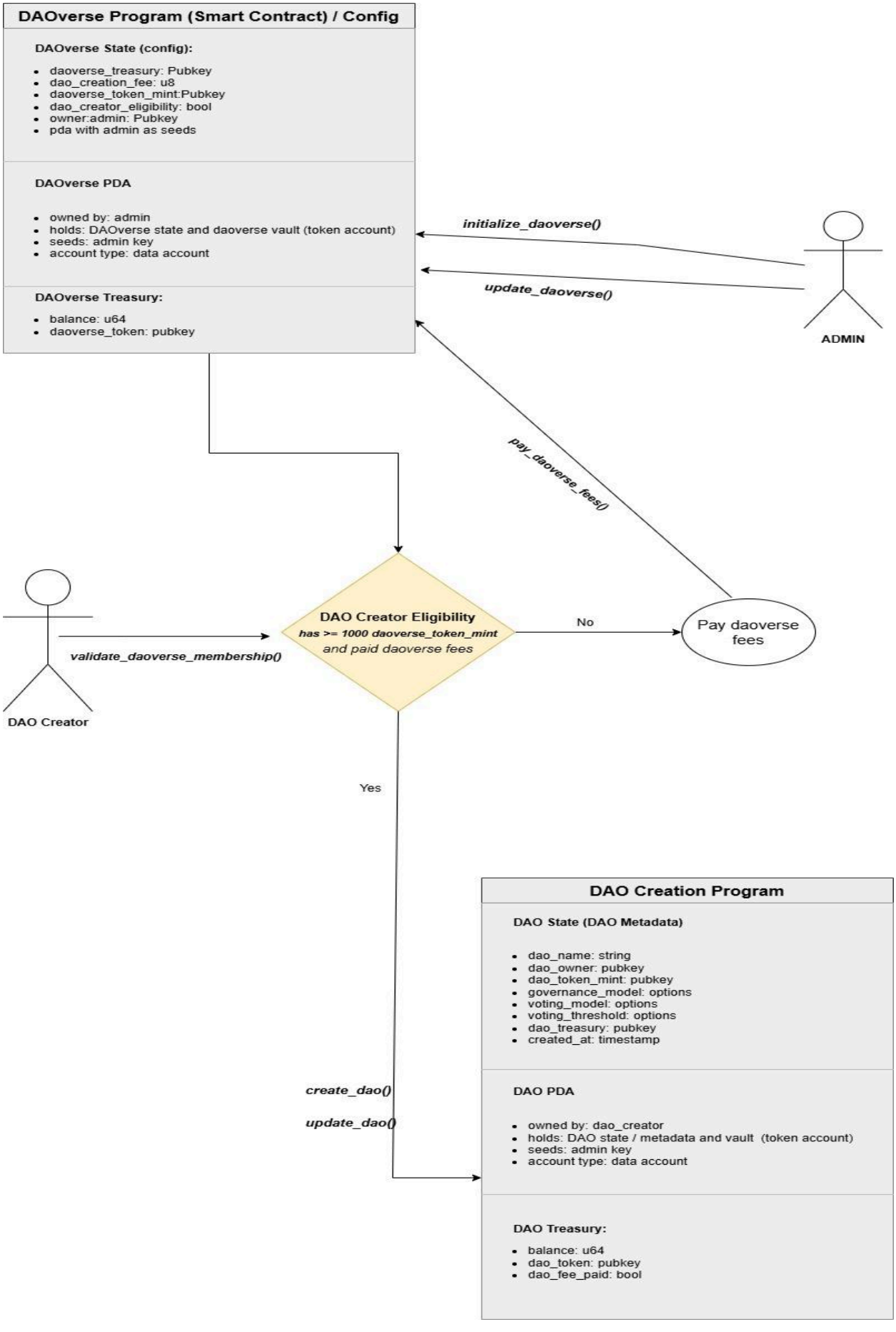
Grey - Architecture Diagram

PROTOCOL POC REQUIREMENTS FOR DAOVERSE

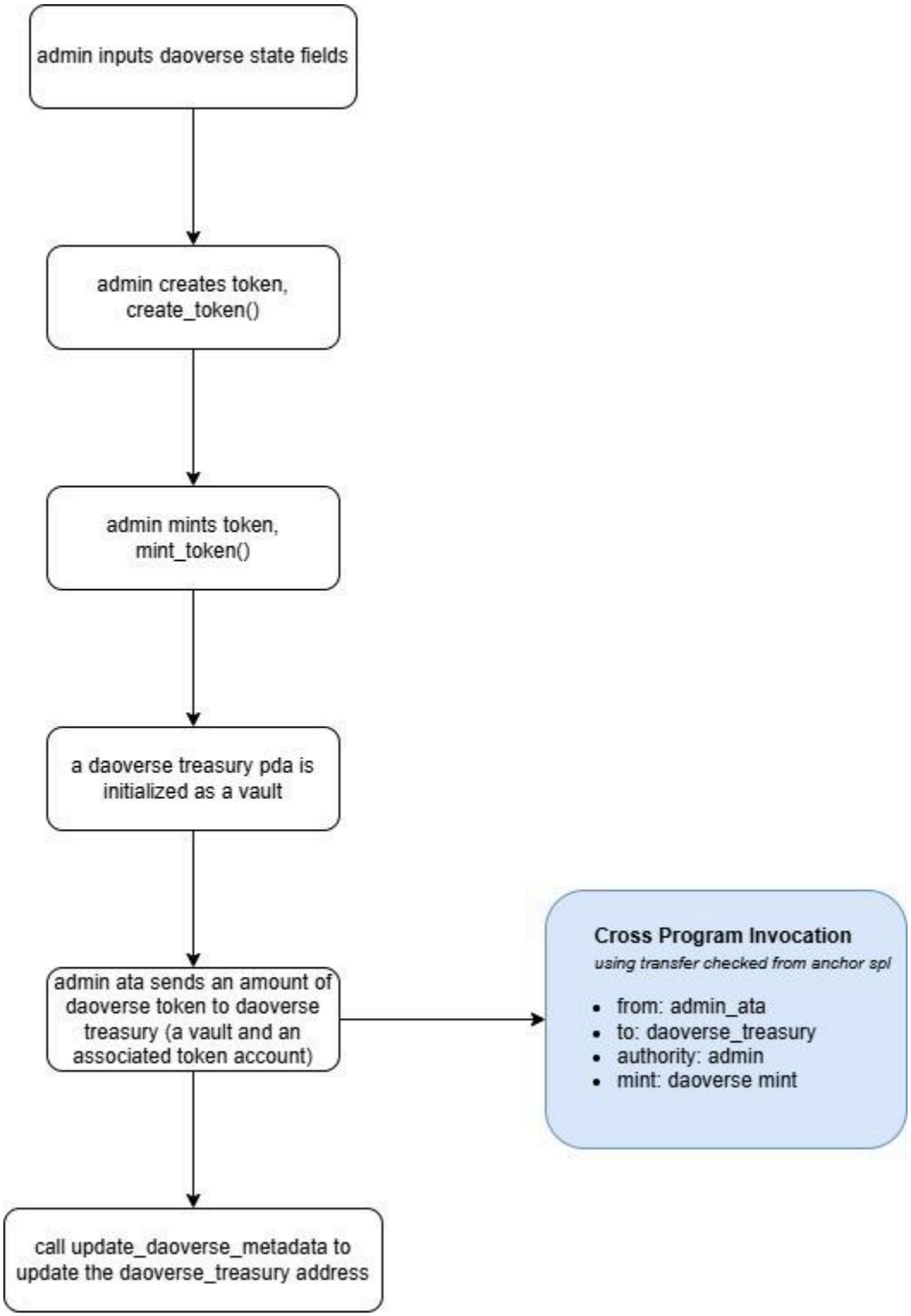
- The protocol shall allow admin have access to manage daoverse
- The protocol shall allow dao creators create dao as well as pay token fees for dao creation.
- The protocol shall allow users join multiple daos on one platform
- The protocol shall allow eligible users create proposals and submit proposals
- The protocol shall allow eligible users to vote on proposals by staking tokens
- The protocol shall allow proposers finalize vote after voting time elapses
- The protocol shall allow voters receive rewards after voting ends
- The protocol shall allow users as proposers to receive funds in milestones if approved.

OVERVIEW OF USER AS DAO CREATOR

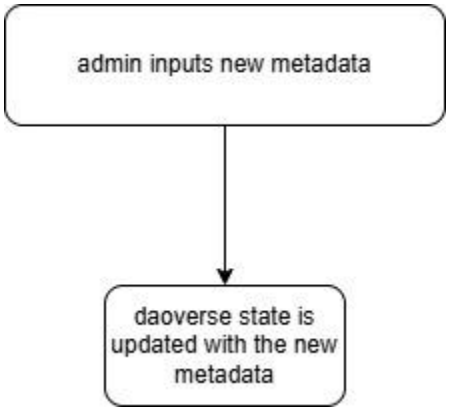

DAOverse Program (Smart Contract) / Config
DAOverse State (config):
daoverse_treasury: Pubkey
dao_creation_fee: u8
daoverse_token_mint:Pubkey
dao_creator_eligibility: bool
owner:admin: Pubkey
pda with admin as seeds
DAOverse PDA
owned by: admin
holds: DAOverse state and daoverse vault (token account)
seeds: admin key
account type: data account
DAOverse Treasury:
balance: u64
daoverse_token: pubkey
initialize_daoverse()
update_daoverse()
ADMIN
pay_daoverse_fees()
DAO Creator
validate_daoverse_membership()
DAO Creator Eligibility
has >= 1000 daoverse_token_mint
and paid daoverse fees
No
Pay daoverse fees
Yes
create_dao()
update_dao()
DAO Creation Program
DAO State (DAO Metadata)
dao_name: string
dao_owner: pubkey
dao_token_mint: pubkey
governance_model: options
voting_model: options
voting_threshold: options
dao_treasury: pubkey
created_at: timestamp
DAO PDA
owned by: dao_creator
holds: DAO state / metadata and vault (token account)
seeds: admin key
account type: data account
DAO Treasury:
balance: u64
dao_token: pubkey
dao_fee_paid: bool

Instructions

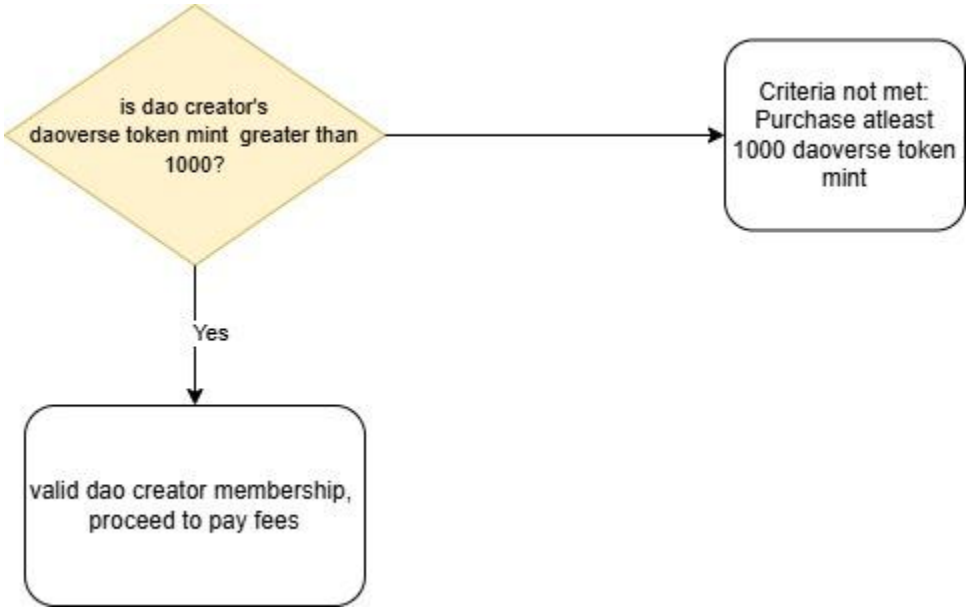
initialize_daoverse()



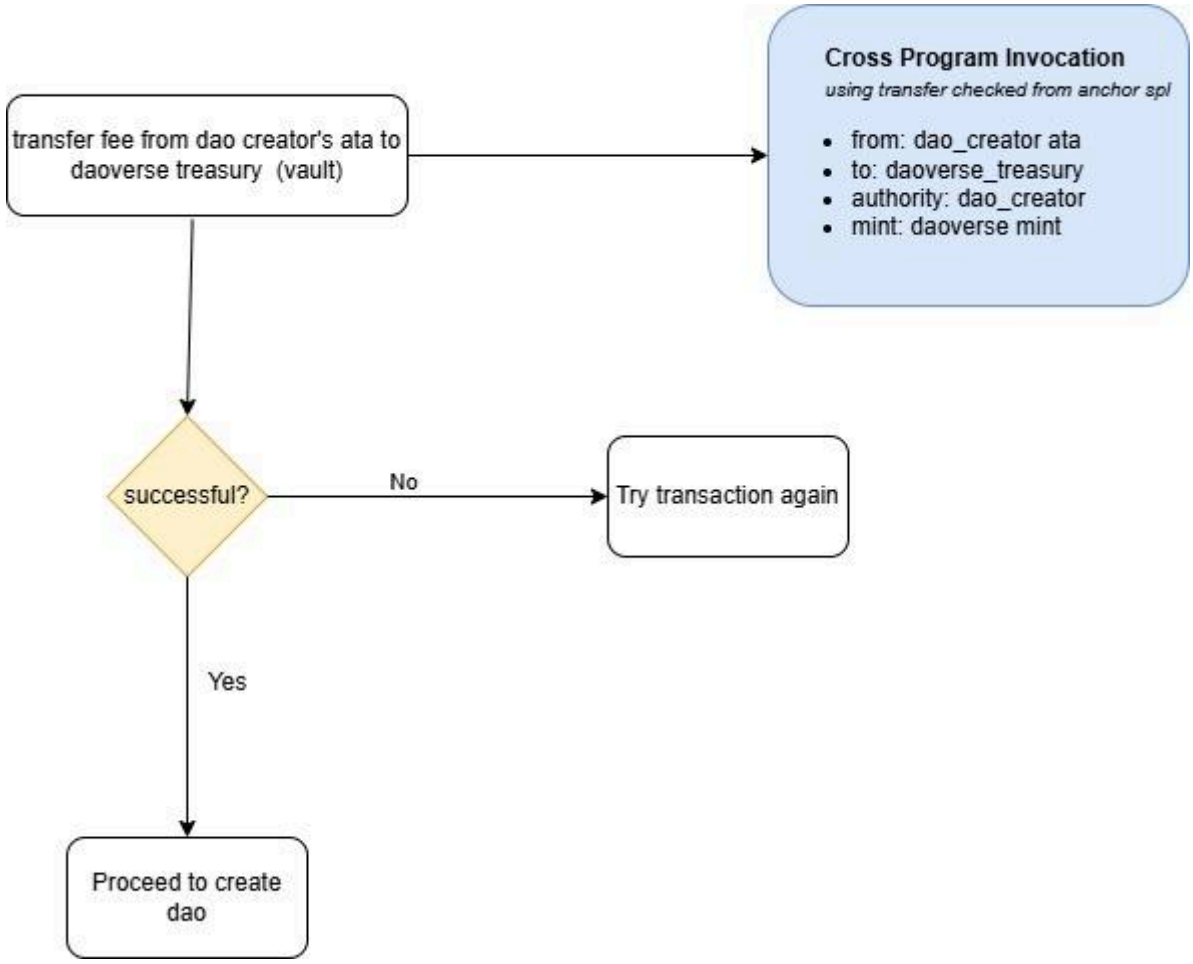
update_daoverse_metadata()



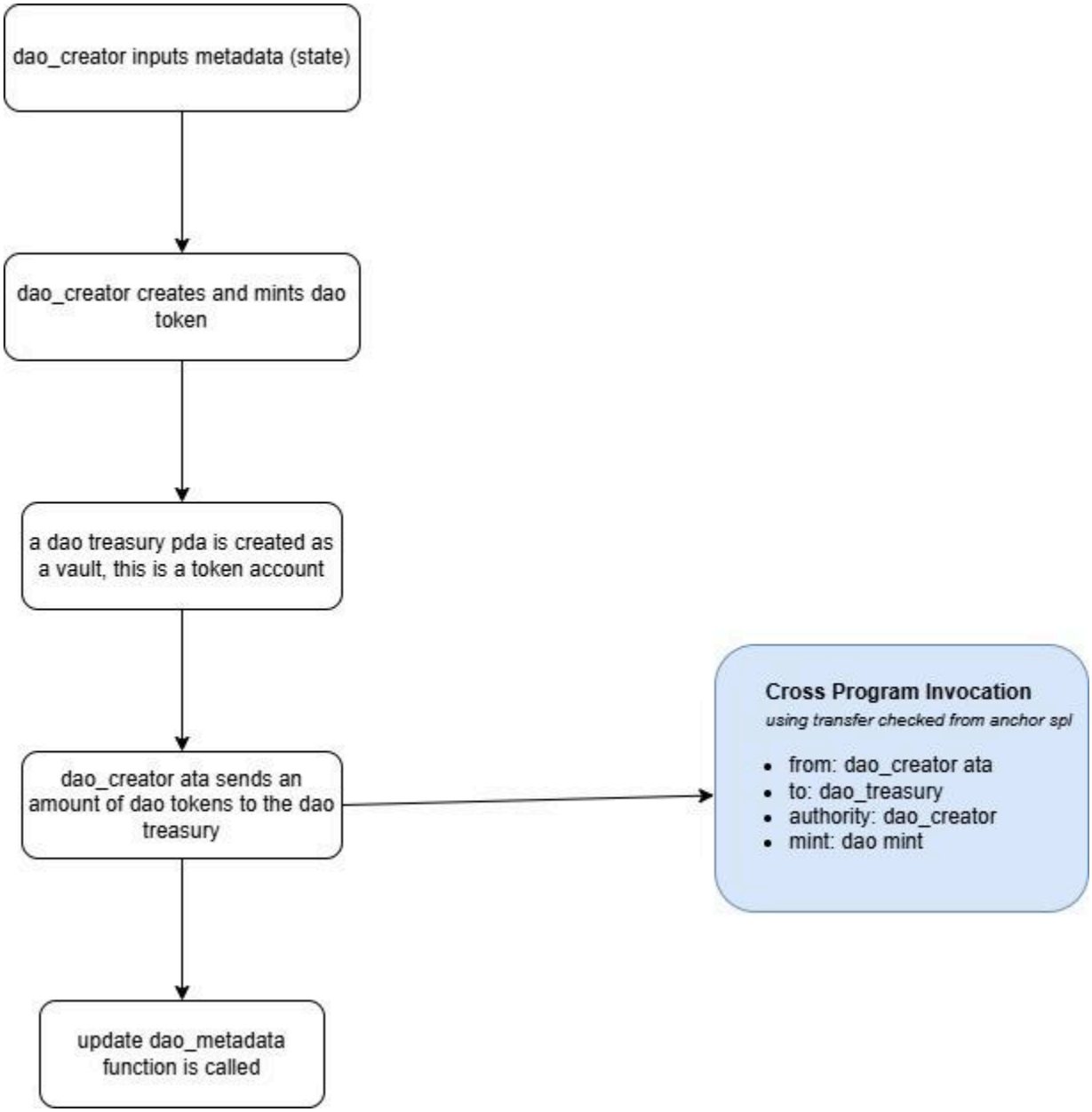
validate_dao_creator()



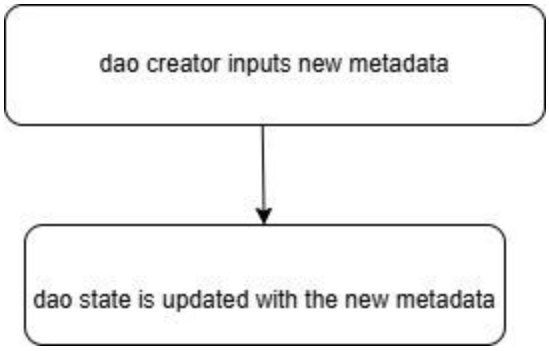
pay_daoverse_fee()



create_dao()

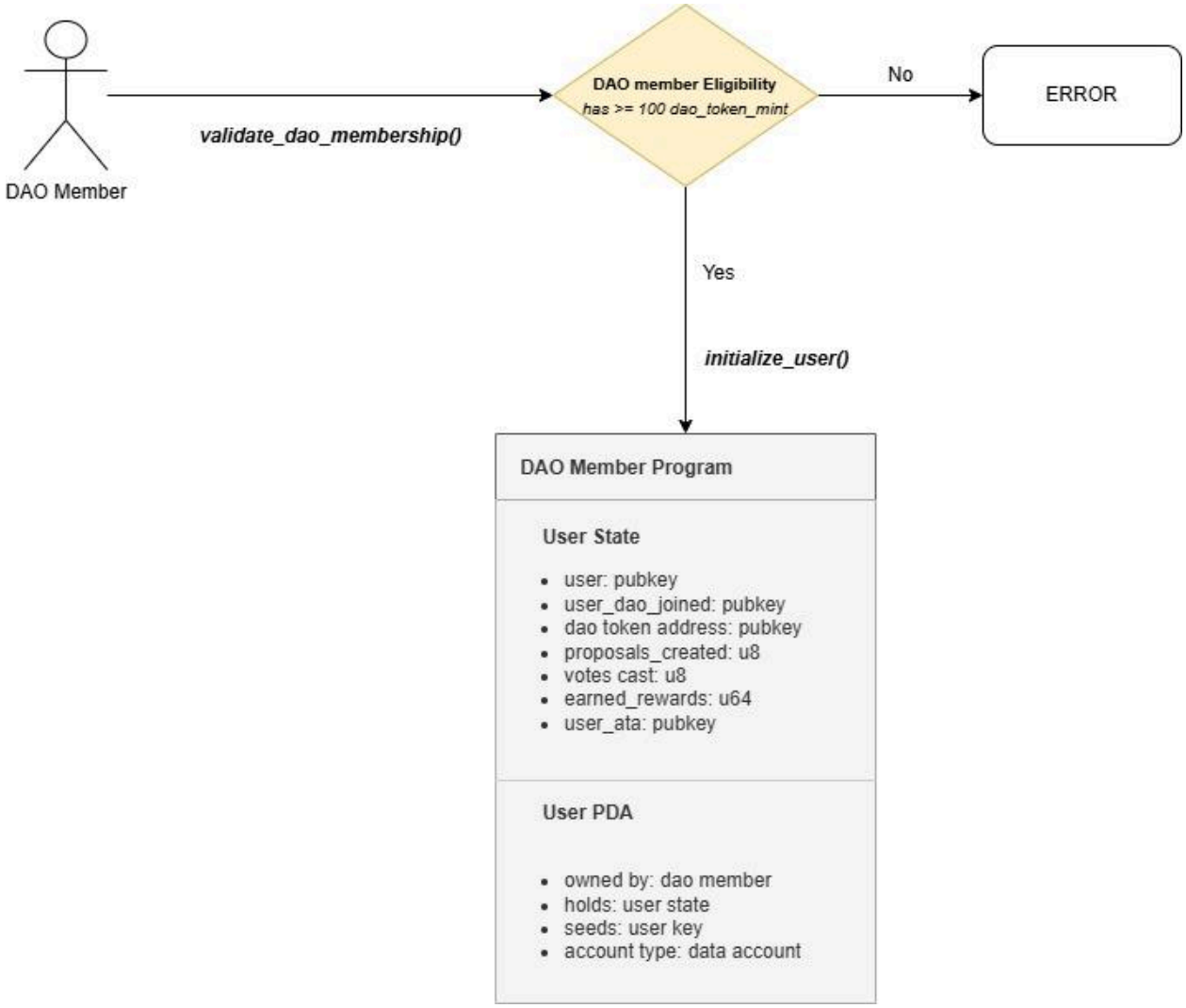


update_dao_metadata()



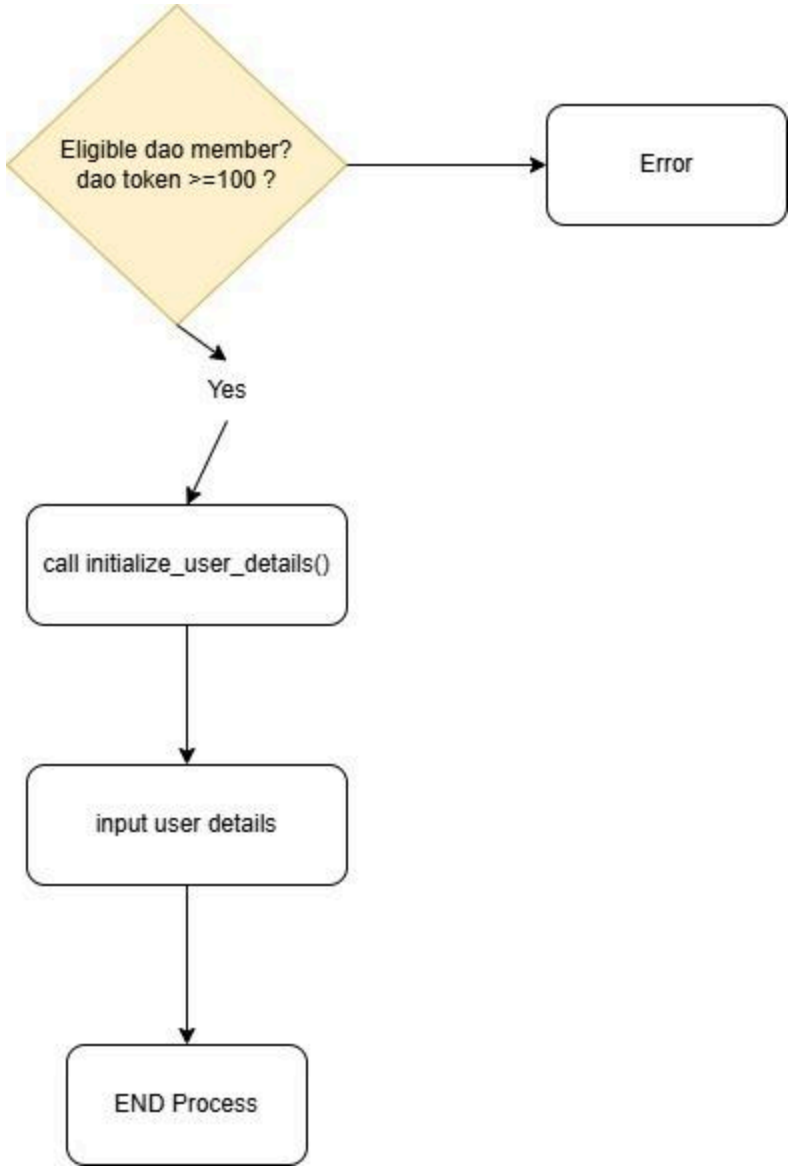
USER INTERACTION FLOW

USER AS DAO MEMBER

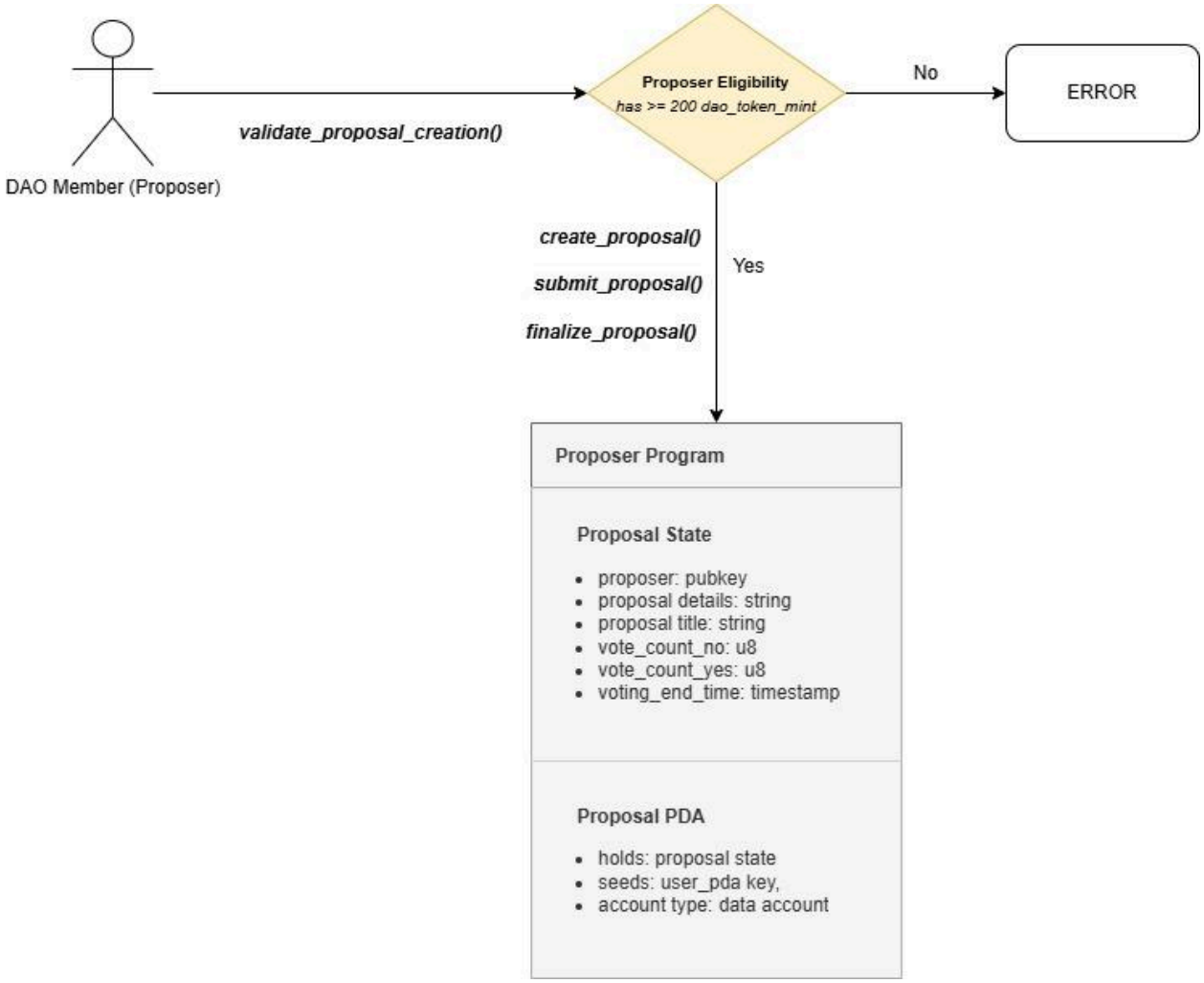


Instructions

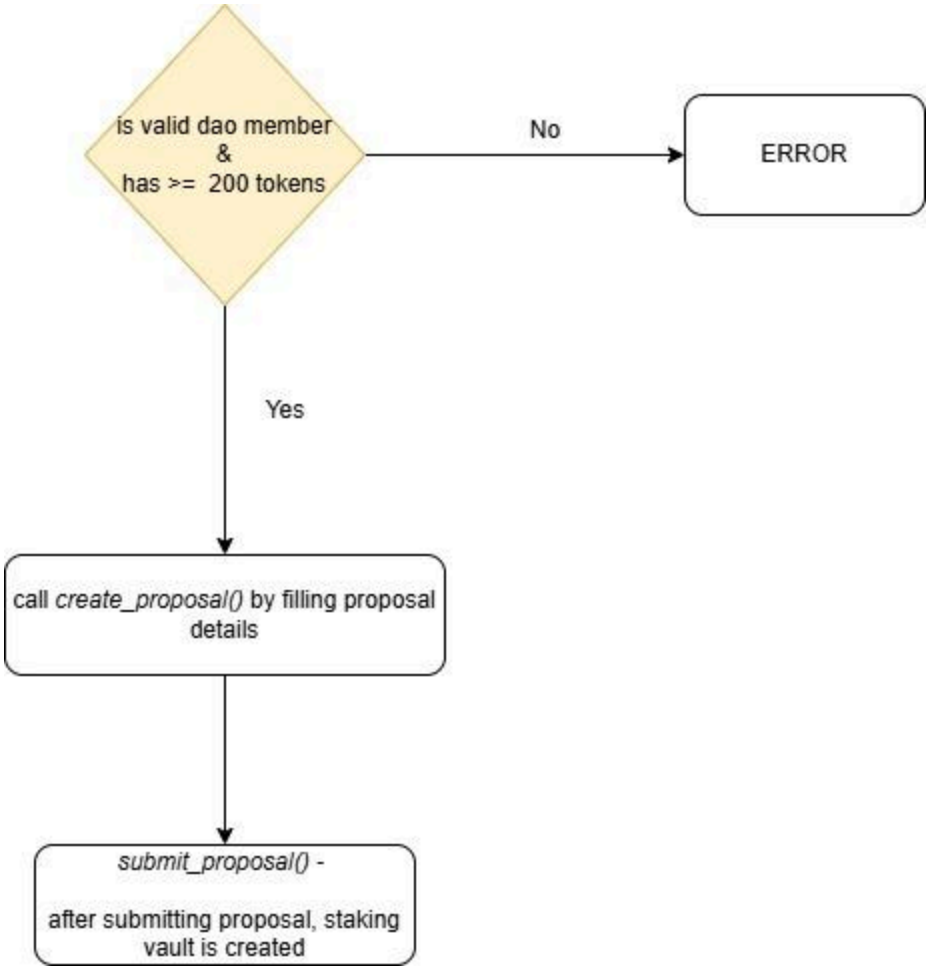
validate_dao_membership(),
initialize user()



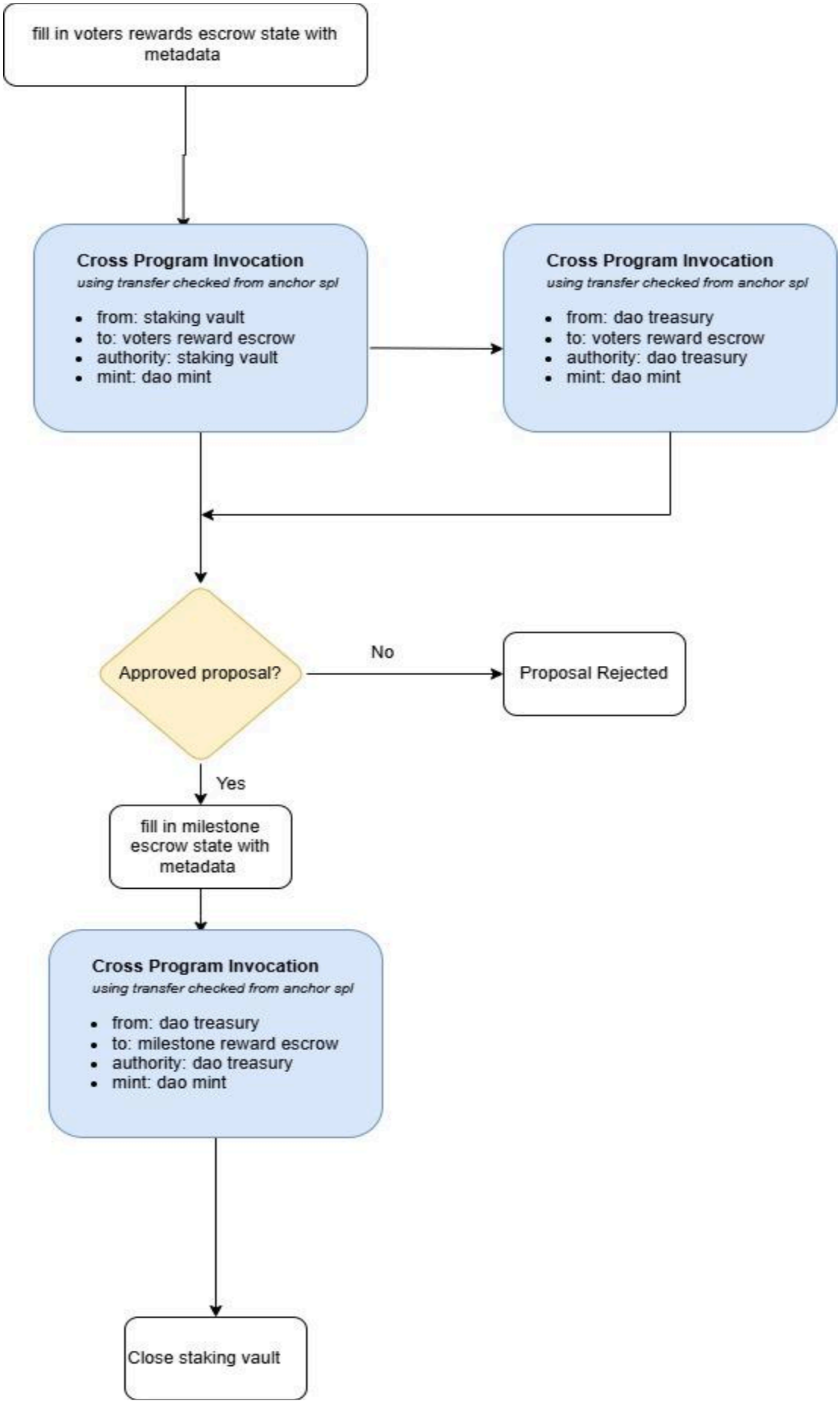
USER AS PROPOSER



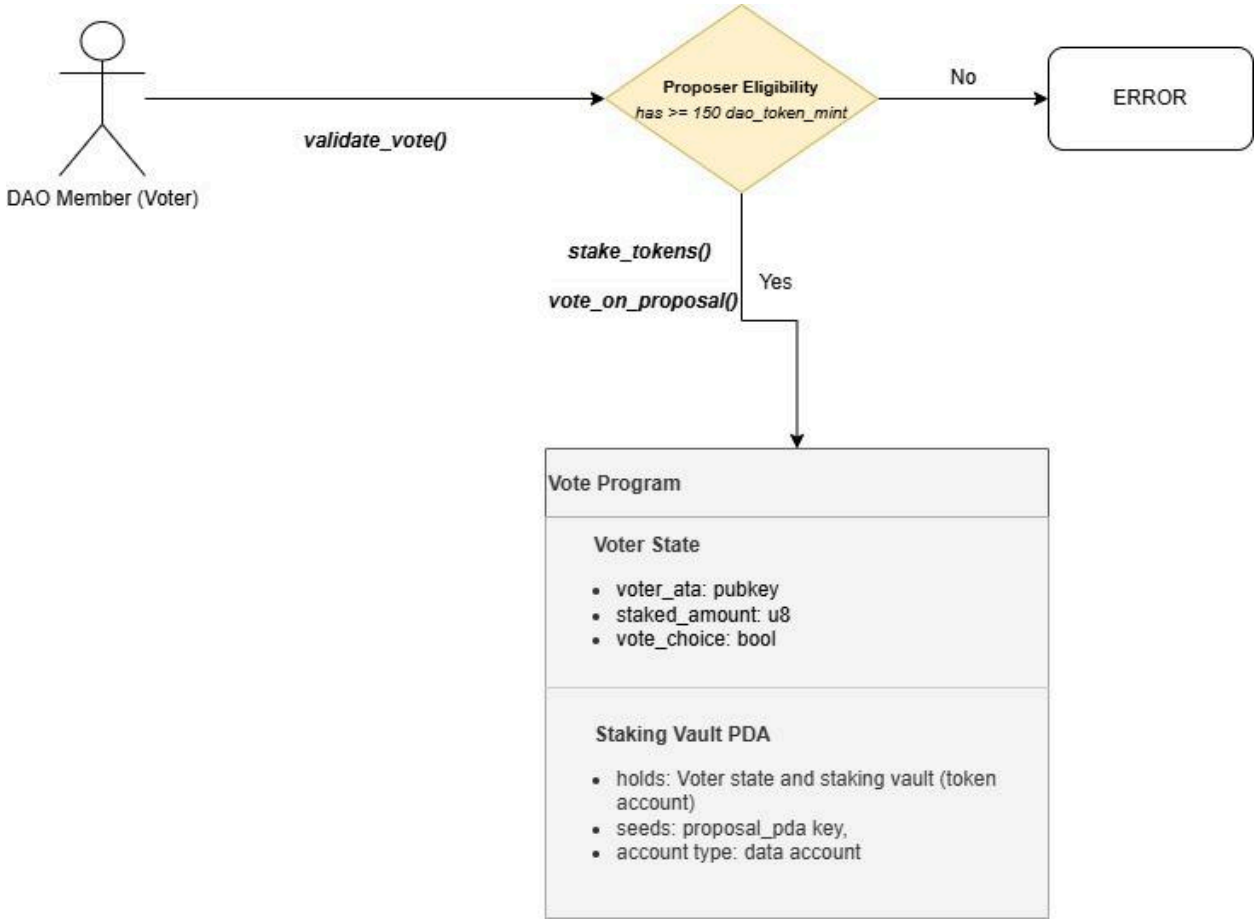
Instructions
create_proposal(),
submit_proposal()



finalize_proposal()

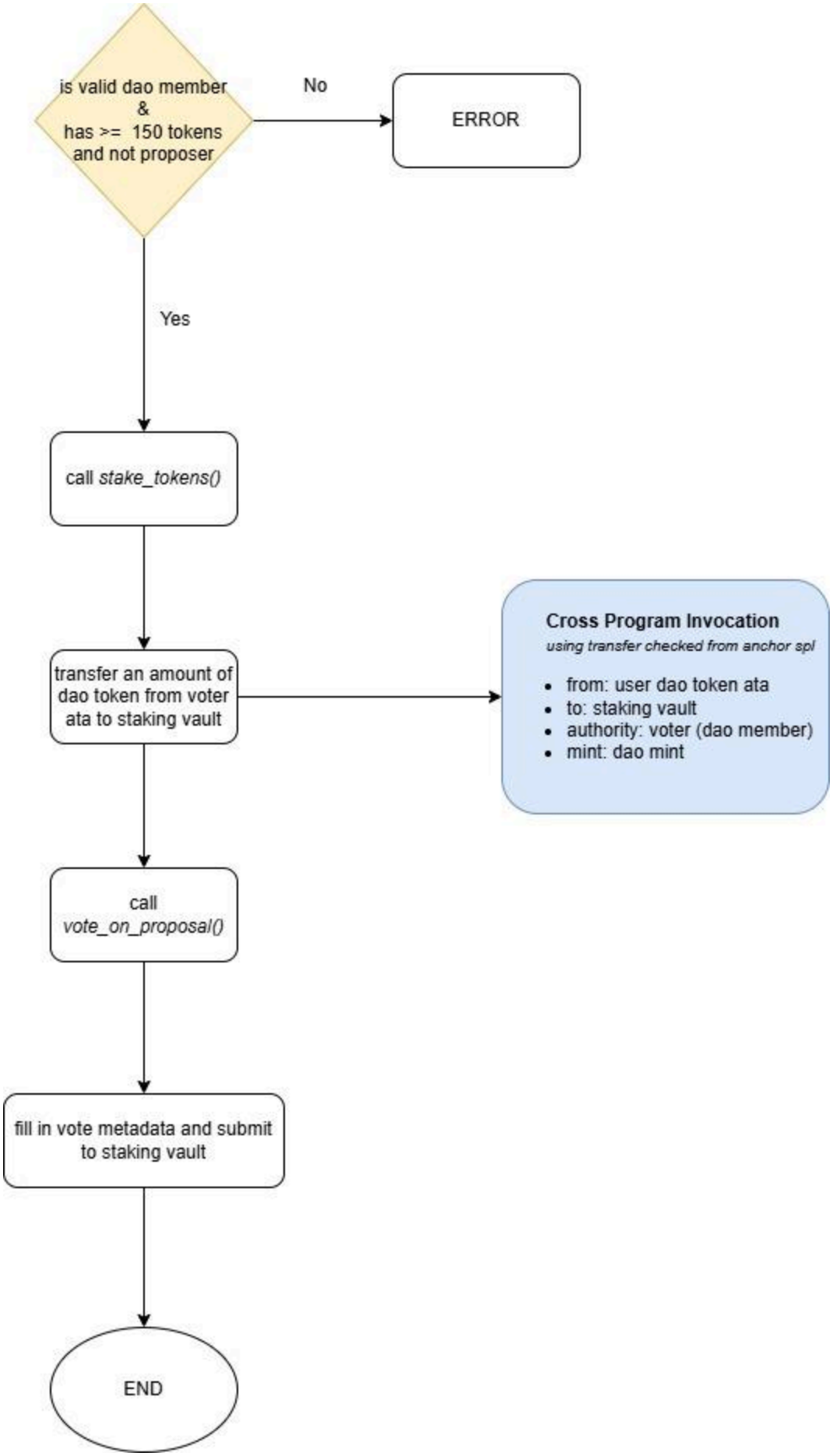

fill in voters rewards escrow state with metadata
Cross Program Invocation
using transfer checked from anchor spl
from: staking vault
to: voters reward escrow
authority: staking vault
mint: dao mint
Cross Program Invocation
using transfer checked from anchor spl
from: dao treasury
to: voters reward escrow
authority: dao treasury
mint: dao mint
Approved proposal?
No
Proposal Rejected
Yes
fill in milestone escrow state with metadata
Cross Program Invocation
using transfer checked from anchor spl
from: dao treasury
to: milestone reward escrow
authority: dao treasury
mint: dao mint
Close staking vault

USER AS VOTER

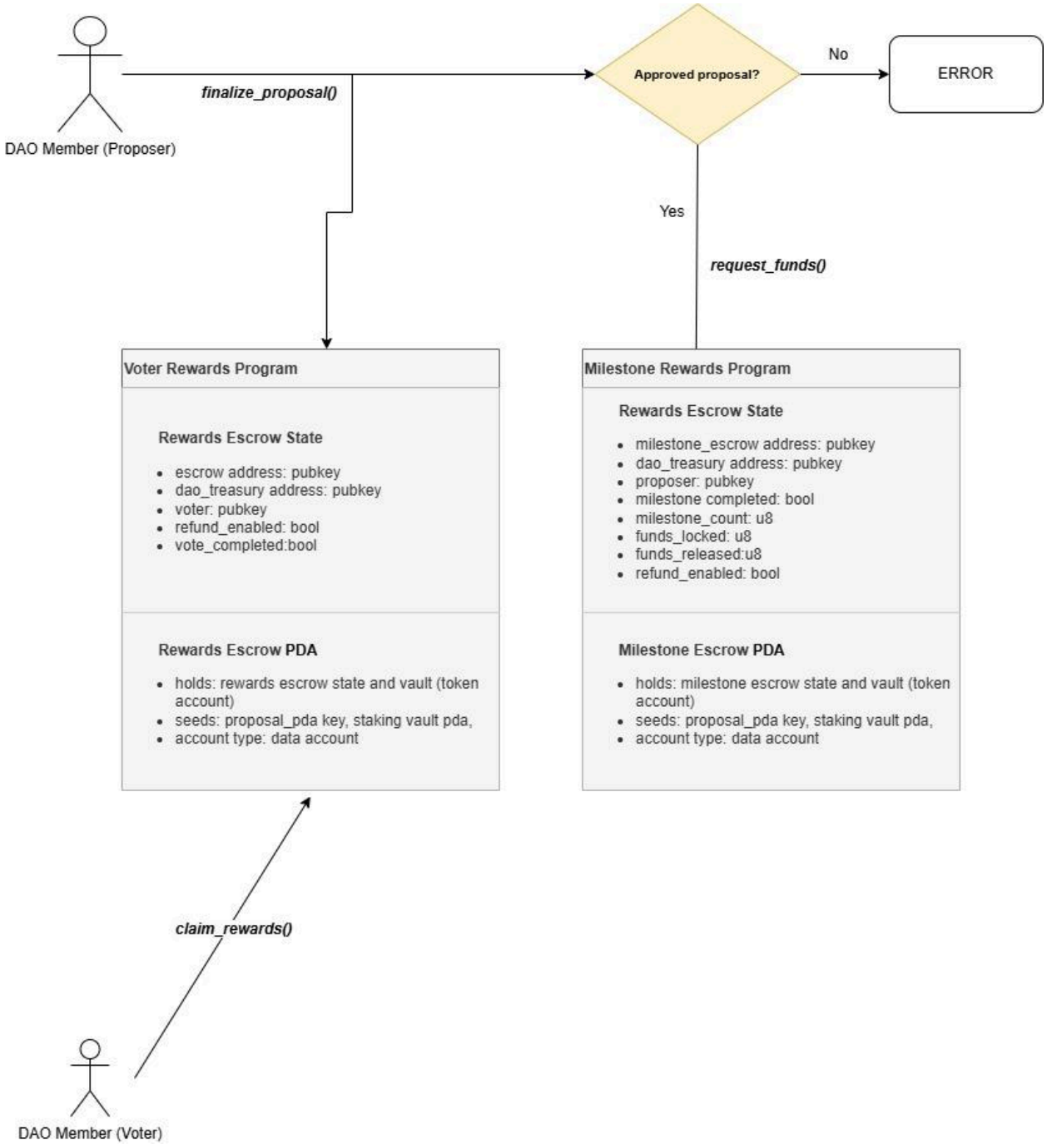


Instructions

validate_voter(), stake_tokens(), vote_on_proposals()

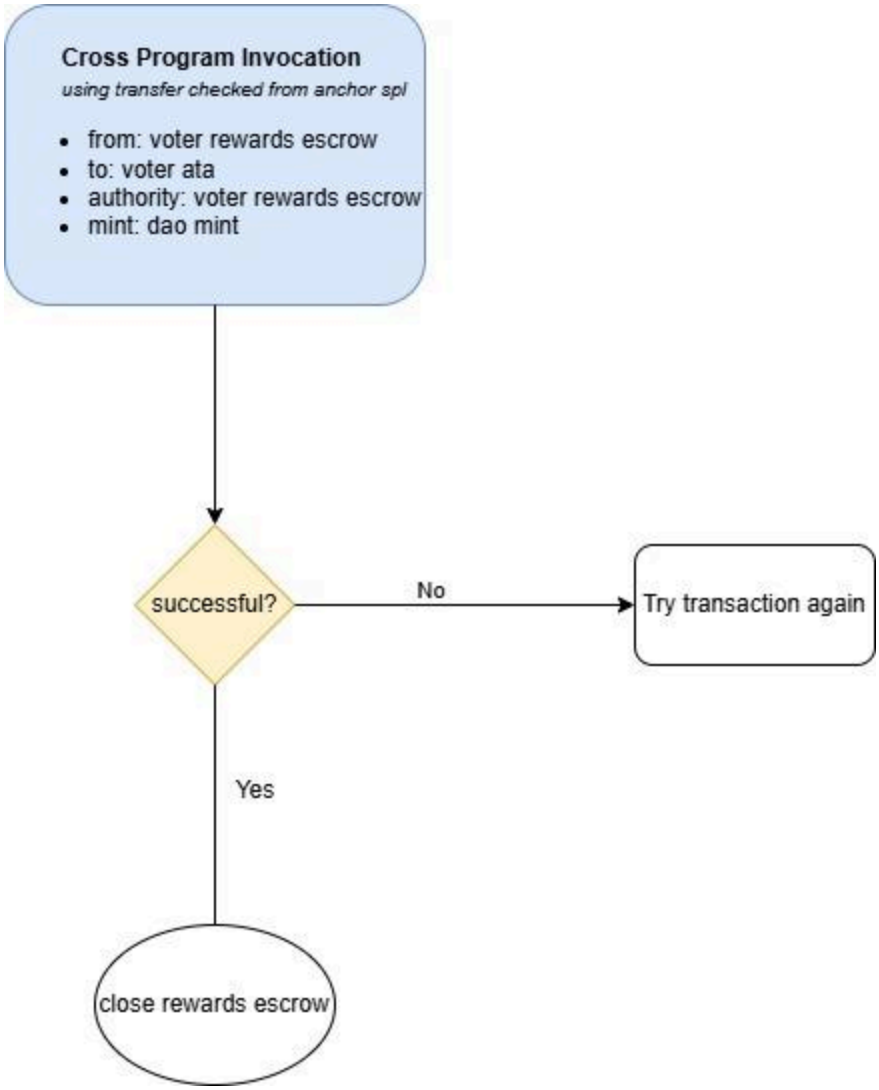

is valid dao member
&
has >= 150 tokens
and not proposer
No
ERROR
Yes
call stake_tokens()
transfer an amount of dao token from voter ata to staking vault
Cross Program Invocation
using transfer checked from anchor spl
from: user dao token ata
to: staking vault
authority: voter (dao member)
mint: dao mint
call vote_on_proposal()
fill in vote metadata and submit to staking vault
END

USER AS REWARDS (for proposer and voter)

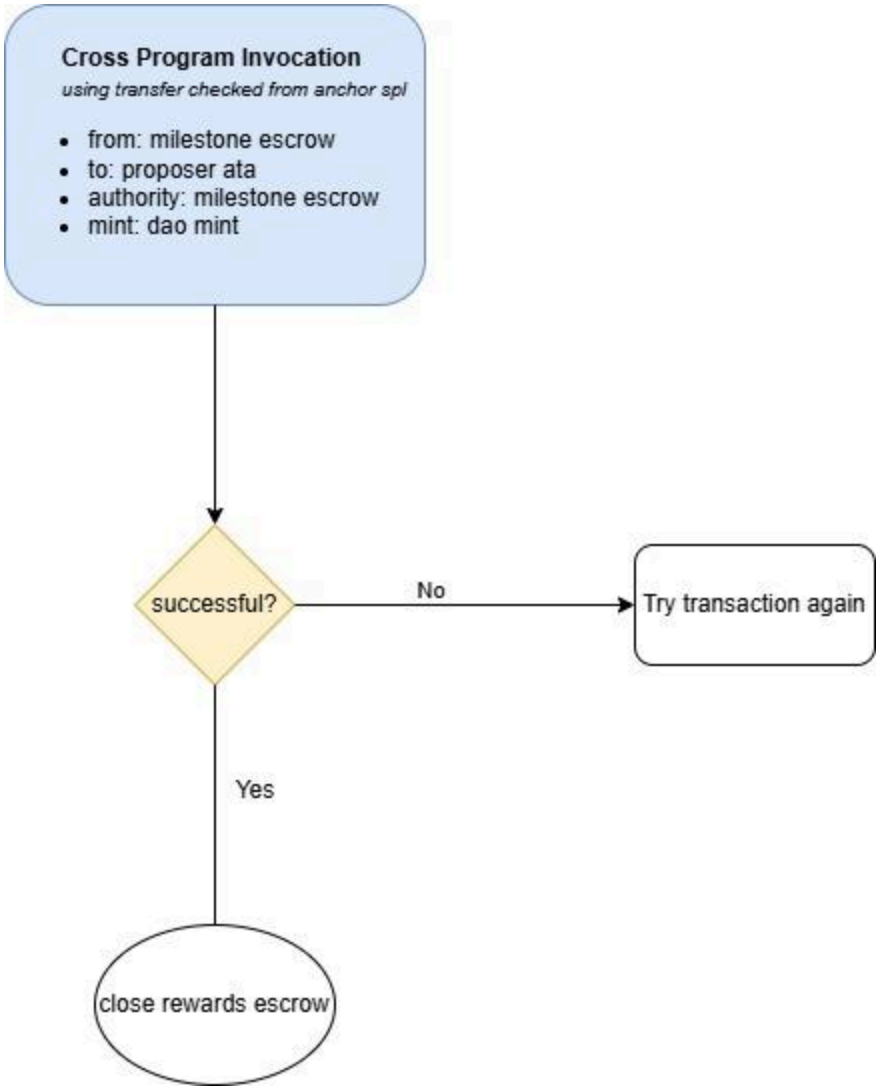


Instructions

claim_rewards()



request_funds()


Cross Program Invocation
using transfer checked from anchor spl
from: milestone escrow
to: proposer ata
authority: milestone escrow
mint: dao mint
successful?
No
Try transaction again
Yes
close rewards escrow