eosden
digital extended networks

EOSDEN is an aggregator of Web3 solutions built on EOS

Organization Structure

| Core Members	Liquidity Pool on Defibox	Smart Contract
Patrick Bernard Schmid ~ NovaCrypto Vladislav Hramtsov ~ CETF CAC ~ Web, Graphics, Writing, Defi	March 7th, 2023 $1'000 EOS – 1'000'000 NCO **Defibox**	Token: NCO 80M Max Supply Multisig

Affiliate Types

| Networkers/Advertisers	Web3 Solution Providers	Content Creators

EOSDEN Online Platform

Collaboration Modules	Affiliates & Users	EOSDEN Cockpit per Solution
- Document Center - WebRTC Chat + Video - MindMapping (**mindweb.io**) - Brainstorming activities - Telegram Bot - EOSDEN Cockpit	- Wallet authentication - User Sybil verification - User rate solutions of affiliates - User reviews	- Closed Source - Partly Open-source - Fully Open-source - 6 star scale rating

Action Plan

March 2023	April 2023	May 2023
Proof of Concept	**Portal**	**Promotion**
First Draft Website: **eosde.net**	- Solutions in Thumbnails - Anchor Wallet Integration - Solution comments from users	- EOSDEN Web3 episodes - EOS Bees Swarm campaigns
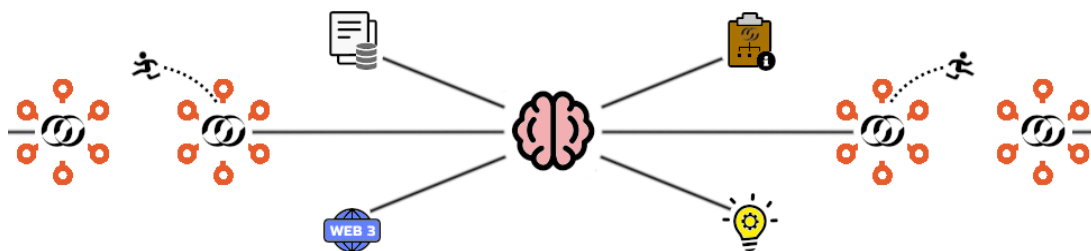
Web3, Information, Ideation, Data, and affiliates/users jumping from one Antelope network to another
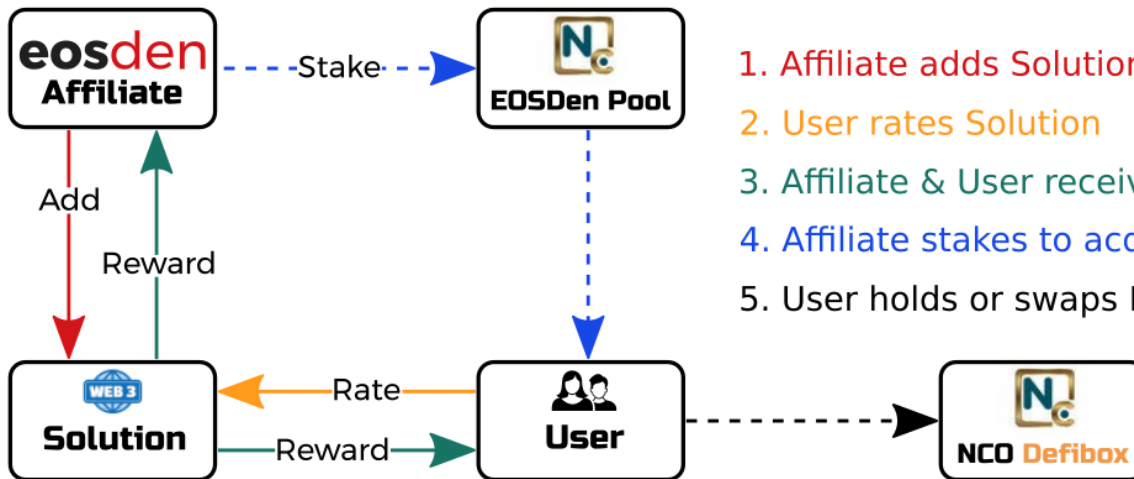
1. Affiliate adds Solution
2. User rates Solution
3. Affiliate & User receives NCO reward
4. Affiliate stakes to acquire user
5. User holds or swaps NCO token on Defibox

EOSDEN phase 1

The focus is on incentivizing affiliates to add solutions and users to give ratings.

Three main features of EOSDEN phase 1

1. **Addition of solution by an affiliate**
   An affiliate can add a solution, but must be approved to become public.
   (nova3mindweb smart contract will create an MSIG proposal with the EOS account name of the affiliate in order to be approved by the EOSDEN core members of the custom permission)

2. **Rating of affiliate/solution by a user**
   2.1 - Overall satisfaction (1-6 star scale)
   2.2 - Written review of an affiliate/solution
   2.3 - One solution per affiliate with an EOS account

3. **Automatic rewards in NCO token**
   3.1 - To the affiliate/solution for completing number 1
   3.2 - To the user for completing number 2

Features of EOSDEN phase 1

1. A user's rating/written review helps an affiliate/solution to iterate or pivot.

2. Build an EOSDEN database of affiliates/solutions (Closed Source - Partly Open-source - Fully Open-source)
   - Each affiliate/solution could have a link to a repository (Github, Notion, etc.)
   - The phase 1 is about rewarding affiliates and users with NCO token

3. The users can see the best affiliates/solutions in their category based on dApp Radar.

EOSDEN phase 2

1. An affiliate/solution will compensate users via staking ** to their related EOSDEN Pool to incentivize users to validate their solution.

2. Payments in NCO token to affiliates/solutions and users will be automatic and able to be swapped on Defibox which should create demand for the NCO token.

3. We see two reasons why an affiliate/solution would incentivize a user:
   - To validate the affiliate/solution and see if it fits the user's needs
   - To grow a user base by promoting an affiliate/solution

Note:
Q: Affiliates have the option to send either NCO or EOS to the affiliate pool.
A: That will be decided in Phase 2 by the EOSDEN core members.
    But best right now seems:
             1. for the affiliate to do an EOS transfer
             2. swap that EOS for NCO (contracts do that automatically)
             3. payments for interaction with affiliate solution to users and EOSDEN done in NCO tokens.
Q: Upon successfully submitting an approved solution, an affiliate has just received an NCO token reward.
A: Yes.
Q:They can use these NCO tokens to stake for users.
A:They can use the NCO tokens that are into their EOSDEN Affiliate Pool to acquire users.
Q: If the affiliate desires to attract more users than their existing NCO reward allows, they can contribute their own EOS to the pool as well.
A: They just do an EOS transfer to their EOSDEN Affiliate Pool (see above point 1).

EOSDEN is an aggregator of Web3 solutions built on EOS

Collaboration Workflow

..
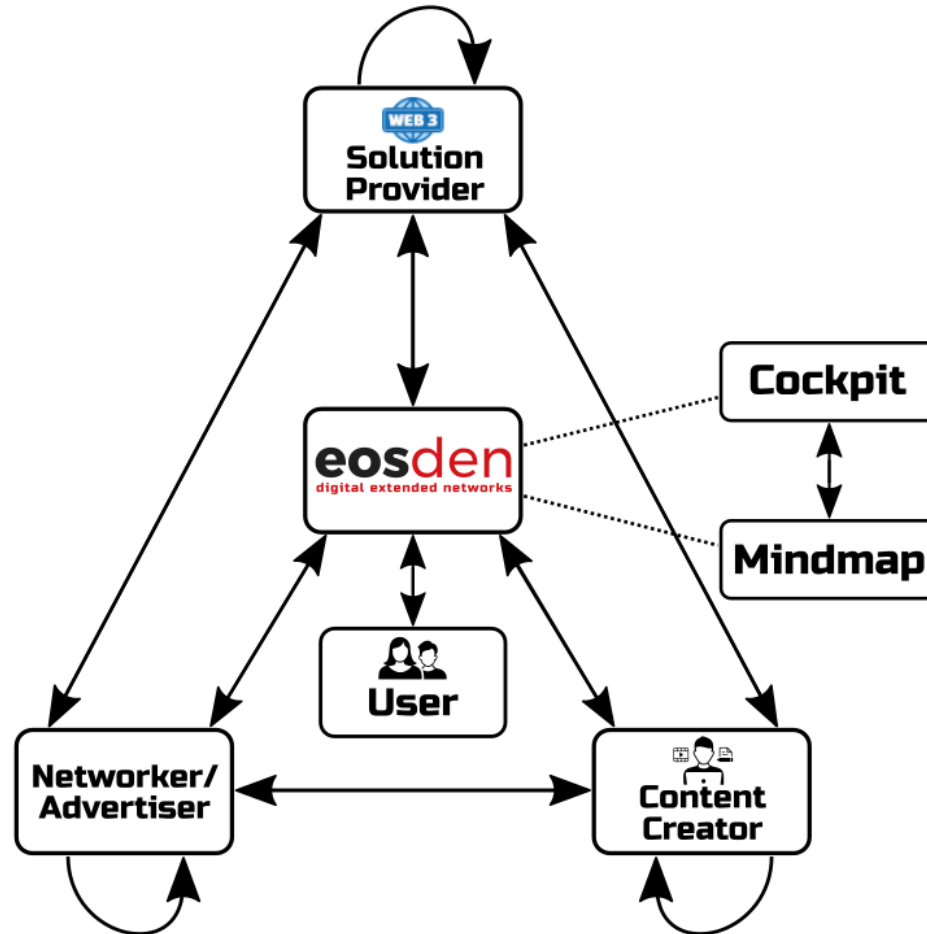
EOSDEN phase 1

1. **Any affiliate** that adds a **solution** (web3 solution provider, content creator, networker/advertiser) on EOSDEN is able to receive an **offer** from any other affiliate to collaborate with.

2. On-chain action -> **eosden::collaborate**

3. **Affiliate** that was offered collaboration on a solution can **accept** the collaboration.

4. **Profile page** of a solution shows all collaborations.

5. An affiliate may receive an e-mail notification when the on-chain action to collaborate is received.

The focus of this use case is to **connect** solutions. If an offer to collaborate is accepted, affiliates can start communicating through the EOSDEN KM (Knowledge Management) Suite or through other channels (e.g., Telegram, Discord, e-mail, etc.).

EOSDEN could serve as a collaborator's introduction.

eosden Platform Core Revenue Stream
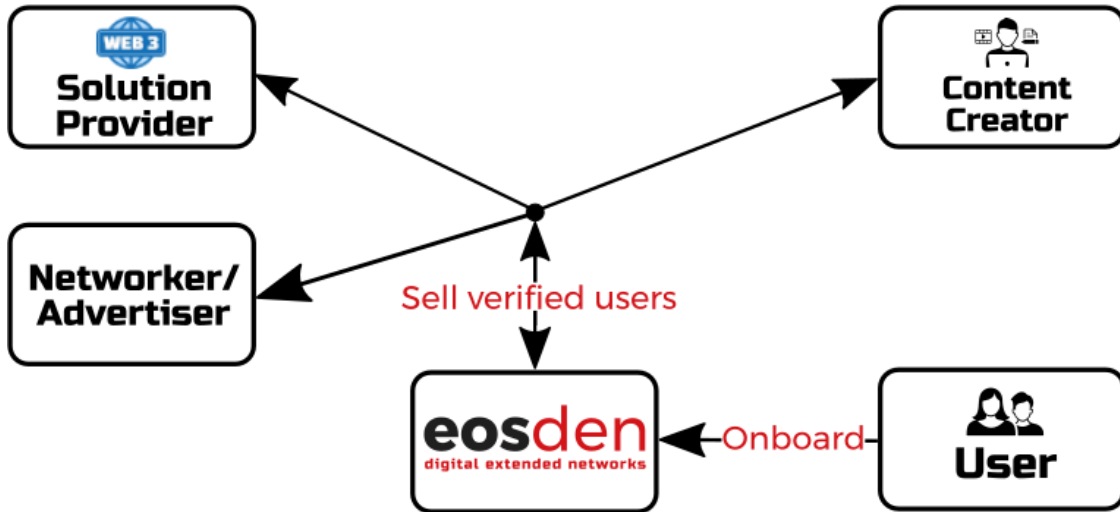
The **eosden** Platform core revenue stream
is to sell verified users to **eosden** affiliates

1. The EOSDEN Platform onboards new users by verifying them.

2. An EOSDEN affiliate/solution buys EOSDEN verified users.

3. An EOSDEN verified user interacts with an EOSDEN affiliate/solution.

4. An EOSDEN verified user is paid if he/she interacts with an affiliate/solution.

5. The EOSDEN platform receives profit.

Action steps to test the EOSDEN Revenue Model by the EOSDEN core members manually in phase 1

The following action steps can validate or invalidate the proposed EOSDEN Revenue Model.

1. EOSDEN Core Members onboards X high quality new users to EOSDEN Platform by verifying them.

2. EOSDEN Core Members search for new affiliates interested in acquiring users.

3. EOSDEN Core Members specify per affiliate/solution the smart contract actions, and how much to pay per action.
     (phase 1: manual, phase 2: will be stored in nova3mindweb per affiliate/solution/action:amount)

4. An EOSDEN affiliate/solution makes the payment to the EOSDEN Platform.
     (phase 1: manual, phase 2: will be stored in nova3mindweb per affiliate/solution/action:amount)

5. EOSDEN Core Members specify amount that goes to the users and what goes to the EOSDEN Platform.
     (phase 1: fixed amount, phase 2: variable amount)

6. EOSDEN Core Members create an EOSDEN Telegram group with X users.
     (only phase 1, phase 2: will be on the website per affiliate/solution/actions)

7. EOSDEN Core Members explain what actions the users have to do.
     (only phase 1 and in Telegram, phase 2: will be on the website per affiliate/solution/actions)

8. Once an EOSDEN verified user has performed an affiliate/solution action, they send the action's transaction ID to the Telegram group.
     (phase 1: manual, phase 2: check the on-chain history of the affiliate/solution smart contract)

9. EOSDEN Core Members manually verify that the correct action was performed by the users
     (phase 1: manual, phase 2: check the on-chain history of the web3 affiliate/solution smart contract)

10. EOSDEN affiliate/solution get the user, user gets paid, EOSDEN gets profit.

Simple example with numbers to showcase the relevance of the EOSDEN Revenue Model

1. An EOSDEN affiliate/solution wants 30 users to perform 1 action.
2. They are willing to pay 3 EOS per user.
3. They pay 3 x 30 = 90 EOS.
4. The EOSDEN platform pays user 2 EOS for the action, 2 x 30 = 60 EOS.
5. The EOSDEN platform profit = 90 - 60 = 30 EOS.

eosden Platform Profit Distribution Mechanism
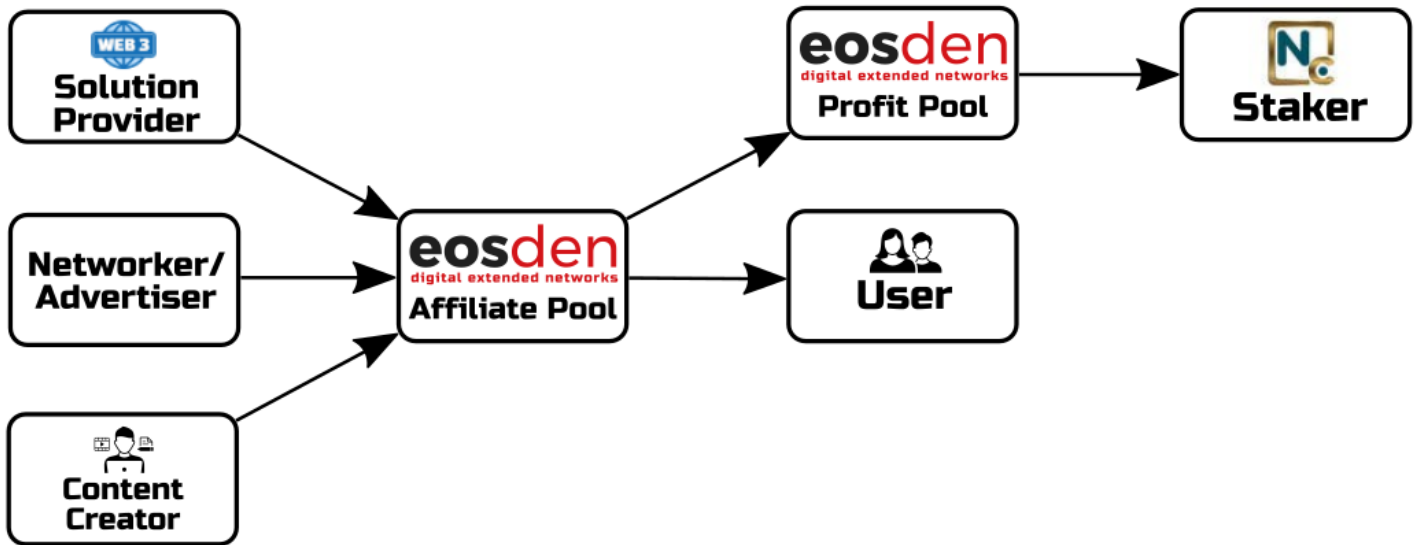
If **eosden** Core Members validate the **eosden** Core Revenue Model in Phase 1, the following NCO tokenomics model could be applied

1. The NCO token utility could establish an EOSDEN Platform profit distribution to the NCO Stakers.

2. If an EOSDEN affiliate/solution wants to acquire users, they would have to stake EOS/NCO to their EOSDEN Pool (TVL in nova3mindweb smart contract).

Example of the EOSDEN Tokenomics Model

1. An EOSDEN affiliate/solution sends 1000 EOS/NCO to their EOSDEN Pool.
2. An EOSDEN affiliate/solution specifies 3 actions they would like EOSDEN verified users to interact with.
3. For 2 actions, an EOSDEN affiliate/solution pays 2 EOS per action and if complex, 3 EOS per action.
4. Each time an EOSDEN verified user performs the specified EOSDEN affiliate/solution actions, EOS is transferred from EOSDEN affiliate/solution's Pool to the EOSDEN verified user and to NCO staker's EOSDEN **Profit Pool**.
5. NCO stakers are able to claim the profits from the NCO staker's Profit Pool.
6. For example, if EOSDEN funders hold 50% of NCO tokens and stakes them, 50% of profits would go to the EOSDEN funders.

Example with numbers to showcase the relevance of the EOSDEN Tokenomics Model

1. 100 users perform all the specified actions.
2. EOS is transferred from the EOSDEN affiliate/solution's pool.
(100 users x 2 actions x 2 EOS/action) + (100 users x 1 Action x 3 EOS/action) = 700 EOS.
3. EOSDEN is configured that 20% of the profit goes to the users and 80% of the profit to EOSDEN.
4. 0.2 x 700 = 140 EOS to users / 0.8 x 700 = 560 EOS to EOSDEN as profit.
5. If the EOSDEN funders holds and stakes 50% of all the NCO tokens, 560 x 0.5 = 280 EOS can be claimed by the EOSDEN funders from the EOSDEN Profit Pool.

All tables and actions will be deployed to nova3mindweb account.
The e-mail for solution approval: **solutionapproval@eosde.net**
For privacy, EOSDEN encrypts the e-mail of the affiliates and users before storing them in the tables of the nova3mindweb smart contract.

1. **eosio.msig::propose** (look into eosjs msig for input parameters)

   Intent of the action:

   Creates a msig for EOSDEN core members to approve and execute a solution. Once the **propose** action is triggered in the EOSDEN UI, a notification is sent to the EOSDEN core members (**solutionapproval@eosde.net).** EOSDEN UI lists all the solutions pending approval. The EOSDEN UI enables the approval and execution of a solution and sends an e-mail to the affiliate once the msig is executed.

2. **eosden::addsolution** (name affiliate, uint_8_t affiliate_type, string description, string solution_name, std::map socials, uint8_t code_type, vector <name> contracts, vector <string> links_to_code, string affiliates_role, string affiliate_email, string category)

   Intent of the action:

   The affiliate adds their solution to EOSDEN. After the affiliate's solution gets approved, the affiliate will be able to send an offer to collaborate with another affiliate. Similarly, other affiliates can send offers to collaborate with the affiliate. By offering to collaborate, the affiliate is signaling their interest in working together. If the collaboration is accepted, the affiliates are rewarded in NCO token.

   When an affiliate's solution is added, only verified users are able to rate the affiliate's solution based on their satisfaction level and leave a review for the affiliate's solution.

   eosden::addsolution is an input parameter of the **eosio.msig::propose** action. Once EOSDEN core members approve and execute the msig, this **eosden::addsolution** action is executed and the affiliate's solution is added to EOSDEN. The affiliate is notified by e-mail.

   nova3mindweb can execute this action on behalf of an affiliate.

3. **eosden::modsolution** (name affiliate, uint_8_t affiliate_type, string description, uint_8_t affiliate_type string solution_name, std::map socials, uint8_t is_code_public, vector <name> contracts, vector <string> links_to_code, string affiliates_role, string affiliate_email, string category)

   Intent of the action:

   The affiliate is able to modify information regarding their solution. Modification can only be done by the affiliate that added the solution.

4. **eosden::delsolution (name affiliate)**

   Intent of the action:

   This action enables the deletion of an affiliate/solution from EOSDEN. Can be deleted by an affiliate that added the solution or nova3mindweb (if by nova3mindwe, the affiliate is notified by e-mail.)

1. eosden::rate (name user, name affiliate, uint8_t rating)

Intent of the action:

This action enables a user to rate an affiliate/solution by giving an overall satisfaction rating to the solution (min=1, max=6). By rating the solution, a user is rewarded in NCO token. In order to rate, a user has to be verified. See the action **eosden::verify**

Points are added to an affiliate/solution when a user gives a rating.

Formula: Total points of an affiliate/solution / count of users rating = Avg. Rating

2. eosden::review (name user, name affiliate, string comment)

Intent of the action:

This action enables a user to leave a review of an affiliate/solution and be rewarded in NCO token. Only a verified user can leave a review.

Reviews should have ratings. EOSDEN needs a way to prioritize reviews for display purposes and favors quality reviews.

Reviews should be time-weighted. Products/services change. They get better, or they get worse. In our space that can happen rather rapidly.

A 1 star review from a year ago should not matter as much as a 6 star review from last week. So, the review weights should degrade over time.

3. eosden::delreview (name user, name affiliate)

Intent of the action:

This action enables users to delete a review of an affiliate's solution or nova3mindweb (user is notified by e-mail.)

4. eosden::modreview (name user, name affiliate)

Intent of the action:

This action enables users to modify a review of an affiliate's solution.

Use of Defibox smart contract table to determine the NCO/USD exchange rate.

1. **eosden::reward (name user, string reward_for)**

Intent of the action:

> This action is placed at the end of **eosden::review**,**eosden::rate**,**eosden::addsolution**,**eosden::collaborate**
>
> This action calculates the amount of tokens issued by each of the four actions above.

2. **eosden::setreward** (uint16_t rate_reward, uint16_t review_reward, uint16_t collab_reward, uint16_t solution_reward)

Intent of the action:

> This action sets the rewards in USD. There is a table that defines USD value for each of the four actions.
>
> This action is triggered when the EOSDEN core members have approved and executed.

EOSDEN is an aggregator of Web3 solutions built on EOS

1. **eosden::collaborate (name affiliate, name collaborate_with)**

Intent of the action:

> This action enables an affiliate to offer collaboration to another affiliate.
>
> The affiliate is signaling their willingness to collaborate with another EOSDEN affiliate. If a collaboration offer is accepted, the two affiliates are rewarded in NCO token.

2. **eosden::endcollab (name affiliate, name collaborate_with)**

Intent of the action:

> This action enables both affiliates to end the collaboration.
>
> An affiliate is signaling to the other that they are no longer interested in collaborating.
>
> Once the action is executed, the "collaborate_with" solution will not appear in the affiliate's collaborations list.

1. **eosden::verify (name user, string email, std::map socials)**

Intent of the action:

> This action enables a user to be verified.
>
>   1. Automatic verification by using external contracts to check the sybil protection
>   2. If 1 is not the EOSDEN UI create an MSIG and the EOSDEN core members have to check the socials of the user by contacting him to his e-mail.

```
/*
Scope : _self
Stores all the users that have completed verification.
*/

TABLE verified_user
{
    name user;
    map<string, string> socials; //email:toto@email.com, twitter:novacryptoltd

    uint64_t primary_key() const { return user.value; }
};

typedef eosio::multi_index<"verifieduser"_n, verified_user> verif_t;
```

```
/*
Scope : _self
This table save all the information in regards to the pending and approved
collaborations.
*/

TABLE collab
{
    name collaborate_with;
    bool accepted;
    bool accepted_by_both;

    uint64_t primary_key() const { return collaborate_with.value }
};

typedef eosio::multi_index<"collab"_n, collab> collab_t;
```

```
/*
Scope : _self
This table stores all the information in regards to the solution. Table gets filled
when addsolution action is triggered.
*/
```

TABLE solution

```
{
    name affiliate;
    string description;
    string solution_name;
    uint_8_t affiliate_type;
    map<string, string> socials; //e.g. Twitter : novacryptoltd, Telegram : novacryptoltd
    uint8_t is_code_public;
    vector<name> contracts; //in case on-chain solution
    vector<string> links_to_code;
    string affiliates_role;
    string category; //check the values from one table, the categories are based on dApp Radar
    uint32_t total_ratings;
    uint32_t nr_of_voters;

    uint64_t primary_key() const { return affiliate.value; }
};

typedef eosio::multi_index<"solutions"_n, solution> solution_t;
```

```
/*
This table has no scope because it is singleton type.
Table stores reward amounts in USD values.
*/
```

TABLE reward

```
{

  uint16_t rate_reward;
  uint16_t review_reward;
  uint16_t collab_reward;
  uint16_t solution_reward;
};

typedef eosio::singleton<"rewards"_n, reward> reward_t;
```

```
/*
Scope: affiliate.value
This table stores the information submitted by the user.
That info is a comment and a rating.
*/

TABLE user
{
    name user;
    string comment;
    uint8_t rating;

    uint64_t primary_key() const { return user.value; }
};

typedef eosio::multi_index<"users"_n, user> user_t
```

```
    /*
Atomicassets::assets_s table
Scope: owner
Table can be iterated through in order to check whether users holds SBT and Pomelo
NFT.
Note that only top 100 contributors hold Pomelo NFTs.
*/


TABLE assets_s
{
        uint64_t        asset_id;
        name            collection_name;
        name             schema_name;
        int32_t         template_id;
        name            ram_payer;
        vector <asset>   backed_token;
        vector <uint8_t> immutable_serialized_data;
        vector <uint8_t> mutable_serialized_data;

        uint64_t primary_key() const { return asset_id; };
};
typedef multi_index <name("assets"), assets_s> assets_t;
```

```
/*
These tables are from Defibox,
and will be used to determine the NCO/USD exchange rate.
*/
struct token
{
    name contract;
    symbol symbol;
};

TABLE pair
{

  uint64_t id;
  token token0;
  token token1;
  asset reserve0;
  asset reserve1;

  uint64_t liquidity_token;
  double price0_last;
  double price1_last;
  uint64_t price0_cumulative_last;
  uint64_t price1_cumulative_last;
  time_point_sec block_time_last;

auto primary_key() const { return id; }
};

typedef eosio::multi_index<"pairs"_n, pair> pairs;
```