Auditing System by ISUNCLOUD

Currently, smart contracts are deployed using Remix, and data is input directly into the deployed smart contracts via Etherscan or Remix.

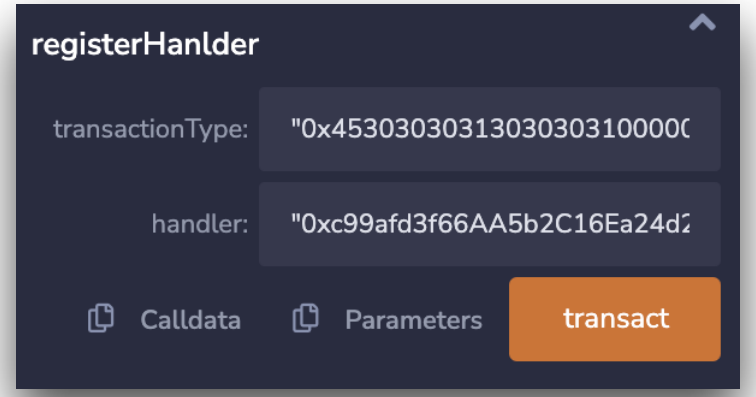
1. System Deployment:

Deploy the smart contracts in the following sequence:

1. parser.sol
2. reports.sol
3. TransactionContract.sol (parser address)
4. Handlers Contracts (TransactionContract address, parser address)
5. Calculating Contracts (TransactionContract address, Parser address , report address)
6. getTransactionTimeSpan.sol (transaction address, parser address, functioning contracts addresses)
7. Router contract(TransactionContract address, getTransactionTimeSpan address)

2. Operation Flow:

1. Users start by registering handlers using the registerHandlers function in the “**router**”, inputting the “**TransactionType**” (of type **bytes32**) and the “**handler’s"** address (of type **address**).

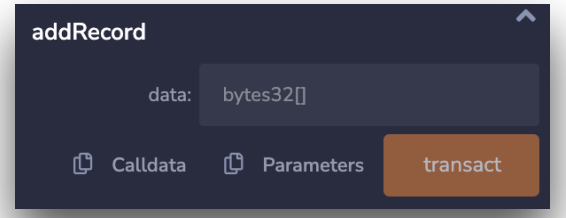


2. Record data using a **bytes32** array in the **addRecord** function of the“**router.sol”** contract, where each element has been multiplied by 10^18. The first element must be the eventID, and the second should specify the event type. Users must omit the timestamp column to prevent fraudulent events; the system will automatically record the current time.”

For example, for the following transaction, the format of the array should be:

[
0x0000000000000000000000000000000000000000000000000000000006669727374,
0x0000000000000000000000000000000000000000000000004530303031303030 31,
0x0000000000000000000000000000000000000000000000021e19e0c9bab2400000,
0x0000000000000000000000000000000000000000000000008ac7230489e80000,
0x0000000000000000000000000000000000000000000000000de0b6b3a7640000,
0x0000000000000000000000000000000000000000000000000e043da617250000
]

	Events
出入金	E00010001 EP001 用戶入金 10000 USDT EP002 內扣手續費 10 USDT EP003 外扣手續費 1 USDT EP004 交易時間 t1 EP005 交易時匯率 1 USDT = 1.01 USD



Which in decimal:

[

“first”,	***(eventID)***
”E00010001”,	***(transactionType)***
10000000000000000000,	***(EP001)***
1000000000000000000,	***(EP002)***
1000000000000000000,	***(EP003)***
1010000000000000000	***(EP005)***

]

, notice that every number has been multipulied by 10^18.

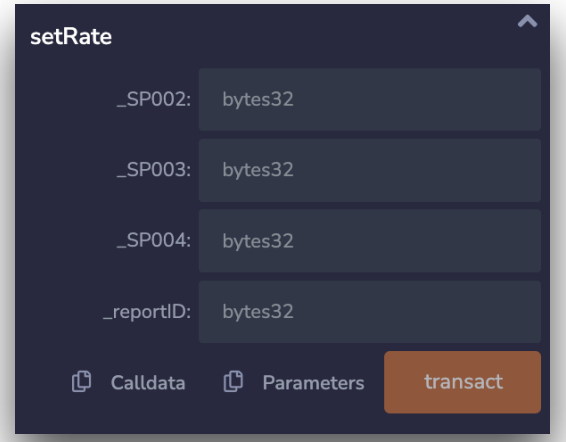
3. In order to create report(s) in a time span, the users first set rates and reportID on "**setRate**" function in "**router.sol**". For example, for the following example, the user should input a bytes32 array as the following format:

[
0x0000000000000000000000000000000000000000000000000dbd2fc137a30000,
0x0000000000000000000000000000000000000000000000056bc75e2d631000000,
0x0000000000000000000000000000000000000000000000000000000000006590,
0x66697273745f7265706f7274000000000000000000000000000000000000000000
]

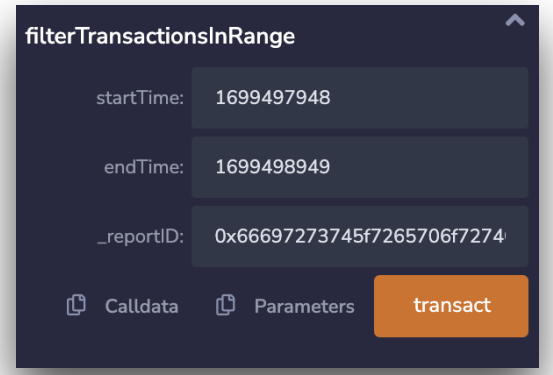
Which in decimal stands for:

[
990000000000000000, (SP002)
1600000000000000000000, (SP003)
2600000000000000000000, (SP004)
"first_report" (reportID)
]

Settlement
SP001 建表時間 t2
SP002 建表時匯率 1 USDT = 0.99 USD
SP003 建表時匯率 1 ETH = 1600 USD
SP004 建表時匯率 1 BTC = 26000 USD



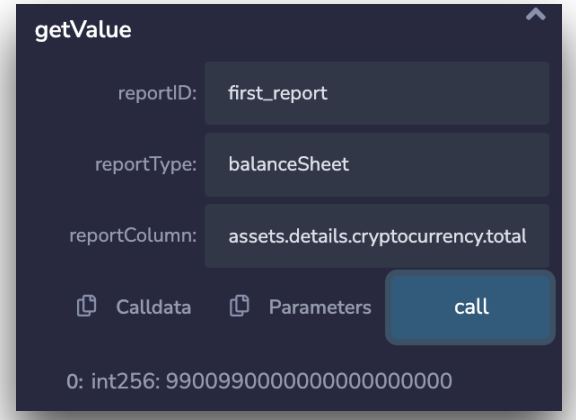
The users then interact with the `**filterTransactionsInRange**` function by inputing **startTime(uint256***)*, ***endTime(uin256***), and ***reportID(bytes32)*** to set a specific time span. This function employs the `**eventID**` as primary keys and using `**reportID**` to organize reports under the `**reportID**`, preventing any disarray. It then retrieves and returns the transactions that occurred within the specified time span, without providing the full dataset.(S07 - S08)

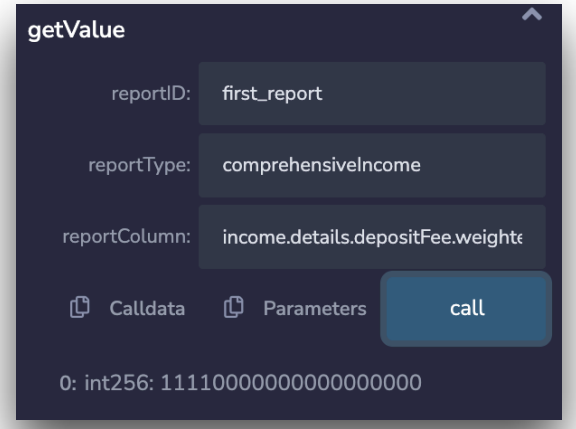


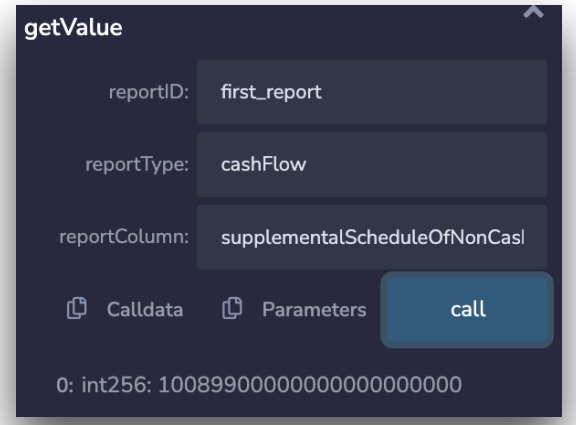
4. Then the system will pass those transactions which had been filtered in a time span to calculating functions (use eventID to determine which one), the calculating functions first use "**Iparser.sol**" function to change **bytes32** into string or **int256** and then calculate data with planned formula.

5. After calculating calculating get a 3D array from "**Ireports.sol**", and then add results into the respective column.

6. We can check the numbers is correct or not by calling the function **getValue(reportID, reportType, reportColumn)** in reports.sol. There are three options to fill in the reportType, '**balanceSheet**', '**comprehensiveIncome**', '**cashFlow**'. Then input the '**reportColumn**' to check the respective column.







3. Interact with smart contracts on Ethereum and checking results

1. First, users should download and deploy the “hardhat” environment locally .
2. Run “***npx hardhat run transformReportAPI.js***”.
3. This program would help users to interact with reports.sol on ethereum by getting reports data.
4. The program will parse the raw data into planned API format.


```
"success": true,
"code": "00000000",
"reason": "ERROR_MESSAGE.SUCCESS",
"data": {
  "id": "first_report",
  "date": 1699411588439,
  "totalAssetsFairValue": 9900.99,
  "totalLiabilitiesAndEquityFairValue": 9900.99,
  "assets": {
    "totalAmountFairValue": 9900.99,
    "details": {
      "accountsReceivable": {
        "totalAmountFairValue": 0,
        "weightedAverageCost": "?",
        "breakdown": {
          "ETH": {
            "currencyType": "CRYPTOCURRENCY",
            "name": "ETH",
            "amount": "?",
            "weightedAverageCost": "?",
            "fairValue": "?"
          },
          "BTC": {
            "currencyType": "CRYPTOCURRENCY",
            "name": "BTC",
            "amount": "?",
            "weightedAverageCost": "?",
            "fairValue": "?"
          },
          "USDT": {
            "currencyType": "CRYPTOCURRENCY",
            "name": "USDT",
            "amount": 0,
            "weightedAverageCost": "?",
            "fairValue": 0
          },
          "USD": {
            "currencyType": "FIAT",
            "name": "USD",
            "amount": "?",
            "weightedAverageCost": "?",
            "fairValue": "?"
          }
        }
      },
```

4. Future Plans:

1. Develop a standardized interface for calculation functions and store it in the interfaces file.
2 . Code each function up to Column 7, encompassing all types of deposits and withdrawals.
3. Implement Mistake Proofing mechanisms, such as require statements.
4. Enhance calculation accuracy, given that Solidity lacks a float variable type.
5. Optimize gas fees through careful data structure design.
6. Create an interface for reports.sol and import only the interface.