

# 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



METAMASK

- Remix browser-IDE

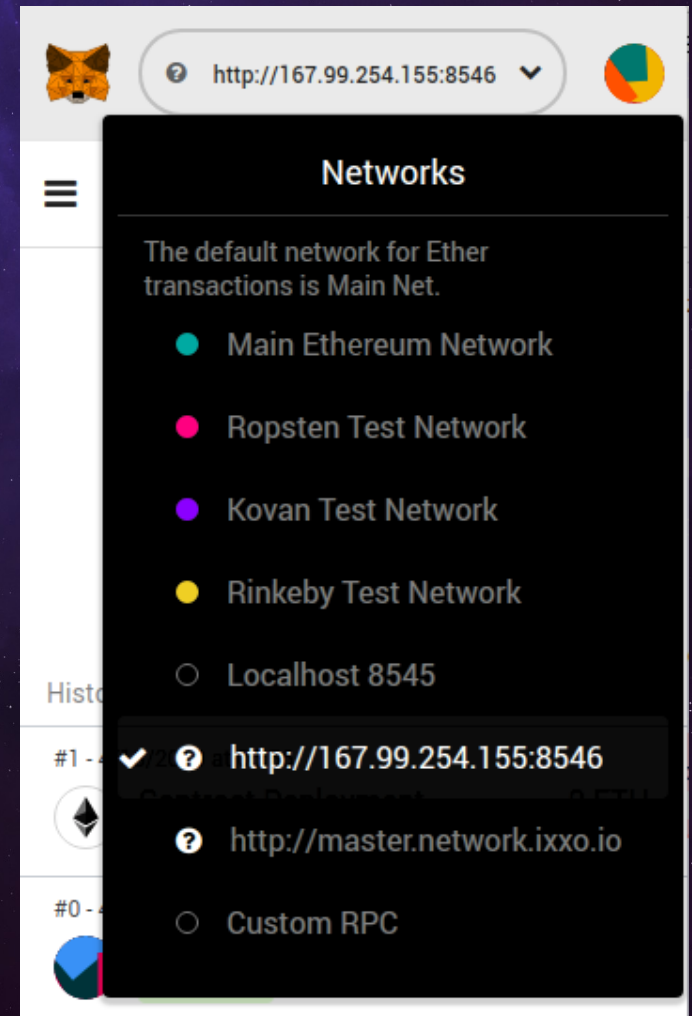


remix



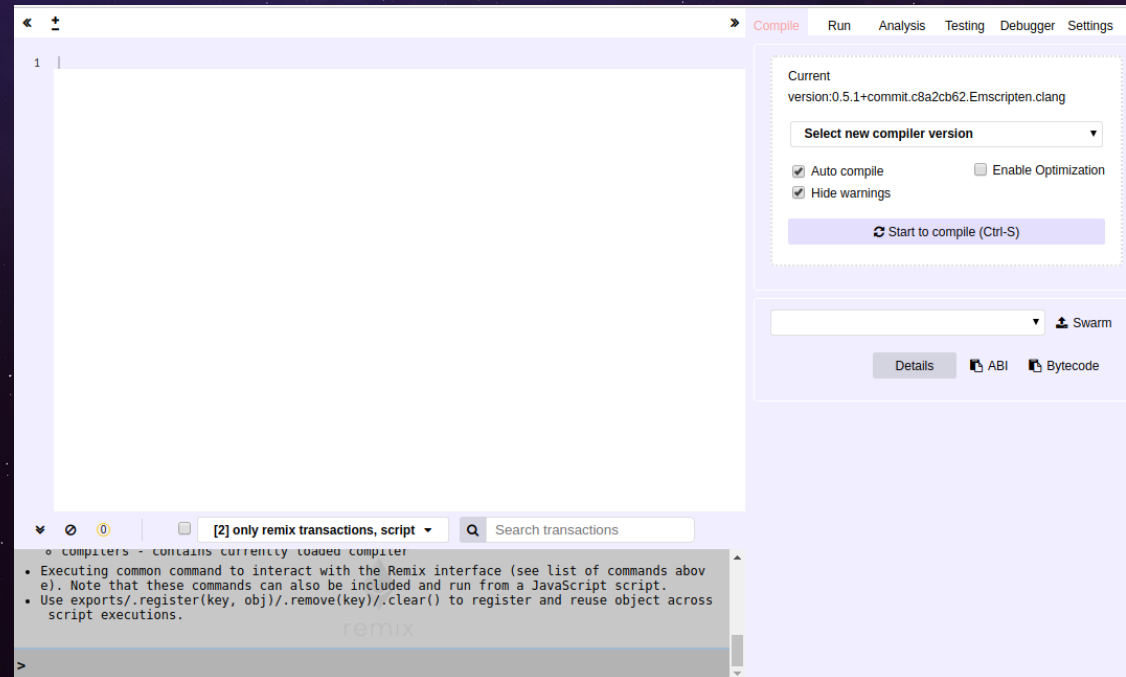
# Connecting to ixxo node

- 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

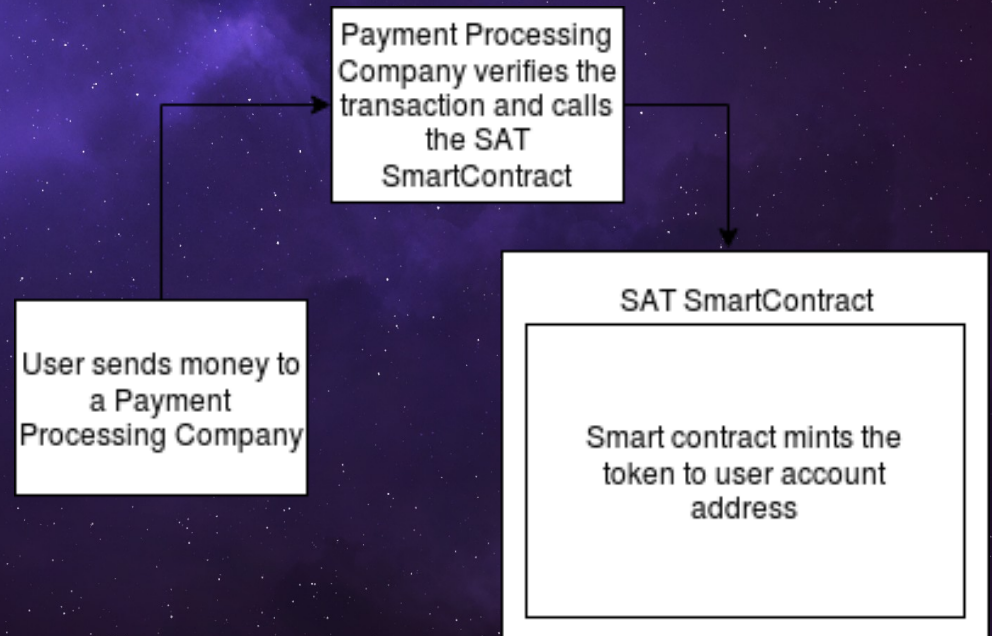
- Open browser and then go to [remix.ethereum.org](https://remix.ethereum.org)
- Remix is not just an IDE, but also provides facility to interact with already deployed contracts.
- We will use contract ABI and contract address to fetch the SAT instance and interact with it.
- Contract address is generated 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 ixco 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.



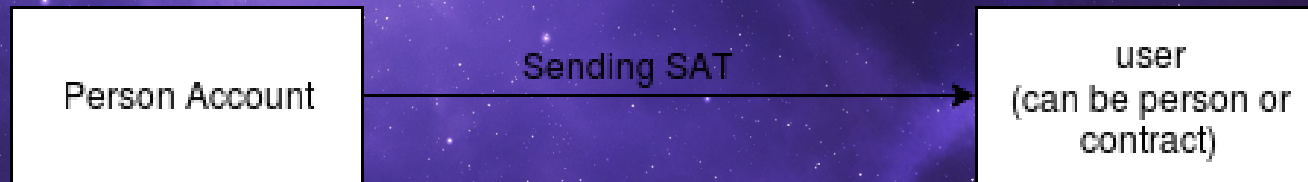
# Sending Of Tokens



- SAT are primary spending allowances in ixco network
- They can also be used as payment mode across the entities in the system
- If an entity sends SAT to other entity ixco 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

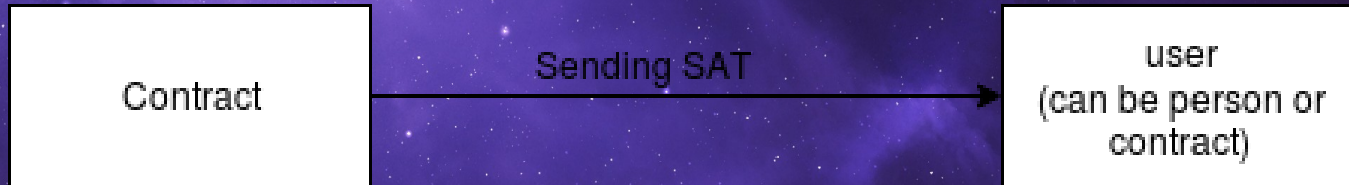


# Tokens originating form Person



- 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

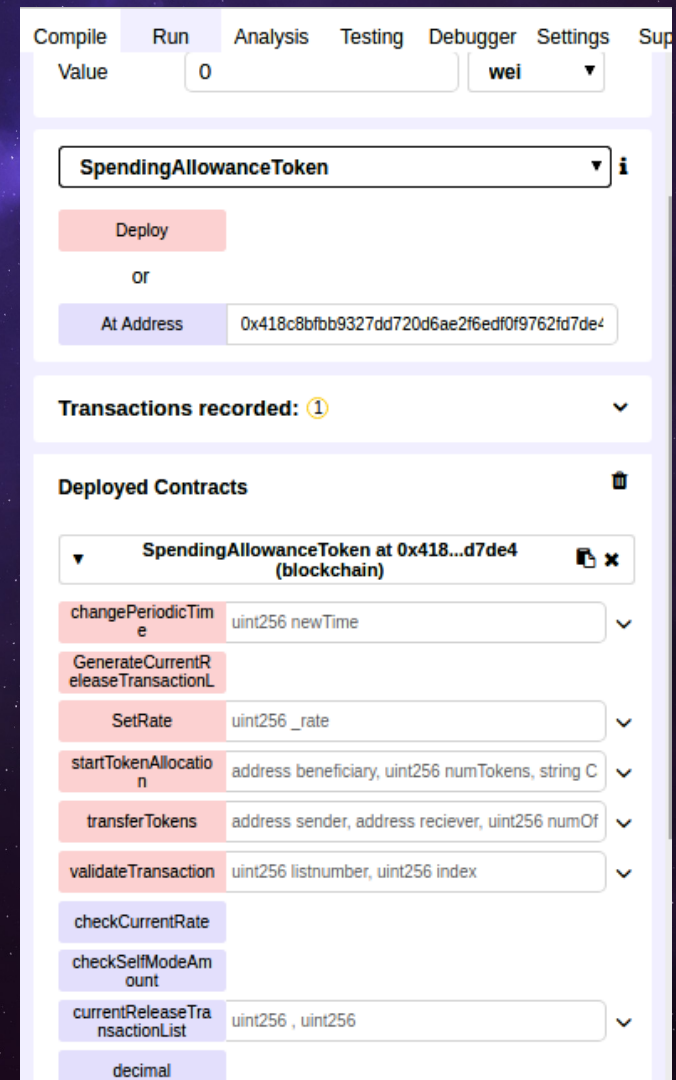
# Test 1 A

- Original Transaction
- Expected Result
- $A \Rightarrow \text{SmartContract1}$
- Transaction should be generated in second period Starting
- $\text{SmartContract1} \Rightarrow B$   
(after 30 seconds)



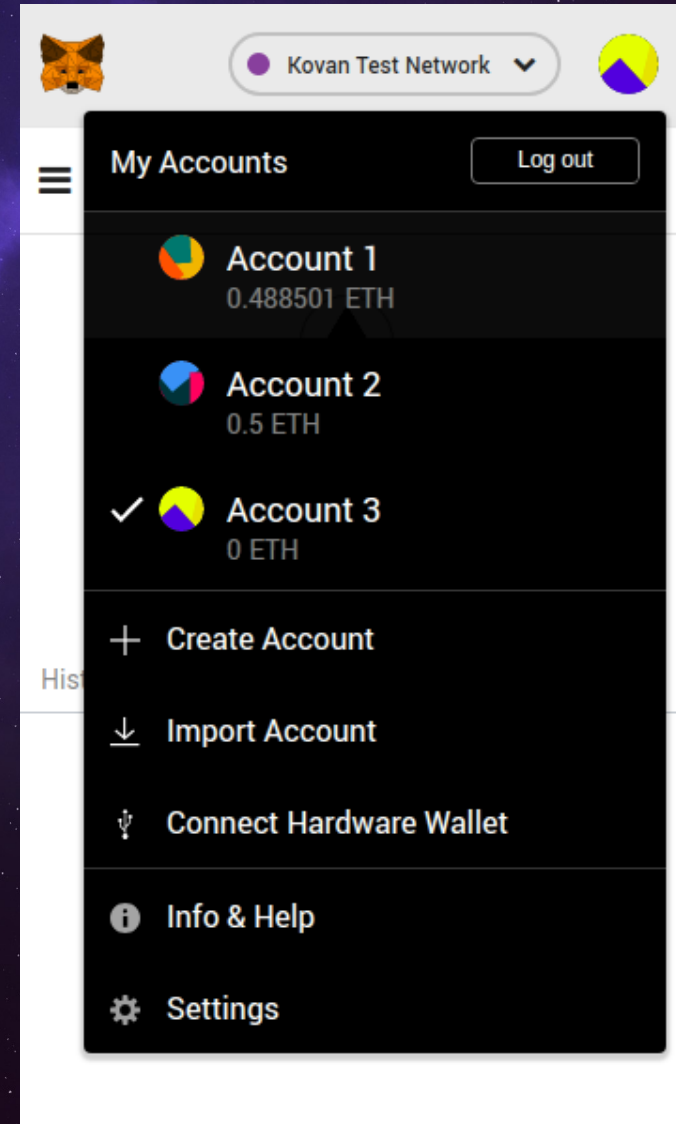
# Test 1 A : step 1

- 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.

**startTokenAllocation**

beneficiary: 0x14723a09acff6d2a60dcdF7aa4aff308fddc160c

numTokens: 1

CashRecieveProof: cashProof

 transact

transferTokens address sender, address reciever, uint256 numOFTokens

validateTransaction uint256 listnumber, uint256 index

checkCurrentRate

heckSelfModeAmour

entReleaseTransactio uint256 , uint256

decimal

etAllPendingRequest uint256 startingIndex

etReleaseTransactio uint256 listnumber, uint256 index

isContract address addr

name

symbol

ensCommissionedTol

totalTokensHeld

0: uint256: 1

# Test 1 A : Step 4.1

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

**transferTokens**

sender:


0x14723a09acff6d2a60dcdF7aa4aff308fddc160c

reciever:

0xe3632b9ab0571d2601e804dfddc65eb51ab19310

numOfTokens:

1



transact

validateTransaction

uint256 listnumber, uint256 index

▼

checkCurrentRate

heckSelfModeAmour

entReleaseTransaction

uint256 , uint256

▼

decimal

etAllPendingRequest

uint256 startingIndex

▼

etReleaseTransaction

uint256 listnumber, uint256 index

▼

isContract

address addr

▼

name

symbol

ensCommissionedToI

totalTokensHeld

0: uint256: 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 .

SetRate	uint256 _rate	▼
startTokenAllocation	address beneficiary, uint256 numTokens, string CashRecieveProof	▼
transferTokens	address sender, address reciever, uint256 numOFTokens	▼
validateTransaction	uint256 listnumber, uint256 index	▼
checkCurrentRate		
heckSelfModeAmour		
entReleaseTransaction	uint256 , uint256	▼
decimal		
etAllPendingRequest	uint256 startingIndex	▼
0: uint256[]: 1,0 1: uint256: 0 2: bool: false		
etReleaseTransaction	uint256 listnumber, uint256 index	▼
isContract	address addr	▼
name		
symbol		
ensCommissionedTol		
totalTokensHeld		

# Test 1 A : Step 4.3

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

SetRate	uint256 _rate	▼
startTokenAllocation	address beneficiary, uint256 numTokens, string CashReclieveProof	▼
transferTokens	address sender, address reciever, uint256 numOFTokens	▼
validateTransaction	uint256 listnumber, uint256 index	▼
checkCurrentRate		
heckSelfModeAmour		
entReleaseTransactio	uint256 , uint256	▼
decimal		
etAllPendingRequest	uint256 startingIndex	▼
0: uint256[]: 1,0 1: uint256: 0 2: bool: false		
getReleaseTransactio	1,0	▼
0: address: 0xe3632B9aB0571d2601e804DfDDc65EB51AB19310 1: uint256: 1 2: uint8: 2 3: address: 0xe3632B9aB0571d2601e804DfDDc65EB51AB19310 4: uint256: 0 5: uint8: 2		
isContract	address addr	▼
name		
symbol		



# 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)

SC1 at 0xe36...19310 (memory)

**SpendingOfTokens**

reciever: 0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db

nbTokens: 1

transact

getAllPendingReques uint256 startingIndex




getReleaseTransactio uint256 listnumber, uint256 index



disAddress


SatLookup


# 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

Account  0x147...c160c (99.99999999999823i)  

Gas limit 6000000  


Value 0 wei 


SC1 



Deploy 0xd25ed029c093e56bc8911a07c46545000cbf37c6 


or

At Address Load contract from Address


Transactions recorded: 9 


Deployed Contracts 


▼ SpendingAllowanceToken at 0xd25...f37c6 (memory)  


changePeriodicTime uint256 newTime 

CurrentReleaseTrans

SetRate uint256 \_rate 

startTokenAllocation address beneficiary,uint256 numTokens,string CashRecieveProof 

transferTokens address sender,address reciever,uint256 numOFTokens 

validateTransaction "1","0" 

checkCurrentRate



# Test 1 A : Result

- On validation all the transaction are chained and released in second period as :
- $A \Rightarrow \text{SmartContract1} \Rightarrow B$  , period2 (Original)
- $A \Rightarrow B$  , period2 (chained)

# Test 1 B

- Original Transaction
  - Expected Result
- 
- 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
- 
- At end of Period1 :
  - $A \Rightarrow C$  (1 SAT)
  - $B \Rightarrow C$  (1 SAT)




# Test 1 B : Step 1


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

Deployed Contracts

▼

SpendingAllowanceToken at 0x35e...450cf (memory)





changePeriodicTime

uint256 newTime

▼

getCurrentPeriod

SetRate

uint256 \_rate

▼

startTokenAllocation

^

beneficiary:


0x14723a09acff6d2a60dcdf7aa4aff308fddc160c

numTokens:

1

CashRecieveProof:

cashProof|



transact

transferTokens

address sender, address reciever, uint256 numOFTokens

▼

validateTransaction

uint256 listnumber, uint256 index

▼

checkCurrentRate

heckSelfModeAmour

entReleaseTransactio

uint256 ,uint256

▼

# Test 1 B : Step 2

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

**Deployed Contracts**

▼ SpendingAllowanceToken at 0x35e...450cf (memory) 📄 ✕

changePeriodicTime

uint256 newTime

▼

getCurrentPeriod

SetRate

uint256 \_rate

▼

startTokenAllocation

"0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db","1","cashProof"

▼

transferTokens

^

sender:

0x14723a09acff6d2a60dcdf7aa4aff308fddc160c

reciever:

0x583031d1113ad414f02576bd6afabfb302140225

numOfTokens:

1

📁

transact

validateTransaction

uint256 listnumber, uint256 index

▼

checkCurrentRate

heckSelfModeAmour

entReleaseTransactio

uint256 , uint256

▼

decimal



# Test 1 B : Step 3

- Validate transaction from A
- Validate Transaction from B

getCurrentPeriod	
SetRate	uint256_rate
startTokenAllocation	"0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db","1","cashProof"
transferTokens	"0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db","0x583031d11"
validateTransaction	3,0
checkCurrentRate	
checkSelfModeAmount	
getReleaseTransaction	uint256 ,uint256
decimal	
getAllPendingRequests	uint256 startingIndex

0: uint256[]: 3,0  
1: uint256: 0  
2: bool: false

---

getReleaseTransaction	3,0
-----------------------	-----

0: address: 0x583031D1113aD414F02576BD6afaBfb302140225  
1: uint256: 1  
2: uint8: 2  
3: address: 0x583031D1113aD414F02576BD6afaBfb302140225  
4: uint256: 0  
5: uint8: 3

# Test 1 B : Results

- Transactions :
  - $A \Rightarrow C$
  - $B \Rightarrow C$
- 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

**Deployed Contracts**

▼

SpendingAllowanceToken at 0x35e...450cf (memory)

changePeriodicTime

uint256 newTime

▼

getCurrentPeriod

SetRate

uint256 \_rate

▼

startTokenAllocation

^

beneficiary:

0x14723a09acff6d2a60dcdf7aa4aff308fddc160c

numTokens:

1

CashRecieveProof:

cashProof

transact

transferTokens

address sender, address reciever, uint256 numOFTokens

▼

validateTransaction

uint256 listnumber, uint256 index

▼

checkCurrentRate

heckSelfModeAmour

entReleaseTransactio

uint256 , uint256

▼



# Test 1 C : Step 2

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

Deployed Contracts

▼ SpendingAllowanceToken at 0x35e...450cf (memory) 📄 ✕

changePeriodicTime

uint256 newTime

▼

getCurrentPeriod

SetRate

uint256 \_rate

▼

startTokenAllocation

"0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db","1","cashProof"

▼

transferTokens

^

sender:

0x14723a09acff6d2a60dcd77aa4aff308fddc160c

reciever:

0x583031d1113ad414f02576bd6afabfb302140225

numOfTokens:

1

📁

transact

validateTransaction

uint256 listnumber, uint256 index

▼

checkCurrentRate

heckSelfModeAmour

entReleaseTransactio

uint256 , uint256

▼

decimal

# Test 1 C : Step 3

- C sends 2 SAT to SmartContract1

Transactions recorded: 7

Deployed Contracts

▼ SpendingAllowanceToken at 0x35e...450cf (memory) [icon] x

changePeriodicTime uint256 newTime ▼

getCurrentPeriod

SetRate uint256 \_rate ▼

startTokenAllocation "0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db","1","cashProof" ▼

**transferTokens** ^

sender: "0x583031d1113ad414f02576bd6afabfb302140225"

reciever: "0x038f160ad632409bfb18582241d9fd88c1a072ba"

numOfTokens: "2"

[icon] transact

validateTransaction 3,0 ▼



# Test 1 C : Step 4


- Smart Contract1 sends 2 SAT to Smart Contract2

SC1 at 0x038...072ba (memory)

SpendingOfTokens

reciever: 0xba8dc1a692093d8abd34e12aa05a4fe691121bb6

nbTokens: 2

 transact

getAllPendingRequests

uint256 startingIndex

getReleaseTransaction

uint256 listnumber, uint256 index

disAddress

SatLookup

SC2 at 0xba8...21bb6 (memory)

SpendingOfTokens

address reciever, uint256 nbTokens

getAllPendingRequests

uint256 startingIndex

getReleaseTransaction

uint256 listnumber, uint256 index

disAddress

SatLookup

# Test 1 C : Step 5

- Smart Contract2 sends 2 SAT to A

▼ SC1 at 0x038...072ba (memory) 📄 ✕

SpendingOfTokens ^

reciever: 0xba8dc1a692093d8abd34e12aa05a4fe691121bb6

nbTokens: 2

👛 transact

getAllPendingReques uint256 startingIndex ▼

getReleaseTransactio uint256 listnumber, uint256 index ▼

disAddress

SatLookup

▼ SC2 at 0xba8...21bb6 (memory) 📄 ✕

SpendingOfTokens 0xba8dc1a692093d8abd34e12aa05a4fe691121bb6, 2 ▼

getAllPendingReques uint256 startingIndex ▼

getReleaseTransactio uint256 listnumber, uint256 index ▼

disAddress

SatLookup



# Test 1 C : Step 6

- Validating transaction from A

▼ SpendingAllowanceToken at 0x35e...450cf (memory) 📄 ✕

changePeriodicTime

uint256 newTime

▼

getCurrentPeriod

SetRate

uint256 \_rate

▼

startTokenAllocation

address beneficiary, uint256 numTokens, string CashRecieveProof

▼

transferTokens

^

sender:

0x038f160ad632409bfb18582241d9fd88c1a072ba

reciever:

0x14723a09acff6d2a60dcdf7aa4aff308fddc160c

numOfTokens:

"2"

🏠

transact

validateTransaction

2,0

▼

checkCurrentRate

heckSelfModeAmour

entReleaseTransactio

uint256 , uint256

▼

decimal

etAllPendingRequest

uint256 startingIndex

▼

# Test 1 C : Result

- Final Transaction
- $A \Rightarrow A$  , 1 SAT (due to chaining , no actual transaction takes place)
- $B \Rightarrow A$  , 1 SAT



# Test 2

- Original Transaction
- $A \Rightarrow B$  1 SAT 10 MIN
- $B \Rightarrow C$  2 SAT 20 MIN
- $C \Rightarrow D$  3 SAT 30 MIN
- $D \Rightarrow A$  4 SAT 40 MIN
- Expected Result
- AT PERIOD 1, WE SHOULD HAVE THE FOLLOWING PAYMENTS
- $D \Rightarrow A$  (1 SAT)
- $C \Rightarrow A$  (1 SAT)
- $B \Rightarrow A$  (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 ()`

**startTokenAllocation**

beneficiary: 0x14723a09acff6d2a60dcdF7aa4aff308fddc160c

numTokens: 1

CashRecieveProof: cashProof

 **transact**

**transferTokens** address sender, address reciever, uint256 numOFTokens

**validateTransaction** uint256 listnumber, uint256 index

**checkCurrentRate**

**heckSelfModeAmour**

**entReleaseTransactio** uint256 , uint256

**decimal**

**etAllPendingRequest** uint256 startingIndex

**jetReleaseTransaction** uint256 listnumber, uint256 index

**isContract** address addr

**name**

**symbol**

**ensCommissionedTol**

**totalTokensHeld**

0: uint256: 1



# Test 2 : Step 2

- Sending 1 SAT from account A to B
- 

**Transactions recorded:** 13

**Deployed Contracts**

▼ SpendingAllowanceToken at 0xd25...f37c6 (memory)

changePeriodicTime uint256 newTime

CurrentReleaseTrans

SetRate uint256 \_rate

startTokenAllocation address beneficiary, uint256 numTokens, string CashRecieveProof

transferTokens

sender: 0x14723a09acff6d2a60dcdf7aa4aff308fddc160c

reciever: 0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db

numOfTokens: 1

transact

validateTransaction "1","0"

checkCurrentRate

heckSelfModeAmour

entReleaseTransactio uint256 ,uint256

decimal

# 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

Transactions recorded: 14

Deployed Contracts

▼ SpendingAllowanceToken at 0xd25...f37c6 (memory) [icon] x

changePeriodicTime uint256 newTime ▼

CurrentReleaseTrans

SetRate uint256 \_rate ▼

startTokenAllocation address beneficiary, uint256 numTokens, string CashRecieveProof ▼

transferTokens ^

sender: 0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db

reciever: 0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db

numOfTokens: 2

[icon] transact

validateTransaction uint256 listnumber, uint256 index ▼

checkCurrentRate



# Test 2 : Step 4

- Sending 3 SAT from C to D

CurrentReleaseTrans

SetRate

uint256 \_rate

▼

startTokenAllocation

address beneficiary, uint256 numTokens, string CashRecieveProof

▼

transferTokens

^

sender:

0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db

reciever:

0x583031d1113ad414f02576bd6afabfb302140225

numOfTokens:

3

transact

validateTransaction

uint256 listnumber, uint256 index

▼

checkCurrentRate

heckSelfModeAmour

entReleaseTransactio

uint256 , uint256

▼

decimal

etAllPendingRequest

uint256 startingIndex

▼

0: uint256[]:

1: uint256: 0

2: bool: false

# 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

changePeriodicTime

uint256 newTime

▼

CurrentReleaseTrans

▼

SetRate

uint256 \_rate

▼

startTokenAllocation

address beneficiary, uint256 numTokens, string CashRecieveProof

▼

transferTokens

▲

sender:

0x583031d1113ad414f02576bd6afabfb302140225

reciever:

0xdd870fa1b7c4700f2bd7f44238821c26f7392148

numOfTokens:

3

transact

validateTransaction

uint256 listnumber, uint256 index

▼

checkCurrentRate

heckSelfModeAmour

entReleaseTransactio

uint256 , uint256

▼

decimal

etAllPendingRequest

uint256 startingIndex

▼

0: uint256[]:

1: uint256: 0

2: bool: false



# Test 2 : Step 6

- Validating transaction from A
- Result :
- Original :  
 $A \Rightarrow B$  (1 SAT)
- After chaining  
 $A \Rightarrow B \Rightarrow C \Rightarrow D \Rightarrow A$   
 $A \Rightarrow A$  (1 SAT)

▼ SpendingAllowanceToken at 0x692...77b3a (memory) 📄 ✕

changePeriodicTime

uint256 newTime

▼

getCurrentPeriod

▼

SetRate

uint256 \_rate

▼

startTokenAllocation

address beneficiary, uint256 numTokens, string CashRecieveProof

▼

transferTokens

"0xdd870fa1b7c4700f2bd7f44238821c26f7392148","0x14723a09ac"

▼

validateTransaction

4,0|

▼

checkCurrentRate

▼

heckSelfModeAmour

▼

entReleaseTransactio

uint256 ,uint256

▼

decimal

▼

etAllPendingRequest

uint256 startingIndex

▼

0: uint256[]: 4,0

1: uint256: 0

2: bool: false

etReleaseTransactio

4,0

▼

0: address: 0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB

1: uint256: 0

2: uint8: 0

3: address: 0x583031D1113aD414F02576BD6afaBfb302140225

4: uint256: 1

# Test 2 : Step 7

- Validating transaction from B
- Result :  
Original  
 $B \Rightarrow C$  (2 SAT)
- Chained
- $B \Rightarrow C \Rightarrow D \Rightarrow A$   
 $B \Rightarrow A$  (1 SAT)

▼ SpendingAllowanceToken at 0x692...77b3a (memory) 📄 ✕

changePeriodicTime	uint256 newTime	▼
getCurrentPeriod		
SetRate	uint256 _rate	▼
startTokenAllocation	address beneficiary, uint256 numTokens, string CashRecieveProof	▼
transferTokens	"0xdd870fa1b7c4700f2bd7f44238821c26f7392148", "0x14723a09ac"	▼
validateTransaction	5 0	▼
checkCurrentRate		
heckSelfModeAmour		
entReleaseTransaction	uint256 , uint256	▼
decimal		
etAllPendingRequest	uint256 startingIndex	▼
0: uint256[]: 5,0 1: uint256: 0 2: bool: false		
etReleaseTransaction	5,0	▼
0: address: 0x583031D1113aD414F02576BD6afaBfb302140225 1: uint256: 0 2: uint8: 0 3: address: 0xdD870fA1b7C4700F2BD7f44238821C26f7392148 4: uint256: 1 5: uint8: 2		



# Test 2 : Step 8

- Validating Transaction from C
- Original Transaction  
 $C \Rightarrow D$
- Chained Transaction  
 $C \Rightarrow D \Rightarrow A$  (3 SAT)  
 $C \Rightarrow A$  (1 SAT)

changePeriodicTime	uint256 newTime	▼
getCurrentPeriod		
SetRate	uint256 _rate	▼
startTokenAllocation	address beneficiary, uint256 numTokens, string CashRecieveProof	▼
transferTokens	"0xdd870fa1b7c4700f2bd7f44238821c26f7392148", "0x14723a09ac"	▼
validateTransaction	6,0	▼
checkCurrentRate		
heckSelfModeAmour		
entReleaseTransaction	uint256 , uint256	▼
decimal		
etAllPendingRequest	uint256 startingIndex	▼
0: uint256[]: 6,0 1: uint256: 0 2: bool: false		
getReleaseTransaction	6,0	▼
0: address: 0xd870fa1b7c4700f2bd7f44238821c26f7392148 1: uint256: 0 2: uint8: 0 3: address: 0x14723a09acff6d2a60dcdf7aA4AF308FDDC160C 4: uint256: 2 5: uint8: 2		

# Test 2 : Step 9

- Validating Transaction from D
- Original Transaction :  
D => A
- Chained Transaction  
D=>A (1 SAT)

▼ SpendingAllowanceToken at 0x692...77b3a (memory) 📄 ✕

changePeriodicTime

uint256 newTime

▼

getCurrentPeriod

SetRate

uint256 \_rate

▼

startTokenAllocation

address beneficiary, uint256 numTokens, string CashRecieveProof

▼

transferTokens

"0xdd870fa1b7c4700f2bd7f44238821c26f7392148", "0x14723a09ac

▼

validateTransaction

7,0

▼

checkCurrentRate

heckSelfModeAmour

entReleaseTransaction

uint256 , uint256

▼

decimal

etAllPendingRequest

uint256 startingIndex

▼

0: uint256[]: 7,0

1: uint256: 0

2: bool: false

etReleaseTransaction

7,0

▼

0: address: 0x14723A09ACf6D2A60DcdF7aA4AF308FDDC160C

1: uint256: 2

2: uint8: 2

3: address: 0x14723A09ACf6D2A60DcdF7aA4AF308FDDC160C

4: uint256: 0

5: uint8: 3



# Test 2 : Result

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

# Test 3

- Original Transaction
  - Expected Result
- 
- $A \Rightarrow B$  10 SAT
  - $B \Rightarrow C$  4 SAT
  - $B \Rightarrow D$  4 SAT
  - $B \Rightarrow E$  2 SAT
  - $C \Rightarrow F$  2 SAT
- 
- AT PERIOD 1 :
    - $A \Rightarrow C$  4 SAT
    - $A \Rightarrow D$  4 SAT
    - $A \Rightarrow E$  2 SAT
    - $A \Rightarrow F$  2 SAT



# Test 3 : step 1

- Allocating 10 SAT to account A

**Transactions recorded:** ①

**Deployed Contracts**

▼ SpendingAllowanceToken at 0x8c1...401f5 (memory)

changePeriodicTime uint256 newTime

getCurrentPeriod

SetRate uint256 \_rate

**startTokenAllocation**

beneficiary: 0x14723a09acff6d2a60dcdF7aa4aff308fddc160c

numTokens: 10

CashRecieveProof: cashProof

transact

transferTokens address sender, address reciever, uint256 numOFTokens

validateTransaction uint256 listnumber, uint256 index

checkCurrentRate

# Test 3 : step 2

- Sending 10 SAT from A to B

Transactions recorded: ②

Deployed Contracts

▼ SpendingAllowanceToken at 0x8c1...401f5 (memory) [icon] [x]

changePeriodicTime uint256 newTime ▼

getCurrentPeriod

SetRate uint256 \_rate ▼

startTokenAllocation address beneficiary, uint256 numTokens, string CashRecieveProof ▼

**transferTokens** ^

sender: 0x14723a09acff6d2a60dcdf7aa4aff308fddc160c

reciever: 0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db

numOfTokens: 10

[icon] [transact]

validateTransaction uint256 listnumber, uint256 index ▼

checkCurrentDate



# Test 3 : Step 3

- Sending SAT from B
  - $B \Rightarrow C$  (4 SAT)
  - $B \Rightarrow D$  (4 SAT)
  - $B \Rightarrow E$  (2 SAT)

**Transactions recorded:** ③

**Deployed Contracts**

▼ SpendingAllowanceToken at 0x8c1...401f5 (memory) ✕

changePeriodicTime

uint256 newTime

▼

getCurrentPeriod

SetRate

uint256 \_rate

▼

startTokenAllocation

address beneficiary, uint256 numTokens, string CashRecieveProof

▼

transferTokens

^

sender:

0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db

reciever:

0x583031d1113ad414f02576bd6afabfb302140225

numOfTokens:

4

transact

validateTransaction

uint256 listnumber, uint256 index

▼

checkCurrentRate

heckSelfModeAmour

entReleaseTransactio

uint256 , uint256

▼

# Test 3 : Step 4

- Sending 2 SAT from account C to F

**Deployed Contracts**

▼ SpendingAllowanceToken at 0x8c1...401f5 (memory) 📄 ✕

changePeriodicTime

uint256 newTime

▼

getCurrentPeriod

SetRate

uint256 \_rate

▼

startTokenAllocation

address beneficiary, uint256 numTokens, string CashRecieveProof

▼

transferTokens

^

sender:

0x583031d1113ad414f02576bd6afabfb302140225

reciever:

0xde5cf0c5078d60062e2bf075dc1cb3f0e466c296

numOfTokens:

2|

🏠

transact

validateTransaction

uint256 listnumber, uint256 index

▼

checkCurrentRate

heckSelfModeAmour

entReleaseTransaction

uint256 , uint256

▼

decimal

etAllPendingRequest

uint256 startingIndex

▼



# Test 3 : Step 5

- Validating transaction from A
- Original Transaction
  - $A \Rightarrow B$  (10 SAT)
- Chained Transaction
  - $A \Rightarrow B \Rightarrow C \Rightarrow F$  (2 SAT)
  - $A \Rightarrow B$  (8 SAT)

changePeriodicTime	uint256 newTime	▼
getCurrentPeriod		
SetRate	uint256 _rate	▼
startTokenAllocation	"0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db","10","sdfferfer	▼
transferTokens	"0x583031d1113ad414f02576bd6afabfb302140225","0xde5cf0c507	▼
validateTransaction	1,0	▼
checkCurrentRate		
heckSelfModeAmour		
entReleaseTransaction	uint256 , uint256	▼
decimal		
etAllPendingRequest	uint256 startingIndex	▼
0: uint256[]: 1,0 1: uint256: 0 2: bool: false		
etReleaseTransaction	1,0	▼
0: address: 0x4B0897b0513fdC7C541B6d9D7E929C4e5364D2dB 1: uint256: 2 2: uint8: 2 3: address: 0xde5Cf0C5078D60062E2bF075dc1CB3F0E466C296 4: uint256: 4 5: uint8: 2		

# Test 3 : Step 6

- Validating transaction from B
- Transaction
  - $B \Rightarrow D$
  - $B \Rightarrow E$

startTokenAllocation	"0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db","10","sdfferfer"	▼
transferTokens	"0x4b0897b0513fdc7c541b6d9d7e929c4e5364d2db","0xdd870fa1b"	▼
validateTransaction	9,1	▼
checkCurrentRate		
heckSelfModeAmour		
entReleaseTransaction	uint256 , uint256	▼
decimal		
etAllPendingRequest	uint256 startingIndex	▼
	0: uint256[]: 9,1 1: uint256: 0 2: bool: false	
etReleaseTransaction	9,1	▼
	0: address: 0xdD870fA1b7C4700F2BD7f44238821C26f7392148 1: uint256: 4 2: uint8: 2 3: address: 0xdD870fA1b7C4700F2BD7f44238821C26f7392148 4: uint256: 0 5: uint8: 3	
isContract	address addr	▼
listNumber		
name		



# Test 3 : Result

- Resulted Transaction :

- $A \Rightarrow C$
- $A \Rightarrow D$
- $A \Rightarrow E$
- $A \Rightarrow F$

Thank You