Users & Summary Of Actions

Admin

- Initialize Program: This action creates the GlobalState PDA, which stores the protocol's core configuration, including the Admin's public key, initial fee structure, the reward rate (SOL per Credit), and establishes the SOL Vault PDA for holding rewards.
- Admin Change Config: This action updates core protocol parameters, allowing the Admin to modify the sol_per_credit rate, adjust the base_deposit required from users, and manage the Whitelist Mint list used by the program.
- Whitelist/Unwhitelist mint: Approve or disapprove a mint from being used
- Admin Withdraw SOL/Token: This action permits the Admin to withdraw SOL from the sol_vault or recover any extraneous, non-protocol tokens accidentally transferred to any of the program's token accounts.

Keeper Bot

- Create Keeper Account: This action registers a bot by creating its unique KeeperState PDA using its wallet key, which tracks the total number of credits the bot has earned for executing transactions.
- Bot Decrease Liquidity (Raydium → Meteora): This action executes the Out-of-Range logic: it uses a CPI to Raydium to withdraw assets, deposits the resulting tokens into the Meteora Vault via a CPI, sets the UserState to deployed, and rewards the bot with decrease_liquidity_credits.
- Bot Increase Liquidity (Meteora → Raydium): This action executes the Back-in-Range logic: it uses a CPI to Meteora to withdraw LP tokens, re-deposits assets into a new Raydium CL position via a CPI, sets the UserState to not deployed, and rewards the bot with increase_liquidity_credits fees may also deducted during this step
- Keeper Withdraw Rewards: This action calculates the bot's total SOL earnings, transfers the SOL from the sol_vault to the keeper's wallet, and resets the Keeper's credit balance to zero.

User

- User Create Position: This action initializes a user's commitment: it creates the UserState PDA, transfers the Raydium CL Position NFT to the program's custody, and deducts the non-refundable Base Deposit SOL fee.

- User Close Position (Final): This action terminates the position: it returns the Raydium Position NFT, any dust funds, the remaining Base Deposit SOL, and closes the UserState account, fees may also deducted during this step.

Program Accounts

GlobalState

Owner: My program
Description:
It stores global wide configuration.

Seeds:
[b"global-state"]

Fields:
- state: u8 → Protocol state flags (bitmask)
- admin: Pubkey → Admin authority address
- sol_vault: Pubkey → SOL vault PDA
- credits_for_decrease_liquidity: u64 → Reward credits for Decrease Liquidity
- credits_for_increase_liquidity: u64 → Reward credits for Increase Liquidity
- sol_per_credit: u64 → Lamports per credit exchange rate
- base_deposit: u64 → Initial refundable deposit
- bump: u8 → Bump for GlobalState PDA
- sol_vault_bump: u8 → Bump for SOL vault PDA

UserState

Owner: My program
Description:
Per-user position and rebalancing state linked to Raydium position NFT.

Seeds:
[b"user-state", self.user_mint.as_ref()]

Fields:
- user: Pubkey → Wallet owner of this position
- user_mint: Pubkey → Raydium position NFT mint
- liquidity: u128 → Amount of liquidity last pulled out of meteora
- token_deployed:TokenDeployed
- lp_amount: u64 → Meteora LP Token balance
- tick_lower_index_in_threshold: i32 → Lower tick threshold for rebalancing
- tick_upper_index_in_threshold: i32 → Upper tick threshold for rebalancing
- tick_lower_index_out_threshold: i32 → Lower tick threshold for rebalancing
- tick_upper_index_out_threshold: i32 → Upper tick threshold for rebalancing
- tick_lower_index: i32 → Lower tick of active Raydium position
- tick_upper_index: i32 → Upper tick of active Raydium position
- bump: u8 → PDA bump
- reserved: [u8;127] → Reserved space

Token deployed(Enum)
Token0

Token1

NotDeployed

KeeperState

Owner: My program
Description:
Tracks the reward credits earned by a keeper bot for performing rebalances.

Seeds:
[b"keeper", self.keeper.as_ref()]

Fields:
- keeper: Pubkey → Keeper bot wallet
- credits: u64 → Total earned credits

WhitelistState

Owner: My program
Description:
Stores a whitelisted mint

Seeds:
[b"whitelist-state", self.mint.as_ref()]

Fields:
- keeper: Pubkey → Keeper bot wallet
- credits: u64 → Total earned credits

Personal Position State

Owner: Raydium Concentrated Liquidity Program
Description:
Represents a user's specific Raydium liquidity position, linked to their Position NFT and used to track liquidity, fees, and rewards.

Seeds:
[b"position", self.nft_mint.as_ref()]

Fields:
- bump: [u8;1] → PDA bump for this position
- nft_mint: Pubkey → Mint of the Raydium position NFT
- pool_id: Pubkey → Associated pool for this position
- tick_lower_index: i32 → Lower tick bound
- tick_upper_index: i32 → Upper tick bound
- liquidity: u128 → Liquidity owned by this position
- fee_growth_inside_0_last_x64: u128 → Token 0 fee growth at last update
- fee_growth_inside_1_last_x64: u128 → Token 1 fee growth at last update
- token_fees_owed_0: u64 → Token 0 fees claimable
- token_fees_owed_1: u64 → Token 1 fees claimable
- reward_infos: [PositionRewardInfo; REWARD_NUM] → Reward tracking array
- recent_epoch: u64 → Last update epoch timestamp
- padding: [u64;7] → Reserved for future expansion

Raydium accounts
These are accounts that the program interacts with directlty reading relevant data from them

Pool State

Owner: Raydium Concentrated Liquidity Program
Description:
Represents the on-chain state of a Raydium liquidity pool (e.g. SOL/USDC), containing configuration, liquidity, and fee/reward tracking data.

Seeds:
[b"pool", self.amm_config.as_ref(), self.token_mint_0.as_ref(), self.token_mint_1.as_ref()]

Fields (core subset):
- bump: [u8;1] → PDA bump for this pool
- amm_config: Pubkey → Pool configuration account
- token_mint_0: Pubkey → Token 0 mint
- token_mint_1: Pubkey → Token 1 mint
- token_vault_0: Pubkey → Token 0 vault (Raydium owned)
- token_vault_1: Pubkey → Token 1 vault (Raydium owned)
- liquidity: u128 → Total active liquidity
- sqrt_price_x64: u128 → Current price in Q64.64 format
- tick_current: i32 → Current pool tick index
- fee_growth_global_0_x64: u128 → Total Token 0 fee growth
- fee_growth_global_1_x64: u128 → Total Token 1 fee growth
- status: u8 → Bitmask of pool feature flags
- reward_infos: [RewardInfo; REWARD_NUM] → Pool reward configurations

Tick Array State (Lower / Upper)

Owner: Raydium Concentrated Liquidity Program
Description:
Stores the initialized state of tick indices defining a liquidity position's range boundaries.

Other raydium accounts
These are accounts that the program does not interact with directly so the specific fields and other details are not relevant

Protocol Position

Owner: Raydium Concentrated Liquidity Program
Description:
Legacy account included for backward compatibility. Used for protocol-level or historical fee tracking.

Tick Array Bitmap Extension

Owner: Raydium Concentrated Liquidity AMM Program
Description:
Auxiliary account extending the pool's tick bitmap, allowing additional tick arrays beyond the fixed primary range to be tracked efficiently.

Meteora accounts
The program does not interact with any metoera specific program accounts directly

Vault Account

Owner: Meteora Vault Program
Description:
The primary account representing a specific yield-bearing vault in the Meteora protocol. It stores the vault's configuration, references to strategy accounts, fee and token vault addresses, and tracks deposits, LP issuance, and profit locking logic.

GlobalState Token Account

Authority: GlobalState PDA
Mint: Whitelisted Pool Mints
Description:
Internal token vaults controlled by your program's GlobalState PDA. Used as intermediate sources or destinations for tokens during Raydium and Meteora CPI calls.

GlobalState LP Token Account

Authority: GlobalState PDA
Mint: Meteora LP Mint
Description:
Holds LP tokens received from Meteora vault deposits. Controlled by the GlobalState PDA for programmatic rebalancing and reward tracking.

Program Fee Token Accounts

Authority: GlobalState PDA
Mint: Whitelisted Pool Mints
Description:
Dedicated vaults that collect protocol fees across whitelisted mints, controlled by the GlobalState PDA.

User Token Account

Authority: User Wallet (Signer)
Mint: Token Whitelisted Pool Mints
Description:
User-owned ATAs that act as the source or destination of funds during deposits, withdrawals, and Raydium liquidity changes.

Token accounts
These are token accounts used across the different programs that are interacted with, the program which it is used by is specified

Raydium Token Vault

Authority: Raydium CLMM Program
Mint: Whitelisted Pool Mints
Description:
Raydium's internal pool token vaults for the CLMM pool. Used by Raydium during liquidity adjustments and swaps.

Token 0 Mint

Authority: Token Mint Authority
Description:
Defines the first asset in the pair (e.g., WSOL). Used by both Raydium and Meteora vault interactions.

Token 1 Mint

Authority: Token Mint Authority
Description:
Defines the second asset in the pair (e.g., USDC). Used by both Raydium and Meteora vault interactions.

Meteora LP Mint

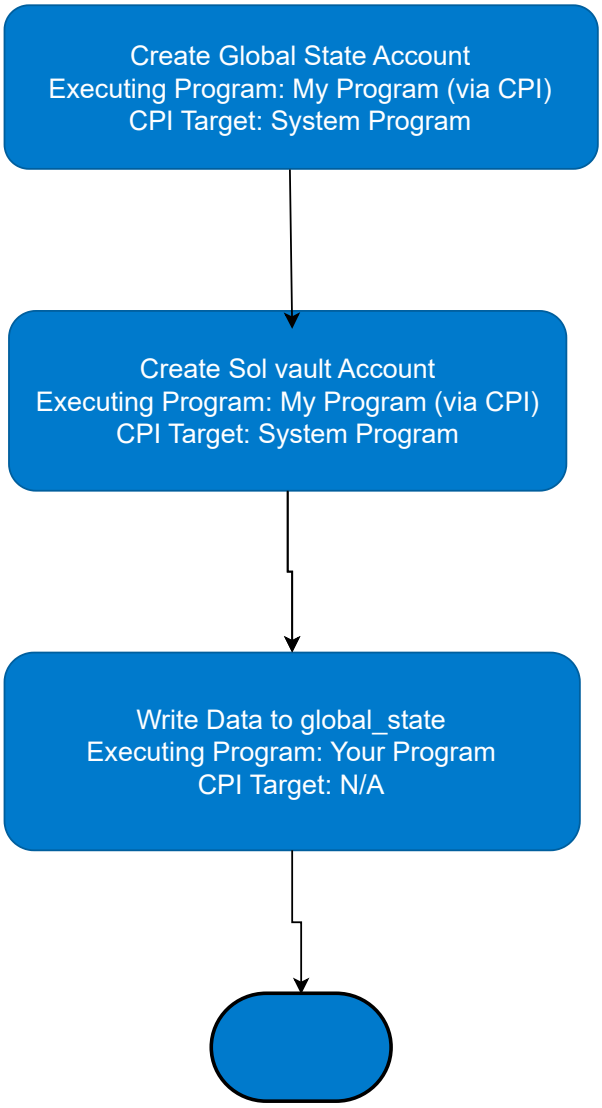
Authority: Meteora Vault Program
Description:
Defines the fungible LP token issued by Meteora vaults to represent user shares in the vault's underlying liquidity.

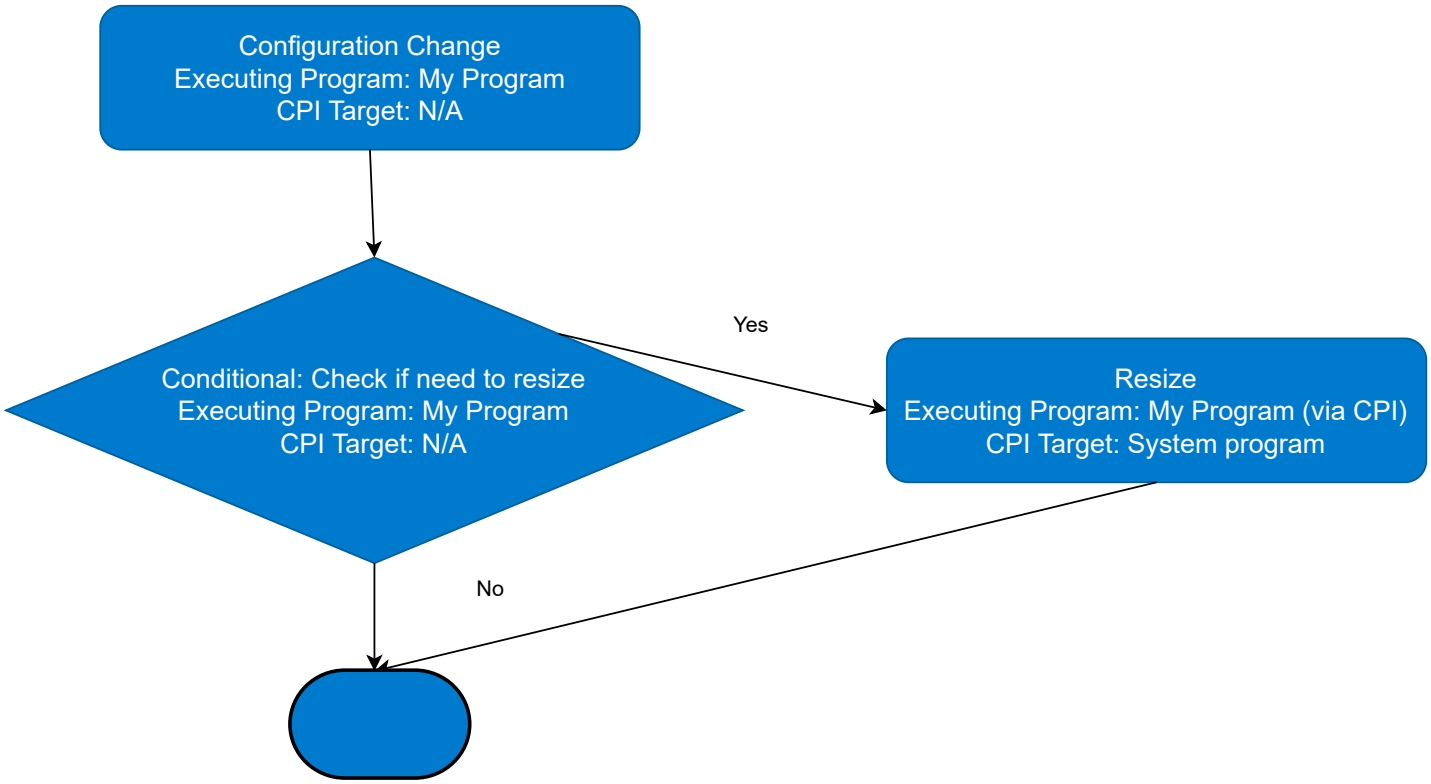
Raydium Position NFT Mint

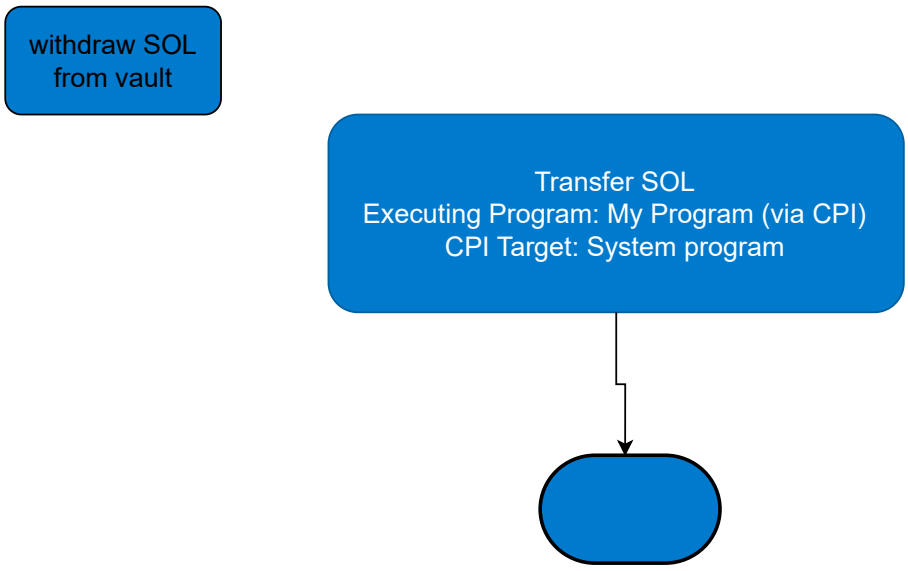
Authority: Raydium CLMM Program
Description:
Defines the non-fungible token representing ownership of a unique Raydium CLMM liquidity position.

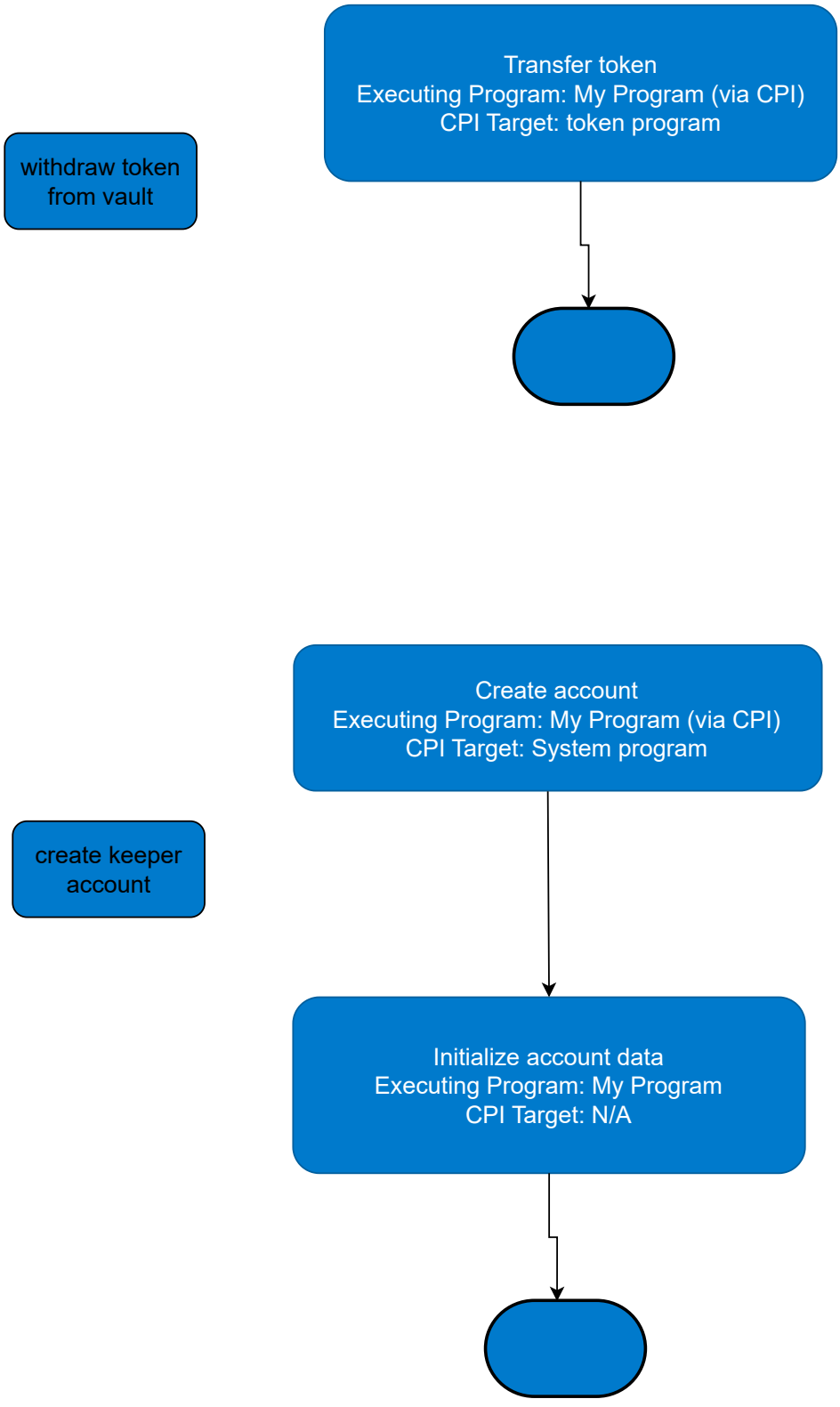
Instructions/Actions

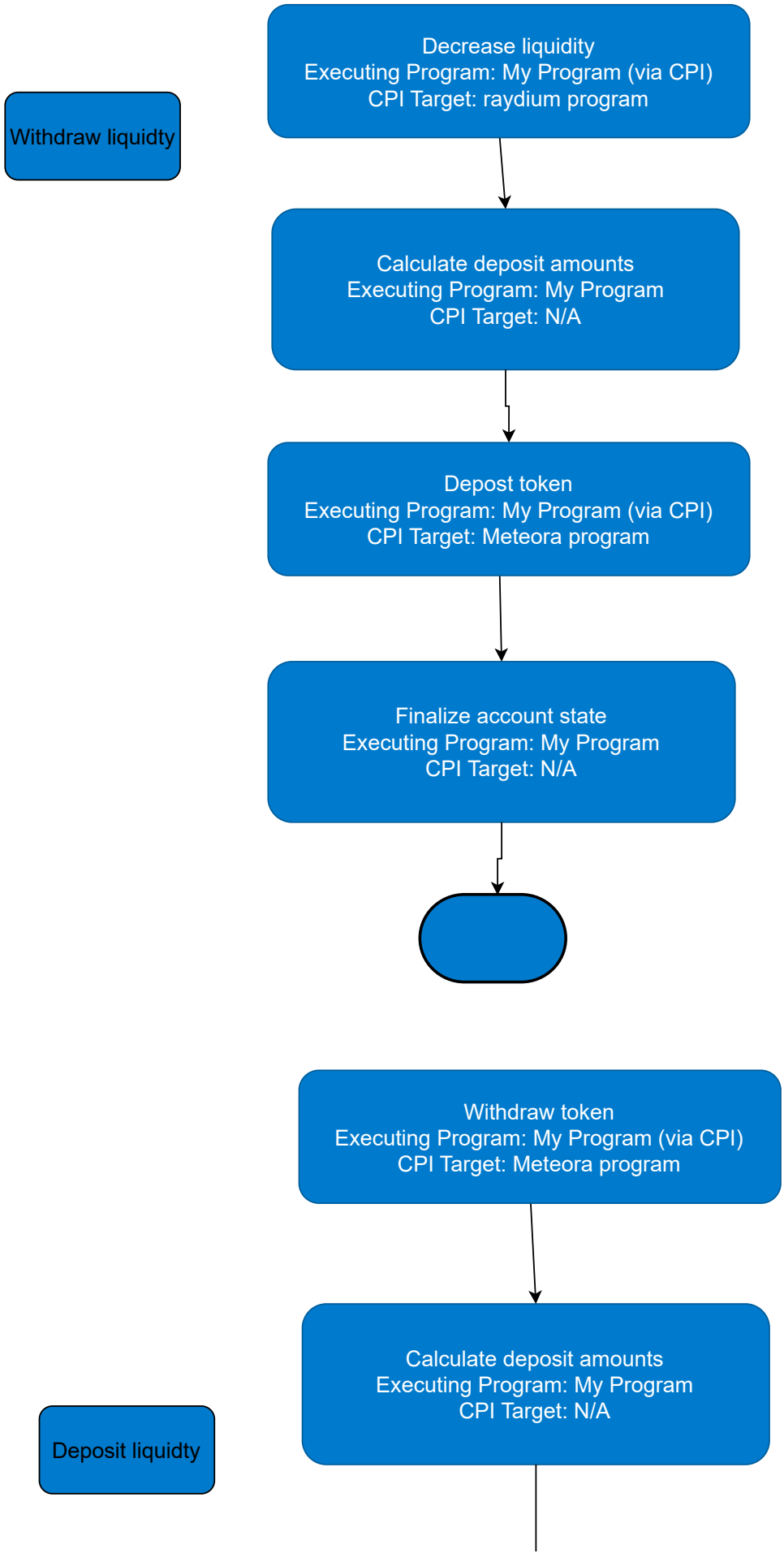
Initialize global config

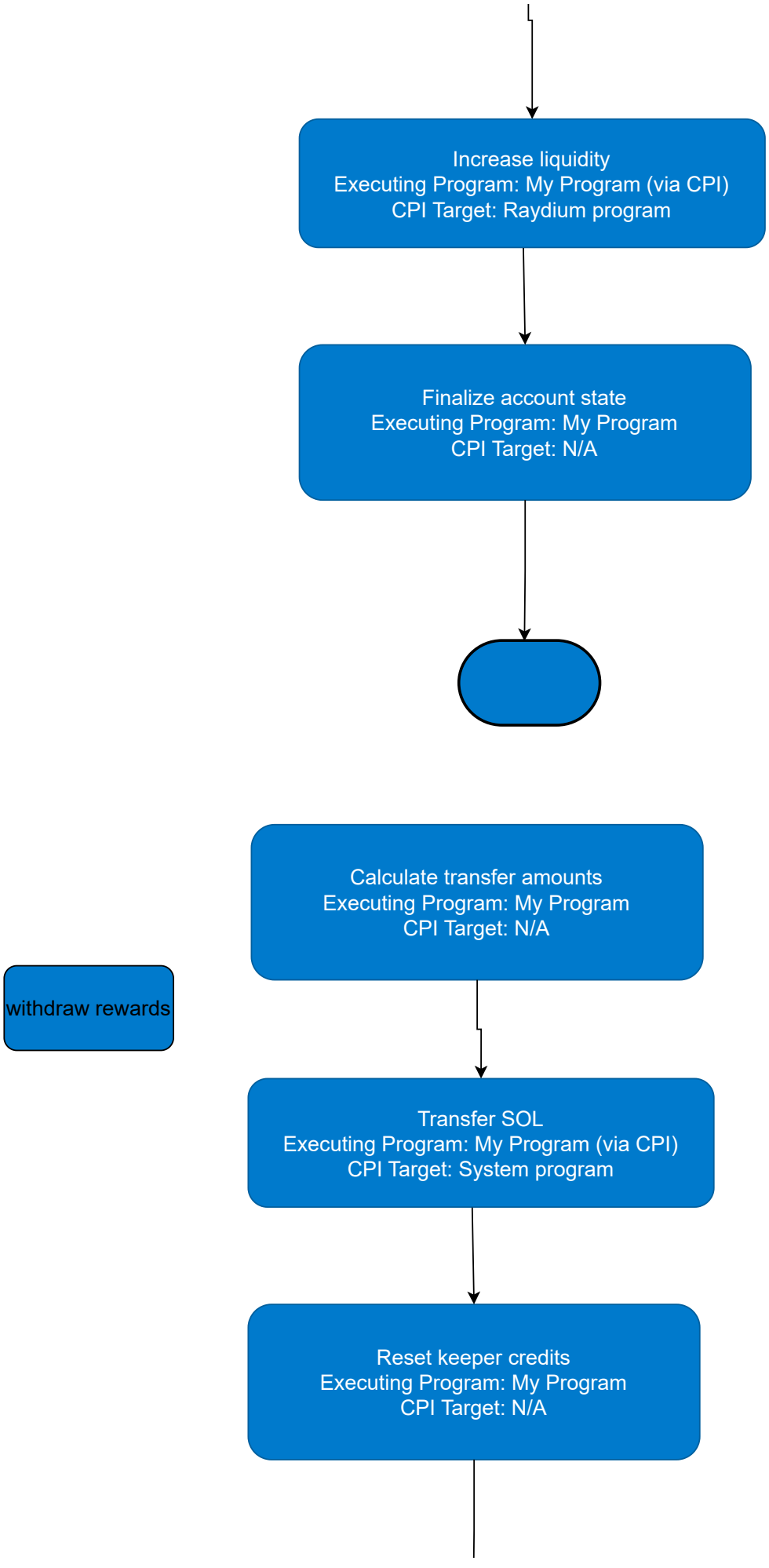


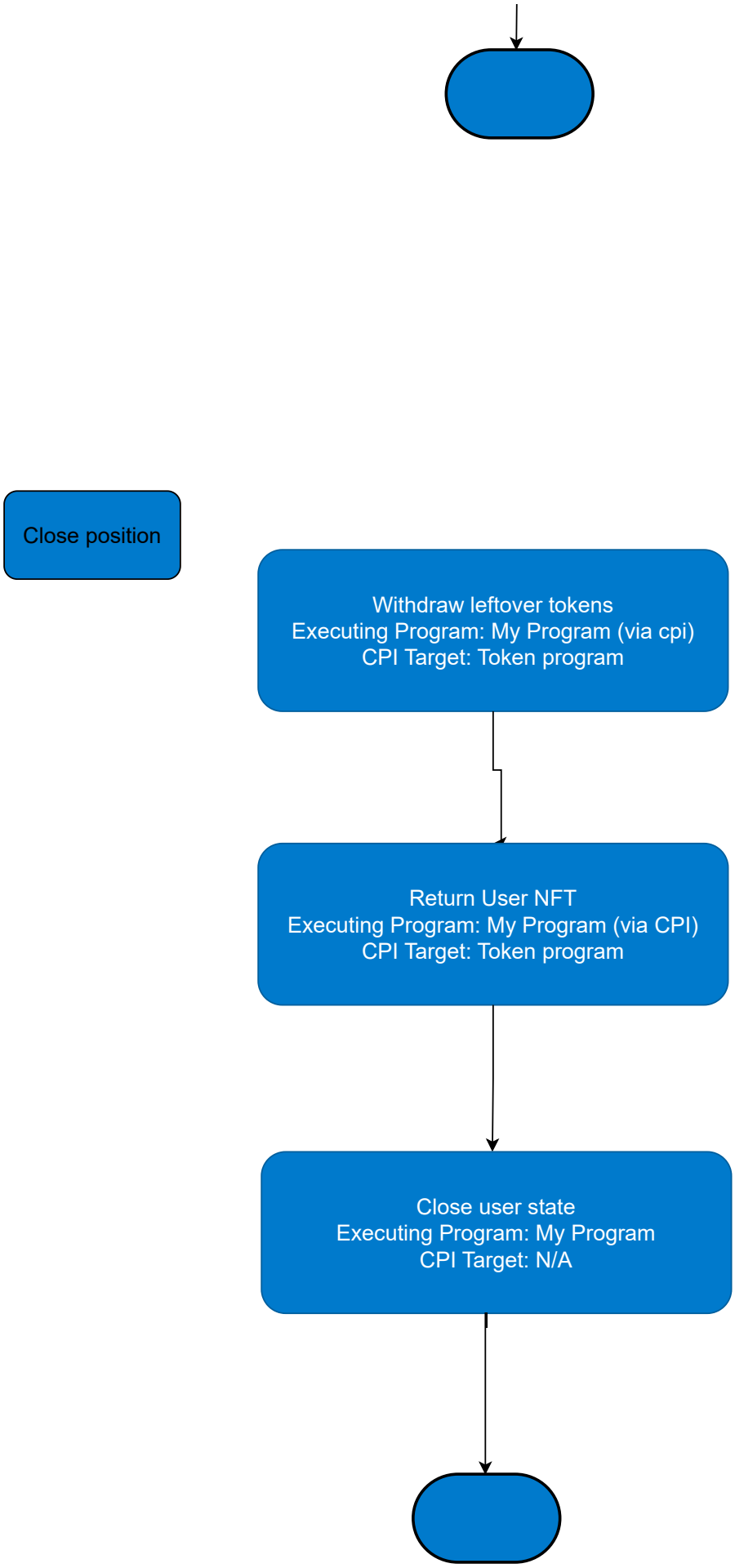

Configuration Change
Executing Program: My Program
CPI Target: N/A
Conditional: Check if need to resize
Executing Program: My Program
CPI Target: N/A
Yes
Resize
Executing Program: My Program (via CPI)
CPI Target: System program
No

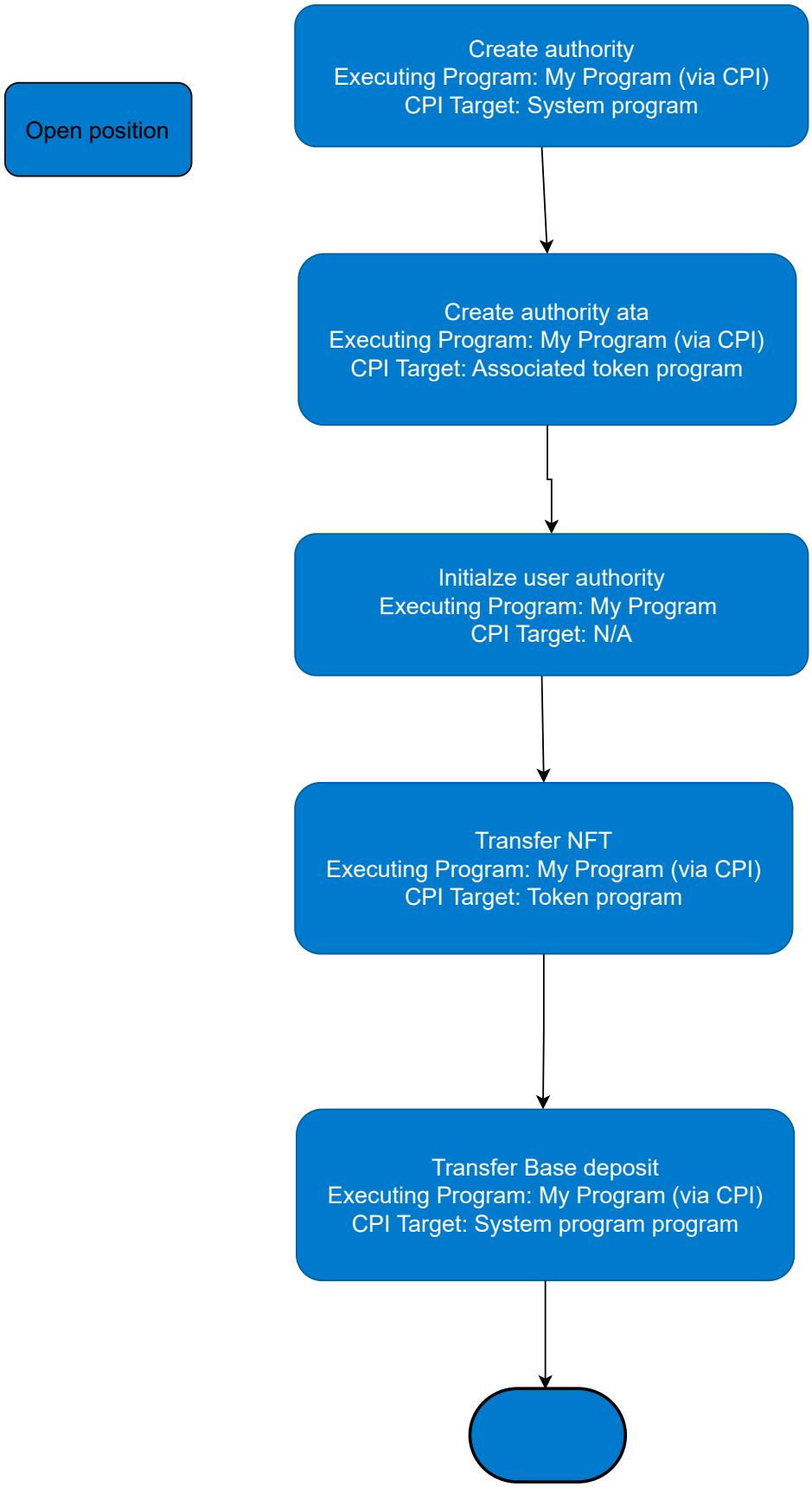

withdraw SOL
from vault
Transfer SOL
Executing Program: My Program (via CPI)
CPI Target: System program

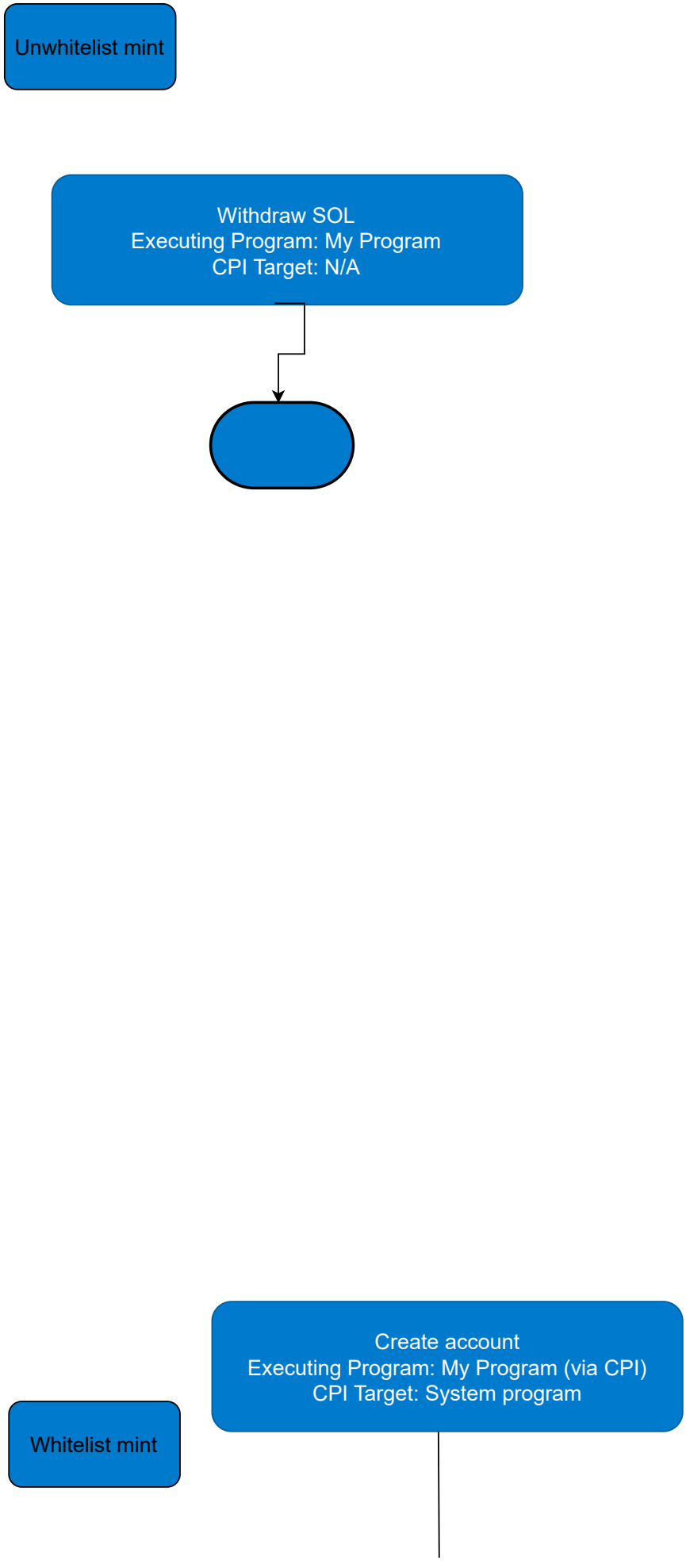

withdraw token from vault
Transfer token
Executing Program: My Program (via CPI)
CPI Target: token program
create keeper account
Create account
Executing Program: My Program (via CPI)
CPI Target: System program
Initialize account data
Executing Program: My Program
CPI Target: N/A

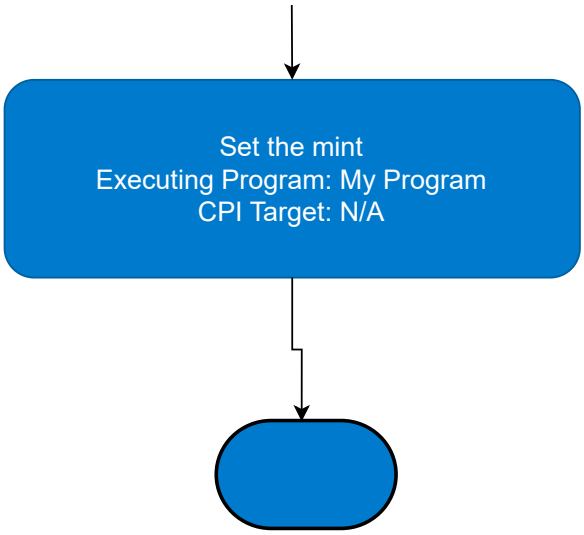

Withdraw liquidty
Decrease liquidity
Executing Program: My Program (via CPI)
CPI Target: raydium program
Calculate deposit amounts
Executing Program: My Program
CPI Target: N/A
Depost token
Executing Program: My Program (via CPI)
CPI Target: Meteora program
Finalize account state
Executing Program: My Program
CPI Target: N/A
Withdraw token
Executing Program: My Program (via CPI)
CPI Target: Meteora program
Calculate deposit amounts
Executing Program: My Program
CPI Target: N/A
Deposit liquidty


Increase liquidity
Executing Program: My Program (via CPI)
CPI Target: Raydium program
Finalize account state
Executing Program: My Program
CPI Target: N/A
withdraw rewards
Calculate transfer amounts
Executing Program: My Program
CPI Target: N/A
Transfer SOL
Executing Program: My Program (via CPI)
CPI Target: System program
Reset keeper credits
Executing Program: My Program
CPI Target: N/A


Close position
Withdraw leftover tokens
Executing Program: My Program (via cpi)
CPI Target: Token program
Return User NFT
Executing Program: My Program (via CPI)
CPI Target: Token program
Close user state
Executing Program: My Program
CPI Target: N/A


Open position
Create authority
Executing Program: My Program (via CPI)
CPI Target: System program
Create authority ata
Executing Program: My Program (via CPI)
CPI Target: Associated token program
Initialze user authority
Executing Program: My Program
CPI Target: N/A
Transfer NFT
Executing Program: My Program (via CPI)
CPI Target: Token program
Transfer Base deposit
Executing Program: My Program (via CPI)
CPI Target: System program program


Unwhitelist mint
Withdraw SOL
Executing Program: My Program
CPI Target: N/A
Create account
Executing Program: My Program (via CPI)
CPI Target: System program
Whitelist mint



Relevant programs

Raydium program
- For giving access to the tokens owned by the user through the increase and decrease liquidity instructions

Meteora program
- For providing a means to earn additional yield on the user's idle liquidity

System program
- For creating accounts and making lamport transfer

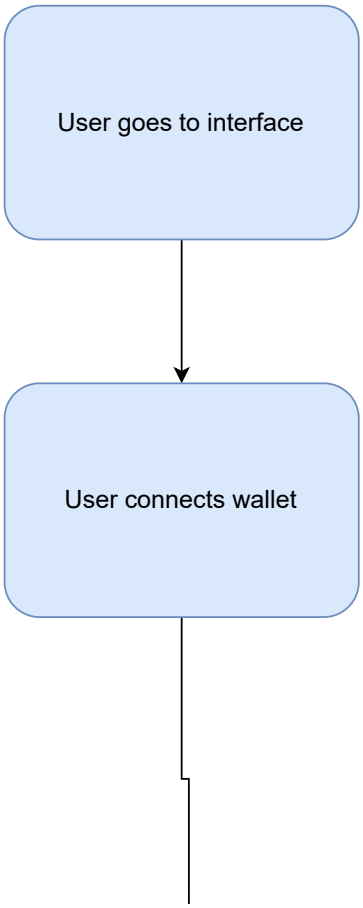
Token/Token 2022 program
- For making token transfers

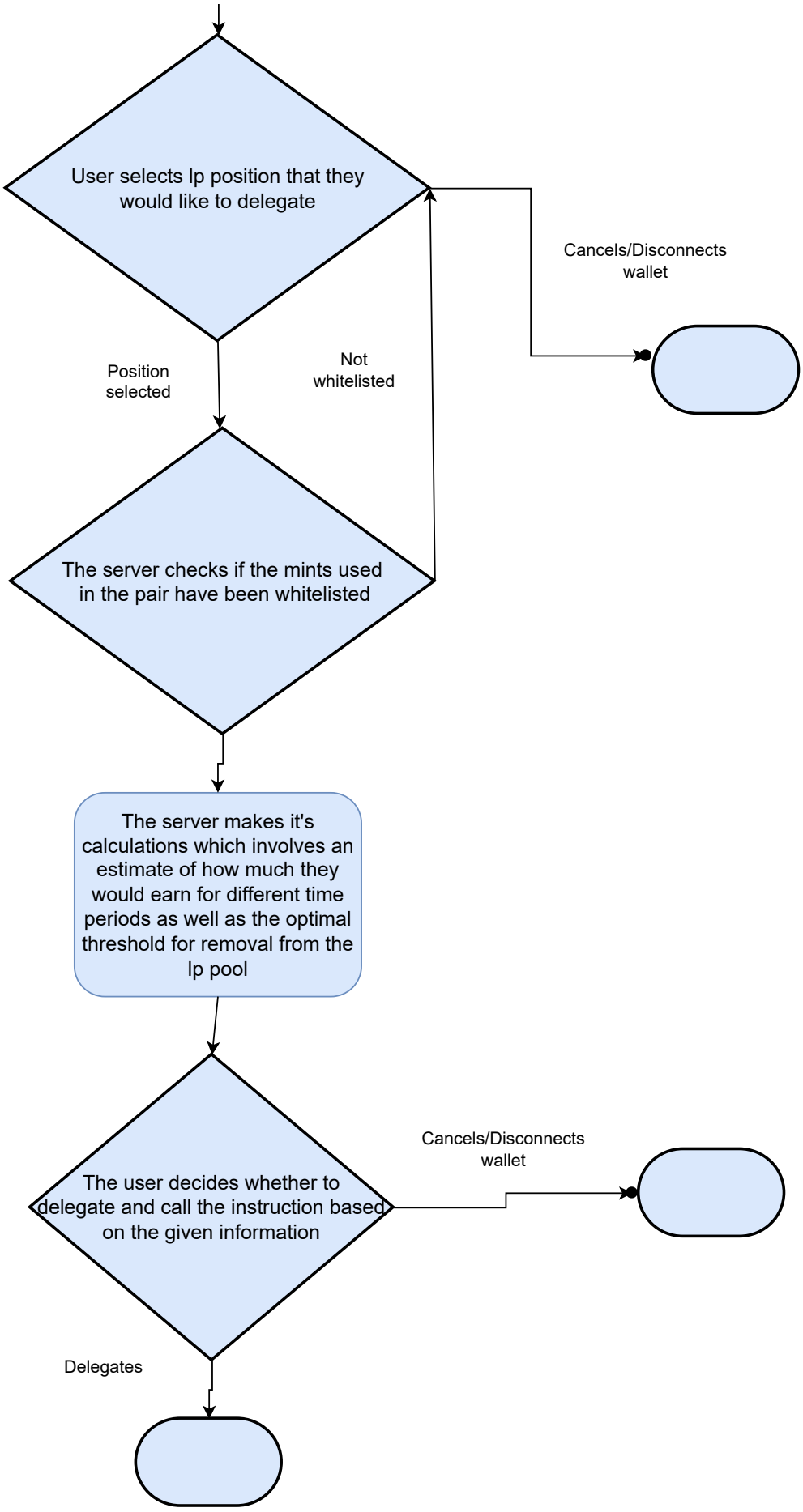
Associated token program
- For creating associated token accounts

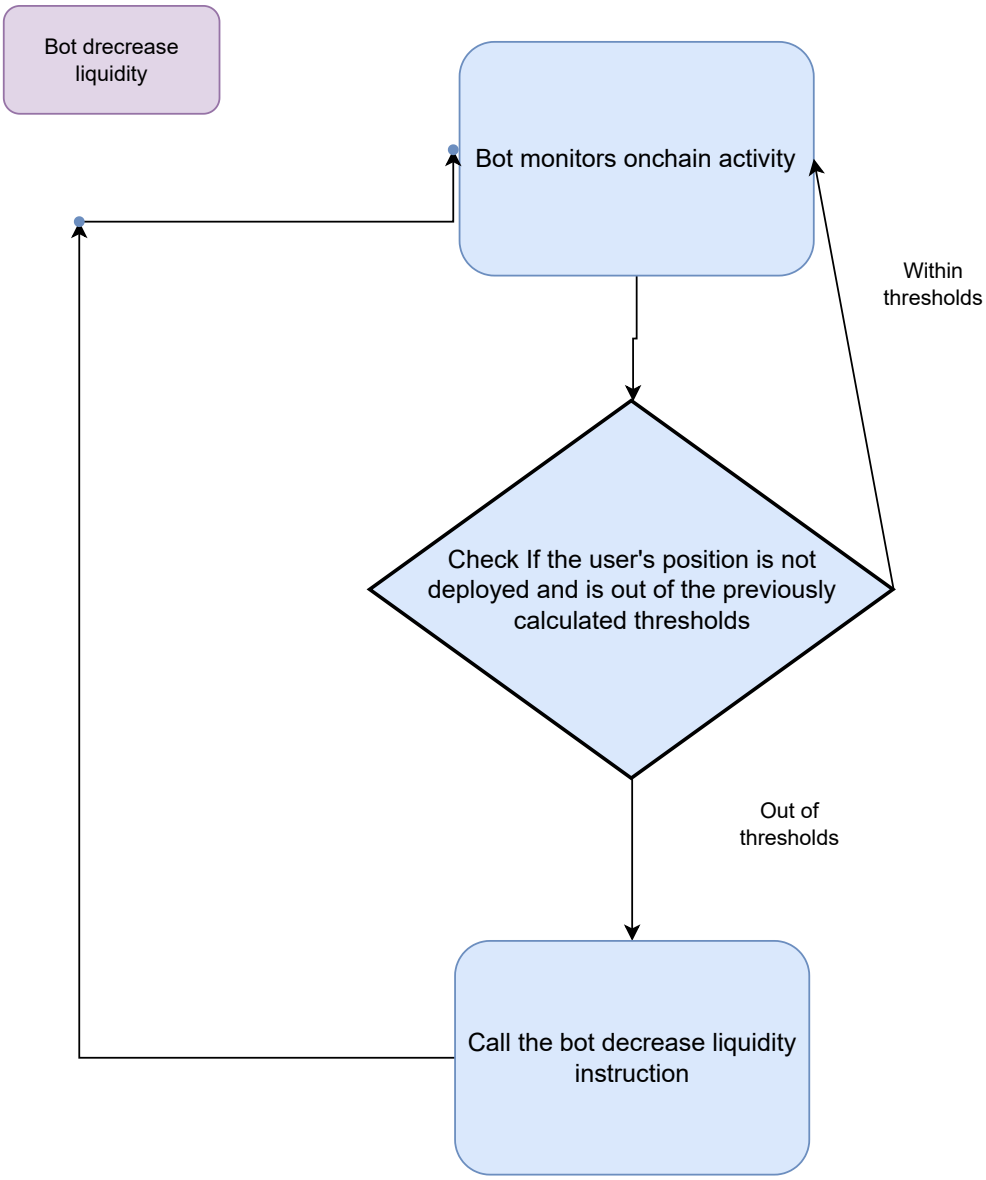
Relevant offchain flows

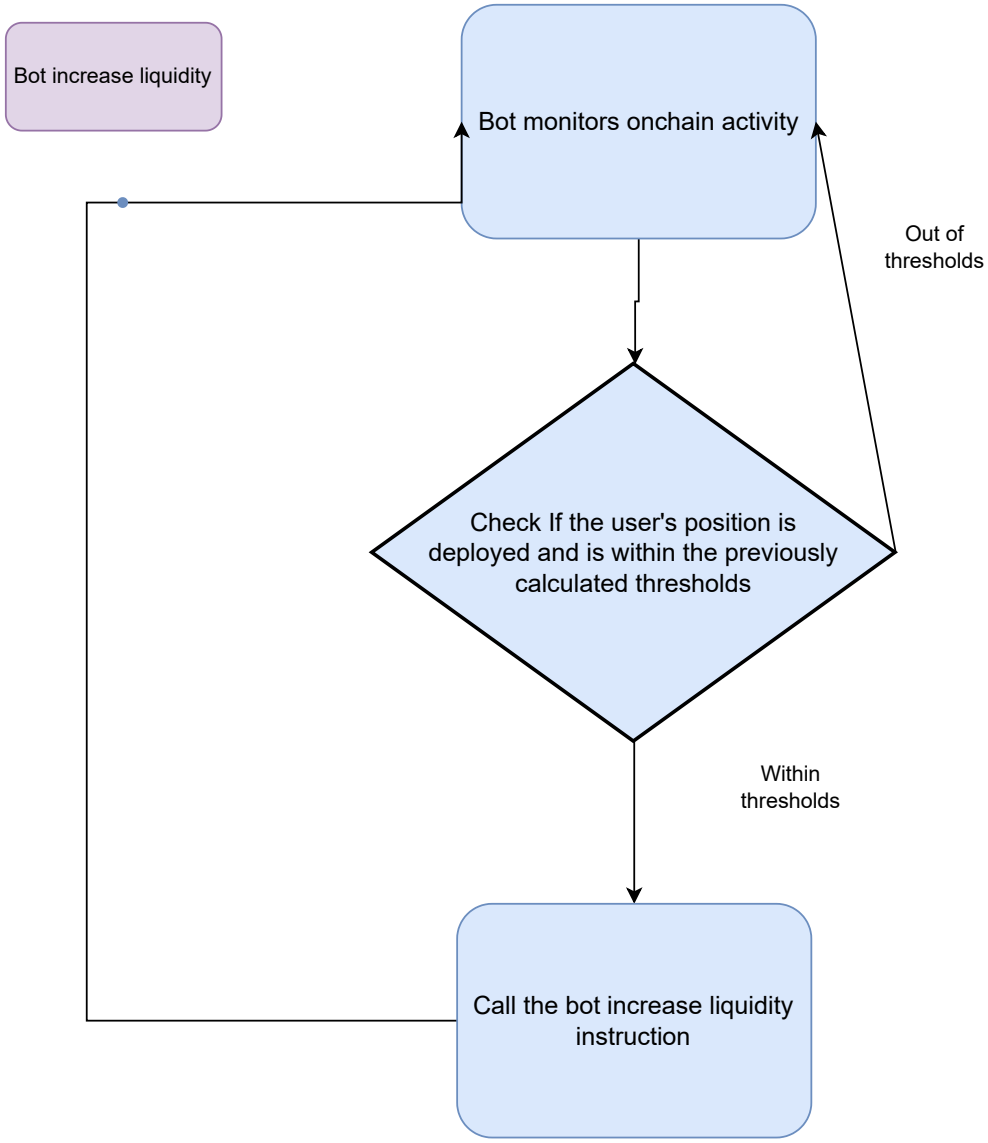
Offchain flows
These are flows that eventually boil down to action/instructions performed on chain, but requiring some off chain coordination

User create position




User selects lp position that they would like to delegate
Cancels/Disconnects wallet
Position selected
Not whitelisted
The server checks if the mints used in the pair have been whitelisted
The server makes it's calculations which involves an estimate of how much they would earn for different time periods as well as the optimal threshold for removal from the lp pool
The user decides whether to delegate and call the instruction based on the given information
Cancels/Disconnects wallet
Delegates


Bot drecrease liquidity
Bot monitors onchain activity
Within thresholds
Check If the user's position is not deployed and is out of the previously calculated thresholds
Out of thresholds
Call the bot decrease liquidity instruction


Bot increase liquidity
Bot monitors onchain activity
Out of thresholds
Check If the user's position is deployed and is within the previously calculated thresholds
Within thresholds
Call the bot increase liquidity instruction