



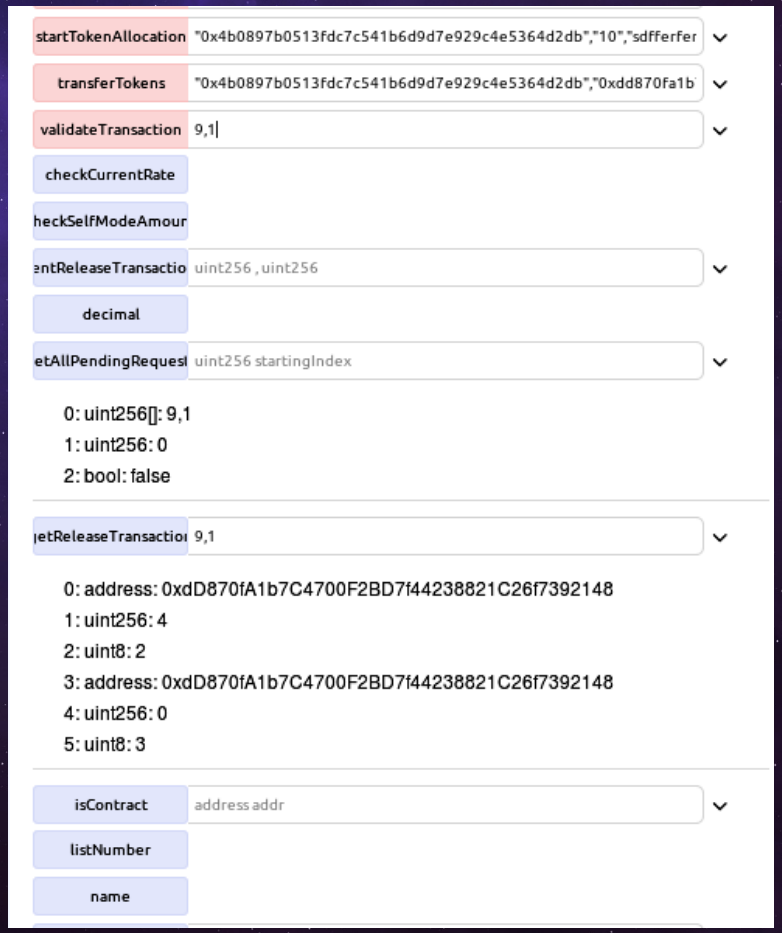
Test 3 : Step 5

- Validating transaction from A
- Original Transaction
  - A=>B (10 SAT)
- Chained Transaction
  - A => B=> C=>F (2 SAT)
  - A => B (8 SAT)



Test 3 : Step 6

- Validating transaction from B
- Transaction
  - B => D
  - B => E



Test 3 : Result

- Resulted Transaction :
  - A => C
  - A => D
  - A => E
  - A => F

Thank You