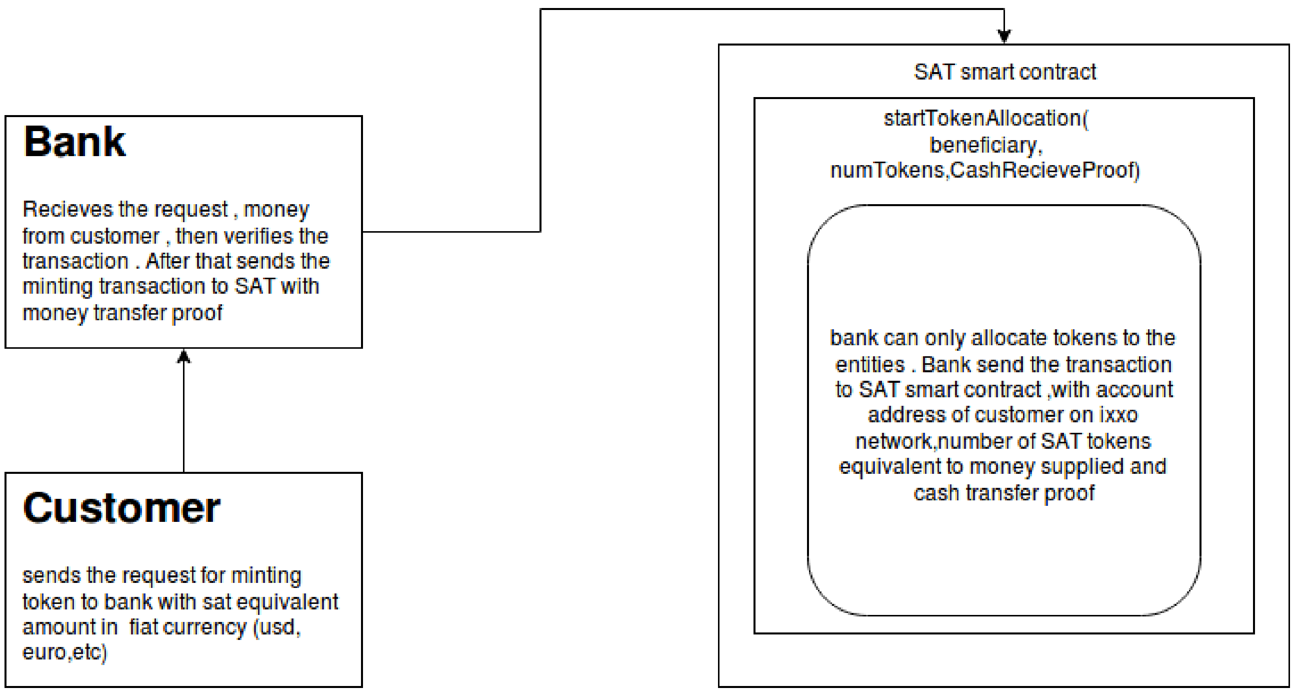
Spending Allowance Token

Here we are going to understand the logic and working of Spending Allowance Token .

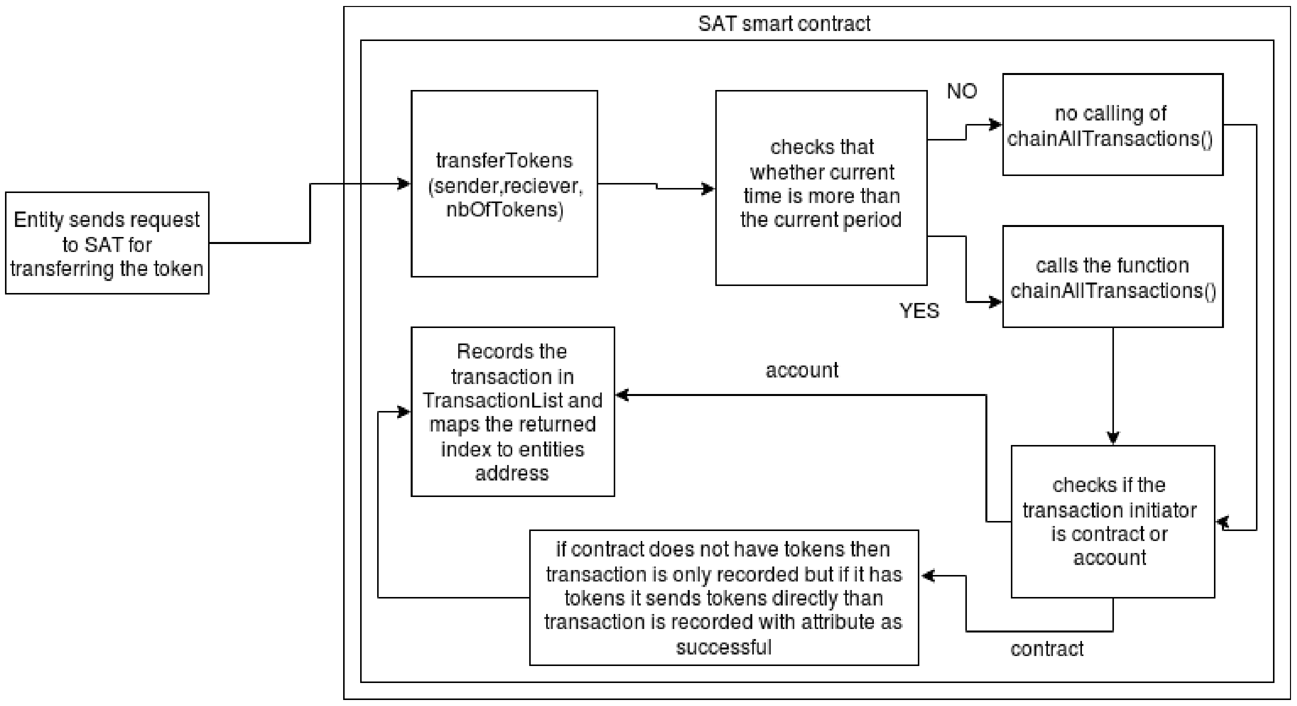
But before that we have to understand what are the various functions in SAT tokens and how they carry out their specified functionality.

Functions in SAT :

1) **startTokenAllocation()** : Function which allocates the mintable stable token to the receiving entity . This function can be taken out by minting entity only (bank).

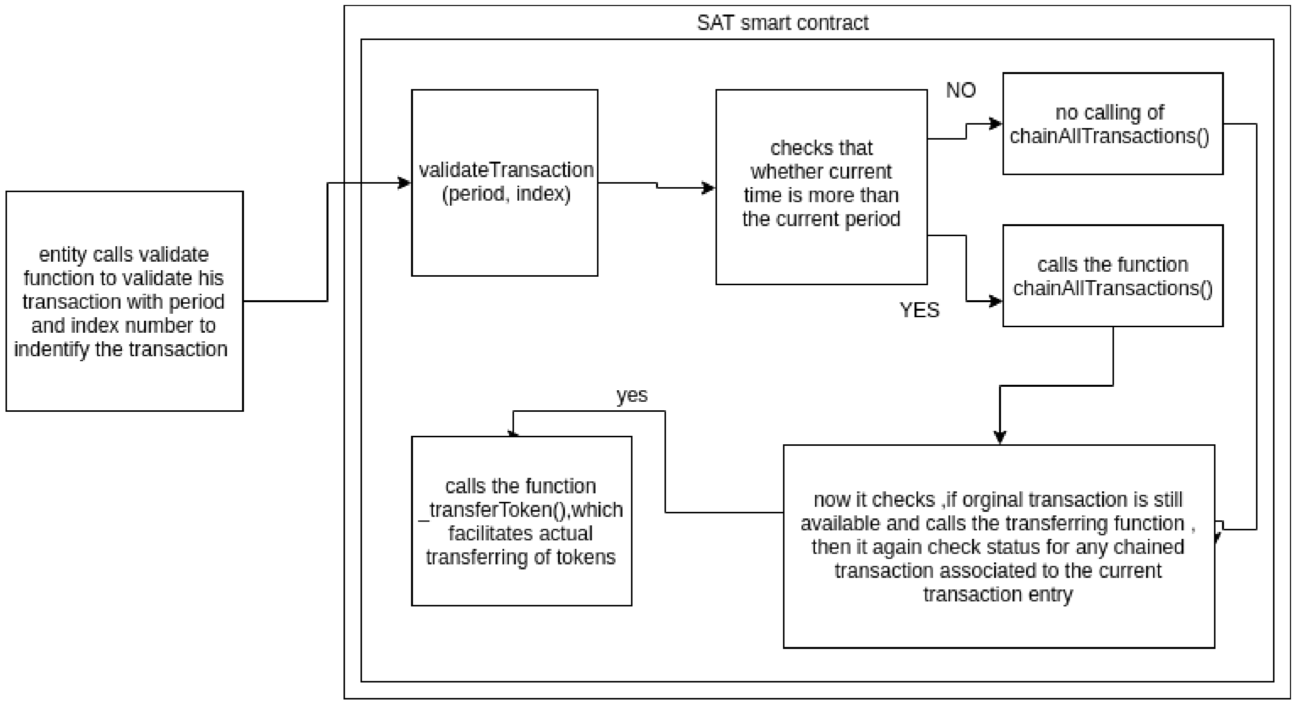

SAT smart contract
startTokenAllocation(beneficiary, numTokens,CashRecieveProof)
bank can only allocate tokens to the entities . Bank send the transaction to SAT smart contract ,with account address of customer on ixxo network,number of SAT tokens equivalent to money supplied and cash transfer proof
Bank
Recieves the request , money from customer , then verifies the transaction . After that sends the minting transaction to SAT with money transfer proof
Customer
sends the request for minting token to bank with sat equivalent amount in fiat currency (usd, euro,etc)

2) **transferTokens()** : this function allows user of SAT token (either account or contract) to transfer tokens in the system



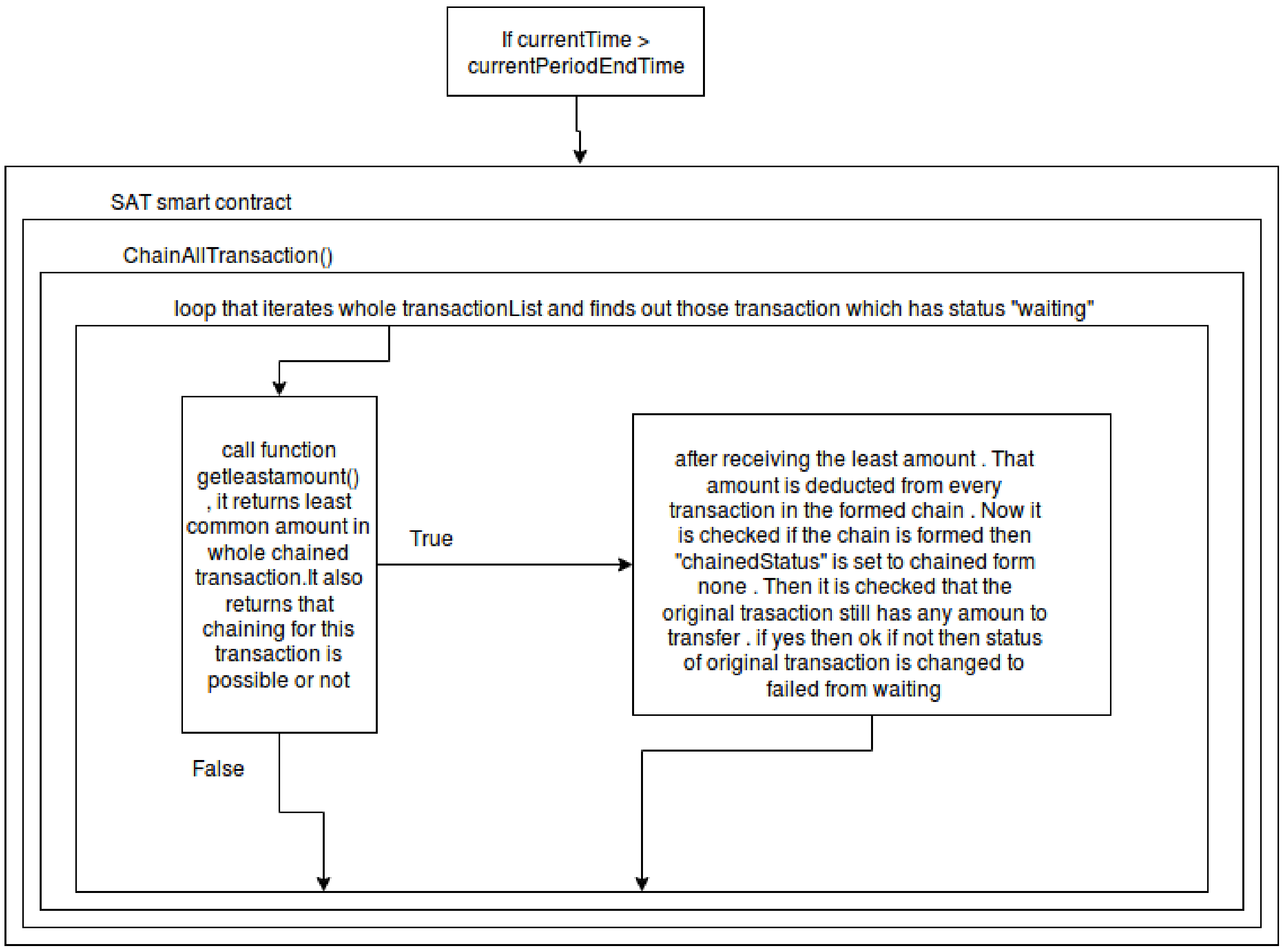
SAT smart contract
Entity sends request to SAT for transferring the token
transferTokens (sender,reciever, nbOfTokens)
checks that whether current time is more than the current period
NO
no calling of chainAllTransactions()
YES
calls the function chainAllTransactions()
checks if the transaction initiator is contract or account
account
Records the transaction in TransactionList and maps the returned index to entities address
contract
if contract does not have tokens then transaction is only recorded but if it has tokens it sends tokens directly than transaction is recorded with attribute as successful

3) **validateTransaction()** : transfer of SAT originated from account address needs to be validated.For this purpose sender sends Period number and Index number as parameter to the function .Both (period number and index) acts as unique identifier to fetch the transaction.



SAT smart contract
entity calls validate function to validate his transaction with period and index number to indentify the transaction
validateTransaction (period, index)
checks that whether current time is more than the current period
NO
no calling of chainAllTransactions()
YES
calls the function chainAllTransactions()
now it checks ,if orginal transaction is still available and calls the transferring function , then it again check status for any chained transaction associated to the current transaction entry
yes
calls the function _transferToken(),which facilitates actual transferring of tokens

4) **chainAllTransaction()** : This function chains the trailing transactions .This function is triggered by time based modifiers . When the current time is for than the current period ending time , this function is called . It checks all the pending transactions and tries to chain them in FIFO (First In First Out) manner.


If currentTime > currentPeriodEndTime
SAT smart contract
ChainAllTransaction()
loop that iterates whole transactionList and finds out those transaction which has status "waiting"
call function getleastamount() , it returns least common amount in whole chained transaction.It also returns that chaining for this transaction is possible or not
True
after receiving the least amount . That amount is deducted from every transaction in the formed chain . Now it is checked if the chain is formed then "chainedStatus" is set to chained form none . Then it is checked that the original trasaction still has any amoun to transfer . if yes then ok if not then status of original transaction is changed to failed from waiting
False

6) **getAllPendingRequests()** : returns the period number and index of those transaction that needs to be validated one by one . This function along with period number and index it also returns if the there is any further transactions which needs attention .

Logic used in SAT tokens for all possible situations in transferring of tokens :

Case 1 : A => B

when an entity sends token to another entity , the transaction is recorded in the list . A can validate before the periods ends this will make a direct payment from A to B . But once the period is over the transaction is chained

case 2 : A => SmartContract

When an entity sends SAT to a smart contract . This type of situation is dealt in same manner as above

case 3 : SmartContract => A

when a SmartContract sends the transaction of sending token . It first checks that whether SmartContract has sufficient tokens or not , if yes then the direct payment is made (since contracts cannot validate the contracts). And if not then transaction is recored in list and when the next ReleaseList is geerated the transaction is chained (if applicable).

case 4 : SmartContract => SmartContract

This type of situation is dealt in same way as above