HERODOTUS

Building with the Herodotus Data Processor

PIA PARK, Product Engineer
(Data Processor Core)

@rkdud007
@piapark_eth

@LambdaZk Week | 2024.07.07

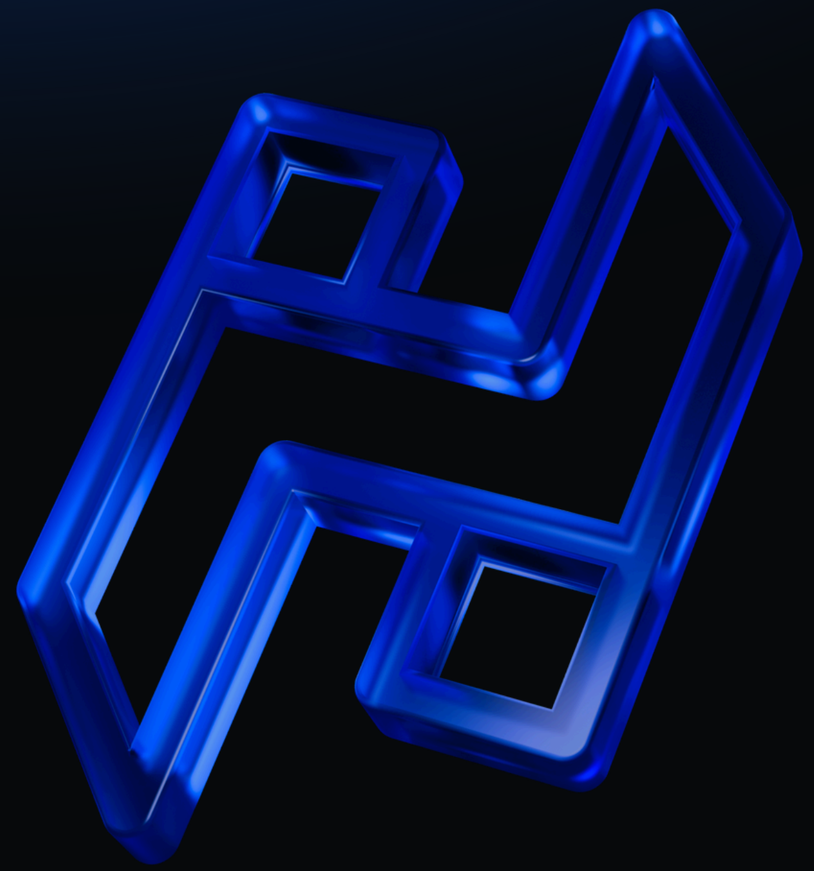
HERODOTUS

THIS IS HOW WE WANT TO ACCESS ONCHAIN DATA

1. From/to any location

2. At any historical time

3. Trust-less (Verifiable)

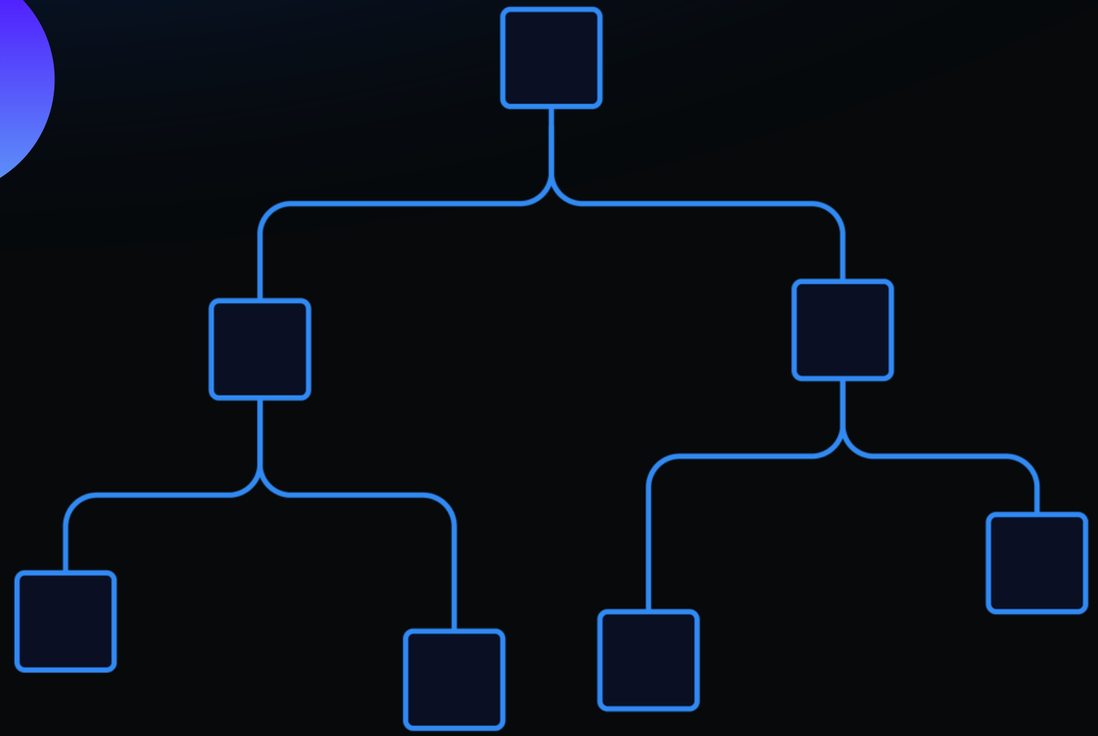


HERODOTUS

WHAT ARE STORAGE PROOFS?

STORAGE PROOFS

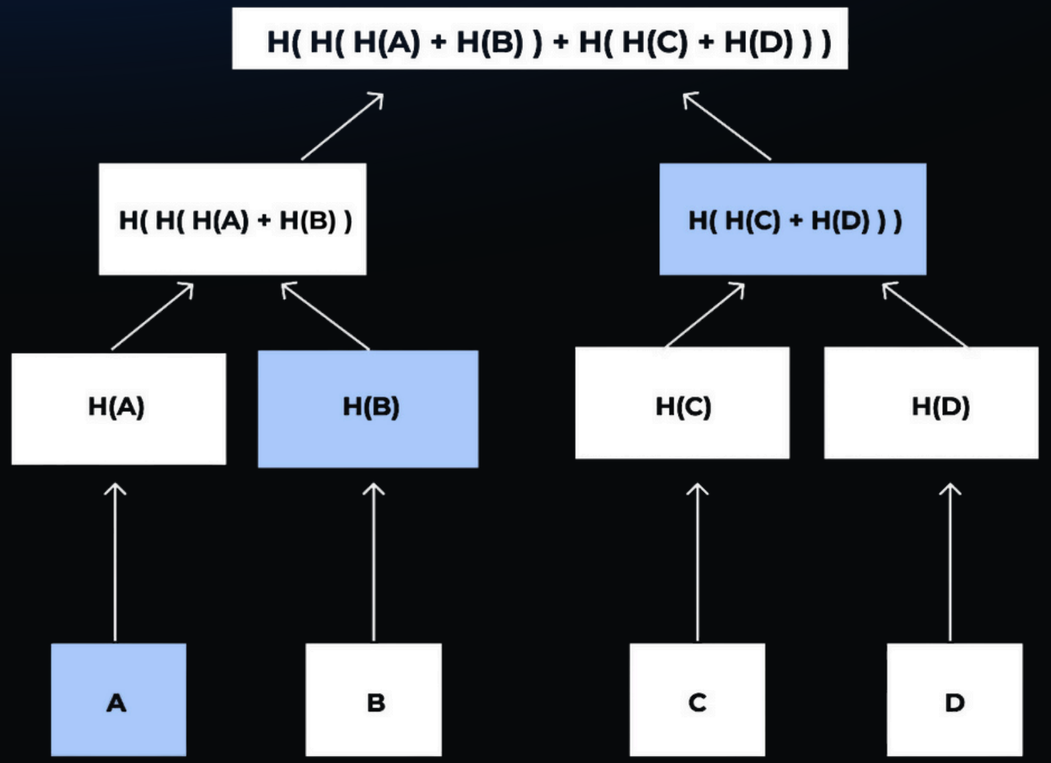
Storage proofs provide smart contracts with synchronous access to current, historical, and cross-chain data across Ethereum layers.



HERODOTUS

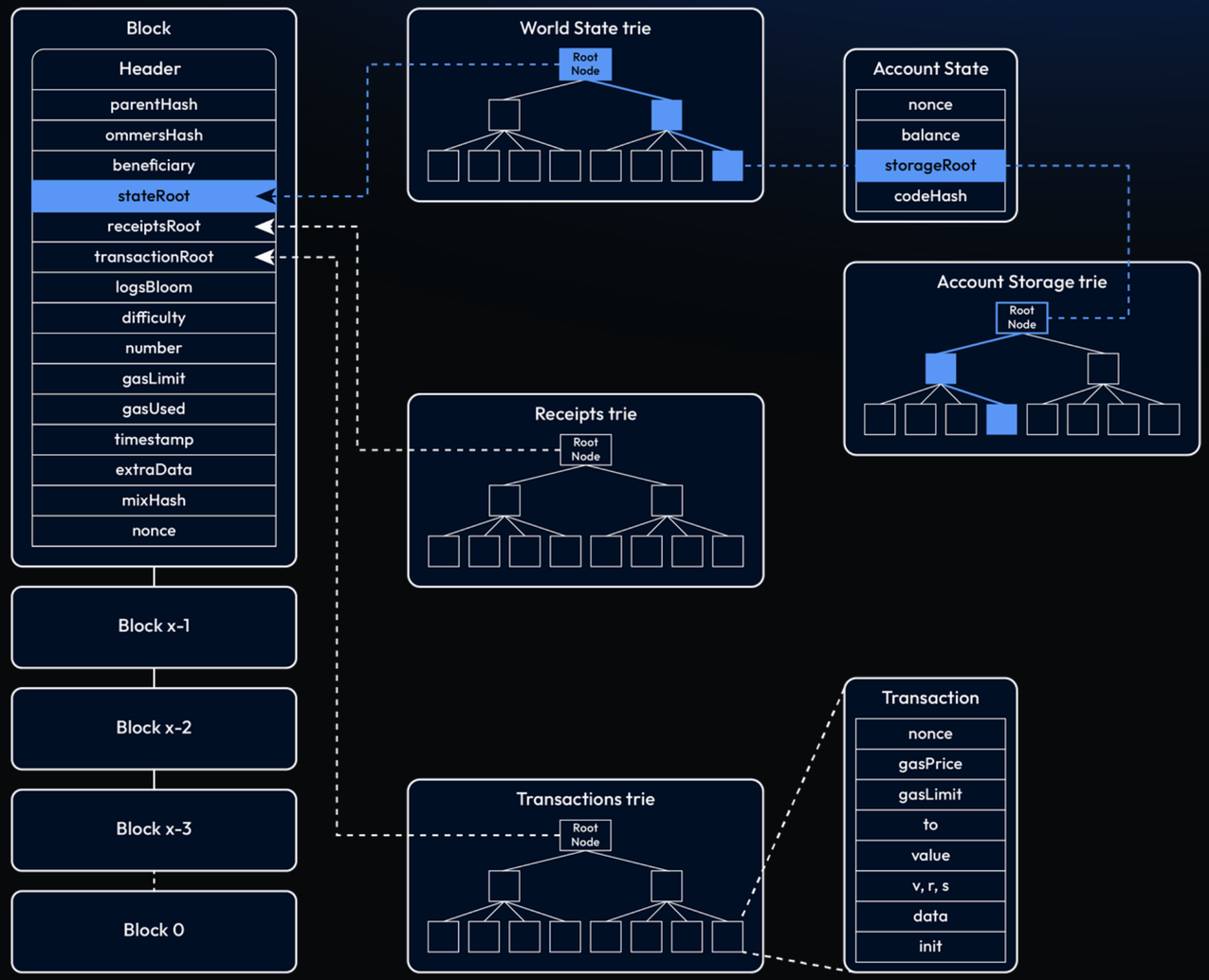
STORAGE PROOFS 101

The core concept behind storage proofs is that existence of any data committed to a stateful blockhain can be proven.



HERODOTUS

SAMPLE STORAGE PROOF FLOW




```
{
0x17C2D875CB...07E348Df07:  100
}
```

1. Select block
2. Prove the existence of an account in block
3. Prove specific data in the account

HERODOTUS

HERODOTUS DATA PROCESSOR

HERODOTUS

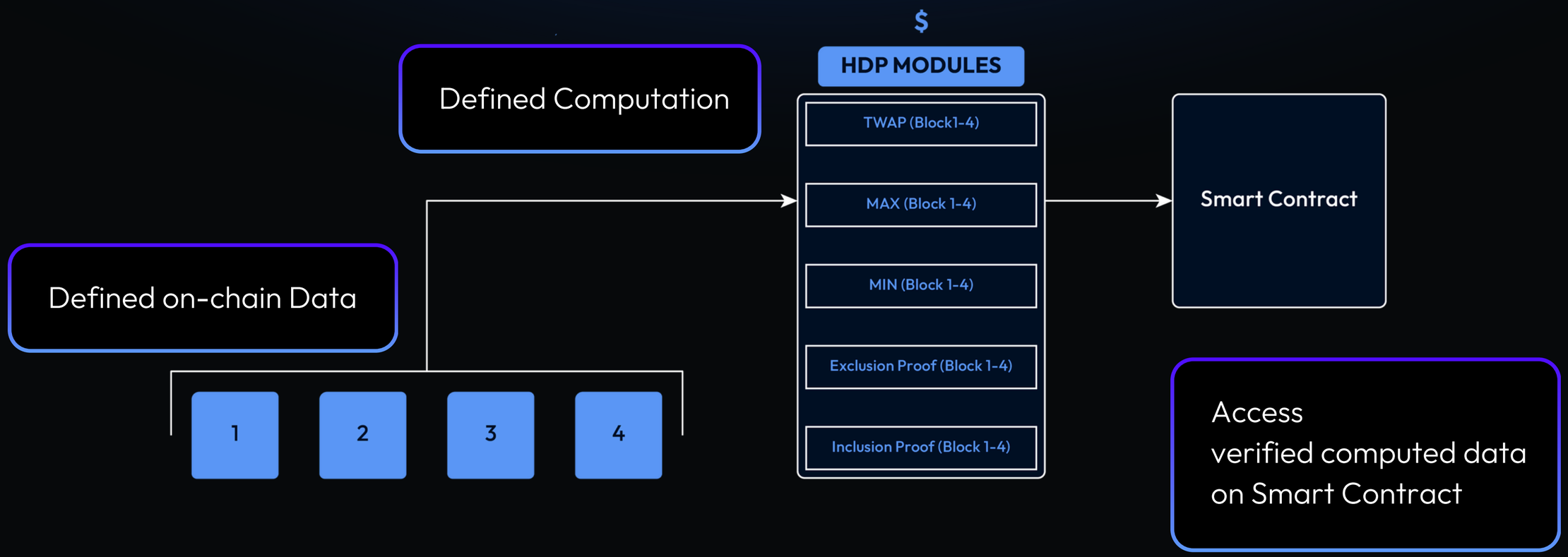
WHAT IS HERODOTUS DATA PROCESSOR?

HDP

HDP is a tool that allows you to easily define **large sets** of on-chain data and then **run compute** over it in a fully sound and proven environment thanks to **STARKs and storage proofs.**

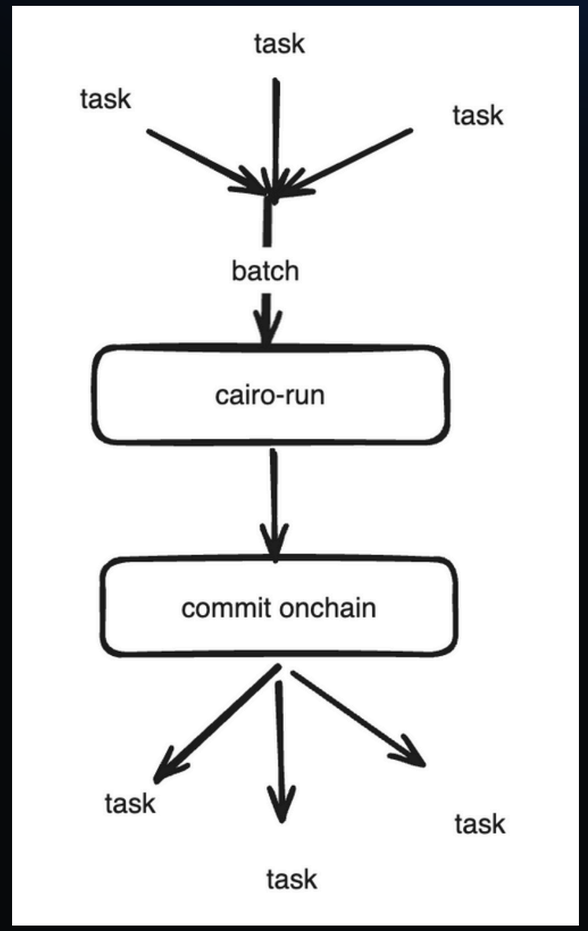
HERODOTUS

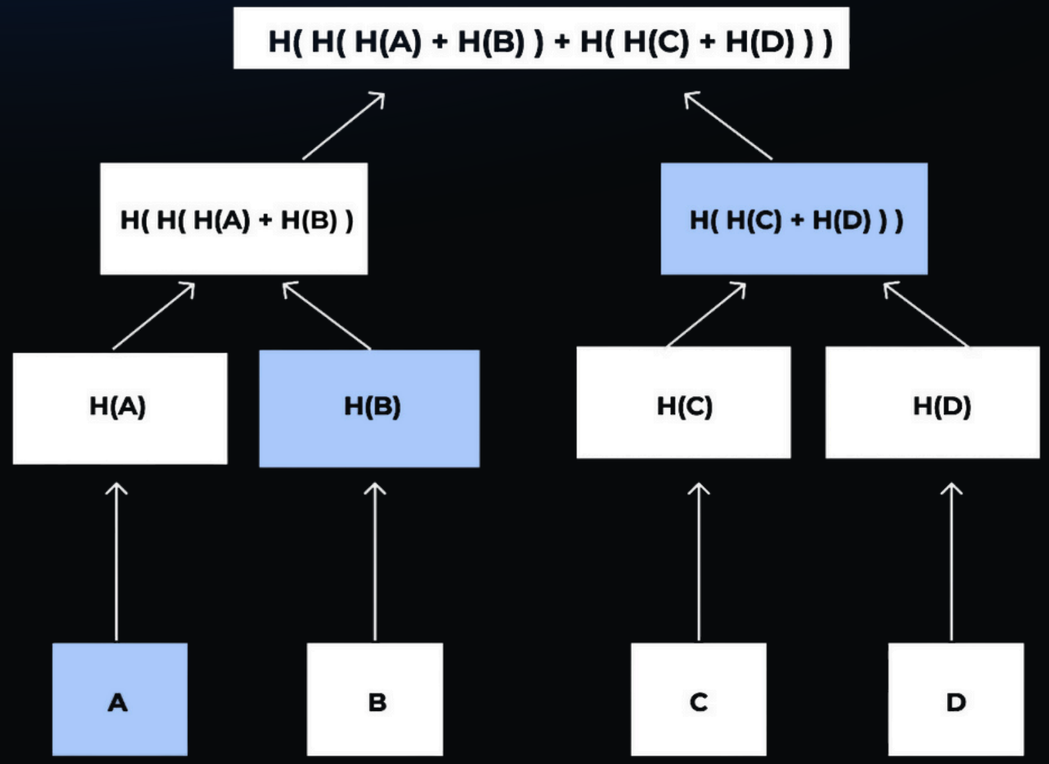
PERFORMING COMPUTE VIA HDP MODULES IN CAIRO VM(OFF CHAIN)



HERODOTUS

BATCH TASKS IN MERKLE TREE





HERODOTUS

WHAT COMPUTATION CAN YOU DEFINE?

AGGREGATE FUNCTION

- AVG
- SUM
- MIN
- MAX
- COUNT_IF

COMPUTE MODULES (CAIRO 1)

- Simple Linear Regression
- Giza ML model (wip)
- ....

HERODOTUS

WHAT ON CHAIN DATA YOU CAN DEFINE?

Feat. The data you want to run computation over

HERODOTUS

1 BlockSampledDatalake

Field Description	SUM	AVG	MIN	MAX	COUNT	SLR
`account.nonce`	✅	✅	✅	✅	✅	✅
`account.balance`	✅	✅	✅	✅	✅	✅
`account.storage_root`	-	-	-	-	-	-
`account.code_hash`	-	-	-	-	-	-
`storage.key` (numeric value)	✅	✅	✅	✅	✅	✅
`storage.key` (hash value)	-	-	-	-	-	-
`header.difficulty`	✅	✅	✅	✅	✅	✅
`header.gas_limit`	✅	✅	✅	✅	✅	✅
`header.gas_used`	✅	✅	✅	✅	✅	✅
`header.timestamp`	✅	✅	✅	✅	✅	✅
`header.base_fee_per_gas`	✅	✅	✅	✅	✅	✅
`header.blob_gas_used`	✅	✅	✅	✅	✅	✅
`header.excess_blob_gas`	✅	✅	✅	✅	✅	✅
`header.nonce`	✅	✅	✅	✅	✅	✅
Other `header` elements	-	-	-	-	-	-

HERODOTUS

API CALL (BLOCK SAMPLED DATALAKE)

/submit-batch-query


```
{
  "deliveryChainId": 11155111,
  "sourceChainId": 11155111,
  "tasks": [
    {
      "datalakeType": "block_sampled",
      "datalake": {
        "blockRangeStart": 5515000,
        "blockRangeEnd": 5515031,
        "sampledProperty": "header.base_fee_per_gas"
      },
      "aggregateFnId": "avg"
    },
    {
      "datalakeType": "block_sampled",
      "datalake": {
        "blockRangeStart": 5515000,
        "blockRangeEnd": 5515031,
        "sampledProperty": "account.0x7f2c6f930306d3aa736b3a6c6a98f512f74036d4.nonce"
      },
      "aggregateFnId": "min"
    }
  ]
}
```

HERODOTUS

2 TransactionsInBlockDatalake

tx.nonce	✅	✅	✅	✅	✅	✅
tx.gas_price	✅	✅	✅	✅	✅	✅
tx.gas_limit	✅	✅	✅	✅	✅	✅
tx.value	✅	✅	✅	✅	✅	✅
tx.v	✅	✅	✅	✅	✅	✅
tx.r	✅	✅	✅	✅	✅	✅
tx.s	✅	✅	✅	✅	✅	✅
tx.chain_id	✅	✅	✅	✅	✅	✅
tx.max_fee_per_gas	✅	✅	✅	✅	✅	✅
tx.max_priority_fee_per_gas	✅	✅	✅	✅	✅	✅
tx.max_fee_per_blob_gas	✅	✅	✅	✅	✅	✅
Other tx elements	-	-	-	-	-	-
tx_receipt.success	✅	✅	✅	✅	✅	✅
tx_receipt.cumulative_gas_used	✅	✅	✅	✅	✅	✅
Other tx_receipt elements	-	-	-	-	-	-

HERODOTUS

API CALL (TRANSACTIONS IN BLOCK)

/submit-batch-query

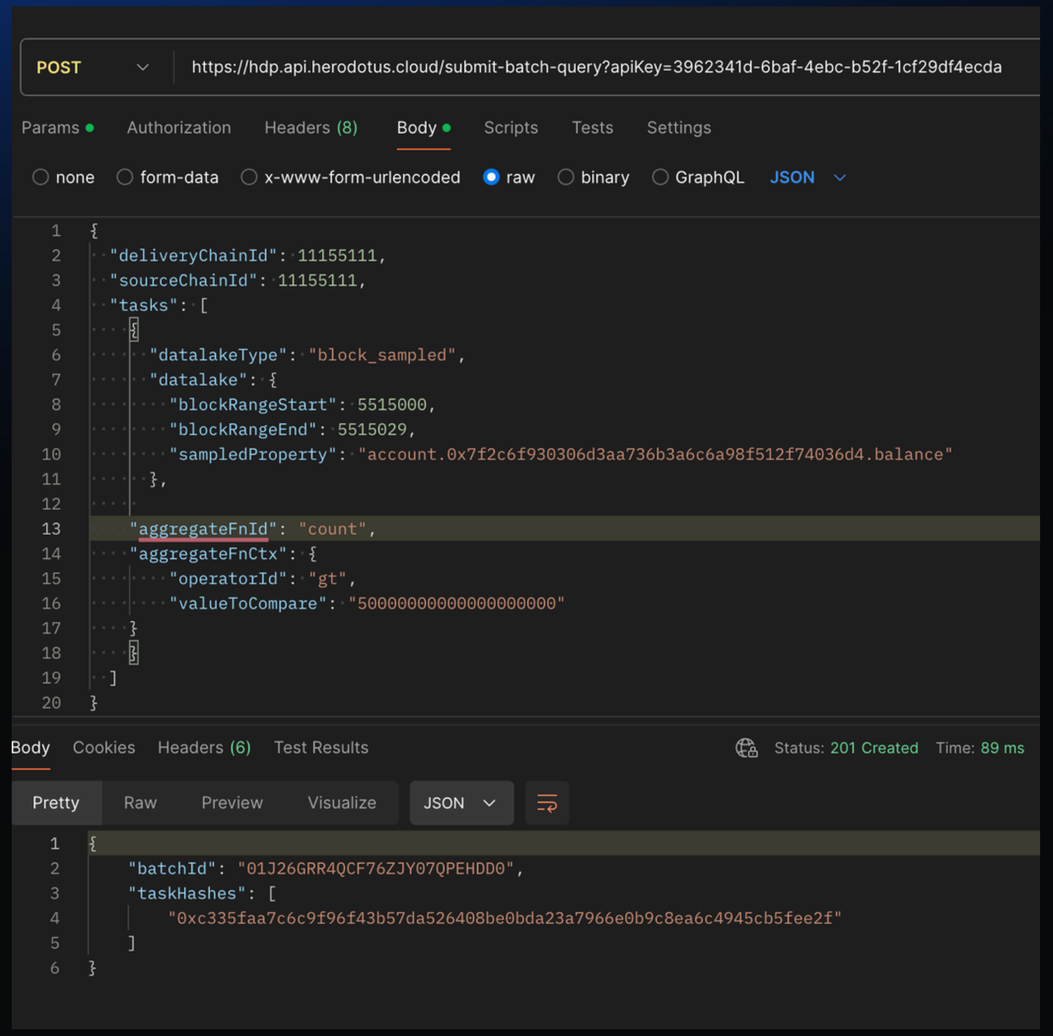

```
{
 "deliveryChainId": 11155111,
 "sourceChainId": 11155111,
  "tasks": [
    {
      "datalakeType": "transactions_in_block",
      "datalake": {
        "targetBlock": 5409986,
        "startIndex": 10,
        "endIndex": 40,
        "increment": 10,
        "includedTypes": {
          "legacy": true,
          "eip2930": true,
          "eip1559": true,
          "eip4844": true
        },
        "sampledProperty": "tx_receipt.success"
      },
      "aggregateFnId": "slr",
      "aggregateFnCtx": {
        "operatorId": "none",
        "valueToCompare": 100
      }
    }
  ]
}
```

HERODOTUS

CASE. COUNTING BALANCE DROPS

COUNTING BALANCE DROPS

To count how often the average balance of an account drops below 50 ETH, you'd use the count_if function. This helps in assesing the frequency of significant balance reduction.

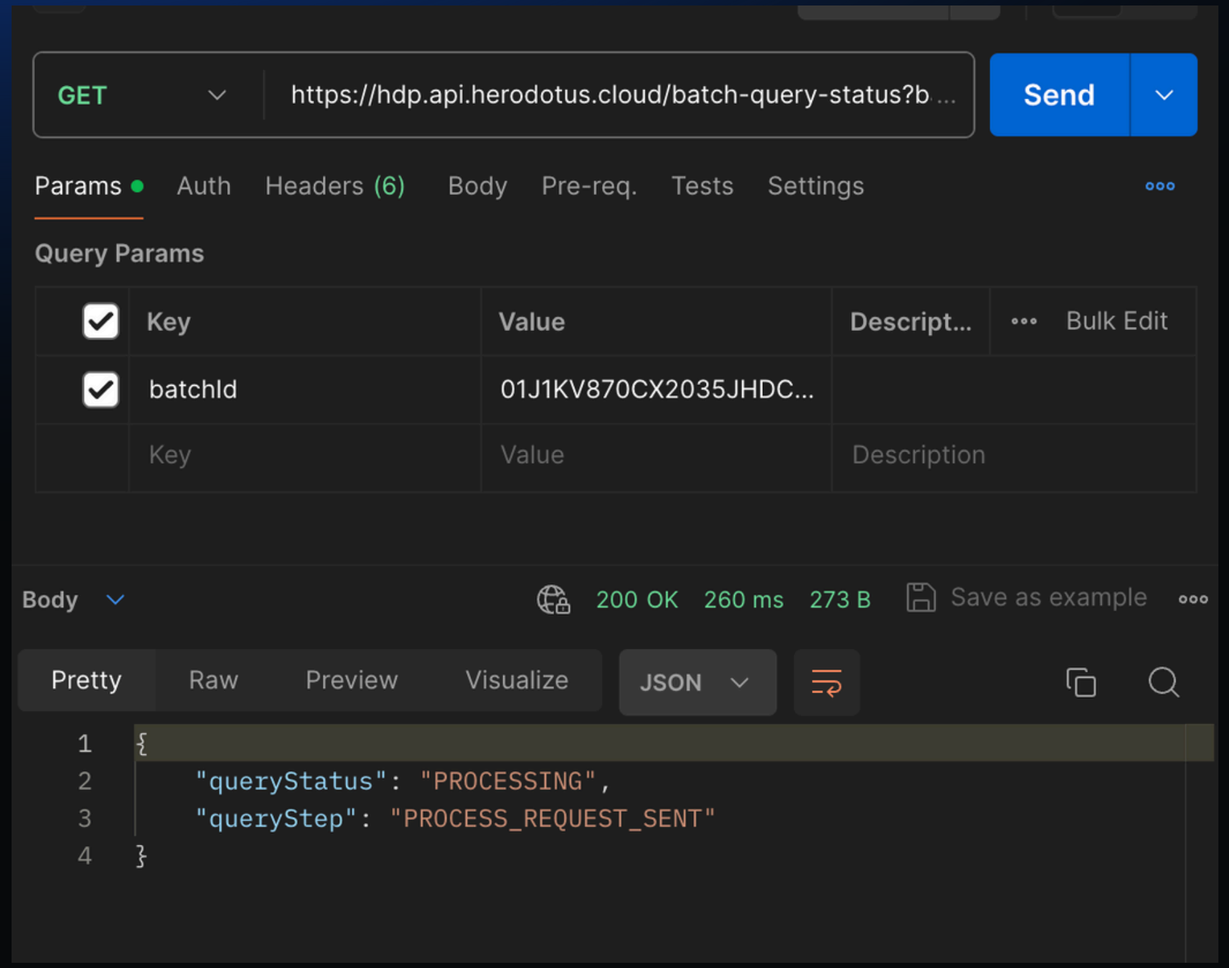


HERODOTUS

CASE. COUNTING BALANCE DROPS

COUNTING BALANCE DROPS

To count how often the average balance of an account drops below 50 ETH, you'd use the count_if function. This helps in assesing the frequency of significant balance reduction.



HERODOTUS

1. Opened:

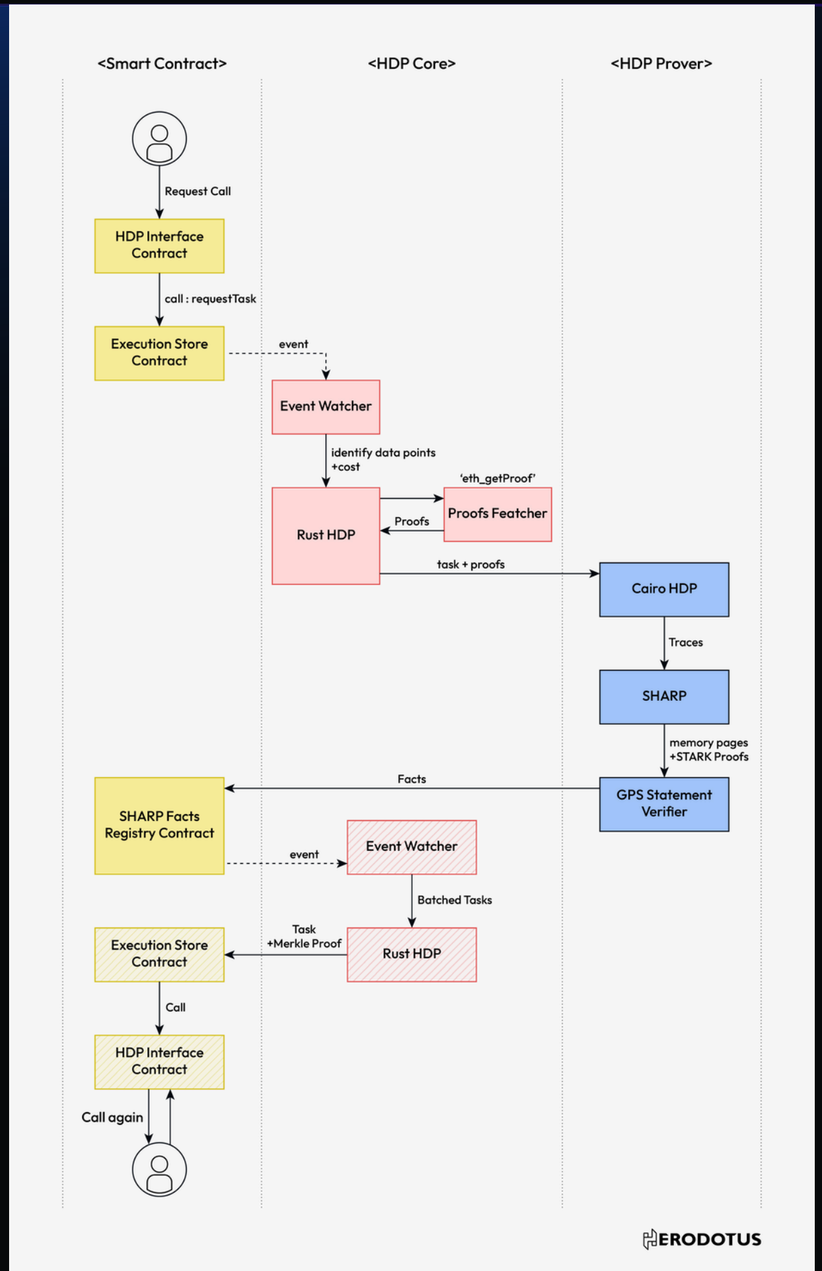
- When the batch is first accepted, it initiates with an opened status.

2. ProofsFetched:

- Successfully fetched proofs from the preprocessor and generated the corresponding PIE object.

3. CachedMmrRoot:

- Successfully cached the MMR root and MMR size used during the preprocessing step to the smart contract.



4. PieSubmittedToSHARP:

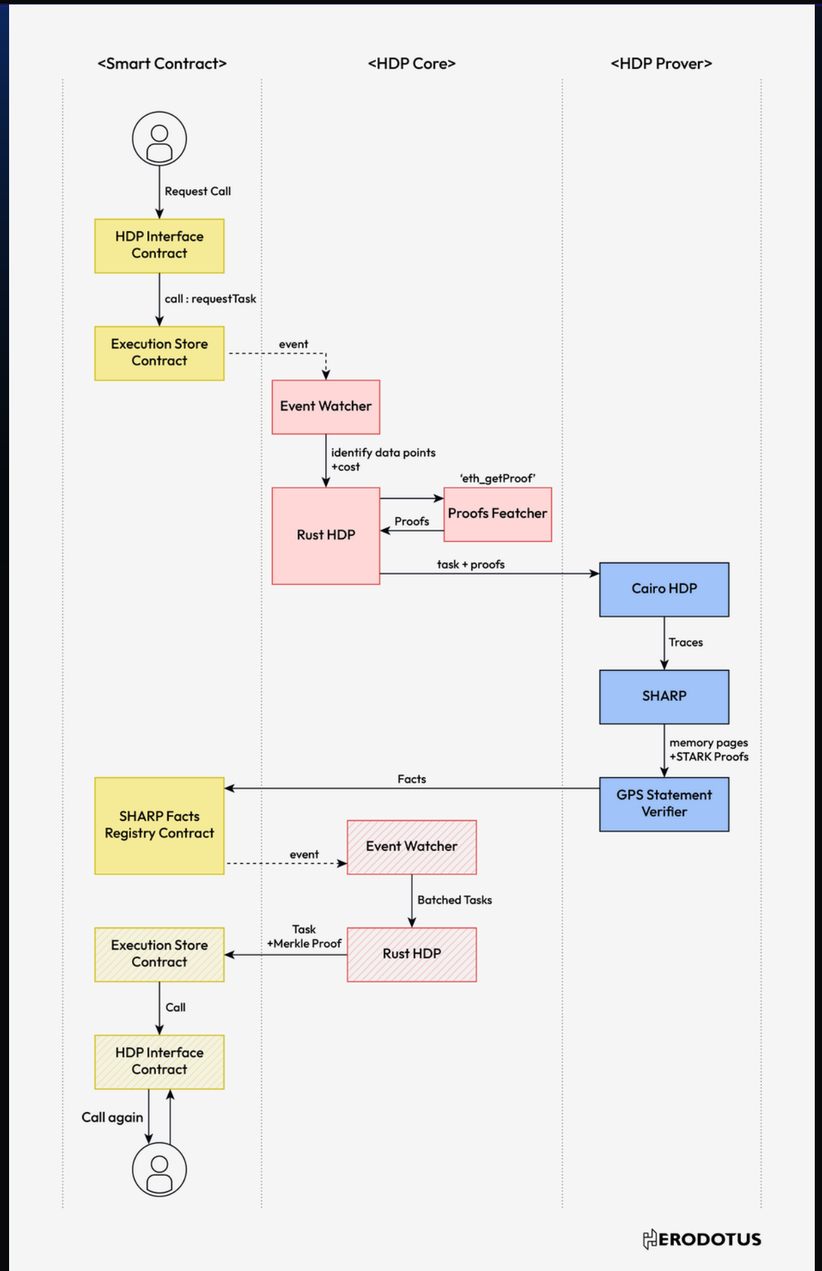
- Successfully submitted the PIE to SHARP.

5. FactRegisteredOnchain:

- The fact hash of the batch is registered in the fact registry contract.

6. Finalized:

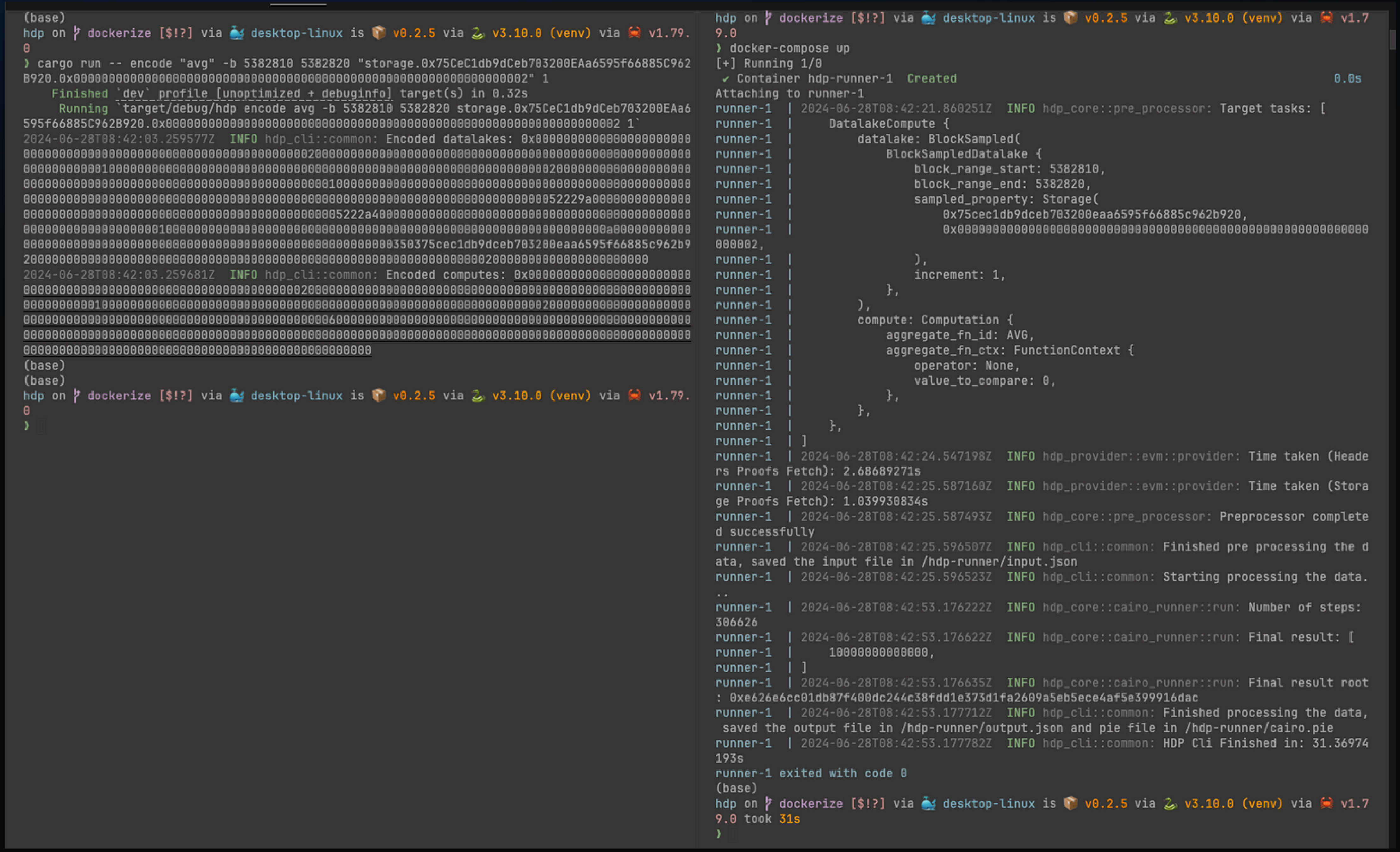
- Successfully authenticated the fact hash and batch, and finalized the valid result on the contract mapping.



HERODOTUS

HDP local environment:

- generate PIE locally
- validate the request is valid
- comprehensive log
- pre setup cairo env from container



HERODOTUS

HDP Interactive CLI:

- one command : hdp start

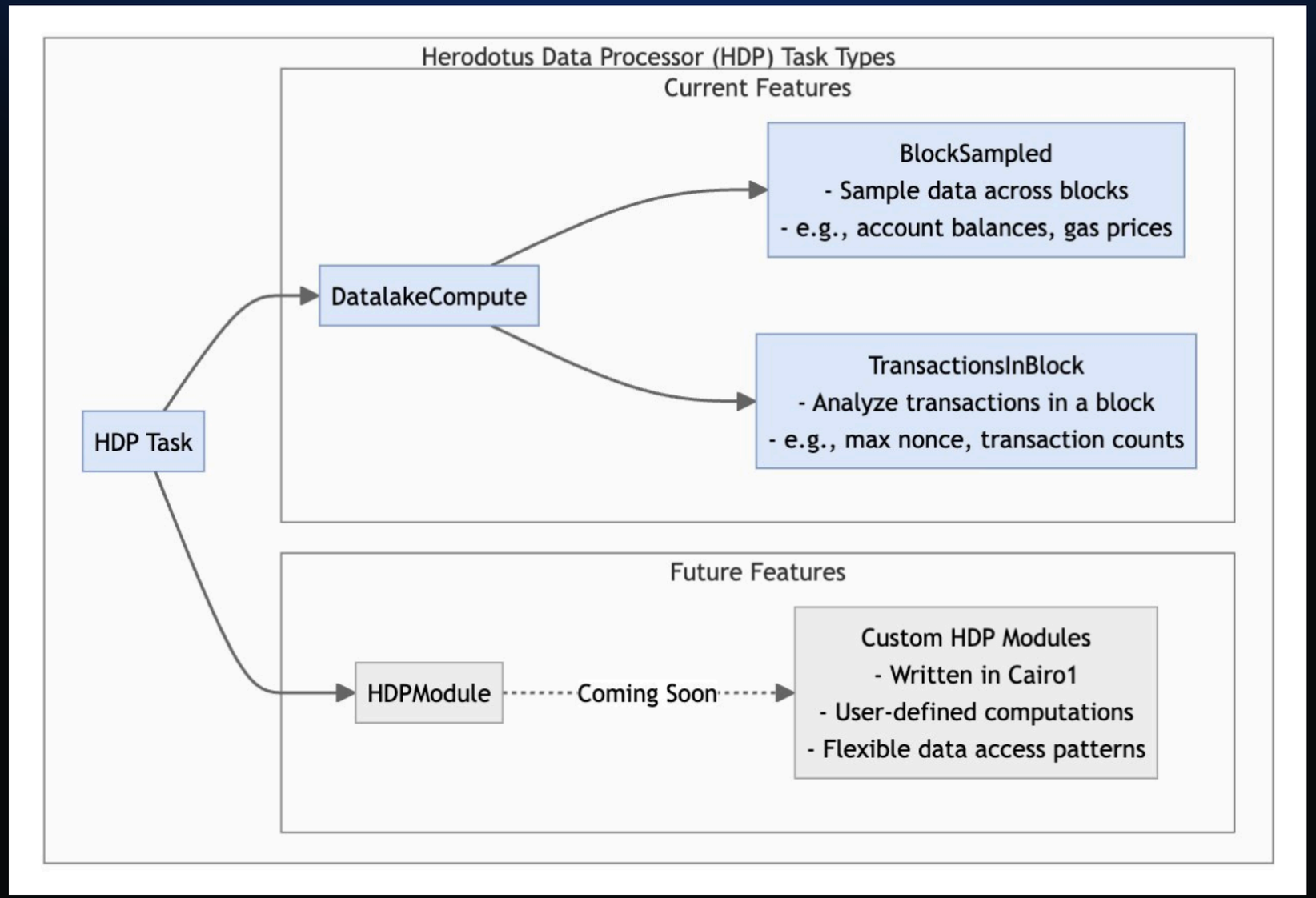
Allows to build request step by step!

```
❯ hdp start
Welcome to Herodotus Data Processor interactive CLI! 🛰️

        _   _   ____    ____
        | | | | |  _ \  |  _ \
        | |_| | | | | | | |_) |
        |  _  | | |_| | |  __/
        |_| |_| |____/  |_|

> Step 1. What's your datalake type? BLOCK_SAMPLED
> Block range start 10
> Block range end 1000
> Increment 1
> Sample Property: Select block sample type HEADER
> Select detail header property NONCE
> Select the aggregation function MIN
> Do you want to run the evaluator? Yes
> Enter RPC URL:
> Enter Chain ID:
> Enter Output file path:  output.json
> Enter Cairo input file path: input.json
> Enter PIE output file path: hdp_pie.zip
2024-06-30T06:19:34.018276Z  INFO hdp_core::pre_processor: Target tasks: [
    DatalakeCompute {
        datalake: BlockSampled(
            BlockSampledDatalake {
                block_range_start: 10,
                block_range_end: 1000,
                sampled_property: Header(
                    Nonce,
                ),
                increment: 1,
            },
        ),
        compute: Computation {
            aggregate_fn_id: MIN,
            aggregate_fn_ctx: FunctionContext {
                operator: None,
                value_to_compare: 0,
            },
        },
    },
]
```

HERODOTUS


Herodotus Data Processor (HDP) Task Types
Current Features
BlockSampled
- Sample data across blocks
- e.g., account balances, gas prices
DatalakeCompute
TransactionsInBlock
- Analyze transactions in a block
- e.g., max nonce, transaction counts
HDP Task
Future Features
Custom HDP Modules
- Written in Cairo1
- User-defined computations
- Flexible data access patterns
HDPModule
Coming Soon

HERODOTUS

HDP version 2

: HDP Runtime with custom module

-> quick demo!

```cairo
#[starknet::contract]
mod contract {
    use hdp_cairo::{HDP, memorizer::account_memorizer::{AccountKey, AccountMemorizerImpl}};
    use starknet::syscalls::call_contract_syscall;
    use starknet::{ContractAddress, SyscallResult, SyscallResultTrait};

    #[storage]
    struct Storage {}

    #[external(v0)]
    pub fn main(
        ref self: ContractState,
        hdp: HDP,
        block_range_start: u32,
        block_range_end: u32,
        address: felt252
    ) -> u256 {
        let mut i: u32 = block_range_start;
        let mut sum: u256 = 0;
        loop {
            if i < block_range_end {
                sum += hdp
                    .account_memorizer
                    .get_balance(
                        AccountKey { chain_id: 11155111, block_number: i.into(), address: address }
                    )
            } else {
                break;
            }
            i += 1;
        };
        sum
    }
}
```

HERODOTUS

```
bash: no job control in this shell
2024-07-07T10:43:27.949840Z  INFO hdp_preprocessor::module_registry: Fetching contract class from module registry... Contract Class Hash: FieldElement {
    inner: 0x02aacf92216d1ae71fbdaf3f41865c08f32317b37be18d8c136d442e94cdd823,
}
2024-07-07T10:43:28.181173Z  INFO hdp_preprocessor::module_registry: Contract class fetched successfully
2024-07-07T10:43:28.191467Z  INFO hdp_preprocessor::module_registry: Program Hash: FieldElement {
    inner: 0x04df21eb479ae4416fbdc00abab6fab43bff0b8083be4d1fd8602c8fbfbd2274,
}
2024-07-07T10:43:28.191538Z  INFO hdp_preprocessor::compile::module: target task: Module {
    class_hash: FieldElement {
        inner: 0x04df21eb479ae4416fbdc00abab6fab43bff0b8083be4d1fd8602c8fbfbd2274,
    },
    inputs: [
        FieldElement {
            inner: 0x00000000000000000000000000000000000000000000000000000000004f21e5,
        },
        FieldElement {
            inner: 0x00000000000000000000000000000000000000000000000000000000004f21e8,
        },
        FieldElement {
            inner: 0x0000000000000000000000013cb6ae34a13a0977f4d7101ebc24b87bb23f0d5,
        },
    ],
    local_class_path: None,
}
2024-07-07T10:43:28.191708Z  INFO hdp_preprocessor::compile::module: 2. Running dry-run...
2024-07-07T10:43:31.307962Z  INFO hdp_cairo_runner::dry_run: Dry-runner executed successfully
2024-07-07T10:43:31.308023Z  INFO hdp_preprocessor::compile::module: 3. Fetching proofs from provider...
2024-07-07T10:43:31.521510Z  INFO hdp_provider::evm::from_keys: Time taken (Headers Proofs Fetch): 208.552542ms
2024-07-07T10:43:31.764274Z  INFO hdp_provider::evm::from_keys: Time taken (Accounts Proofs Fetch): 242.728208ms
2024-07-07T10:43:31.764826Z  INFO hdp_preprocessor: 1️⃣ Preprocessor completed successfully
2024-07-07T10:43:31.767396Z  INFO hdp_cli::common: Finished pre processing the data, saved the input file in /hdp-demo/hdp_input.json
2024-07-07T10:43:31.767408Z  INFO hdp_cli::common: Starting processing the data...
2024-07-07T10:43:41.820991Z  INFO hdp_cairo_runner::run: Number of steps: 94661
2024-07-07T10:43:41.821049Z  INFO hdp_cairo_runner::run: Cairo run output: CairoRunOutput {
    tasks_root: 0xa51a6cc7bc840aec540121e72180d35414db998f26ecb1eda7c8896baaf451b7,
    results_root: 0x5fe7dd5abe72bbb124c33cbd4b25dfc1ccebe944a918dfd5a93015f85589e117,
    results: [
        0,
    ],
}
2024-07-07T10:43:41.821114Z  INFO hdp_processor: 2️⃣ Processor completed successfully
2024-07-07T10:43:41.821204Z  INFO hdp_cli::common: Finished processing the data, saved the output file in /hdp-demo/output.json and pie file in /hdp-demo/cairo.pie
2024-07-07T10:43:41.821240Z  INFO hdp_cli::common: HDP Cli Finished in: 13.874028131s
```

HERODOTUS

Contract

A contract is a program and an instance of a class on Starknet.

Overview | Transactions 0 | Events 0 | Account Calls 0 | Portfolio | Class Code/History | Read/Write Contract | Token Transfers | NFT Events | Storage Slots

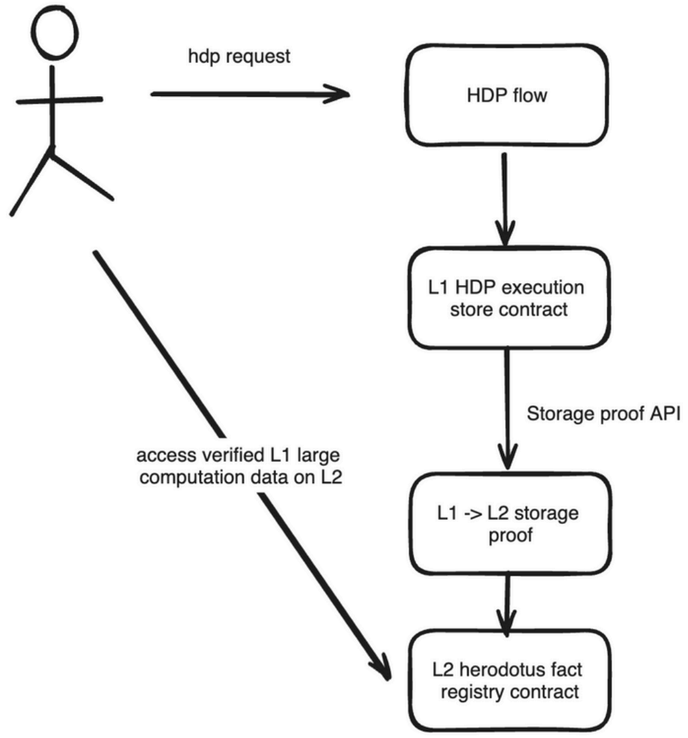
Contract Address	0x009d750d3373617962343257af19a3a5b051ba8a3ae097bcc16f2e64bf5900e7
Class Hash	0x02aacf92216d1ae71fbdaf3f41865c08f32317b37be18d8c136d442e94cdd823
ETH Balance	0 ETH View All Tokens →
Deployed By Contract Address	0x01172c7024f026c9bf89b47e39be72f5ed7713982f6ddc3e38976a769ab997ad
Deployed At Transaction Hash	0x01b9d1f77f36b06ff9c403654354b8b327431ce29fc405e75f9c1a2ceca8f791
Deployed At	July 3, 2024 at 11:33:31 AM GMT+2
Class Version	Cairo 2

HERODOTUS

USE HDP TO ACCESS DATA CROSS CHAIN

L1 -> L2

Just by changing the delivery chain id,
can send these data to other L2 by using
Storage Proof API by Herodutus.

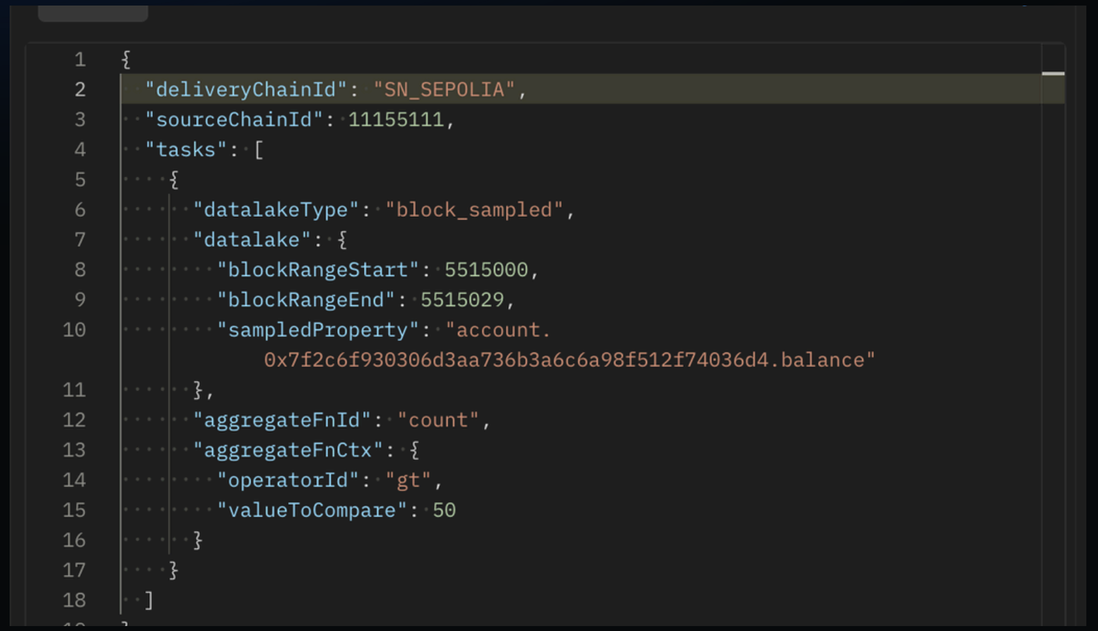


HERODOTUS

USE HDP TO ACCESS DATA CROSS CHAIN

L1 -> L2

Just by changing the delivery chain id, can send these data to other L2 by using Storage Proof API by Herodutus.



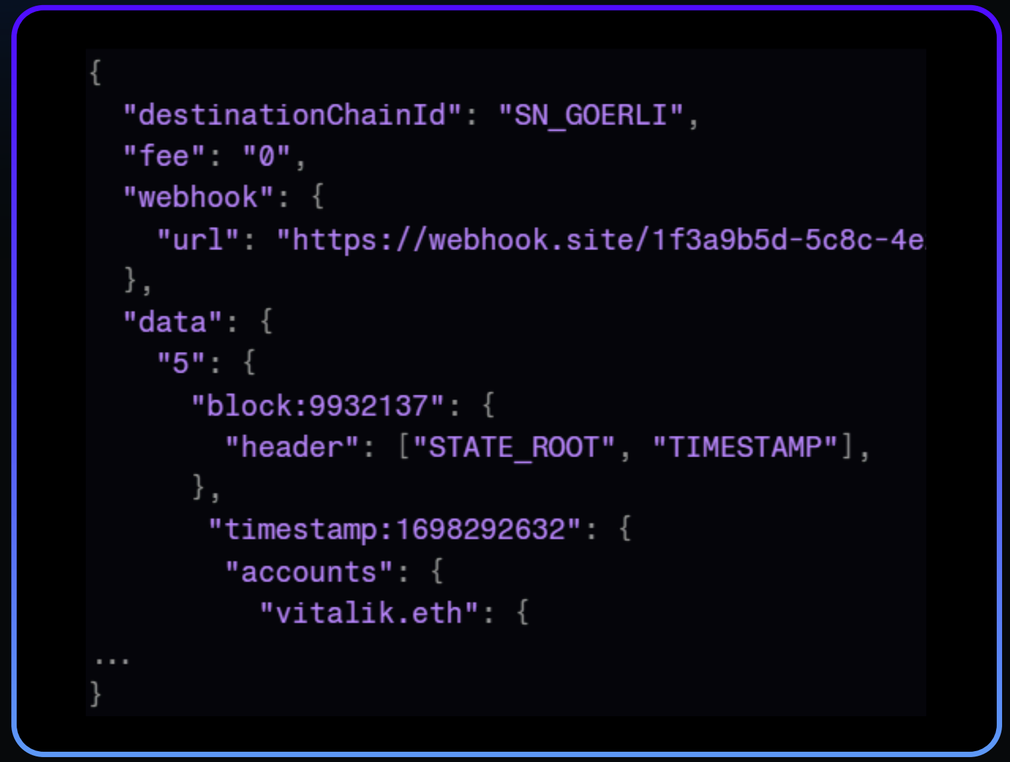
```
{
  "deliveryChainId": "SN_SEPOLIA",
  "sourceChainId": 11155111,
  "tasks": [
    {
      "datalakeType": "block_sampled",
      "datalake": {
        "blockRangeStart": 5515000,
        "blockRangeEnd": 5515029,
        "sampledProperty": "account.
            0x7f2c6f930306d3aa736b3a6c6a98f512f74036d4.balance"
      },
      "aggregateFnId": "count",
      "aggregateFnCtx": {
        "operatorId": "gt",
        "valueToCompare": 50
      }
    }
  ]
```

HERODOTUS

WHAT IS API?

API

The Storage Proof API enables developers to request storage proofs. There is no need to understand cryptography, zk proofs, circuits. The API mutualizes costs associated with generating storage proofs and saves developers significant time so they can focus on building.



HERODOTUS

WHAT IS TURBO?

TURBO

Herodotus Turbo is a smart contract interface for the Storage Proof API. Even though our API is simple, developers still need to think about its existence. Turbo abstracts away these complexities and enables smart contracts to make arbitrary on-chain data queries in only one line of code!

```
let dai_balance_on_mainnet = herodotus
    .read_variable(MAINNET_CHAIN_ID,
                   TARGETTED_BLOCK,
                   DAI_ADDR,
                   'balanceOf',
                    key);
```

HERODOTUS

WHAT IS TURBO?

TURBO

Herodotus Turbo is a smart contract interface for the Storage Proof API. Even though our API is simple, developers still need to think about its existence. Turbo abstracts away these complexities and enables smart contracts to make arbitrary on-chain data queries in only one line of code!

HERODOTUS

CHECK OUT HDP DOCS & REPO

DOCS

HDP REPO

HERODOTUS

THANK YOU!

@piapark_eth

@HerodotusDev

HERODOTUS.DEV