Migration to terra 2: A perfect guide

No comments

This article aims to bring the necessary guidelines for migrating from origin or classic Terra to new Terra or Terra2. This migrating includes the updates in main-net, test-net, exchanges, functions, dApps, starting a node, faucets and everything that help the developers update their scripts with the new Terra chain. It is important to mention that the classic or origin Terra still works the way it used to. But in Terra2 we have some modifications that solve the problem related to the classic Terra. Of course this guide is not meant to create a full tutorial on how to interact with Terra2 main-net or test-net and other interactions, but to continue the detailed guidelines that we provided for the original Terra (Terra1), with the difference that they are going to be used for the new Terra chain.

What is terra?

Terra is a DeFi protocol on the Blockchain that powers the universal payment systems which is based on fiat currencies using the fiat-based stablecoins. This protocol uses a mixture of wide adoption of stable coins and at the same time censorship resistance of Bitcoin. Terra is built on Cosmos SDK and Tendermint. The main language that the Terra smart contracts are written in, is Rust. However, there are SDKs for python and JavaScript and creating the smart contracts called WASM Contracts.

What does terra2 look like?

Most of what we know about the original Terra Or Terra Classic is true about Terra2 as well. Although the differences that Terra2 has with Terra Classic, is the main focus of this article and you need to read it fully until the end to understand the differences, we can summarize them here in the beginning. The new Terra chain is also a Cosmos chain with the difference that it doesn't have oracle, stable coins (such as UST, KRT, EUT, etc), treasury and the market modules as opposed to the original chain. Moreover, Luna will be the main staking asset of the chain (notice that this Luna is the related to the new Terra chain and the token of the Terra Classic will be named LUNC). And last but not the least, the official name that is considered for the new Terra chain is Terra without any post or pre phrases. These are the main features of Terra2 mentioned by Terra official documentation website.

Changes made on the original Terra:

The only changes that have been made on the Terra Classic is the names of tokens and the new conventions set for calling them. Consequently, we have no changes in the back-end and codes behind the scenes except that the market swaps have been disabled as well as the minting and burning functions. The new naming changes include: 1. Luna has become Luna Classic (LUNC). 2. Terra stable coins have been renamed from (UST, KRT, EUT, etc) to (USTC, KRTC, EUTC, etc) Arashtad.com Design and development solutions arashtad

Terra 2 main-net and test-net:

The new Terra chain has a new main-net and test-net name. If you have read our articles about the Terra Classic which was called Terra at the time, you can remember that the main-net used for it was called Columbus-5 and the test-net named Bombay-12. With the new Terra chain, we have a new main-net called Phoenix-1 and the test-net with the name of Pisco-1.. It is important to notice that for Terra2 main-net Phoenix -1 the LCD changes from (https://lcd.terra.dev) which was used to for Columbus-5 to (https://phoenix-lcd.terra.dev). Also the FCD changes from (https://fcd.terra.dev) to (https://phoenix-fcd.terra.dev). Now, let's see if the new settings works for our recent Terra Classic scripts. Before you run the codes, you should reinstall the Terra Python SDK, as there are new updates made on the SDK. To do so enter the following command in the terminal to perform the installation: `pip install -U terra_sdk` And then create a file called test.py and paste the below scripts in it:

```
from terra_sdk.key.mnemonic import MnemonicKey
from terra_sdk.client.lcd import LCDClient
terra = LCDClient(
     url="https://pisco-lcd.terra.dev/",
     chain_id="pisco-1"
)
mk = MnemonicKey().__dict__
wallet = terra.wallet(mk)
print(mk)
```

To run the above script, paste the following command in the terminal:

```
Python test.py
```

Result:

```
{'public_key': SimplePublicKey(key=b'\x02s\xf0\xca\x8c$A\x86;\xbc\xbeE\xff8\xf3\x96\x98\xdb/R\x1f\x89\x11\r \xf7\xc5O|\xe3#\x1e\x98'), 'raw_address': b'\xb3^^\xde\xfce\xe1\xa5\x97&\x1f\x88\x85\xa4\x95}\xc7\x1aP\x8d', 'raw_pubkey': b'\xebZ\xe9\x87!\x02s\xf0\xca\x8c$A\x86;\xbc\xbeE\xff8\xf3\x96\x98\xdb/R\x1f\x89\x11\r \xf7\xc5O|\xe3#\x1e\x98', 'private_key': b'\x89\tq\xe5\xee\x14k\x82\xd0\xe5\xd4\x85\x9aW\x1e\x98\xf5E\xd2\xa4]\xed\x99\x9f\xd9O\x8b\xe9n? \x98', 'mnemonic': 'tube have blanket praise panther popular owner brand brick arena combine follow refuse frog normal seed alarm awkward spend vicious inject solution soul artwork', 'coin_type': 330, 'account': 0, 'index': 0}
```

Using the mnemonic key given above, we can get some test ULUNAs from Terra Faucet (The way we did on our Python Terra SDK article) and also create another account. Finally, we can create and sign the transaction of transferring money from account 1 to account 2.

Estimating Gas fees on Terra2:

On all the blockchains, the transactions need efforts from computational resources. In the proof of work architectures, the computational resources are the miners and in other mechanisms, there are other resources. The computational processing applied for transactions is quantified in the units called gas. To estimate the gas fee for a transaction, we use the very same script that we used when

```
import requests
import json
gas_price_dict = requests.get("prices").json()
print(gas_price_dict["uluna"])
```

Result:

```
5.665
```

Migrating CosmWasm contracts on Terra:

The CosmWasm smart contracts in Local Terra remain the same but with a little bit of difference. If you can remember from the article of Local Terra, we had some .rs files including msg.rs and contract.rs. We also had 3 kinds of messages in the Local Terra: 1. Instantiate 2. Execute 3. Query Now, in the new Terra chain, we have also got the Migrate message added. We will add this migrate method to the msg.rs like below:

```
#[derive(Serialize, Deserialize, Clone, Debug, PartialEq, JsonSchema)]
pub struct MigrateMsg {}
```

Also, in the contract.rs file we have 2 changes for adding the migration:

```
#[cfg_attr(not(feature = "library"), entry_point)]
pub fn migrate
(_deps: DepsMut, _env: Env, _msg: MigrateMsg) -> StdResult {
     Ok(Response::default())
}
```

And

```
use
 crate::msg::{CountResponse, ExecuteMsg, InstantiateMsg, QueryMsg, MigrateMsg
```

Deprecated functionalities on terra2:

On the new Terra chain, we will see that some of the functions are deprecated. Meaning that they cannot be used anymore. Some of these functions are as follows: 1. oracle functions:

```
const oracleParams = await lcd.oracle.parameters();
const marketParams = await lcd.market.parameters();
```

2. swap function used in Python Terra SDK:

```
const swap = new MsgSwap('terra...9fj',new Coin('uluna','1000000'),
'uusd')
```

We used the MsgSwap function to swap tokens on Terra Classic Network. With the expiration of the stablecoins on Terra2, the MsgSwap function is deprecated as well 3.compute tax function:

```
pub fn compute_tax(deps: Deps, coin: &Coin) -> StdResult {
     let terra_querier = TerraQuerier::new(&deps.querier);
     let tax_rate = Decimal256::from((terra_querier.query_tax_rate()
?).rate);
     let tax_cap = Uint256::from((terra_querier.query_tax_cap(coin.
denom.to_string())?).cap);
     let amount = Uint256::from(coin.amount);
     Ok(std::cmp::min(amount * Decimal256::one() - amount /
(Decimal256::one() + tax_rate),tax_cap,))
}
```

Conclusion

In this article, we have got familiar with Terra Network, what it is doing, how it does what it does, and the distinctions of this chain. Furthermore, we have got to know the new features of Terra2 with the new updates that have been made on this new fork of the original Terra or Terra Classic. In addition to that, we have taken a look at the Terra network's new names from the Terra Classic tokens to the Terra2. The other thing that we have done is to run the Terra2 Pisco-1 test-net to see if works properly. Moreover, we have seen what scripts have been added to the CosmWasm smart contracts and how to update them. And last but not the least, we have taken a look at the deprecated functions that can no more be used in the new Terra chain such as the swap function, oracles, and taxes.

Join Arashtad Community

Follow Arashtad on Social Media

We provide variety of content, products, services, tools, tutorials, etc. Each social profile according to its features and purpose can cover only one or few parts of our updates. We can not upload our videos on SoundCloud or provide our eBooks on Youtube. So, for not missing any high quality original content that we provide on various social networks, make sure you follow us on as many social networks as you're active in. You can find out Arashtad's profiles on different social media services.

Get Even Closer!

Did you know that only one universal Arashtad account makes you able to log into all Arashtad network at once? Creating an Arashtad account is free. Why not to try it? Also, we have regular updates on our newsletter and feed entries. Use all these benefitial free features to get more involved with the community and enjoy the many products, services, tools, tutorials, etc. that we provide frequently.

SIGN UP NEWSLETTER RSS FEED