hack-sdk-java

Java version SDK for hackathon.

Requirements

Java 1.8 or higher.
Maven

Installation

Directly clone the project and use the project folder as your working folder
You may find IntelliJ Idea extremely helpful.

Usage

Initialization

All APIs are provided in OgSolver.java and can be called once the OGSolver is inited. For SpringBoot applications, OgSolver will be autowired as a Bean. Parameters are specified in application.yaml. Modifiy the **token** and **privkey** before your initialization.

Basic Queries

OgSolver includes all usages of APIs provided. For example, you may call queryAllTxsInPool() to get all transactions that is currently in the pool and to be picked up as parents.

```
// Here OgSolver is built using parameters. In SpringBoot applications these
parameters will be injected. Use Autowired instead to get an instance of
OgSolver.
Account account = new Account("privkeyhex");
OgSolver os = new OgSolver(url, token, account.getPrivateKey());
System.out.println(os.queryAllTxsInPool());
```

The results will be in QueryPoolTxsResp format

To send a transaction, call sendTx(): (you may need to get the latest working nonce in order to fire a correct tx)

```
QueryNonceResp qnr =
ogSolver.queryNonce(this.ogSolver.getAccount().getAddress());
Transaction tx = new Transaction();
tx.setParents(new String[]{seq.getHash()});
tx.setFrom(ogSolver.getAccount().getAddress());
tx.setTo(null);
tx.setNonce(qnr.getNonce() + 1);
tx.setGuarantee(BigInteger.valueOf(1));
tx.setValue(BigInteger.valueOf(0));

os.sendTx(tx)
```

Message Queue Events

To get the latest transaction (including those txs that have not been confirmed) and sequencers, we provide a message queue to annoucne the new tx events. Once a new Tx/Seq is broadcasted, you will be notified if you subscribe the Kafka topic:

```
@KafkaListener(topics = "${app.topic}", groupId = "${api.token}")
public void listen(@Payload String json) throws Exception {
    logger.info(json);
    JSONObject obj = JSON.parseObject(json);
    switch (obj.getIntValue("type")) {
        case TXTYPE_SEQUENCER:
            SequencerResp seq = obj.getObject("data", SequencerResp.class);
            handleSequencer(seq);
            break;
        case TXTYPE_TX:
            TransactionResp tx = obj.getObject("data", TransactionResp.class);

            handleTx(tx);
            break;
     }
}
```

A further processing function should be there to handle this tx:

```
private void handleTx(TransactionResp tx) {
    logger.info("received tx: " + tx.getHash());
}
```

You may find the code in HackathonApplication.java