Solmate Project User Manual

Since:Thursday, December 30, 2021

Version: 1

Table of Content

This project is really computationally heavy, that's the challenge we tackled.

1. Facts About Solana Chain

- Trasaction size limit: 1232 bytes
- Compute unit limit per contract execution: 200k
- Account size limit: 10 MB

2. Architecture

2.1. System Architecture

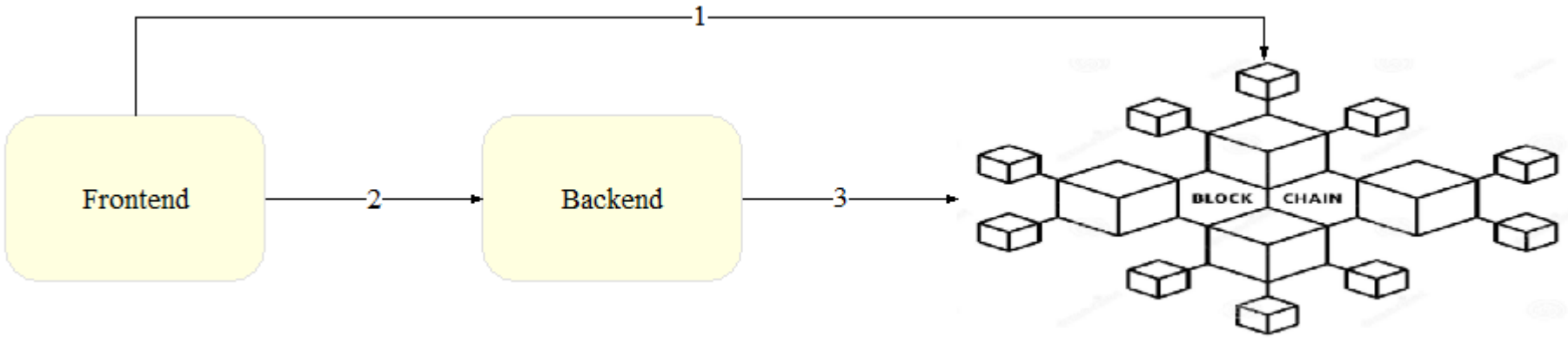


Figure 0 System Architecture

#1, if signed by staker, just simulation, not really commit the transaction

#2, wire transferable objects signed by stakers

#3, host wallet is always a signer if necessary

- This architecture is highly recommeded with respect to securities, it involves two phases of signing across network, it's better to deploy frontend and backend on different servers.
- The frontend is not really part of my job, not a good UX but I have to test, what's more, lots of code there can be reused.

2.2. Core Data Model

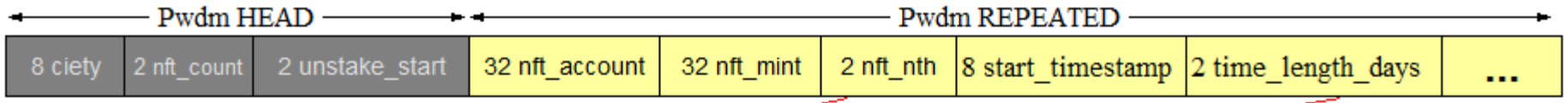


Figure 1 Per Wallet Data Model, denoted as Pwdm

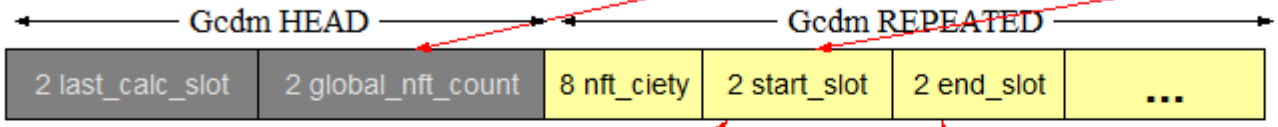


Figure 2 Global Ciety Data Model, denoted as Gcdm

Gsdm REPEATED

2 nft_count	2 nft_last_ind	1 nft_nth				
3	2	1	0	1		
4	3	1	0	0	1	
...						
5	4	0	1	0	0	1

Figure 3 Global Slots Data Model, denoted as Gsdm

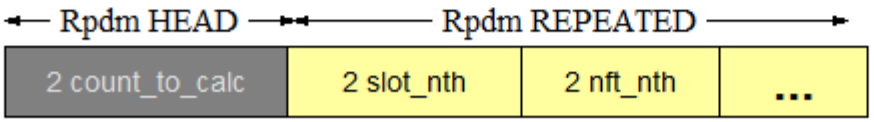


Figure 4 Global Reward Prep Data Model, denoted as Rpdm

The first byte of all HEADs is semantically served as is_initialized

Core Data Model Layout		
Gsdm	is_initialized(1) + 365 * (nft_count(2) + nft_last_ind(2) + nft_nth(1) * 7000)	2556461
Gcdm	is_initialized(1) + last_calc_slot(2) + global_nft_count(2) + (ciety_of_nth_nft(8) + start_slot(2) + end_slot(2)) * 7000	84005
Pwdm	is_initialized(1) + ciety(4) + nft_count(2) + unstake_start(2) + (nft_account(32) + nft_pda(32) + nft_n_th(2) + start_timestamp(8) + time_length_days(2)) * 100	7609
Rpdm	is_initialized(1) + count_to_calc(2) + 365 * (slot_nth(2) + nft_nth(2))	1463

- Without this model, basically it's practically infeasible to implement this project, if using some frequently mentioned stuff, like Anchor, I guess.
- We have to come to this low level data model all due to Solana's compute unit limit by squeezing the performance byte by byte to reduce the time complexity.
- All yellow blocks are repated segment
- Pwdm
  - Ciety, a staker's rewards, but only after staker's viewing
  - Nft_count, the total number of staked NFTs of a staker
  - Unstake_start, an index from which it starts to unstake a certain number of NFTs, it's wise to assume any blockchain may crash, for this reason it has to remember where to start unstake
  - Nft_Nth is a pointer which points to the index of repeated blocks in Gcdm where stores rewards of each NFT
- Gcdm
  - Last_calc_slot, in preparation for reward calc, Rpdm always starts from it down to current slot, Rpdm stores a reward calc plan
  - Global_nft_count, total number of NFTs of all stakers
  - Nft_ciety, stores the rewarded ciety on a specific NFT
- Gsdm
  - Nft_count, the total number of NFTs staked on a slot
  - Nft_last_ind, the last index of NFT that's staked on a slot

- 0/1 indicates if an NFT staked on that index on a slot
- Rpdm
  - Slot_nth, slot index
  - Nft_nth, nft index on slot nft_nth
-

3. Deliverables & Reusables

3.1. App_frontend

- Src/solmate/v2_*
- Src/solmate/read_chain_common.js
- Src/solmate/read_chain_inspector.js

3.2. projects

- App_backend
- Program_stake
- Program_token

4. Configuration

4.1. App_backend/solmate/v2_config.js

```
const TOTAL_NFTS = 7000;
/**
 * ========================================
 * Gcdm data layout:
 * ========================================
 * is_initialized,   1 byte
```

```
 * last_stake_slot,  2 bytes
 * last_calc_slot,   2 bytes
 * global_nft_count, 2 bytes
 *
 * ======the following is repeated per nft===========
 * nft_ciety,  8 bytes
 * start_slot, 2 bytes
 * end_slot,   2 bytees
 */
const GCDM_SIZE = 84005; // = 1 + 2 + 2 + (8 + 2 + 2) * 7000

/**
 * ========================================
 * Gsdm data layout:
 * ========================================
 * is_initialized, 1 byte
 *
 * ======the following is repeated per slot===
 * nft_count,    2 bytes
 * nft_last_ind, 2 bytes
 * nft_nth,      1 byte, 7000 bytes in total
 */
const GSDM_SIZE = 2556461; // for 365 days, = 1 + 365 * (2 + 2 + 7000 * 1)

/**
 * ========================================
```

```
 * Rpdm data layout:
 * ========================================
 * is_initialized, 1 byte
 * count_to_calc,  2 bytes
 * ===below repeated==
 * slot_nth,       2 bytes
 * nft_nth,        2 bytes
 */
const RPDM_SIZE = 1463; // for 365 days, = 1 + 2 + 365 * (2 + 2)
exports.config={
    network: 'devnet',

    // Redeploy a Solana program doesn't need to change these addresses,
    // but you can if needed
    stake_program_addr: 'HAoqALFTiyibSRgh63qUWX1sdxt82R3hCqSwAmk8Pysw',
    token_program_addr: '7Pfq1Lz24RWDrsJxCYaFuw3J9vVwPvcNiPeu38ssLrRo',

    gsdm_size: GSDM_SIZE,
    gcdm_size: GCDM_SIZE,
    rpdm_size: RPDM_SIZE,

    // if you want to replay for some reasons, you need to change the number suffix
    // and re-initialize,
    // they are used to derive keys for accounts from hosst or staker wallet
    seed_config: 'seed_config_25',
    seed_ciety: 'seed_ciety_25',
```

```
    seed_gsdm: 'seed_gsdm_25',
    seed_gcdm: 'seed_gcdm_25',
    seed_pwdm: 'seed_pwdm_25',
    seed_rpdm: 'seed_rpdm_25',

    // $CIETY distribution in the begining is performed off-chain, since it's one time job.
    // but, the remaining is locked on-chain with an account owned by the smart contract
    total_supply: 1000000000,
    to_community_percent: 0.2,
    to_community_address: 'AJz6nffDtCBtquZj4gSHMj9YjHDhETZe2RpA2S9Pfqf4',
    to_team_percent: 0.05,
    to_team_addresses: ['DnUspL6hJpp7edGF2LGt4vSEUwttYxvVR3LtuEHs3QM8', '2qwZumjVu8CCasFHxL31YhzrMwN3ubkM8XzupGxPZ6u1'],
    to_rewards_percent: 0.75,

    // below on-chain
    stake_min_days: 7,
    stake_max_days: 365,
    emission_total_days: 365*2,
    ciety_decimals: 9,
    start_emission: 100000,
    decay_frequency_seconds: 60 * 1,
    total_nfts: TOTAL_NFTS,
    max_stake_per_wallet: 100,
    max_stake_each_time: 5,
    debug: 1,
```

```
    nft_author: '9MJzW1oEzvjHnmdLdoGRGr1i4hu82g7eEEnxvmifcDZD',
}
```

4.2. App_frontend/src/solmate/v2_config

```
const TOTAL_NFTS = 7000;
/**
 * Pwdm data layout:
 * ====================================
 * is_initialized, 1 byte
 * ciety, 4 bytes
 * nft_count, 2 bytes
 * unstake_start, 2 bytes
 *
 * =====the following is repated========
 * nft_account, 32 bytes
 * nft_pda, 32 bytes
 * nft_n_th, 2 bytes
 * start_timestamp, 8 bytes
 * time_length_days, 2 bytees
 */
const pwdm_size = 7609;

export const config = {
    network_url: 'https://api.devnet.solana.com',
    network: 'devnet',
    stake_program_addr: 'HAoqALFTiyibSRgh63qUWX1sdxt82R3hCqSwAmk8Pysw',
```

```
    token_program_addr: '7Pfq1Lz24RWDrsJxCYaFuw3J9vVwPvcNiPeu38ssLrRo',
    host_wallet_addr: '9MJzW1oEzvjHnmdLdoGRGr1i4hu82g7eEEnxvmifcDZD',

    // during initializing, you can see an address named as ciety_mint_addr
    // you should put it here
    ciety_mint_addr: "dbTA4wXXKdyXnD1gKGbsbqXhfk3JgJ8Mef1Qpv2Hz4e",

    // if you want to replay for some reasons, you need to change the number suffix
    // and re-initialize, these must be the same as those in backend configuration,
    // they are used to derive keys for accounts from hosst or staker wallet
    seed_config: 'seed_config_25',
    seed_ciety: 'seed_ciety_25',
    seed_gsdm: 'seed_gsdm_25',
    seed_gcdm: 'seed_gcdm_25',
    seed_pwdm: 'seed_pwdm_25',
    seed_rpdm: 'seed_rpdm_25',

    pwdm_size: pwdm_size,
    total_nfts: TOTAL_NFTS,

    // Under the Solana's compute unit limit, 10 NFTs can be unstaked at one time,
    // unfortunately Solana also has another limit, transaction size limit = 1232 bytes,
    // DONT'T TOUCH THIS! Both Stake & Unstake are limited by this
    unstake_batch_size: 2,
};
```

5. Initialize

5.1. Prerequisites

Make sure the host wallet has some SOLs, at least 30.

5.2. Operation

- Run app_backend/solmate/v2_initialize.js
- On the command line terminal console, copy the value of 'ciety_mint_addr', and replace the value of 'ciety_mint_addr' in app_frontend/src/solmate/v2_config.js
- Go to the site, check out CONFIG and CIETY on Inspector page.

NOTE: in this case, the host wallet is the only payer and signer.

6. Start and Stop Reward Trigger

- App_backend/./trigger_start.sh
- App_backend/./trigger_stop.sh

7. Stake

7.1. Prerequisites

- Staker's wallet has some SOLs, 5 at least
- Staker's wallet has some valid NFTs

7.2. Operation

- Maker sure you are using a staker's wallet
- Go the site, on the Stake page, connect to the staker's wallet
- After NFTs show up, select NFTs you want to stake then click Stake button
- Switch to Inspector page
- Connect to the staker's wallet

- Check out PWDM, GCDM, GSDM
- If you wanto stake again, you have to refresh the page and repeat operations above
- You can’t stake again after unstaking
- Keep it in mind, it’s possible to faile because of invalid NFT

NOTE:

- In this use case, the staker is the payer and approver because we have to create an account (PWDM) holding staking info for each staker. But, these SOLs will be returned back to the staker’s wallet after unstaking.
- The number of NFTs that will be staked at one time is limited by configration ‘unstake_batch_size’, because Solana has a limit on transaction size, 1231 bytes, the same restrictions for unstaking.

8. Extend

8.1. Prerequisites

8.2. Operation

- Page = Extend
- Click Connect button
- Click Extend button
- Go to Inspector
- Connect wallet
- Check out PWDM, GCDM and GSDM, if not expired, you should be able to see the below as expected:
  - PWDM, longer time_length_days
  - GCDM, larger gap between start_slot and end_slot
  - GSDM, larger slot number, NOTE: it only show the slots from intialization time down to current time instead of the max slot
-

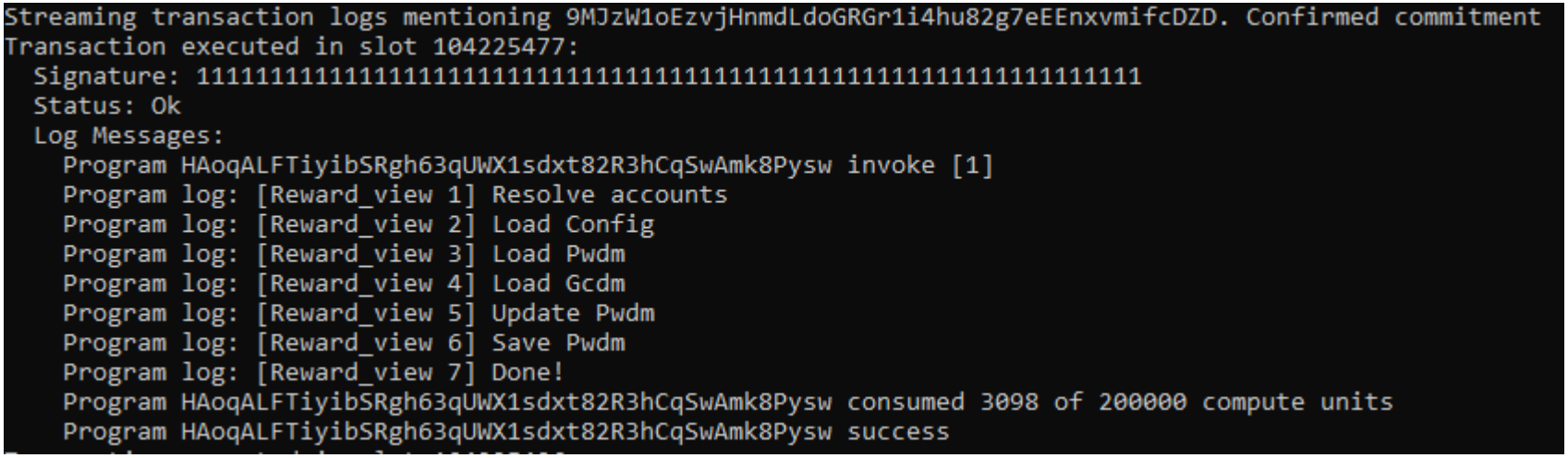
NOTE:

In this use case, the staker is the payer and signer because it has to modify PWDM.

9. Reward

The Reward Trigger calculate rewards per NFT, it doesn't update each staker's reward in their PWDMs which is updated only when a staker views his rewards. Also each Pwdm is owned by a staker, it can't be updated by reward calc trigger.

The following screenshot is a on-chain log for viewing 4 NFTs, it consumed 3098 compute units, it's quite small, roughly it will cost 15,000 compute untis if a staker has staked 2000 NFTs, for this reason we didn't add any protection in order to avoid the violation of Solana's compute unit limit in the assumption that a staker won't stake more tha 2000 NFTs in total.



10. Unstake

10.1. Prerequisites

All NFTs of a staker have to be on due

10.2. Observation

After unstaking, you should see more NFTs and SOLs in staker's wallet.

Also $CIETY shows up in the wallet

10.3. Operation

- Check out current SOLs and NFTs in the staker's wallet
- Page = Unstake
- Connect to the staker's wallet
- Click Unstake button
- Check out current SOLs and NFTs in the staker's wallet again
-

What happens:

- The contract unlock all NFTs of a staker and send them back to staker's wallet.
- Close staker's PWDM
- Return SOLs of PWDM back to the staker's wallet
- Close all the accounts which hold the staker's NFTs during staking
- Calculate rewards
- Send $CIETY to staker's wallet

Actually this process goes in batch, the batch size is 'unstake_batch_size', also it's wise to assume a chain may crash, also Pwdm.unstake_start make this system more robust, if a staker is not lucky during unstaking, say Solana crashed, that's not quite uncommon, it does't matter, because Pwdm.unstake_start tells where to start.

11. Security & Biz Rules

- It's better to deploy the frontend and the backend separately on different servers, because we have two phase siging, the host wallet always signs at the backend and the staker signs in browser.
- When initializing, the on-chain project configuration stores the host wallet key, after that, any calls to the contract will be checked as this:
  - Assume the first account passed in is the host wallet
  - It must be a signer
  - It must be the same as the one remembered by the on-chain configuration

    - The on-chain configuration must be intialized
    - CONCLUSION: NO ONE CAN ATTACK YOUR CONTRACT AS LONG AS YOUR SECRET DOESN'T LEAK OUT
- The contract check if NFTs to be staked are valid by comparing their authority with on-chain pre-configured one
- max_stake_each_time, stake_min_days, stake_max_days and max_stake_per_wallet are all checked on-chain
- Both $CIETY and NFTs are locked into the contract, they can't be transfered outside of the contract
- A staker can't unstake his NFTs if any one of his NFTs is still in its staking period