DEX

Nikita Puzankov

Senior Rust Dev
puzankov@soramitsu.co.jp

High Level Scenario

Given Liquidity Provider has 1 BTC in Pool

And Liquidity Provider has 20 ETH in Pool

And Buyer has 1 BTC Asset

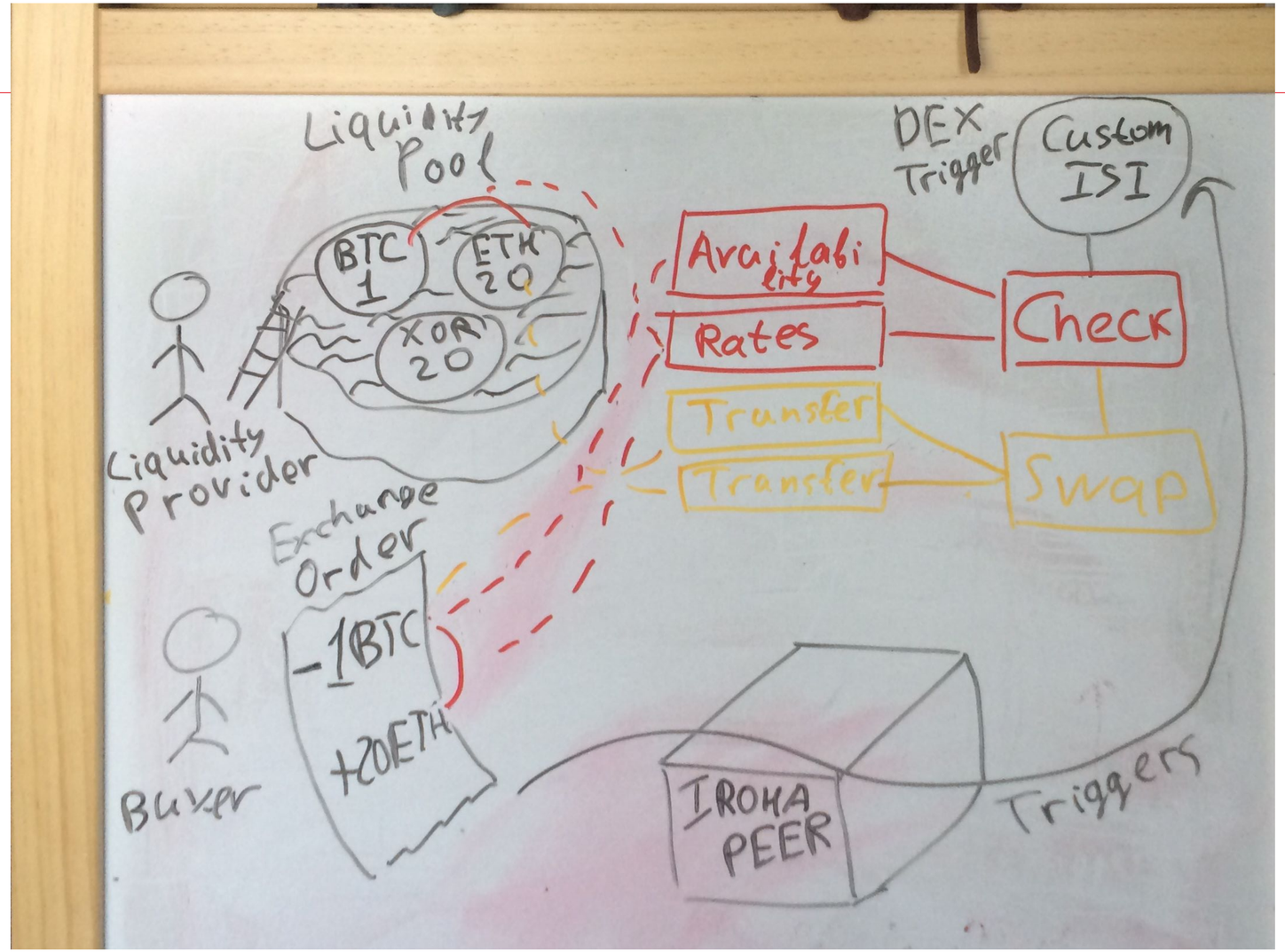
And DEX module is on

When Buyer registers Exchange Order (-1BTC, +20ETH)

Then DEX transfers 1BTC from Buyer to Liquidity Pool

And DEX transfers 20ETH from Liquidity Pool to Buyer

Sequence


Liquidity Pool
BTC 1
ETH 20
XOR 20
Liquidity Provider
DEX Trigger
Custom ISI
Availability
Rates
Transfer
Transfer
Check
Swap
Exchange Order
-1BTC
+20ETH
Buyer
IROHA PEER
Triggers

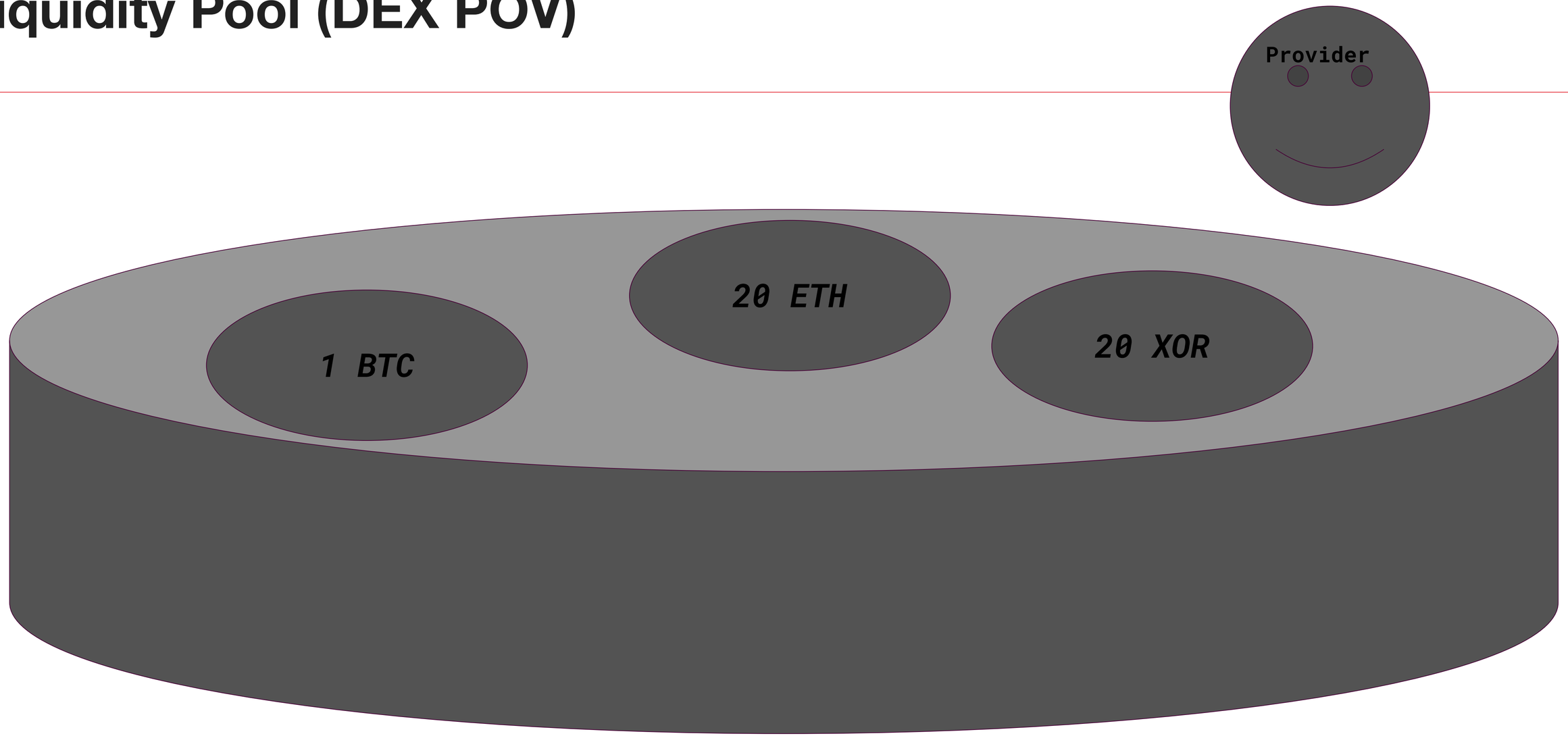
Agenda

. Trigger Registration
. Pool Representation
. Trigger Invocation and Processing
. Assumptions
. Questions from Users

Register Trigger

```
isi::Register<Peer, Instruction> {
  object: DexTrigger::new(),
  destination_id: Peer::current(),
}
```

Liquidity Pool (DEX POV)

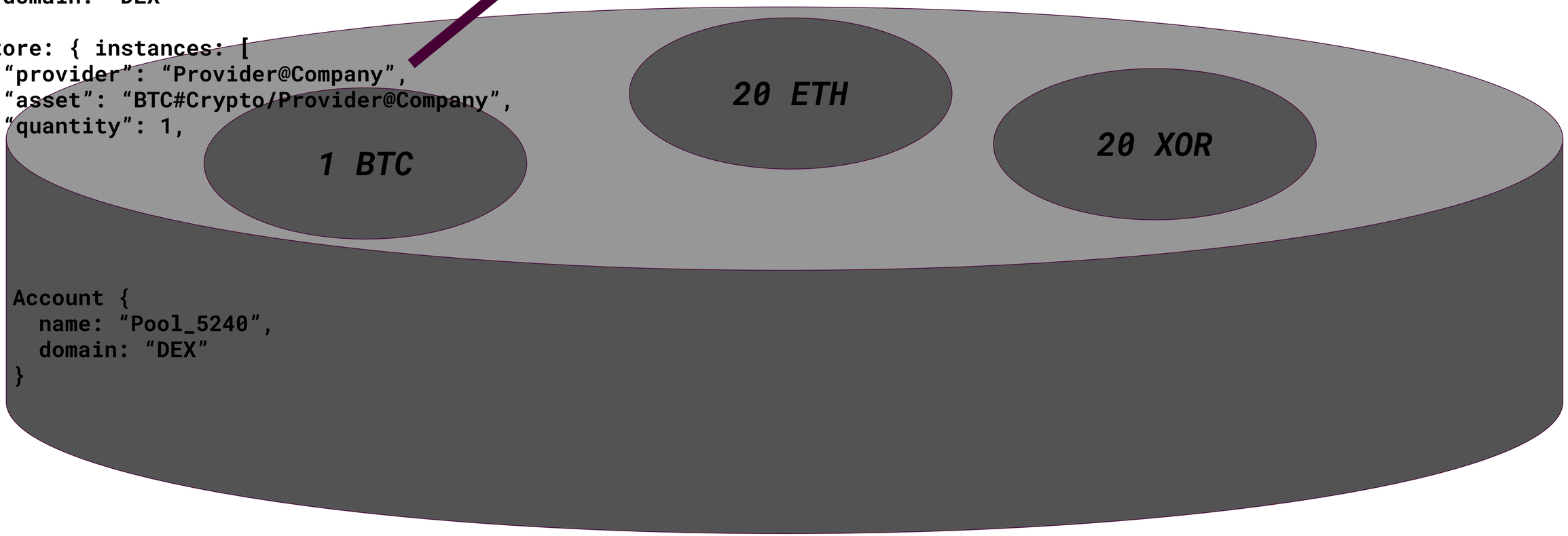


Liquidity Pool (IROHA POV)

```
Asset {
  definition: {
    name: “Pool_Asset”,
    domain: “DEX”
  },
  store: { instances: [
    “provider”: “Provider@Company”,
    “asset”: “BTC#Crypto/Provider@Company”,
    “quantity”: 1,
  ]
  }
}
```

```
Asset {
  definition: {
    name: “BTC”,
    domain: “Crypto”
  },
  quantity: 2
}
```

```
Account {
  name: “Provider”,
  domain: “Company”
}
```



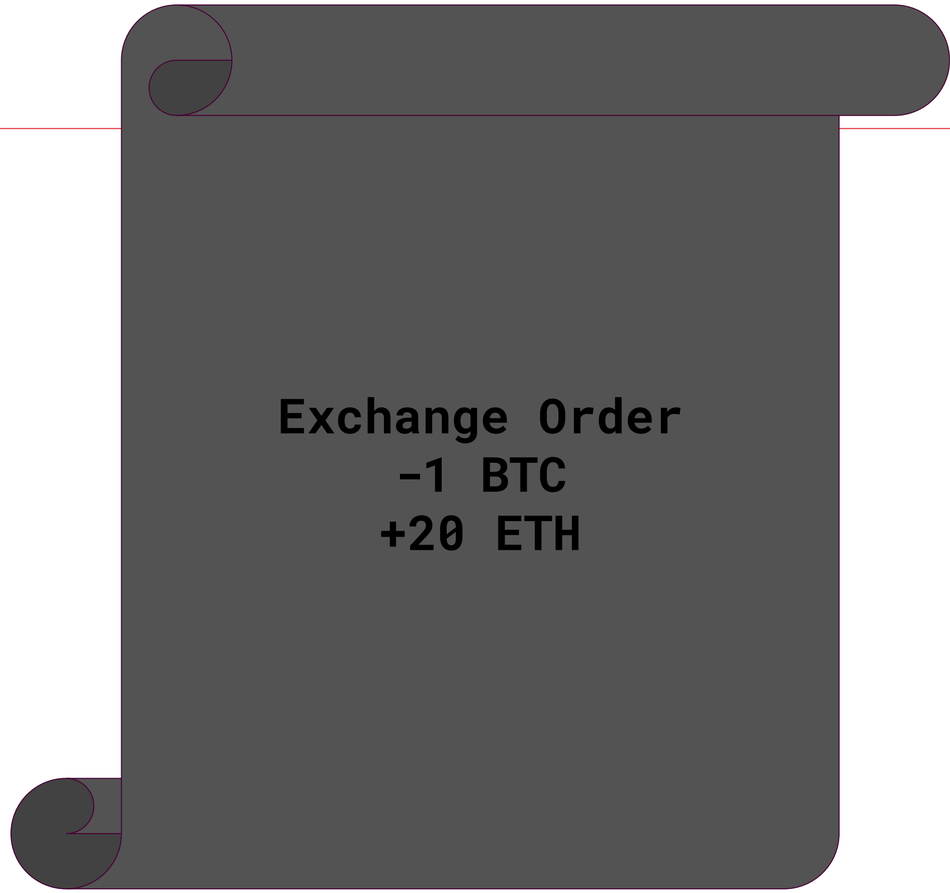

```
Account {
  name: “Pool_5240”,
  domain: “DEX”
}
```

Trigger Invocation

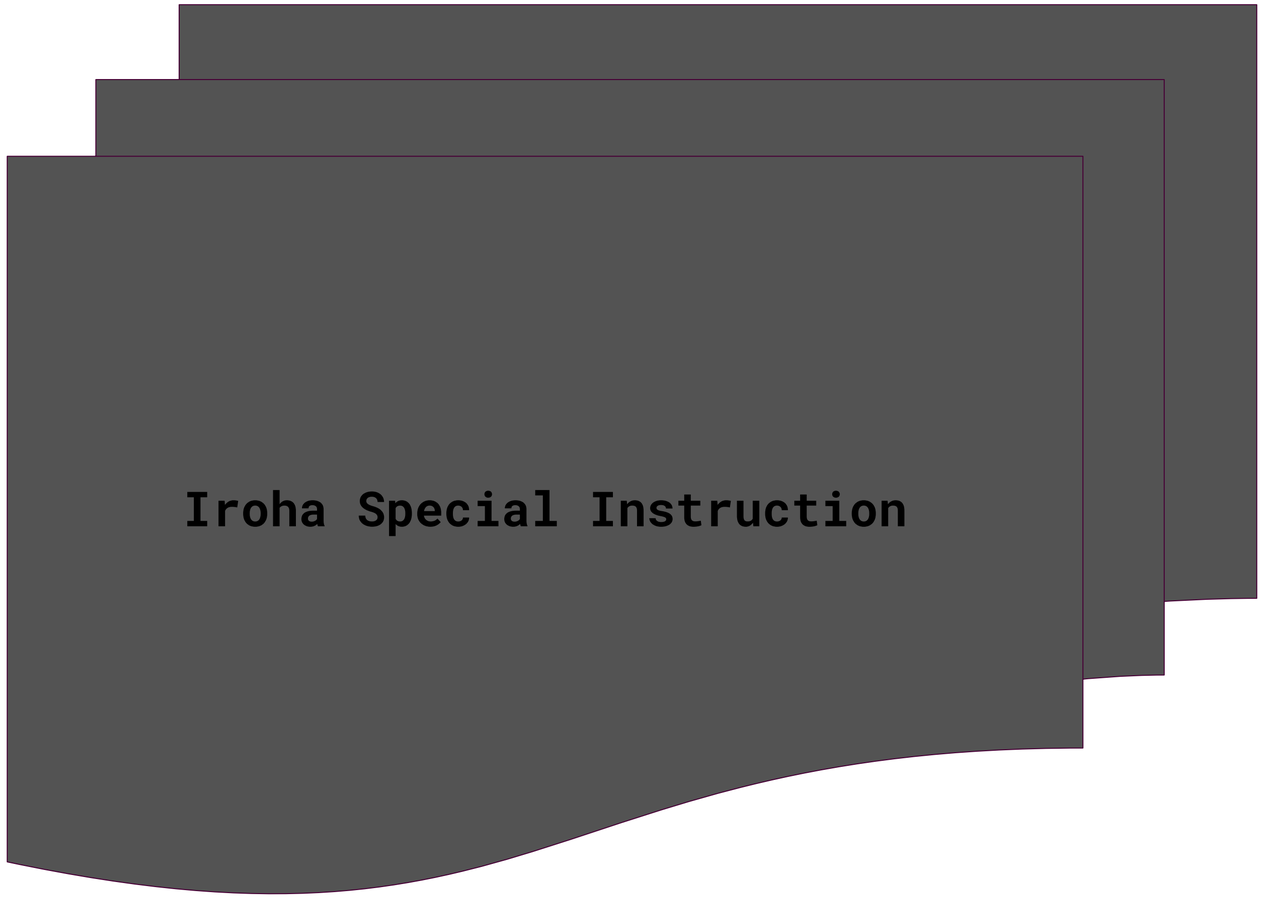
```
Trigger {
  condition: Instruction,
  action: Instruction
}
```

Trigger Invocation

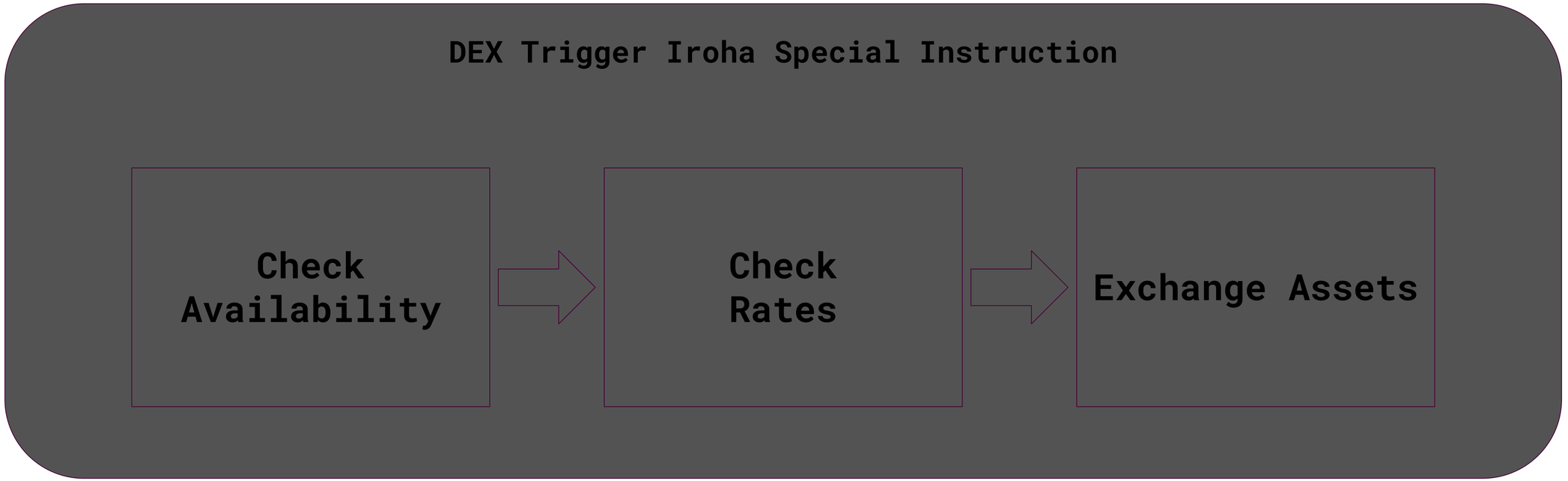
```
Trigger {
  condition: Instruction,
  action: Instruction
}
```



Trigger Processing



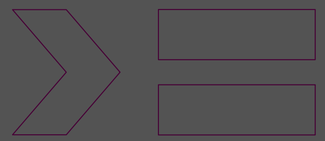
Trigger Processing



Trigger Processing

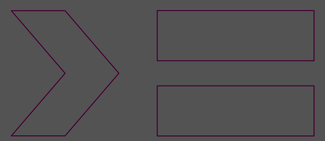
Check Availability Iroha Special Instruction

```
IrohaQuery::GetAsset(
  Pool_Asset#DEX/DEX@DEX
)[
  ETH#Crypto/Provider@Company
]
```



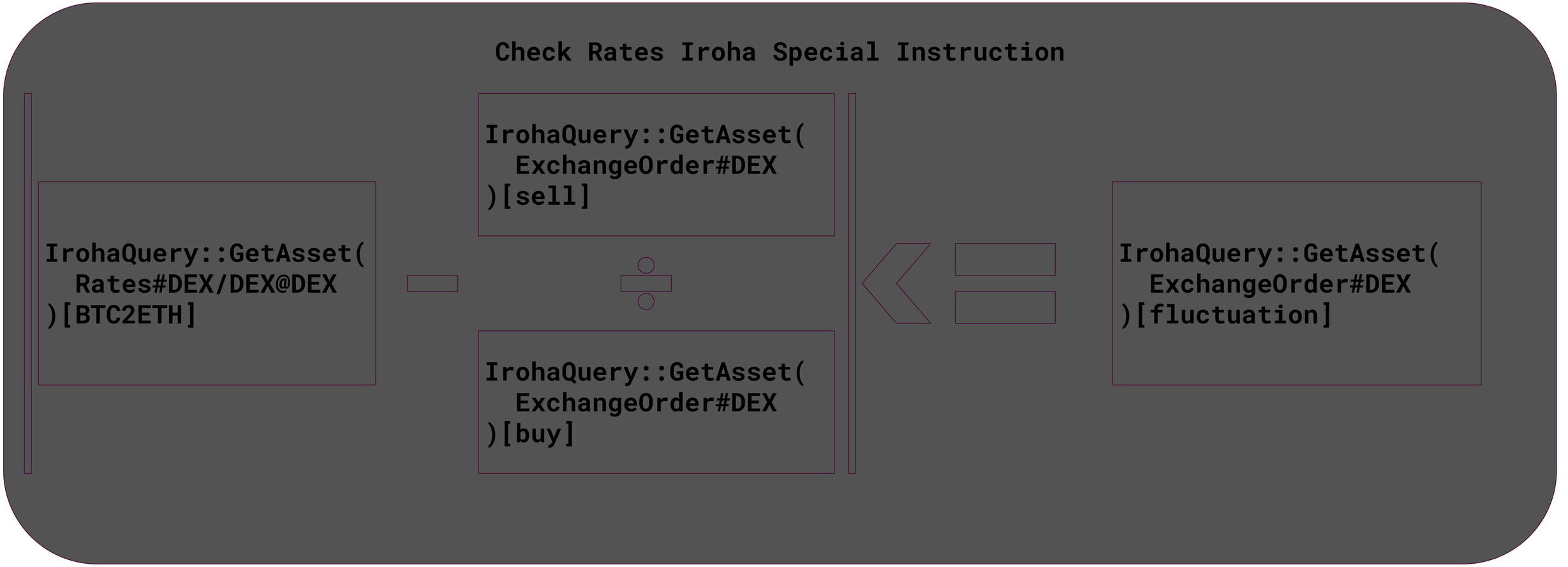
```
IrohaQuery::GetAsset(
  ExchangeOrder#DEX/Buyer@Company
)[buy]
```

```
IrohaQuery::GetAssetQuantity(
  BTC#Crypto/Buyer@Company
)
```



```
IrohaQuery::GetAsset(
  ExchangeOrder#DEX/Buyer@Company
)[sell]
```

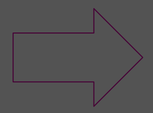
Trigger Processing



Trigger Processing

Exchange Assets Iroha Special Instruction

```
Transfer(
  source: BTC#Crypto/Buyer@Company,
  destination: Pool_Asset#DEX/DEX@DEX[
    BTC#Crypto/Provider@Company
  ]
)
```



```
Transfer(
  source: Pool_Asset#DEX/DEX@DEX[
    ETH#Crypto/Provider@Company
  ],
  destination: ETH#Crypto/Buyer@Company
)
```

Assumptions

* Liquidity Provider owns Assets equal or greater than provided in Liquidity Pool

* Liquidity Provider sets appropriate exchange rates

[1](https://uniswap.org/docs/v1/frontend-integration/pool-liquidity/#add-liquidity)

[2](https://soramitsu.atlassian.net/wiki/spaces/PLS/pages/1716944966/XYK+Liquidity+Pools)

Questions from Users

To have the ability to pass exchange-related data like 'exchanging asset ID' and etc. to the algorithm (from asset-store, for example).

Composite ISI can be used.

To be able to manipulate the given data. For example, to calculate the exchanged currency amount.

Math ISI can be used.

To store new data. For example, to store the calculated amount.

Assets can be used.

To load saved data and pass it as an argument for an ISI. E.g. passing the calculated amount to the `TransferAsset` ISI.

Assets + Queries can be used.

To have a guarantee that all the stored data is consistent, so the user would not have to verify all the data by itself.

Iroha World State View and Blockstore are consistent storages.

[About Triggers]: It should be persisted, so no one could change the order of the instructions.

Triggers will be stored under the Peer entity.

[About Triggers]: It should be guaranteed that the creator of the algorithm is not permitted to call these instructions by itself.

Triggers are executed by declared conditions, not by direct request from the client side.

Questions?

Nikita Puzankov

Senior Rust Dev
puzankov@soramitsu
.co.jp