NodeJS Server

Implementation language : JavaScript

Framework : NodeJS

Major Files : service.js, dataTable.json, mine.json

The data server is the server which maintains the list of nodes in the network. This server deals with the adding and returning records of information on various nodes in the network. The service.js is the server code. dataTable.json contains the list of all the nodes in the network and mine.json contains the details of the tokens that are yet to be mined.

Service.js

This code contains the various routes/methods. Some of it's core functions are:

- Add a new node to the network
- Maintain a list of records of the nodes in the network
- Return this list of all nodes in the network to other nodes, to keep them in sync
- Maintain the list of tokens to be mined

dataTable.json

This file contains the various records of nodes in the network which in the following format:

```
[{"peerid":"Qmb9ghACUGzLYueVea3ihHaKtU6doJrj8swffkP9CZkGbT","didHash":"QmW8e8XMd9gm
kGgxgsgPWEhkKLuE4o4TFhvKXyLAELBRYu","walletHash":"QmPHo6Aopr4nhVEaSBBYE9kdcPCrBaU84
XJkzvyCQAu7pG"},
{"peerid":"QmQEftV3oU7sWpxitiTHK5J8BSSVV7KNP9xbixqDqLu9qj","didHash":"QmfWAwbWmR51k
8gV1AY12qMV3X9h6SthmESCQm9VwPFVLx","walletHash":"Qmds8PiBQ6zj3tSmRSoGSaZMsDbKCLCakH
i9K5UCZGTJ53"}………………]
```

peerID: IPFS peerID of a node

didHash: Rubix DID of a node

walletHash: WalletID of a node

This data is crucial for authentication of blockchain transactions. The walletHash and didHash of a node are crucial information needed for authenticating blockchain transactions and signatures of various nodes. Hence this information in dataTable.json is synced with all the nodes in the network.

mine.json

The file contains various an array of the tokens that are yet to be mined. The records are in the following format:

[{"level":1,"token":1},{"level":1,"token":2}………]

NOTE: This folder contains Navy project specific files and functions that were required for that project and hence can be removed later on

mine.json can be generated using the java code/jar provided. The jar is prebuilt and can be accessed here: MineJsonCreate/target/MineJsonCreate-1.0-SNAPSHOT.jar

How to use?

Java -jar original-MineJsonCreate-1.0-SNAPSHOT.jar <level> <startingtoken> <endingtoken>

Once run, you will get mine.json file which needs to be added to the nodeJS server.

NOTE: Refer to Rubix's excel sheet for check the number of tokens allowed in a particular token level.