Gaia-X Hackathon #4
Gaia-X DLT Network: Including DLT components in the Gaia-X Framework of Self-Descriptions and Federation Services

–

Albert Peci, Frederic Schwill, Alexander Eger, Kai Meinke, Moritz Kirstein – deltaDAO AG

Monday, 20 June 2022

01

Gaia-X Testnet Overview

What is the Gaia-X Testnet?

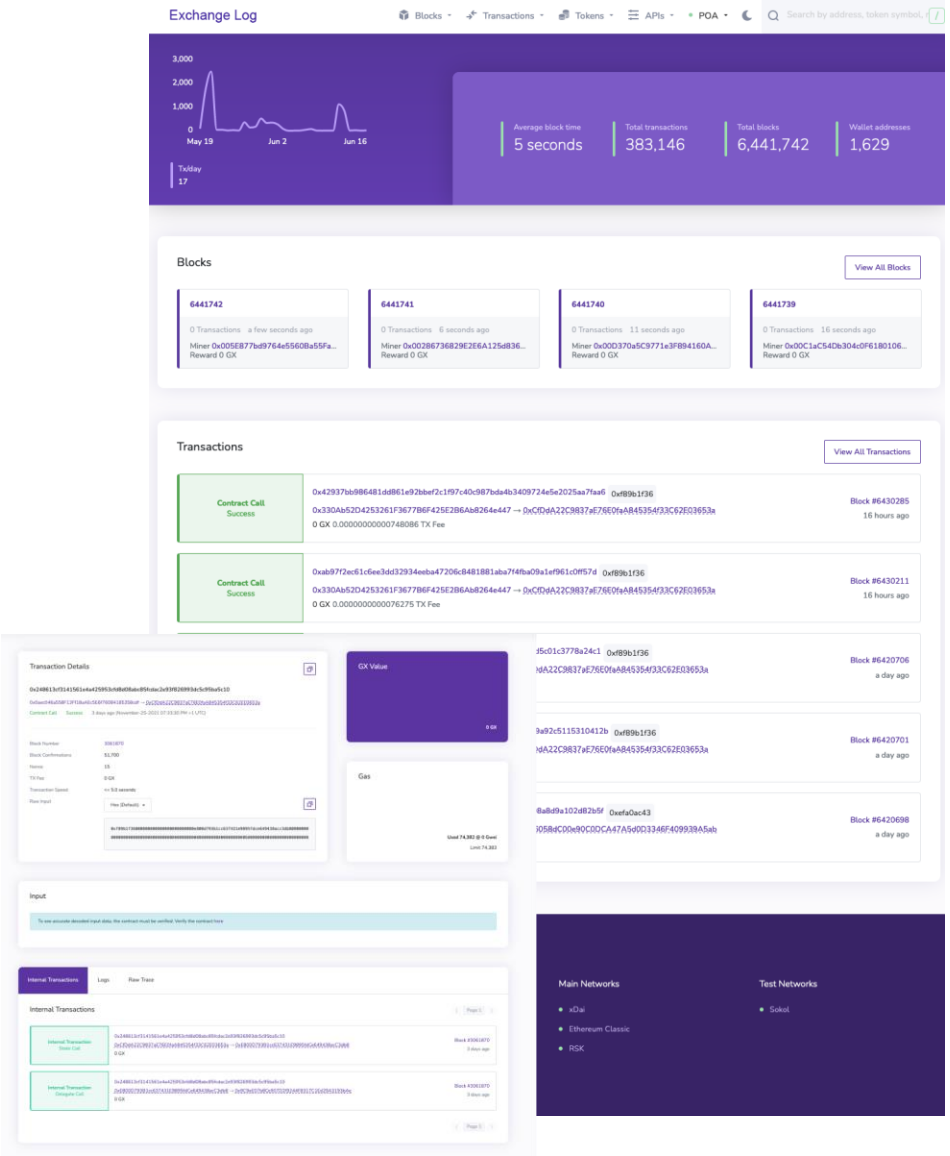
- Gaia-X Testnet is an **Open Ethereum** based **Proof of Authority** network
- **Aura** *(Authority Round)* is used as consensus algorithm
- It is **Ethereum Virtual Machine (EVM)** compatible
- It is a public-permissioned distributed ledger. All transactions are transparent and can be viewed at https://exchangelog.minimal-gaia-x.eu/, but only selected nodes are responsible for consensus.
- Due to its open nature, everyone can run their own archive node and analyze the network. See our stats example at https://stats.minimal-gaia-x.eu/
- Everyone can submit transactions. Fees are paid with a custom Gaia-X Test Token. They can be retrieved via a Faucet: https://faucet.gx.gaiaxtestnet.oceanprotocol.com/

What does Proof of Authority (PoA) mean?

gaia-x

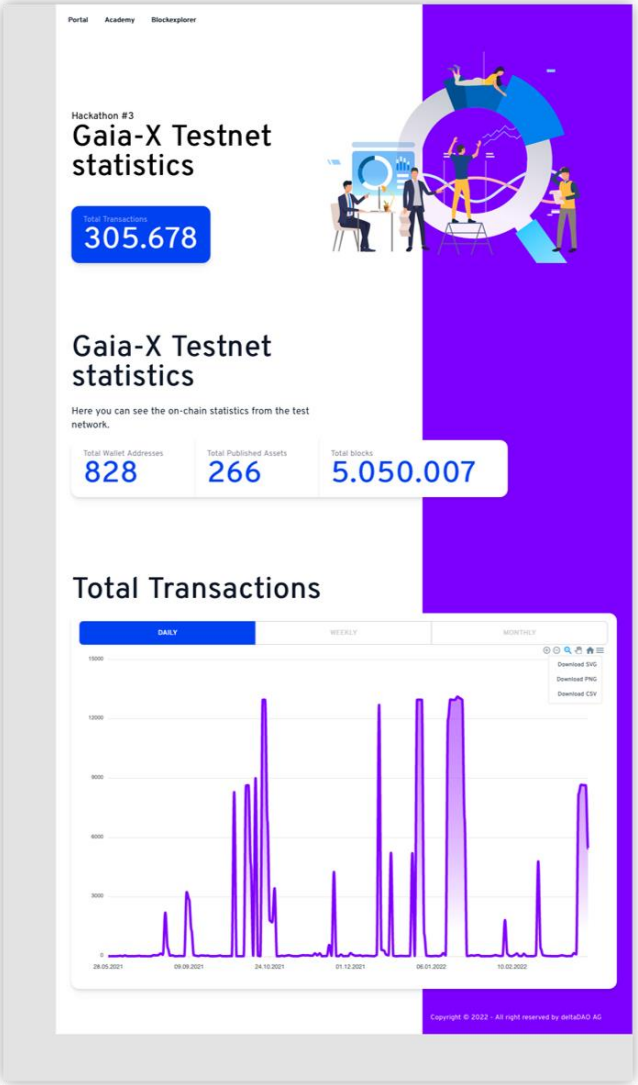
- PoA is a replacement for Proof-of-Work (PoW), which can be used for private chain setups. It is a crucial element to reach consensus in a distributed ledger.
- It does not depend on nodes solving arbitrarily difficult mathematical problems (like Proof-of-Work), but instead uses a set of “authorities”.
- For consortium setting there are no disadvantages of PoA network as compared to PoW: less computationally intensive, more performant due to lower transaction acceptance latency and more predictable (blocks are issued at steady time intervals)
- **Aura *(Authority Round)*** is one of the Blockchain consensus algorithms available in OpenEthereum.
- On each step, each honest node will propagate the chain with the highest score it knows about to all other nodes.

Gaia-X Exchange Logging Service Explorer



- Gaia-X test network Explorer offers an interface to investigate and analyze the Gaia-X test network transactions
- Explorers usually use archive nodes to sync with the network
- Usually, a custom database is used to optimize for queries like address, token symbols, names, transactions hashes and block numbers
- The current explorer distinguishes between Contract Calls, Token Transfers, and failed transactions.
- **Explorer**: https://exchangelog.minimal-gaia-x.eu/

Gaia-X Testnet Stats



Gaia-X Testnet Stats uses a custom script to analyze the distributed ledger for transactions and published assets.

Total Transactions
382.808

Total Wallet Addresses
1.418

Total Published Assets
637

Total blocks
6.441.627

Repositories

https://stats.minimal-gaia-x.eu/
https://github.com/deltaDAO/gaia-x-testnet-statistics-api
https://github.com/deltaDAO/gaia-x-testnet-statistics
https://github.com/deltaDAO/gaia-x-snapshot

gaia-x

02

How to Setup an Archive Node and Describe it?

Gaia-X Testnet Client

- **OpenEthereum** is an implementation of the Ethereum protocol written in Rust, a systems programming language. It is developed and actively maintained by OpenEthereum DAO.
- OpenEthererum is used as a **client** to run **EVM compatible distributed ledgers**.
- Gaia-X Testnet OpenEtheruem **client requirements**:
  - multi-core CPU, 8GB RAM and an SSD drive and at least 200GB free space and a decent DSL connection is required
- A list of **boot nodes**, a **config.toml** to connect to the network and a **chain.spec** to define the properties of the chain that the client should connect to: https://github.com/oceanByte/gaia-x-node-setup

Gaia-X Testnet Client Setup

Create a new folder
> mkdir gaia-x-node && cd gaia-x-node

Clone config files
> git clone https://github.com/oceanByte/gaia-x-node-setup

Download the openethereum client https://github.com/openethereum/openethereum/releases
> wget https://github.com/openethereum/openethereum/releases/download/v3.2.6/openethereum-linux-v3.2.6.zip
unzip the client
> unzip openethereum-linux-v3.2.6.zip

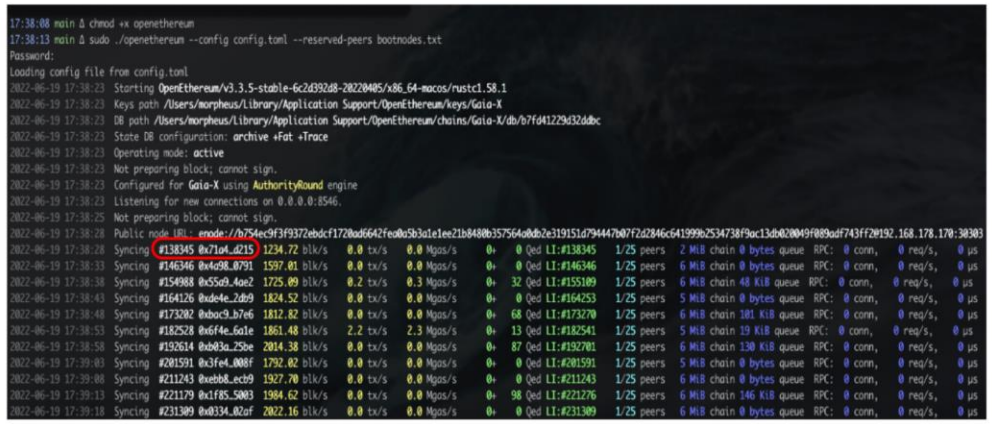
Run your client with the config file
> chmod +x openethereum
> sudo ./openethereum --config config.toml --reserved-peers bootnodes.txt

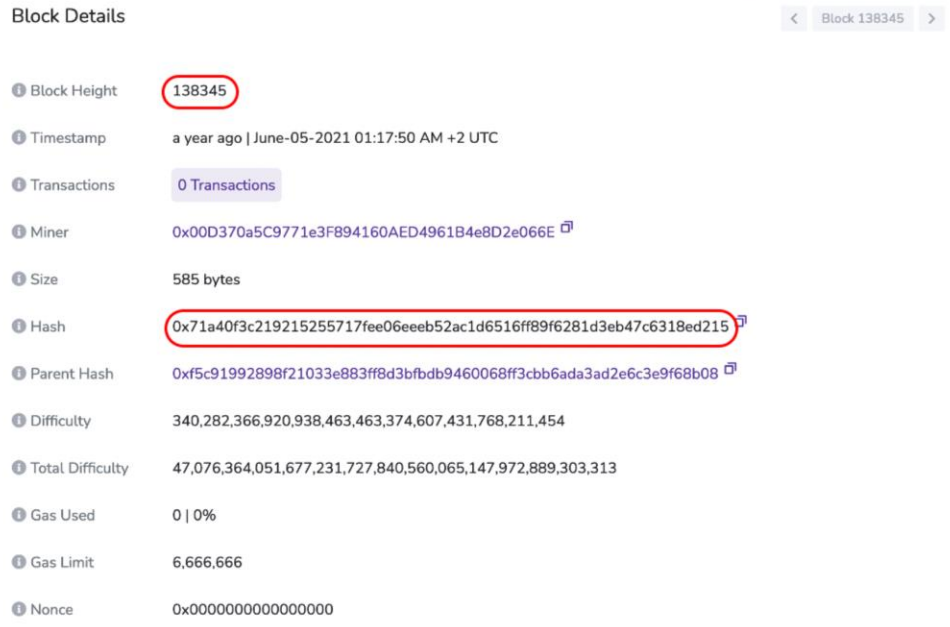
Gaia-X Testnet Config

```
[parity]
chain = "./chain.spec"

[network]
warp = false

[rpc]
apis = ["web3","eth","net","parity","traces"]
processing_threads = 8
server_threads = 16
interface = "0.0.0.0"
cors=["all"]

[websockets]
port = 8546
interface = "0.0.0.0"
max_connections = 1000
apis = ["web3","eth","net","parity","pubsub","traces"]
origins = ["all"]
hosts = ["all"]

[footprint]
tracing = "on"
pruning = "archive"
fat_db = "on"
cache_size_db = 12000
```

- The OpenEthereum client can be configured via a config.toml
- You can define exactly who and how someone can query data from your node
- Nodes can be used in very different scenarios ranging from archive nodes for data analyses and applications like Explorers to light nodes used to transmit transactions into the network to validator nodes responsible for consensus
- Find a full list of config settings here: https://openethereum.github.io/Configuring-OpenEthereum

Gaia-X Testnet Sync





- Once started the Client will connect to the provided list of peers and start downloading all blocks
- It will store all transaction data in a local database that can be queried via the API that is exposed by the client
- Sync speed depends on the number of connected peers and internet speed
- The client can be paused and resumed later
- You can verify that you are downloading the correct blocks by comparing the hash of a specific block number via an Explorer like https://exchangelog.minimal-gaia-x.eu/ or another node

Gaia-X Testnet JSON-RPC Methods

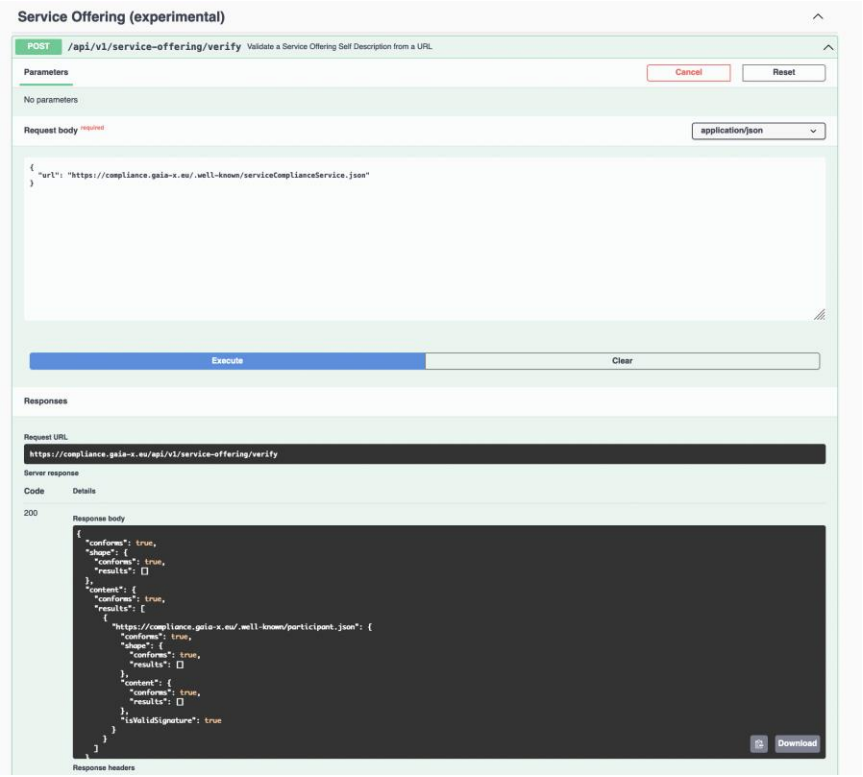
- OpenEthereum exposes a JSON-RPC interface that you can use to query the chain
- JSON-RPC is a stateless, light-weight remote procedure call (RPC) protocol
- OpenEthereum supports HTTP and WebSockets
- When requests are made that act on the state of Gaia-X Testnet, the last parameter determines the height of the block
- There are several datatypes that are passed over JSON. When encoding QUANTITIES (integers, numbers): encode as hex, prefix with “0x

```
18:33:34 main Δ curl --data '{"method":"eth_blockNumber","params":[],"id":1,"jsonrpc":"2.0"}'
-H "Content-Type: application/json" -X POST localhost:8545
{"jsonrpc":"2.0","result":"0x1ab321","id":1}
```

Example call to query latest block using HTTP

curl --data '{"method":"eth_blockNumber","params":[],"id":1,"jsonrpc":"2.0"}' -H "Content-Type: application/json" -X POST localhost:8545

How to Create and Validate Self Descriptions?




```
07:42:01 main Δ node index.js
Loaded ./config/self-description.json
Hashed canonized SD 43f98805a5eafab77467572e66dc1caad8afd285409e768e0513b52143b96a24
SD signed successfully (local)
Verification successful (local)
./output/1655703738115_self-signed_ServiceOfferingExperimental.json saved
./output/1655703738115_did.json saved

Checking Self Description with the Compliance Service...
SD signed successfully (compliance service)
Verification successful (compliance service)
./output/1655703738115_complete_ServiceOfferingExperimental.json saved
```

- We are using the SD signer to create our Self Descriptions: https://github.com/deltaDAO/self-description-signer
- First, create your Self Description based on examples https://gitlab.com/gaia-x/lab/compliance/gx-compliance/-/tree/main/src/tests/fixtures or using the Trust Framework as reference at https://gaia-x.gitlab.io/policy-rules-committee/trust-framework/
- Next, adjust your Self Description based on the validation results from the Compliance Service
- Once done you can upload to a place of your choice and share your Self Description and others can verify it:
  - Using the Compliance Swagger UI: https://compliance.gaia-x.eu/docs/#/Participant/ParticipantController_verifyParticipant
  - Using a command line tool (or your favorite tool) like curl: *curl 'https://compliance.gaia-x.eu/api/v1/service-offering/verify' -X POST -H 'Content-Type: application/json' --data-raw $'{ "url": "https://www.delta-dao.com/.well-known/serviceDLTvalidatorDeltaDAO.json" }'*

Using Self Descriptions to describe the Gaia-X Testnet


```
3         "@context": [
4             "http://www.w3.org/ns/shacl#",
5             "http://www.w3.org/2001/XMLSchema#",
6             "http://w3id.org/gaia-x/resource#",
7             "http://w3id.org/gaia-x/participant#",
8             "http://w3id.org/gaia-x/service-offering#"
9         ],
10        "@type": [
11            "VerifiableCredential",
12            "ServiceOfferingExperimental"
13        ],
14        "@id": "https://delta-dao.com/.well-known/serviceDLTvalidatorDeltaDAO.json",
15        "credentialSubject": {
16            "id": "https://delta-dao.com/.well-known/serviceDLTvalidatorDeltaDAO.json",
17            "gx-service-offering:providedBy": {
18                "@value": "https://delta-dao.com/.well-known/participant.json",
19                "@type": "xsd:string"
20            },
21            "gx-service-offering:name": {
22                "@value": "Gaia-X Test Network Validator Node",
23                "@type": "xsd:string"
24            },
25            "gx-service-offering:description": [
26                {
27                    "@value": "Gaia-X Test Network Validator Node",
28                    "@type": "xsd:string"
29                }
30            ],
31            "gx-service-offering:chainID": {
32                "@value": "2021000",
33                "@type": "xsd:string"
34            },
35            "gx-service-offering:chainSymbol": {
36                "@value": "GX",
37                "@type": "xsd:string"
38            },
39            "gx-service-offering:rpcAddress": {
40                "@value": "https://rpc.gaiaxtestnet.oceanprotocol.com",
41                "@type": "xsd:anyURI"
42            },
43            "gx-service-offering:chainExplorer": {
44                "@value": "https://blockscout.gaiaxtestnet.oceanprotocol.com",
45                "@type": "xsd:anyURI"
46            },
47            "gx-service-offering:nodeClient": {
48                "@value": "OpenEthereum",
49                "@type": "xsd:anyURI"
50            },
51            "gx-service-offering:nodeClientVersion": {
52                "@value": "3.2.6",
53                "@type": "xsd:anyURI"
54            },
```


- Using the Experimental Service Offering to describe the validator nodes and network itself
- Validator nodes and network Self Descriptions as service SDs:
  - https://www.delta-dao.com/.well-known/serviceDLTvalidatorBDB.json
  - https://www.delta-dao.com/.well-known/serviceDLTvalidatorDeltaDAO.json
  - https://www.delta-dao.com/.well-known/serviceDLTnetwork.json
- Using Participant Self Descriptions to define the provider of the service:
  - https://delta-dao.com/.well-known/participantBigchainDB.json
  - https://delta-dao.com/.well-known/participant.json

Useful Resources

- **Compliance Service Documentation:** https://gitlab.com/gaia-x/lab/compliance/gx-compliance
- **Compliance Service Guide:** https://compliance.gaia-x.eu/guide/
- **Compliance Service:** https://compliance.gaia-x.eu/docs/
- **SD Signer:** https://github.com/deltaDAO/self-description-signer
- **Gaia-X Testnet Node setup**: https://github.com/oceanByte/gaia-x-node-setup
- **Gaia-X Faucet**: https://faucet.gx.gaiaxtestnet.oceanprotocol.com/
- **Client Documentation** https://openethereum.github.io/index
- **OpenEthereum Client Releases:** https://github.com/openethereum/openethereum/releases
- **Gaia-X Testnet Explorer**: https://exchangelog.minimal-gaia-x.eu/
- **Gaia-X Testnet Stats:** https://stats.minimal-gaia-x.eu/
- **JSON-RPC methods**: https://openethereum.github.io/JSONRPC-eth-module