EUREKA

🔴 Prerequisites

❗IMPORTANT, please complete the following❗

1. Read the Polkadot x EasyA notion page, this should be like your “rule book” for the hackathon, refer to this when in doubt
2. Complete the Polkadot crash course on the EasyA app, you can get the app from easya.io
3. Setup the required software on your computer:

    1. VSCode*
    2. Github*
    3. Git*
    4. Metamask wallet*
    5. TRAC Test tokens*1
    6. WND Test tokens, WND Config*
    7. ChatGPT*
    8. Lovable*
    9. Metamask tools**
    10. Docker***
    11. Canva (frontend requirement)
    12. NPM/NVM (frontend requirement)
    13. Python (backend requirement)
    14. PAPI (frontend requirement)
    15. Substrate API Sidecar (frontend requirement)
    16. Polkadot SDK (backend requirement) // To build your own blockchain
    17. Subxt (backend requirement) - RUST **
    18. Pallet-Revive (backend requirement)
    19. DKG SDK (backend requirement)
    20. Cursor (optional but useful for development)

    * required;
    *1 NOTE this will show as 0 in metamask wallet even after you have tokens due to UX bug with Metamask
    ** need both Metamask SDK and Wallet API
    *** only required if you want to run the app on your laptop (other option is for us to deploy to a server)

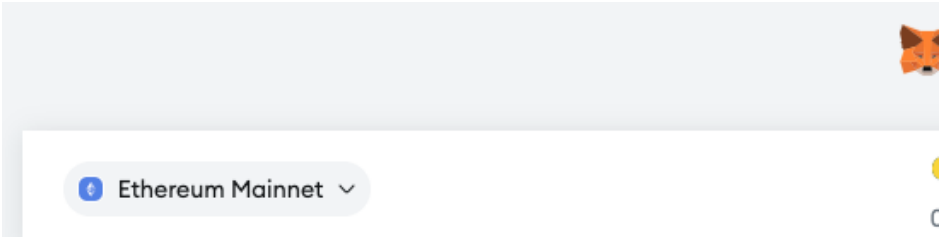
Please put your wallet Addresses below, we all need to get test tokens to test our app so create a test metamask account (DON'T USE YOUR MAIN CRYPTO WALLET)

Name	WND (Polkadot) Address	TRAC (NeuroWeb) Address	ETH (Sepolia) Address
Daniel	0x65B85C546fbb0211aCd676a852397588360fAC37	0x65B85C546fbb0211aCd676a852397588360fAC37	0x65B85C546fbb0211aCd676a852397588360fAC37
Yuri			
Jack	0xeb202166015976623cDe87d4f2cAeF41abdb7177		0xeb202166015976623cDe87d4f2cAeF41abdb7177
Gustavo	0x5a7Ef77ED01cd50e094F6cbc65e3416010a0B6B2		0x5a7Ef77ED01cd50e094F6cbc65e3416010a0B6B2
Faris			

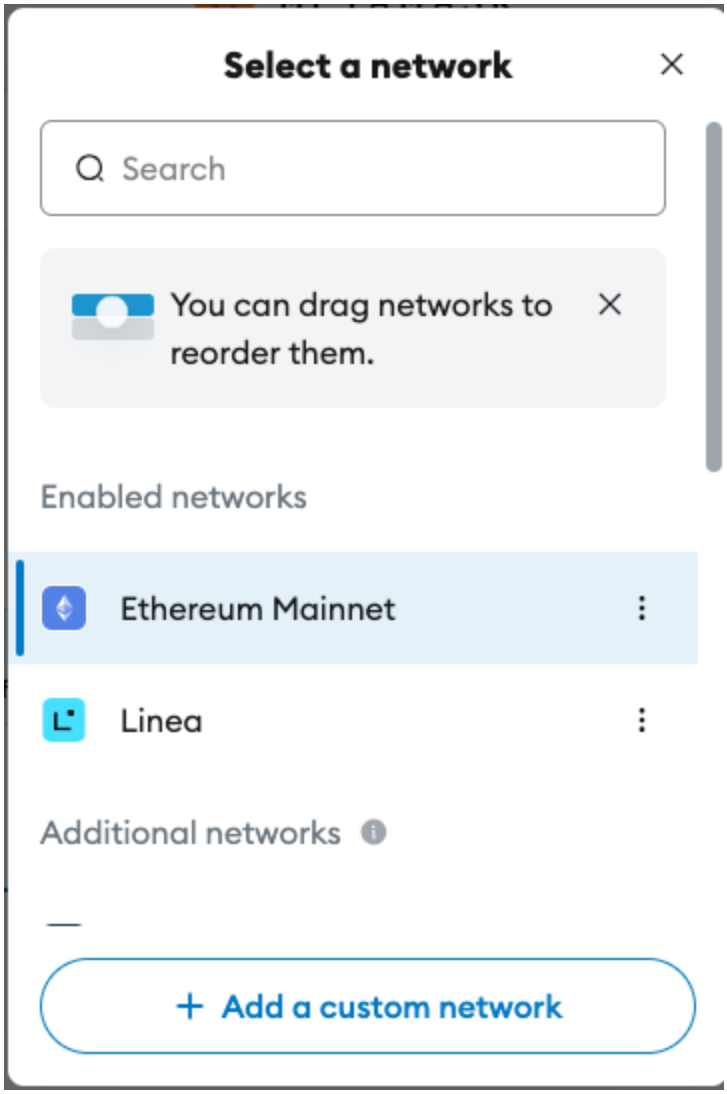
Guide to add Tokens to Meta Mask Wallet

Follow these steps to add WND to Meta Mask Wallet:

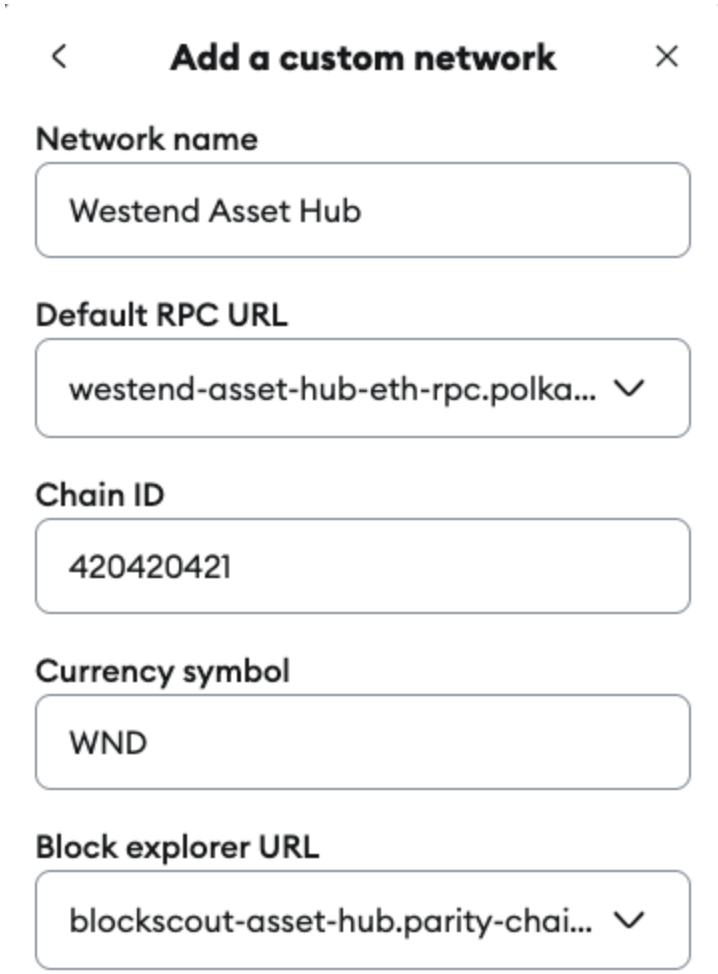
1. Click on "Ethereum Mainnet" button



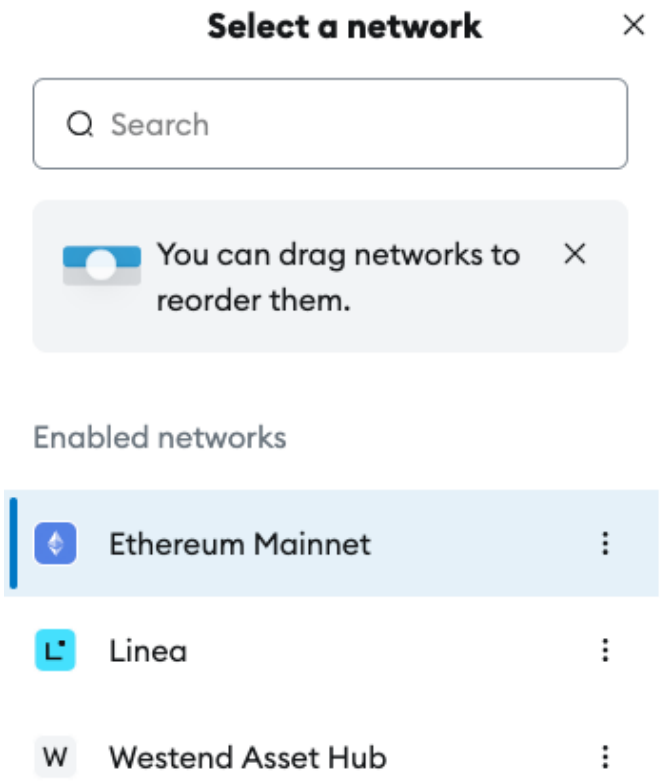
2. Click on "Add a custom network" button



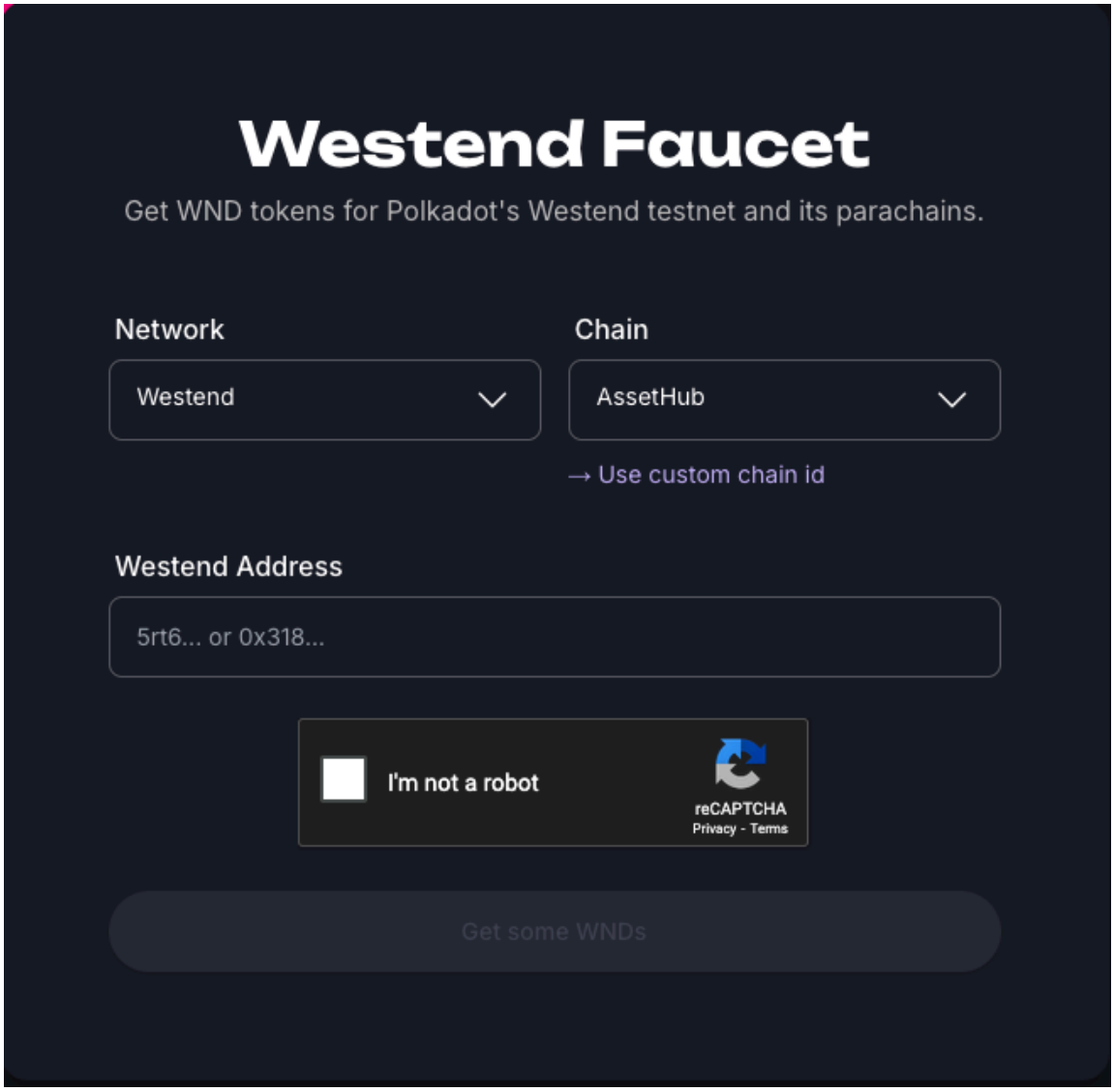
3. Fill out the details using https://contracts.polkadot.io/connect-to-asset-hub/


Add a custom network
Network name
Westend Asset Hub
Default RPC URL
westend-asset-hub-eth-rpc.polka...
Chain ID
420420421
Currency symbol
WND
Block explorer URL
blockscout-asset-hub.parity-chai...

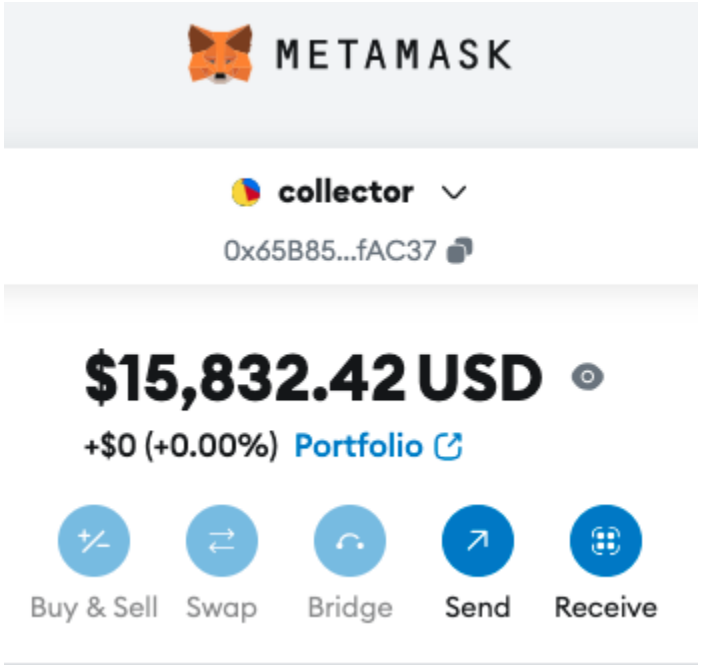
4. Now select “Westend Asset Hub” network from your wallet



5. Fill out the address on [West End Faucet]

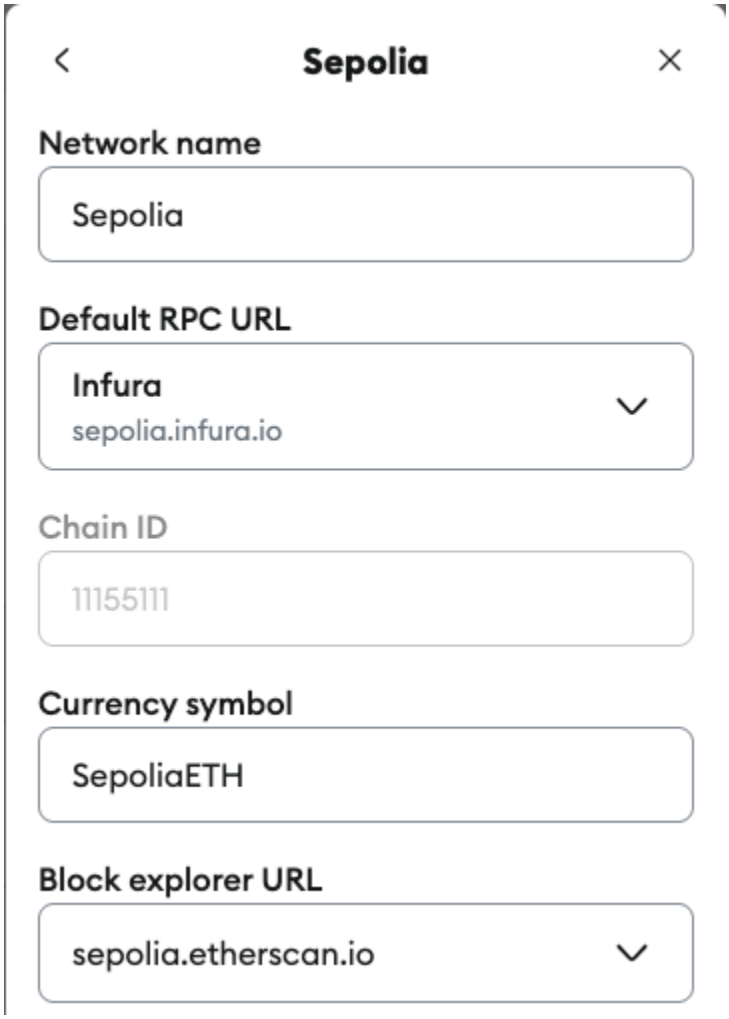


6. Congrats you're rich!



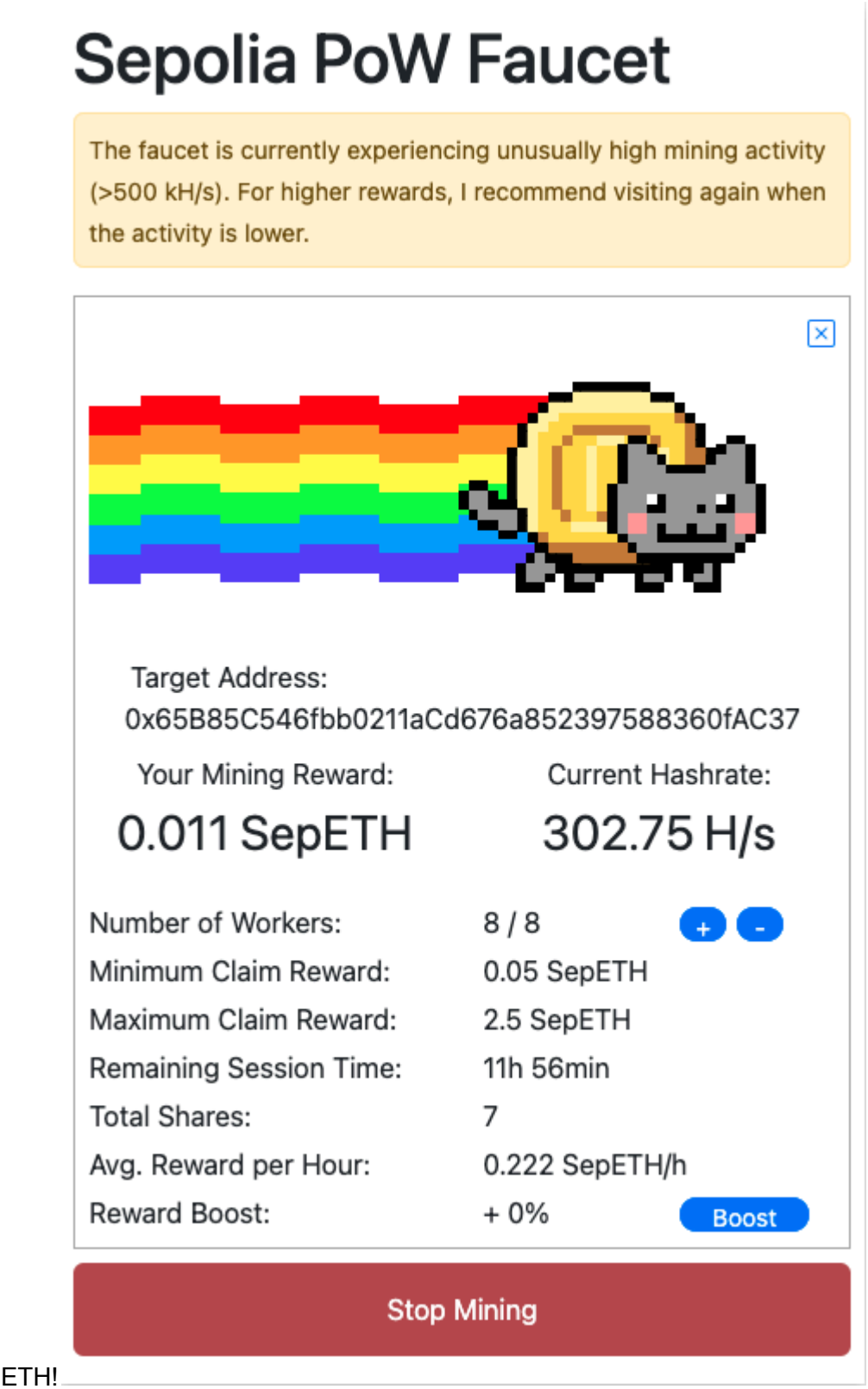
Follow these steps to add test ETH to your wallet

1. Add custom network and fill it out with Sepolia Faucet info

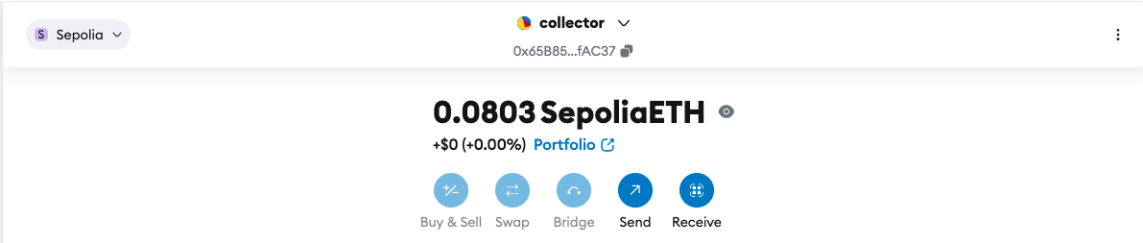


2. Get test tokens from [Sepolia Test Faucet]

3. Once you've put your wallet address and filled out the captcha you should start receiving ETH!

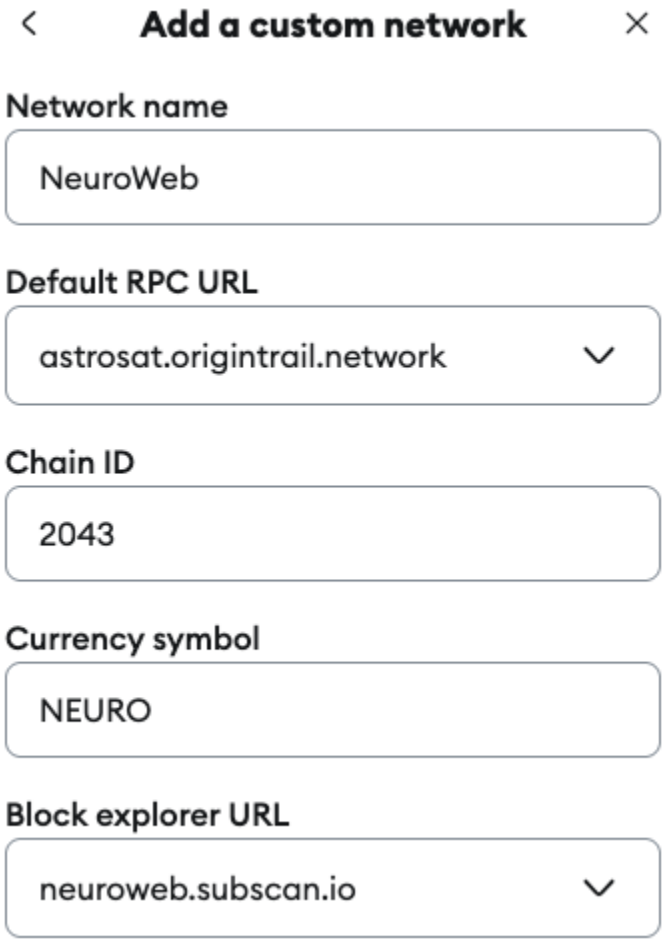


Note, you won’t receive any tokens until you click “Stop Mining” and then click on “Claim Reward”!
You should now have some Sepolia in your wallet!



Adding Neuro Web Tokens is more involved, follow the steps below to get them:

1. Add a new network with the following properties



2. Then You need access to the [OriginTrail discord server] , then in #faucet-bot channel type !help command for info

3. In #faucet-bot channel execute the following command: !fundme_neuroweb <wallet_address>
4. After some time you should have tokens in your wallet

Problem Statement

1. Alice receives a fake bill from Bob for her utility bill, to her ADDRESS with an AMOUNT for company X
2. Alice believes this to be the true utility bill and she pays the money but instead of it going to company X it goes Bob, so Alice has fallen victim to FRAUD
3. We need to solve this problem, but instead of trying to do like a bank where we take action after the FRAUD we try to prevent it from happening
4. We want to solve this using blockchain but the problem is blockchain is public, so instead of storing the private data on chain we just store a hash
5. We generate the appropriate hash using the smart contract

Smart Contract

The smart contract is a self-executing program stored on a blockchain that automatically enforces and executes the terms of an agreement once predefined conditions are met

Properties of smart contract:

- 🧠 **Logic-Based**: It contains the business logic (like “if X happens, then do Y”).
- 🔒 **Trustless**: No need for a middleman — the blockchain ensures everything runs as coded.
- 📜 **Immutable**: Once deployed, it can’t be altered (unless there’s an upgrade mechanism built in).
- 🧾 **Transparent**: Everyone can see the code and how it operates on public blockchains.

NFTs

We can thus represent the given utility bill defined in the problem statement as a Non Fungible Token (NFT). This is because we want each contract to be unique, non replicable and thus using NFTs to represent utility bills is a perfect use case for NFTs.

ERC-721

Ethereum defines NFTs using the ERC-721 standard, this is the specific standard we’ll be using in our solidity smart contracts.

ERC-721 Key Properties

The key properties of ERC-721 are summarized below

- 🎯 **Uniqueness**:
  Each token is distinct and has a unique tokenId. No two are the same.
- 👤 **Ownership**:
  Tracks who owns each token using the ownerOf(tokenId) function.
- 🔄 **Transferability**:
  Tokens can be transferred between addresses using transferFrom() or safeTransferFrom().
- 📦 **Metadata**:
  Each token can point to off-chain metadata via tokenURI(tokenId) (e.g., artwork, attributes).
- 🛡️ **Approval System**:
  Owners can approve others to manage their tokens (approve() and setApprovalForAll()).
- 🧩 **Interoperability**:
  Recognized by marketplaces and dApps that support the ERC-721 standard.

You can view the full properties from EIP-721

Utility NFT (UNFT)

How do we use NFTs to solve the problem defined in the problem statement? Well one solution would be to provide the information required for each contract in the metadata attribute; the metadata would be stored off chain and only the unique identifier (hash) of the contract and the NFT would be stored on chain.

To solve our problem the company issuing the invoice mints an NFT, this NFT has a hash which is published on-chain. The metadata associated with the given NFT is stored off-chain, and is what contains the sensitive data associated with the bill.

How can the customer access this metadata without repeating the problem defined in the problem statement? Well there are two ways: one would be for the user to receive a code from the company for each bil via email, this code would then be used in the API to access the metadata; the second more complex solution would be to have a keypair associated with each company the customer has a bill from, then each time the company issues a bill they send the metadata encrypted and then the customer can decrypt it using their key (this avoid the need to issue codes each time a bill is issued which reduces the attack vector). We should initially do this off-chain, using the former suggested solution, (due to the time constraints of the hackathon) however as future steps we could look into implementing more secure methods such as ZKPs and meta data storage using DKG.

Suggested attributes to include in the metadata are:

- Invoice number: unique identifier for the invoice
- Invoice date: the date the invoice was issued
- Payment due date: the date the payment is expected to be made
- Seller/supplier information: name, address, and contact information
- Buyer/customer information: name, address, and contact information
- Description of goods or services: a clear and concise description of what is being sold
- Quantity: the number of items or services being invoiced
- Unity Price: the price of each individual item or service
- Subtotal: the total cost of the goods or services before any taxes or discounts
- Taxes: any applicable taxes, such as VAT
- Discounts: any applicable discounts
- Total amount due: the final amount the customer owes
- Payment terms: specify the timeframe for payment
- Accepted payment methods: methods that the seller accepts
- Company logo
- Additional information of instructions

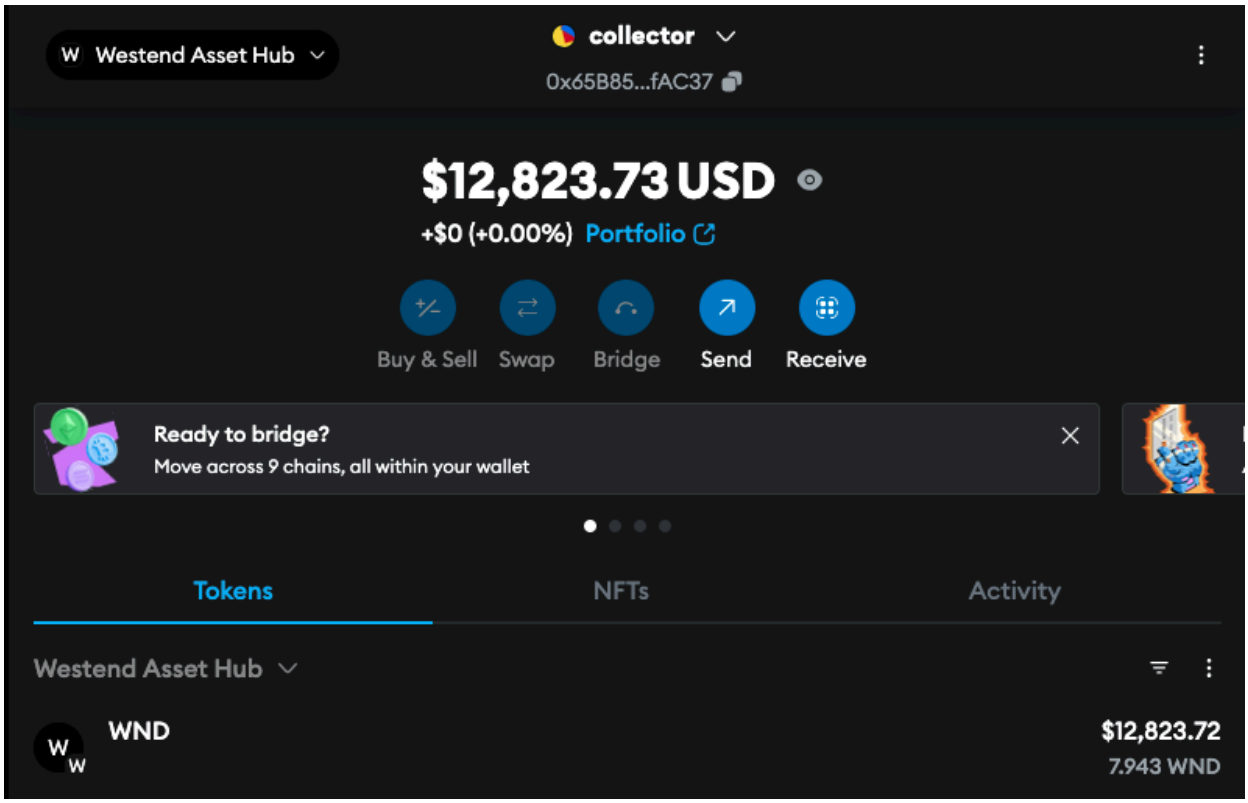
Since all of this is private sensitive data, it's important that this information is stored, retrieved and transferred in a safe, trustable and verifiable way. Using traditional web2 methods where the data is stored off-chain in private servers and queried with API is the only feasible option in the given time window, however we plan to move to more secure solutions using ZKPs and DKG in the future.

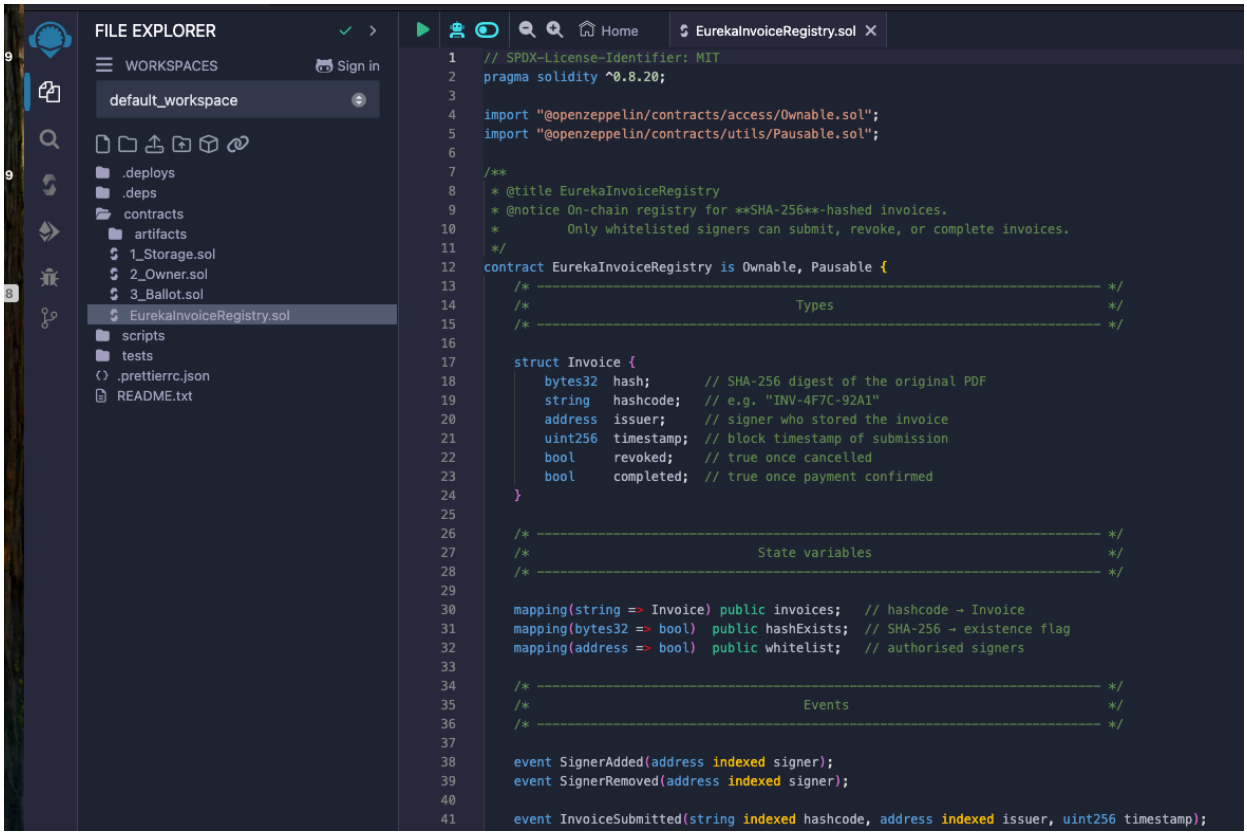
Implementation

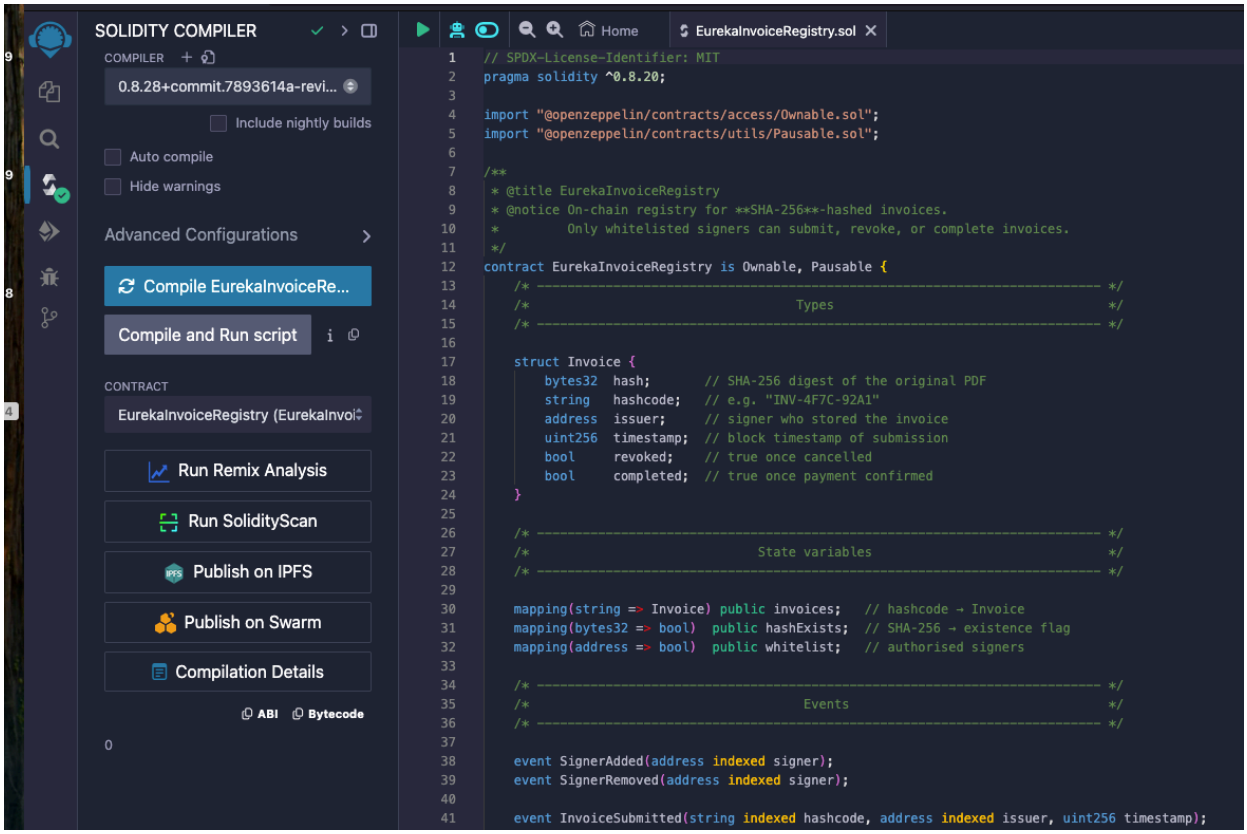
1. Write smart contract, we'll use solidity because of familiarity (DONE)
2. Deploy the contract to Westend (DONE)
3. Execute contract methods to publish invoice on-chain (DONE)
4. Get data from blockchain using API (PENDING)
5. Show results from API calls to the user, so that they can see that their PDF has been verified using the blockchain (PENDING)

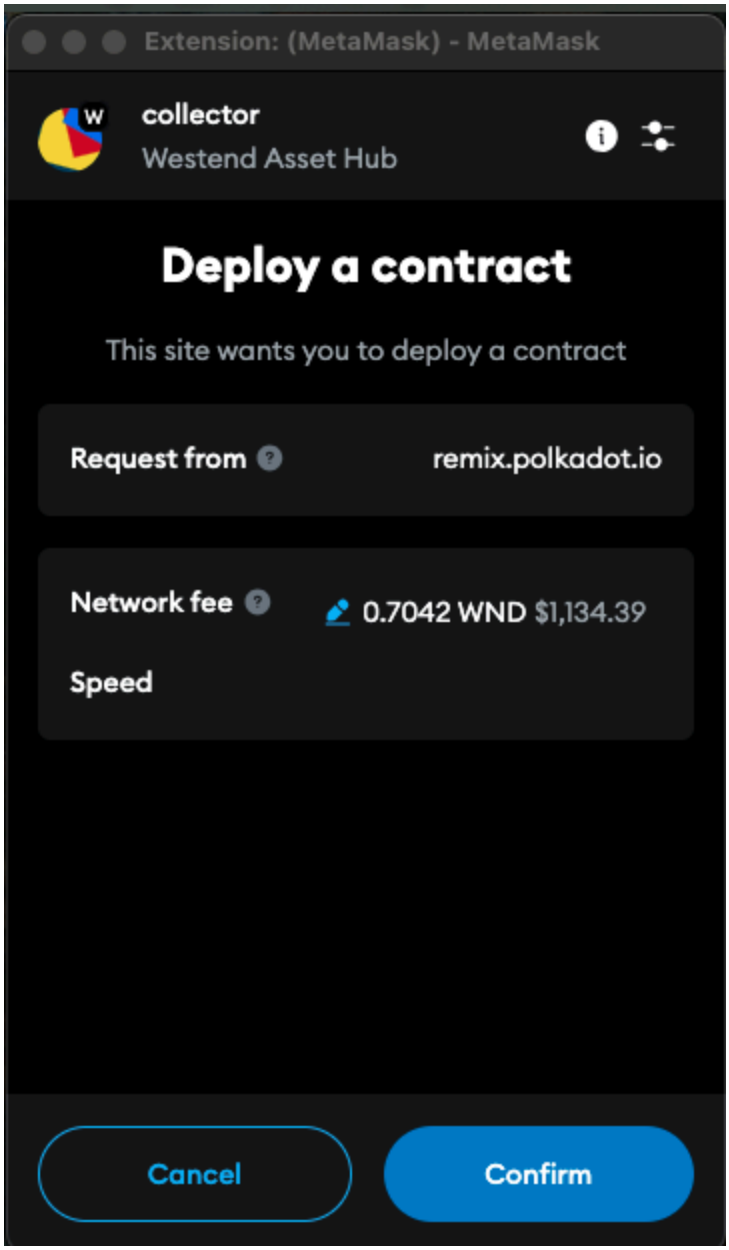
Screenshot Testing Proofs

Screen shots to keep track and provide proof of testing!

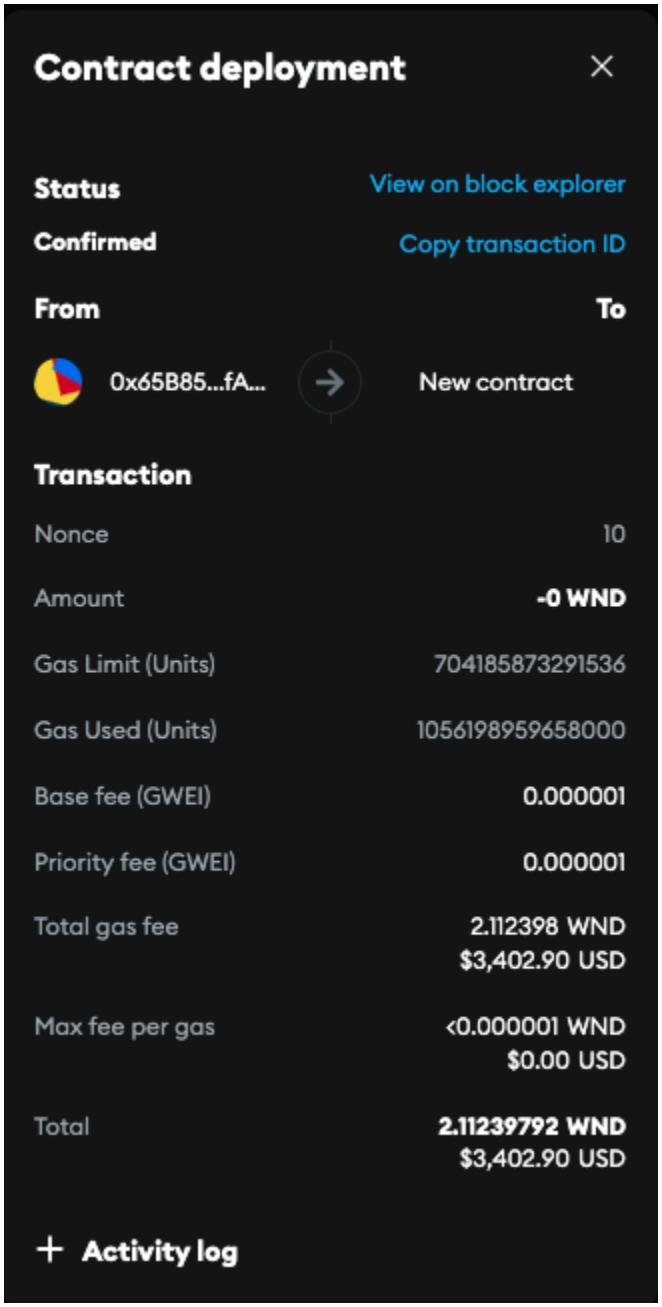

Westend Asset Hub
collector
0x65B85...fAC37
$12,823.73 USD
+$0 (+0.00%) Portfolio
Buy & Sell
Swap
Bridge
Send
Receive
Ready to bridge?
Move across 9 chains, all within your wallet
Tokens
NFTs
Activity
Westend Asset Hub
WND
$12,823.72
7.943 WND

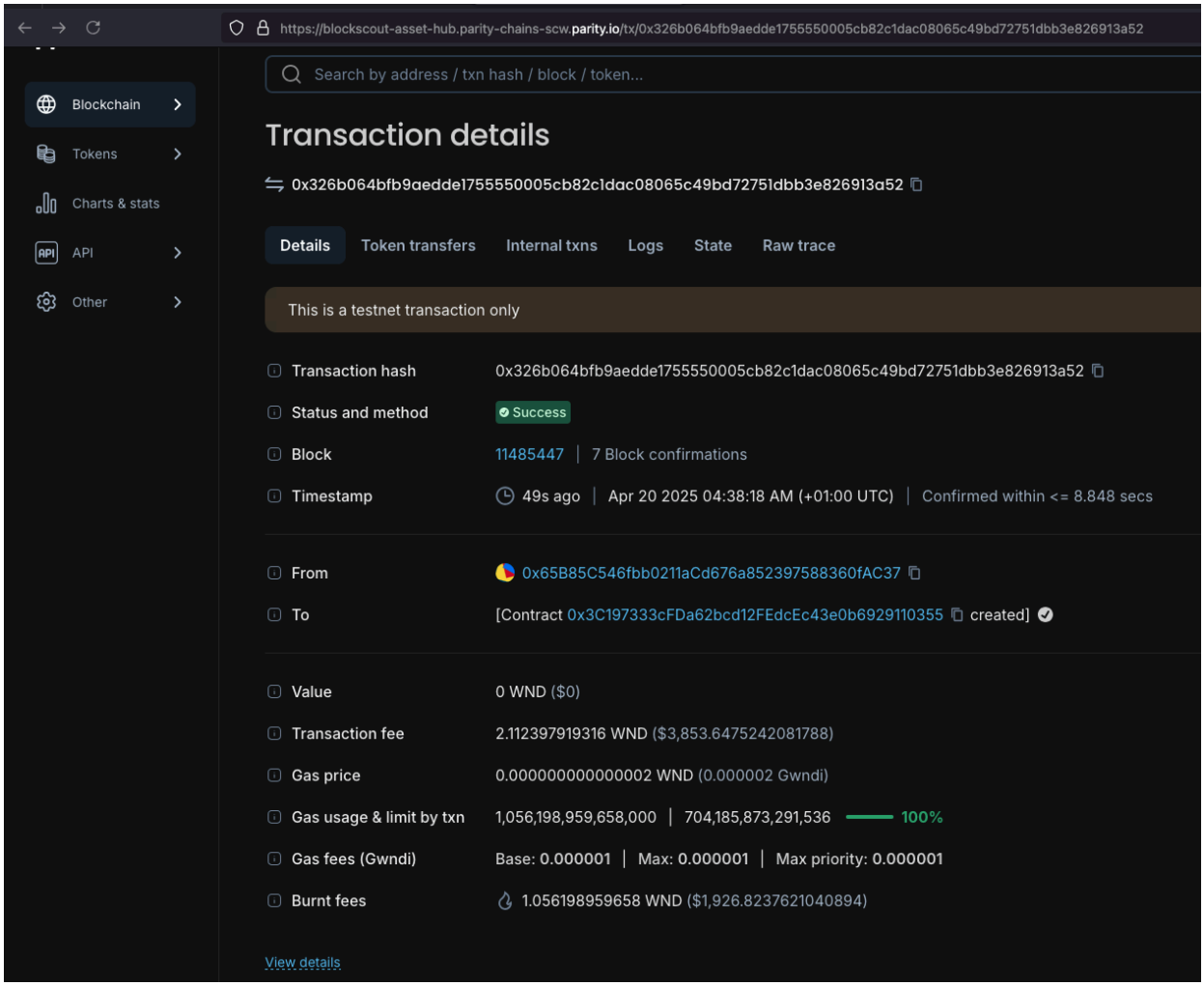

FILE EXPLORER
WORKSPACES
Sign in
default_workspace
.deploys
.deps
contracts
artifacts
1_Storage.sol
2_Owner.sol
3_Ballot.sol
EurekaInvoiceRegistry.sol
scripts
tests
.prettierrc.json
README.txt
Home
EurekaInvoiceRegistry.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;
import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/utils/Pausable.sol";
/**
* @title EurekaInvoiceRegistry
* @notice On-chain registry for **SHA-256**-hashed invoices.
* Only whitelisted signers can submit, revoke, or complete invoices.
*/
contract EurekaInvoiceRegistry is Ownable, Pausable {
/* ---------- Types ---------- */
struct Invoice {
bytes32 hash; // SHA-256 digest of the original PDF
string hashcode; // e.g. "INV-4F7C-92A1"
address issuer; // signer who stored the invoice
uint256 timestamp; // block timestamp of submission
bool revoked; // true once cancelled
bool completed; // true once payment confirmed
}
/* ---------- State variables ---------- */
mapping(string => Invoice) public invoices; // hashcode → Invoice
mapping(bytes32 => bool) public hashExists; // SHA-256 → existence flag
mapping(address => bool) public whitelist; // authorised signers
/* ---------- Events ---------- */
event SignerAdded(address indexed signer);
event SignerRemoved(address indexed signer);
event InvoiceSubmitted(string indexed hashcode, address indexed issuer, uint256 timestamp);

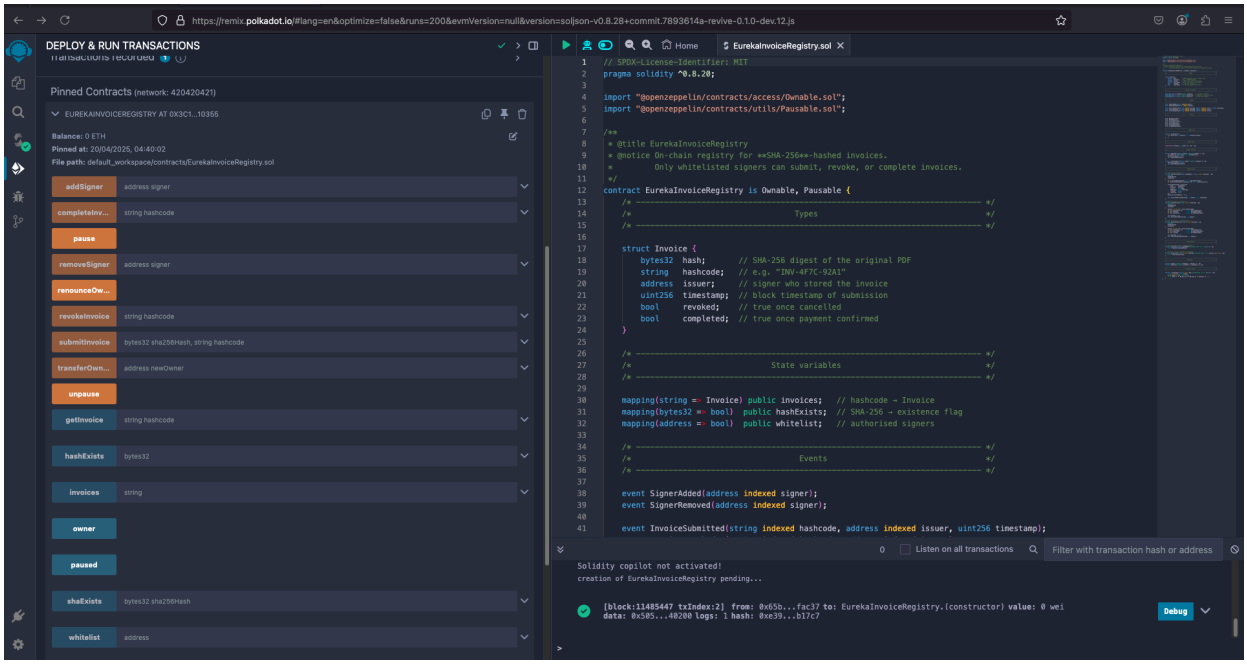

SOLIDITY COMPILER
COMPILER
0.8.28+commit.7893614a-revi...
Include nightly builds
Auto compile
Hide warnings
Advanced Configurations
Compile EurekaInvoiceRe...
Compile and Run script
CONTRACT
EurekaInvoiceRegistry (EurekaInvoi
Run Remix Analysis
Run SolidityScan
Publish on IPFS
Publish on Swarm
Compilation Details
ABI
Bytecode
0
Home
EurekaInvoiceRegistry.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;
import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/utils/Pausable.sol";
/**
* @title EurekaInvoiceRegistry
* @notice On-chain registry for **SHA-256**-hashed invoices.
*   Only whitelisted signers can submit, revoke, or complete invoices.
*/
contract EurekaInvoiceRegistry is Ownable, Pausable {
/* ---------- Types ---------- */
struct Invoice {
bytes32 hash; // SHA-256 digest of the original PDF
string hashcode; // e.g. "INV-4F7C-92A1"
address issuer; // signer who stored the invoice
uint256 timestamp; // block timestamp of submission
bool revoked; // true once cancelled
bool completed; // true once payment confirmed
}
/* ---------- State variables ---------- */
mapping(string => Invoice) public invoices; // hashcode → Invoice
mapping(bytes32 => bool) public hashExists; // SHA-256 → existence flag
mapping(address => bool) public whitelist; // authorised signers
/* ---------- Events ---------- */
event SignerAdded(address indexed signer);
event SignerRemoved(address indexed signer);
event InvoiceSubmitted(string indexed hashcode, address indexed issuer, uint256 timestamp);


Extension: (MetaMask) - MetaMask
collector
Westend Asset Hub
Deploy a contract
This site wants you to deploy a contract
Request from
remix.polkadot.io
Network fee
0.7042 WND $1,134.39
Speed
Cancel
Confirm


Apr 20, 2025
Contract deployment
Confirmed
-0 WND
-$0.00 USD


Contract deployment
Status
View on block explorer
Confirmed
Copy transaction ID
From
To
0x65B85...fA...
New contract
Transaction
Nonce
10
Amount
-0 WND
Gas Limit (Units)
704185873291536
Gas Used (Units)
1056198959658000
Base fee (GWEI)
0.000001
Priority fee (GWEI)
0.000001
Total gas fee
2.112398 WND
$3,402.90 USD
Max fee per gas
<0.000001 WND
$0.00 USD
Total
2.11239792 WND
$3,402.90 USD
Activity log

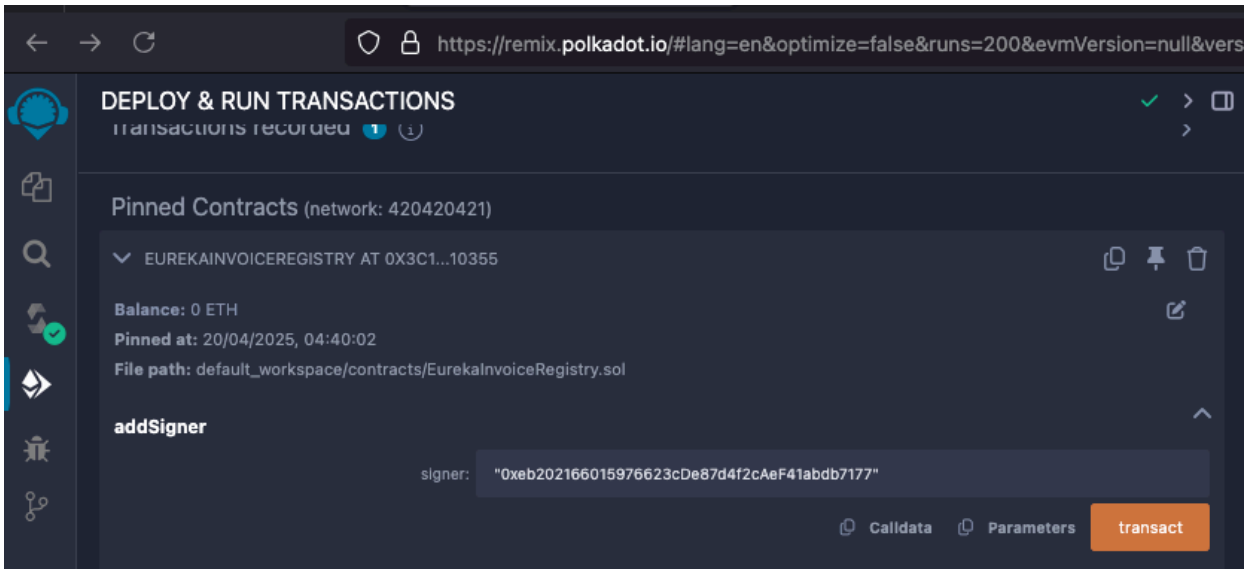

https://blockscout-asset-hub.parity-chains-scw.parity.io/tx/0x326b064bfb9aedde1755550005cb82c1dac08065c49bd72751dbb3e826913a52
Search by address / txn hash / block / token...
Blockchain
Tokens
Charts & stats
API
Other
Transaction details
0x326b064bfb9aedde1755550005cb82c1dac08065c49bd72751dbb3e826913a52
Details
Token transfers
Internal txns
Logs
State
Raw trace
This is a testnet transaction only
Transaction hash
0x326b064bfb9aedde1755550005cb82c1dac08065c49bd72751dbb3e826913a52
Status and method
Success
Block
11485447 | 7 Block confirmations
Timestamp
49s ago | Apr 20 2025 04:38:18 AM (+01:00 UTC) | Confirmed within <= 8.848 secs
From
0x65B85C546fbb0211aCd676a852397588360fAC37
To
[Contract 0x3C197333cFDa62bcd12FEdcEc43e0b6929110355 created]
Value
0 WND ($0)
Transaction fee
2.112397919316 WND ($3,853.6475242081788)
Gas price
0.000000000000002 WND (0.000002 Gwndi)
Gas usage & limit by txn
1,056,198,959,658,000 | 704,185,873,291,536 100%
Gas fees (Gwndi)
Base: 0.000001 | Max: 0.000001 | Max priority: 0.000001
Burnt fees
1.056198959658 WND ($1,926.8237621040894)
View details

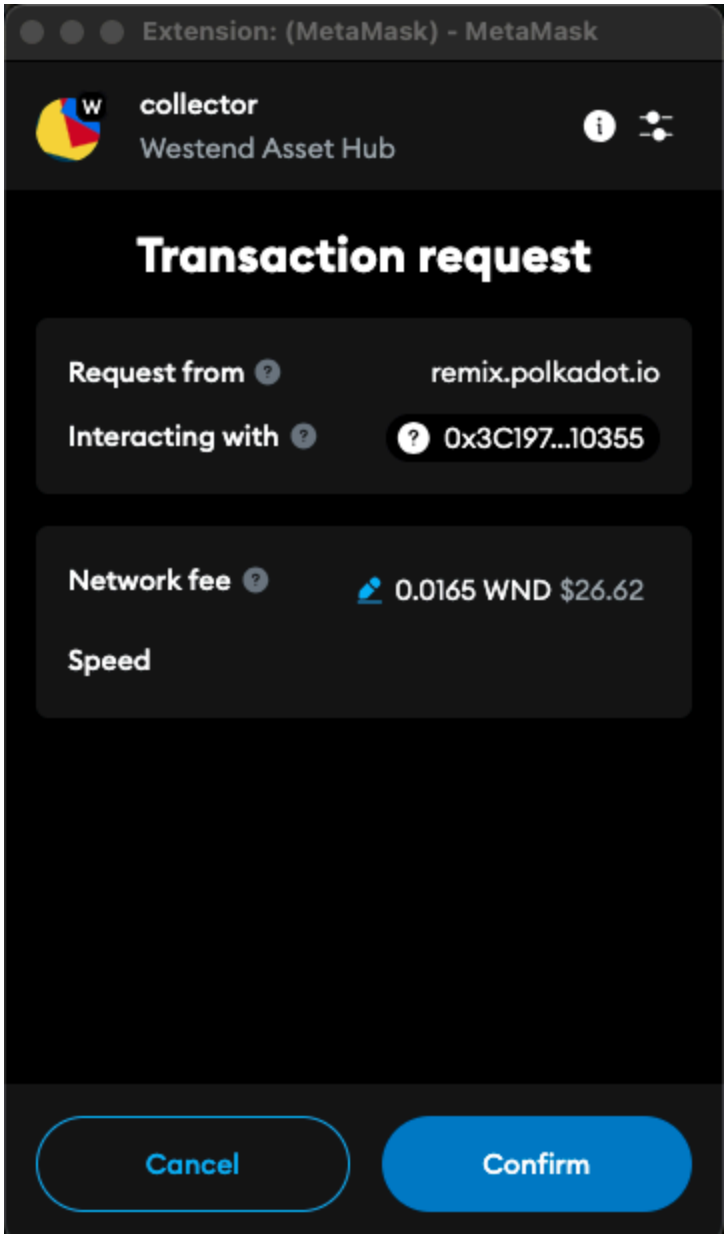

https://remix.polkadot.io/#lang=en&optimize=false&runs=200&evmVersion=null&version=soljson-v0.8.28+commit.7893614a-revive-0.1.0-dev.12.js
DEPLOY & RUN TRANSACTIONS
Pinned Contracts (network: 420420421)
EUREKAINVOICEREGISTRY AT 0X3C1...10355
addSigner
completeInv...
pause
removeSigner
renounceOw...
revokeInvoice
submitInvoice
transferOwn...
unpause
getInvoice
hashExists
invoices
owner
paused
shaExists
whitelist
Home
EurekaInvoiceRegistry.sol
// SPDX-License-Identifier: MIT
pragma solidity ^0.8.20;
import "@openzeppelin/contracts/access/Ownable.sol";
import "@openzeppelin/contracts/utils/Pausable.sol";
/**
* @title EurekaInvoiceRegistry
* @notice On-chain registry for «SHA-256»-hashed invoices.
*   Only whitelisted signers can submit, revoke, or complete invoices.
*/
contract EurekaInvoiceRegistry is Ownable, Pausable {
/* ---------- Types ---------- */
struct Invoice {
bytes32 hash; // SHA-256 digest of the original PDF
string hashcode; // e.g. "INV-4F7C-92A1"
address issuer; // signer who stored the invoice
uint256 timestamp; // block timestamp of submission
bool revoked; // true once cancelled
bool completed; // true once payment confirmed
}
/* ---------- State variables ---------- */
mapping(string => Invoice) public invoices; // hashcode → Invoice
mapping(bytes32 => bool) public hashExists; // SHA-256 → existence flag
mapping(address => bool) public whitelist; // authorised signers
/* ---------- Events ---------- */
event SignerAdded(address indexed signer);
event SignerRemoved(address indexed signer);
event InvoiceSubmitted(string indexed hashcode, address indexed issuer, uint256 timestamp);
Listen on all transactions
Filter with transaction hash or address
Solidity copilot not activated!
creation of EurekaInvoiceRegistry pending...
[block:11485447 txIndex:2] from: 0x65b...fac37 to: EurekaInvoiceRegistry.(constructor) value: 0 wei
Debug

Testing contract methods

addSigner

Calldata: 0xeb12d61e000000000000000000000000eb202166015976623cde87d4f2caef41abdb7177
Parameters: 0x000000000000000000000000eb202166015976623cde87d4f2caef41abdb7177

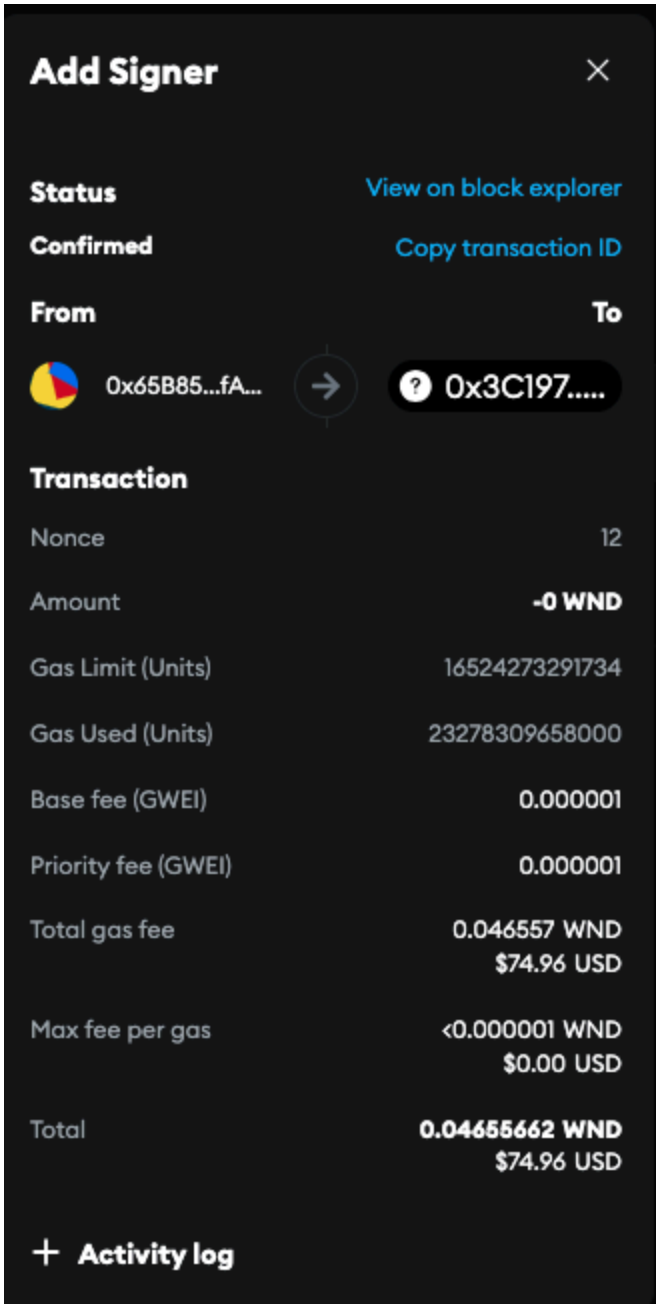


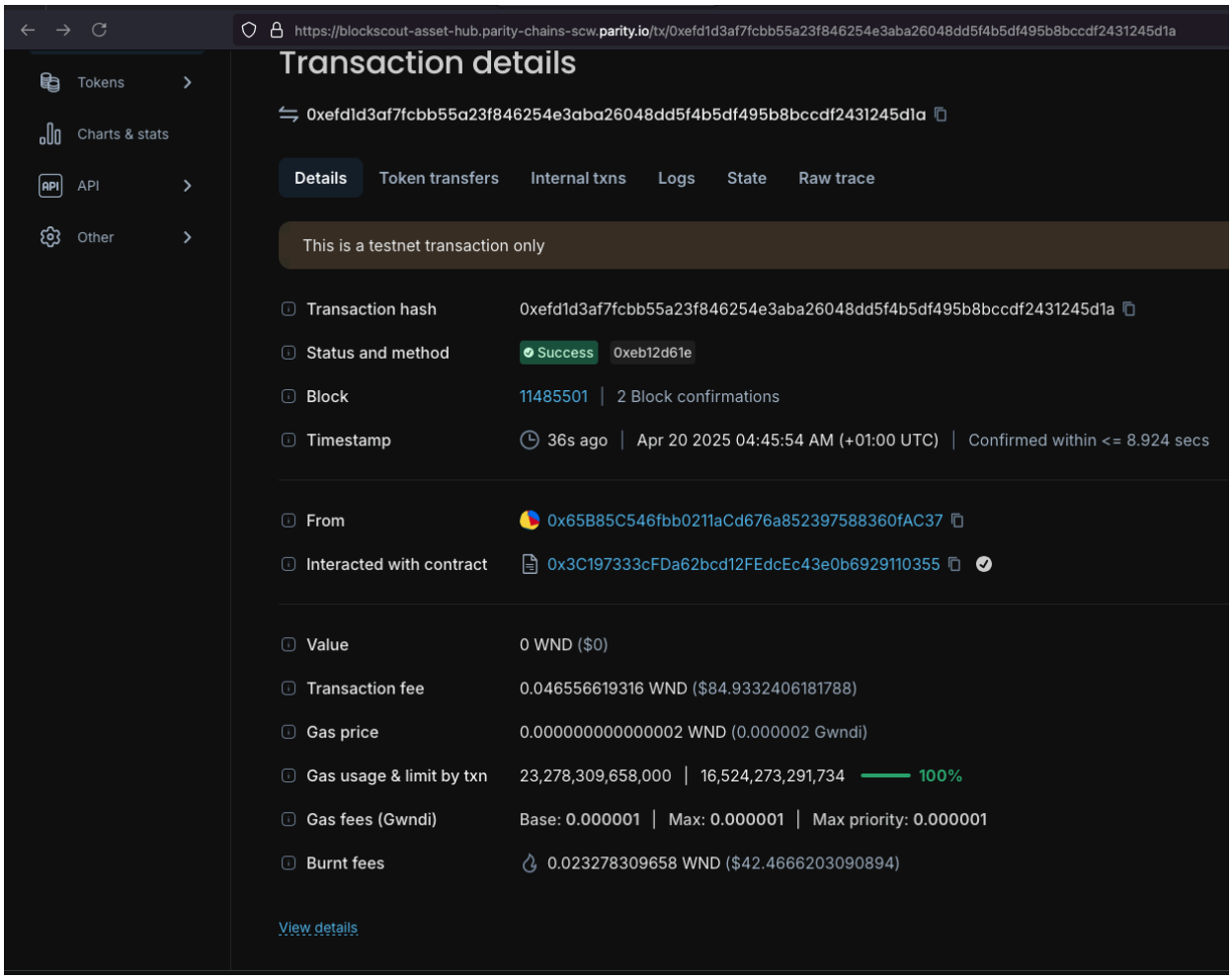


Apr 20, 2025

Add Signer
Confirmed

-0 WND
-$0.00 USD


Add Signer
Status
View on block explorer
Confirmed
Copy transaction ID
From
To
0x65B85...fA...
0x3C197.....
Transaction
Nonce
12
Amount
-0 WND
Gas Limit (Units)
16524273291734
Gas Used (Units)
23278309658000
Base fee (GWEI)
0.000001
Priority fee (GWEI)
0.000001
Total gas fee
0.046557 WND
$74.96 USD
Max fee per gas
<0.000001 WND
$0.00 USD
Total
0.04655662 WND
$74.96 USD
Activity log

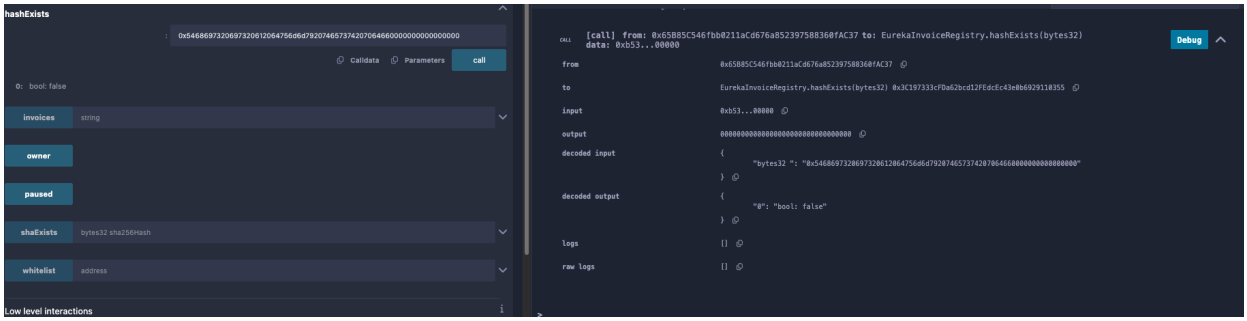


Need fill value out, cannot be zero for function calls that require gas fees

hashExists

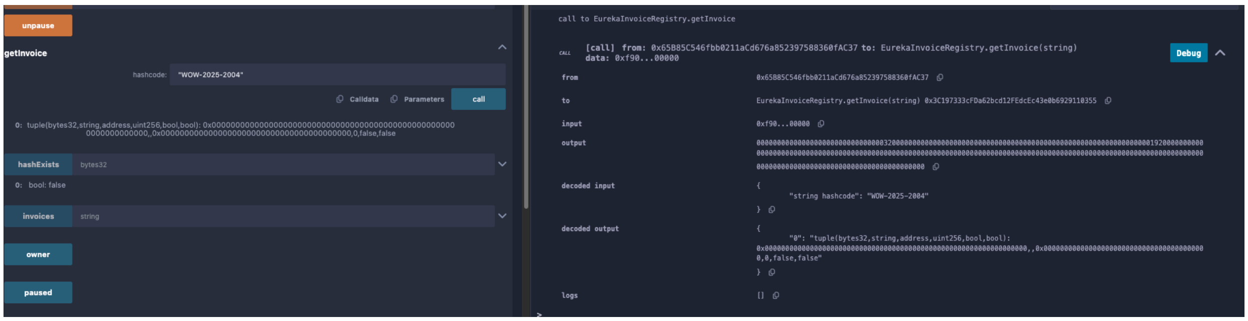
Calldata: 0xb535b5c3546869732069732061206475 6d6d792074657374207064660000000000000000

Parameters: 0x5468697320697320612064756d6d792074657374207064660000000000000000



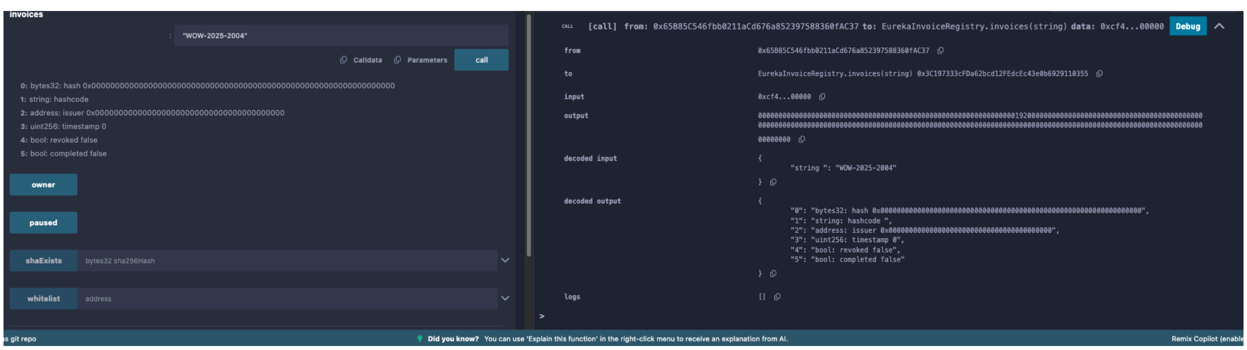
getInvoice

Calldata:
0xf906a440000000000000000000000000000000000000000000000000000000000000002000000000000000000000
00000000000000000000000000000000000000000000d574f572d323032352d32303034000000000000000000000000
00000000000000
Parameters:
0x00000000000000000000000000000000000000000000000000000000000000200000000000000000000000000000
000000000000000000000000000000000000d574f572d323032352d323030340000000000000000000000000000000
0000000



Invoices

This function just fetches the invoice given the string, and we passed the same call data as above, (non-sense data) so we get back empty invoice since no invoice with given hashcode exists



addSigner

calldata:0xeb12d61e00000000000000000000000065b85c546fbb0211acd676a852397588360fa
c37
Parameters: 0x00000000000000000000000065b85c546fbb0211acd676a852397588360fac37

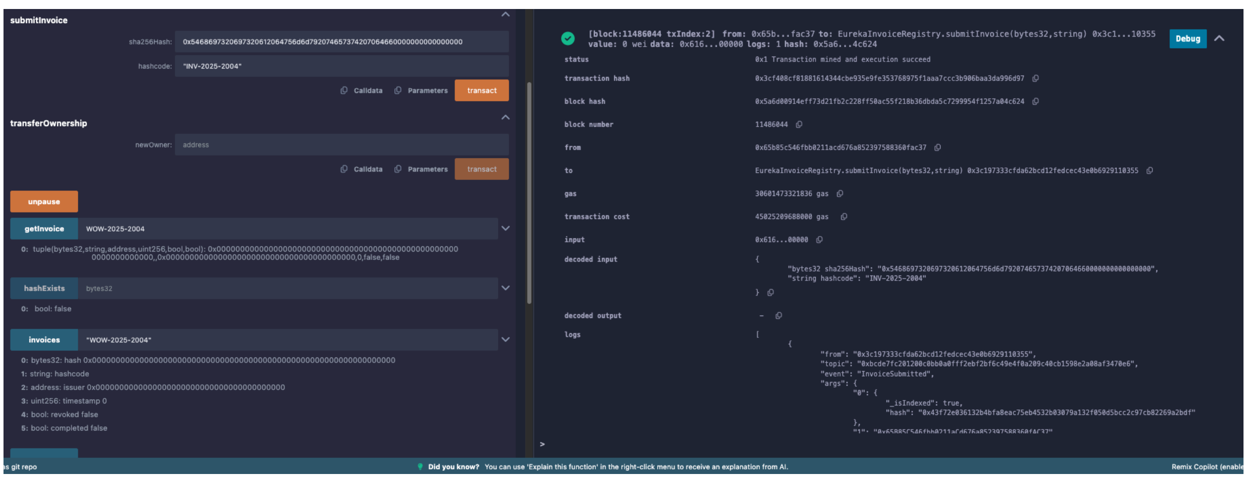
submitInvoice

This is the important function, before we had no invoices, however a lot of things can go wrong here!

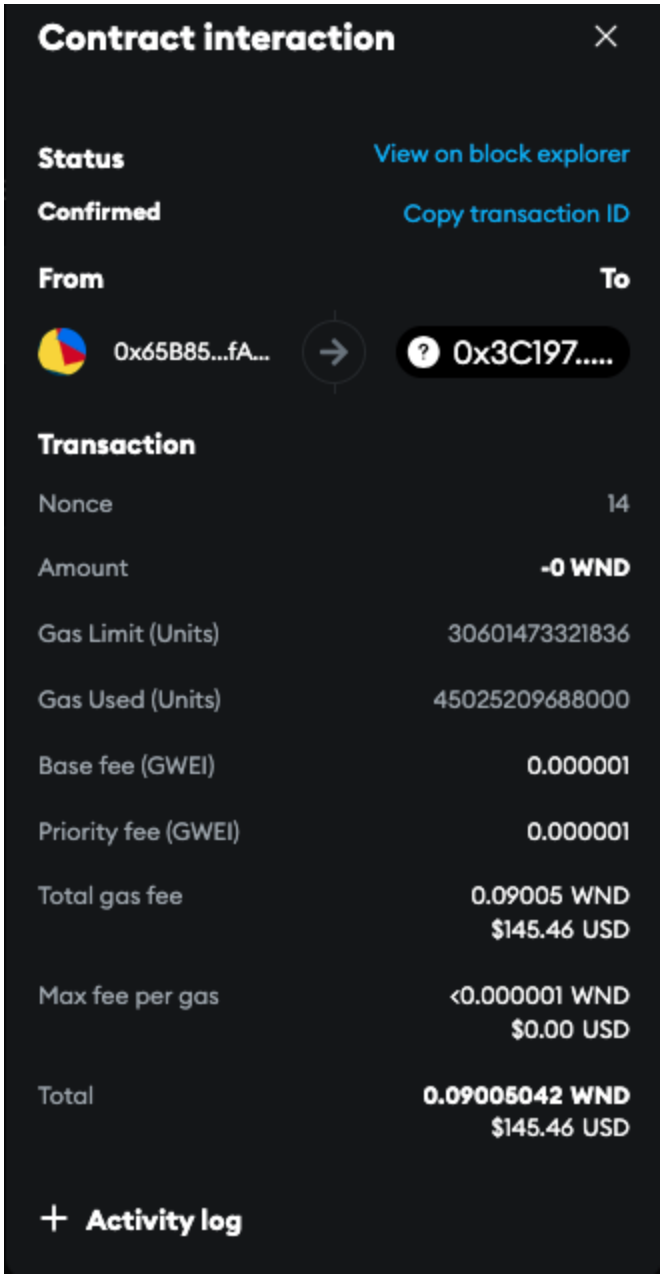
Be careful, not quoting the string will give error encoding arguments, also using a non zero value or not selecting an estimated gas fee with limit 3000000 will give the infamous rpc gas error message!
This function has to have valid sha256 and also hashcode quoted otherwise again infamous rpc gas error message.

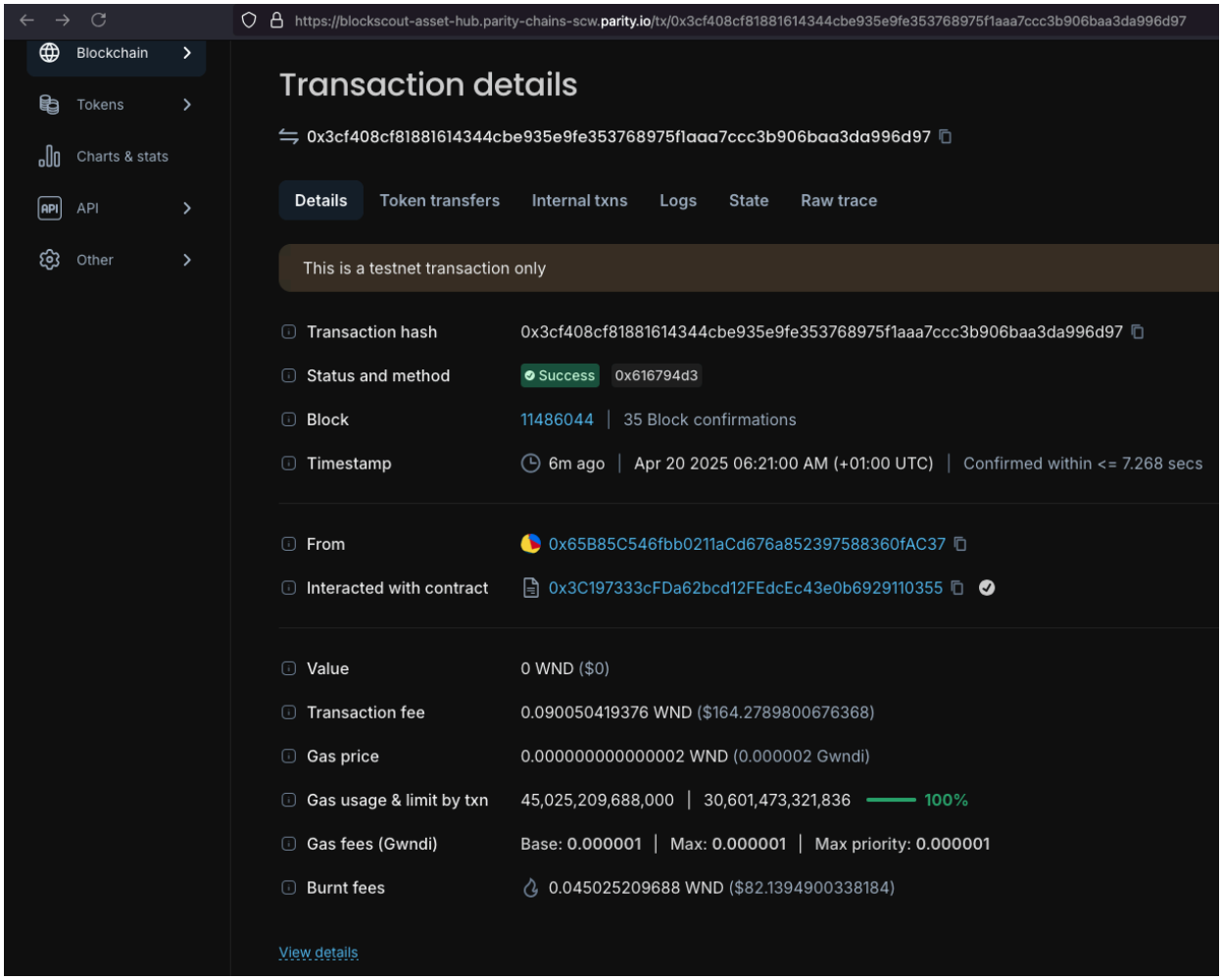
Calldata:
0x616794d35468697320697320612064756d6d79207465737420706466000000000000000000
000000000000000000000000000000000000000000000000000000000000400000000000000
0000000000000000000000000000000000000000000000000d494e562d323032352d32303034
00000000000000000000000000000000000000

Parameters:
0x5468697320697320612064756d6d7920746573742070646600000000000000000000000000
0000000000000000000000000000000000000000000000000000400000000000000000000000
00000000000000000000000000000000000000000d494e562d323032352d3230303400000000
000000000000000000000000000000



Now the contract is on the blockchain!


Contract interaction
Status
View on block explorer
Confirmed
Copy transaction ID
From
To
0x65B85...fA...
0x3C197.....
Transaction
Nonce
14
Amount
-0 WND
Gas Limit (Units)
30601473321836
Gas Used (Units)
45025209688000
Base fee (GWEI)
0.000001
Priority fee (GWEI)
0.000001
Total gas fee
0.09005 WND
$145.46 USD
Max fee per gas
<0.000001 WND
$0.00 USD
Total
0.09005042 WND
$145.46 USD
Activity log



Creating more invoices

collector

Westend Asset Hub

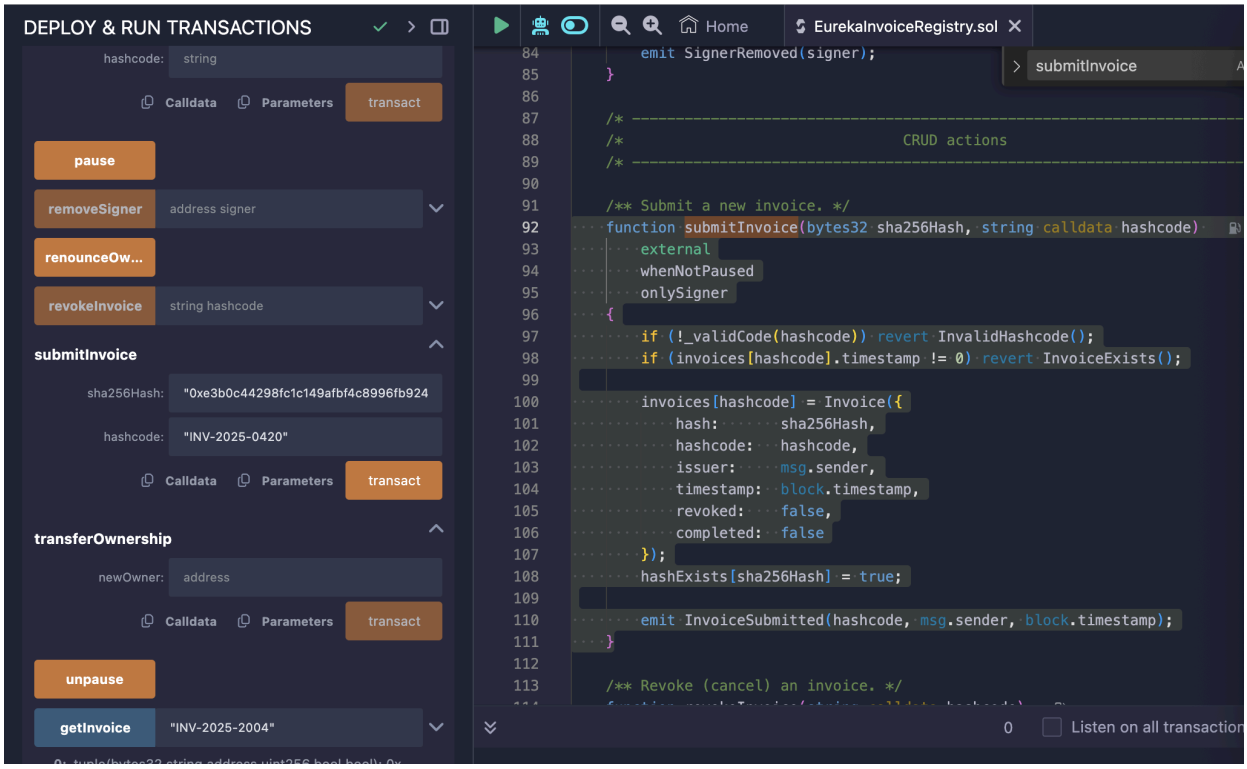
Transaction request

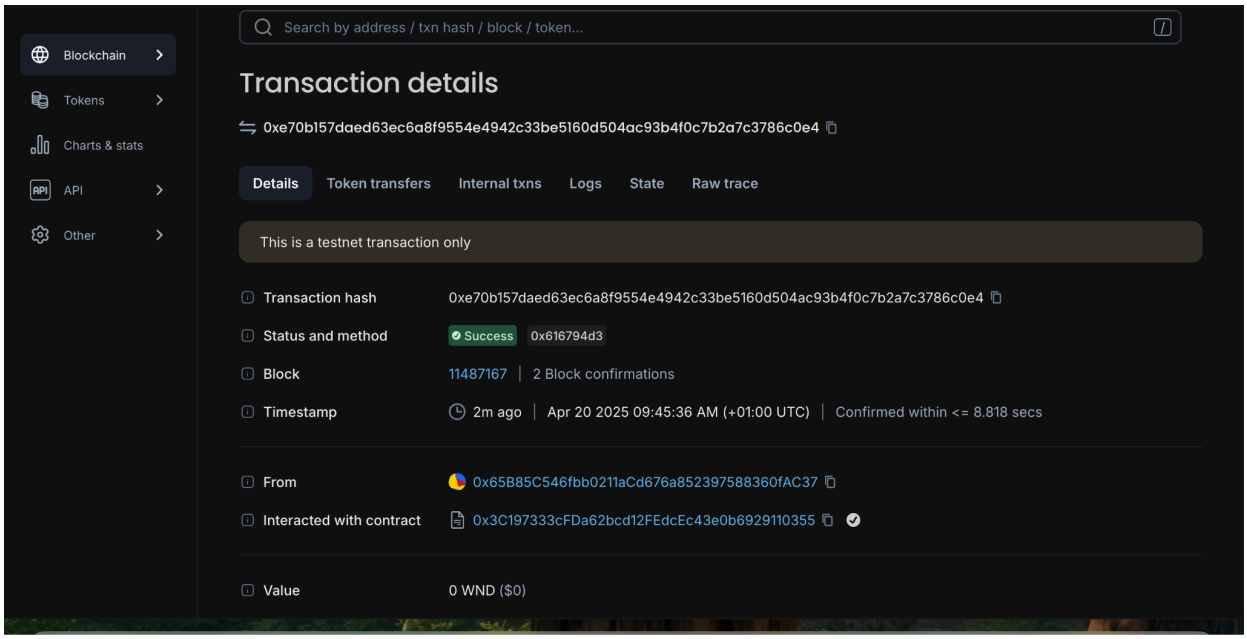
Request from ? remix.polkadot.io

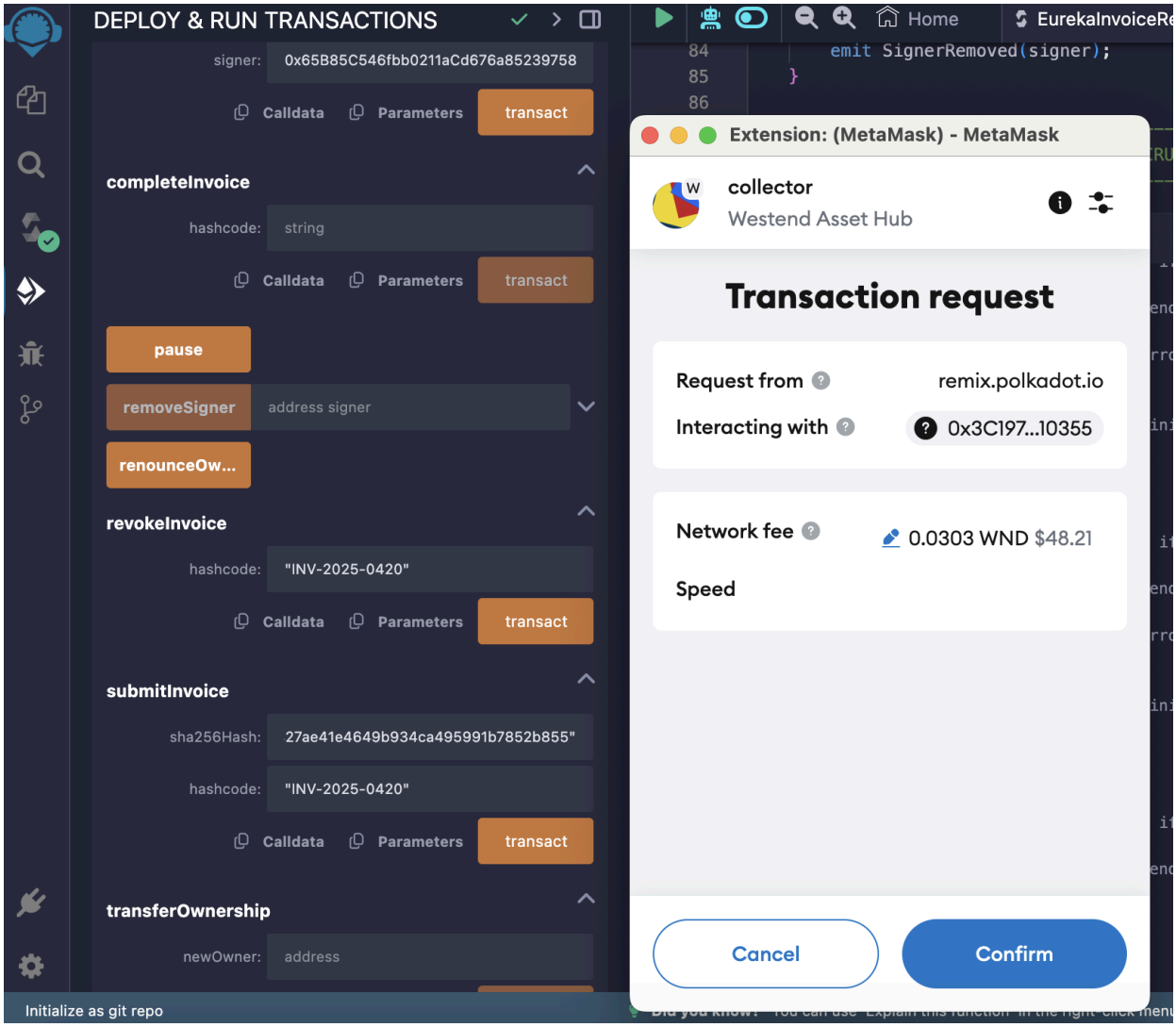
Interacting with ? 0x3C197...10355

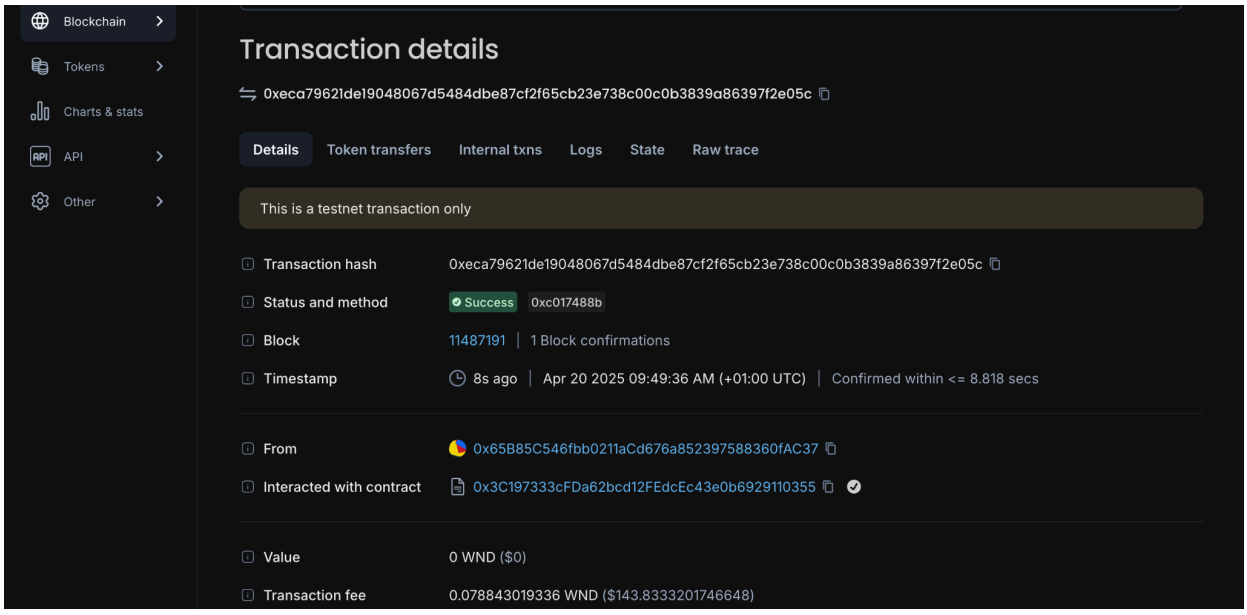
Network fee ? 0.0306 WND $48.72

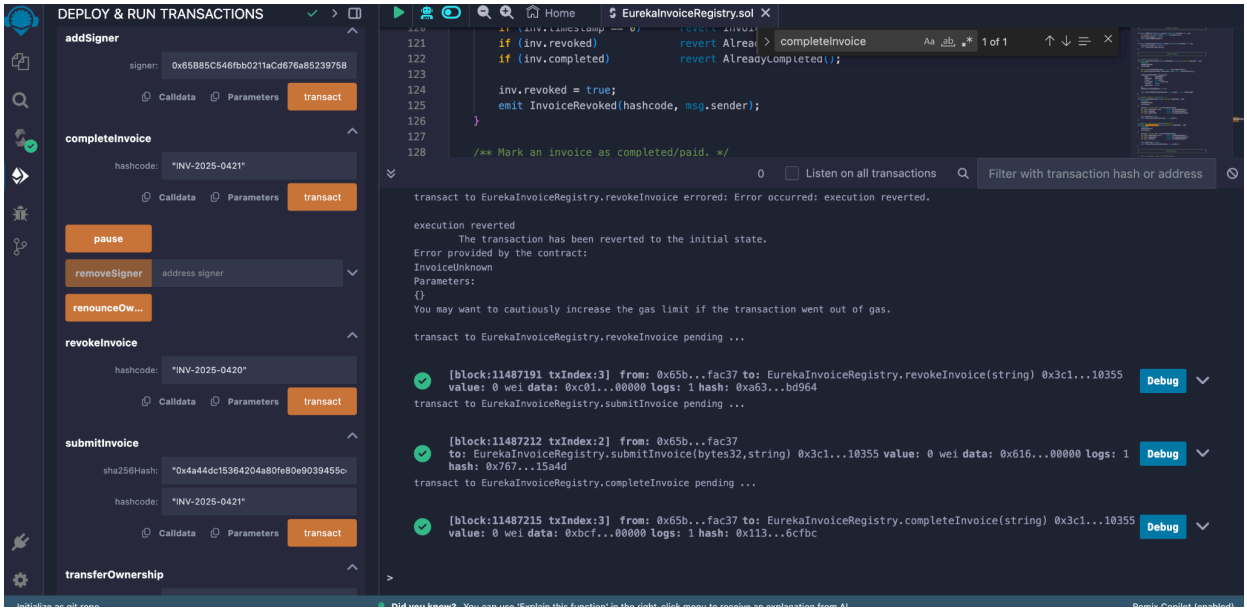
Speed


DEPLOY & RUN TRANSACTIONS
hashcode: string
Calldata
Parameters
transact
pause
removeSigner
address signer
renounceOw...
revokeInvoice
string hashcode
submitInvoice
sha256Hash: "0xe3b0c44298fc1c149afbf4c8996fb924
hashcode: "INV-2025-0420"
Calldata
Parameters
transact
transferOwnership
newOwner: address
Calldata
Parameters
transact
unpause
getInvoice
"INV-2025-2004"
Home
EurekaInvoiceRegistry.sol
submitInvoice
emit SignerRemoved(signer);
}
/* CRUD actions
/** Submit a new invoice. */
function submitInvoice(bytes32 sha256Hash, string calldata hashcode)
external
whenNotPaused
onlySigner
{
if (!_validCode(hashcode)) revert InvalidHashcode();
if (invoices[hashcode].timestamp != 0) revert InvoiceExists();
invoices[hashcode] = Invoice({
hash: sha256Hash,
hashcode: hashcode,
issuer: msg.sender,
timestamp: block.timestamp,
revoked: false,
completed: false
});
hashExists[sha256Hash] = true;
emit InvoiceSubmitted(hashcode, msg.sender, block.timestamp);
}
/** Revoke (cancel) an invoice. */
0
Listen on all transaction


Search by address / txn hash / block / token...
Blockchain
Tokens
Charts & stats
API
Other
Transaction details
0xe70b157daed63ec6a8f9554e4942c33be5160d504ac93b4f0c7b2a7c3786c0e4
Details
Token transfers
Internal txns
Logs
State
Raw trace
This is a testnet transaction only
Transaction hash 0xe70b157daed63ec6a8f9554e4942c33be5160d504ac93b4f0c7b2a7c3786c0e4
Status and method Success 0x616794d3
Block 11487167 | 2 Block confirmations
Timestamp 2m ago | Apr 20 2025 09:45:36 AM (+01:00 UTC) | Confirmed within <= 8.818 secs
From 0x65B85C546fbb0211aCd676a852397588360fAC37
Interacted with contract 0x3C197333cFDa62bcd12FEdcEc43e0b6929110355
Value 0 WND ($0)


DEPLOY & RUN TRANSACTIONS
signer: 0x65B85C546fbb0211aCd676a85239758
Calldata
Parameters
transact
completeInvoice
hashcode: string
Calldata
Parameters
transact
pause
removeSigner
address signer
renounceOw...
revokeInvoice
hashcode: "INV-2025-0420"
Calldata
Parameters
transact
submitInvoice
sha256Hash: 27ae41e4649b934ca495991b7852b855"
hashcode: "INV-2025-0420"
Calldata
Parameters
transact
transferOwnership
newOwner: address
Initialize as git repo
Home
EurekaInvoiceRe
84 emit SignerRemoved(signer);
85 }
86
Extension: (MetaMask) - MetaMask
collector
Westend Asset Hub
Transaction request
Request from
remix.polkadot.io
Interacting with
0x3C197...10355
Network fee
0.0303 WND $48.21
Speed
Cancel
Confirm


Blockchain
Tokens
Charts & stats
API
Other
Transaction details
0xeca79621de19048067d5484dbe87cf2f65cb23e738c00c0b3839a86397f2e05c
Details
Token transfers
Internal txns
Logs
State
Raw trace
This is a testnet transaction only
Transaction hash
0xeca79621de19048067d5484dbe87cf2f65cb23e738c00c0b3839a86397f2e05c
Status and method
Success
0xc017488b
Block
11487191
1 Block confirmations
Timestamp
8s ago
Apr 20 2025 09:49:36 AM (+01:00 UTC)
Confirmed within <= 8.818 secs
From
0x65B85C546fbb0211aCd676a852397588360fAC37
Interacted with contract
0x3C197333cFDa62bcd12FEdcEc43e0b6929110355
Value
0 WND ($0)
Transaction fee
0.078843019336 WND ($143.8333201746648)



Full invoices in json format

PENDING to verify invoice

```
export const exampleInvoice: Invoice = {
  hash: "0x5468697320697320612064756d6d792074657374207064660000000000000000",
  hashcode: "INV-2025-2004",
  issuer: "0x65B85C546fbb0211aCd676a852397588360fAC37",
  timestamp: 1745126460,
  revoked: false,
  completed: false
};
```

REVOKED invoice

```
{
        "0": "bytes32: hash
0xe3b0c44298fc1c149afbf4c8996fb92427ae41e4649b934ca495991b7852b855",
        "1": "string: hashcode INV-2025-0420",
        "2": "address: issuer 0x65B85C546fbb0211aCd676a852397588360fAC37",
        "3": "uint256: timestamp 1745138832",
        "4": "bool: revoked true",
        "5": "bool: completed false"
}
```

COMPLETED invoice

```
{
        "0": "bytes32: hash
0x4a44dc15364204a80fe80e9039455cc1608281820fe596e5a4f6f15b6a36d475",
```

```
        "1": "string: hashcode INV-2025-0421",
        "2": "address: issuer 0x65B85C546fbb0211aCd676a852397588360fAC37",
        "3": "uint256: timestamp 1745139192",
        "4": "bool: revoked false",
        "5": "bool: completed true"
}
```


Useful Resources Links

Useful website to check chain details: https://chainlist.org/
Reference docs for contracts on Polkadot:
https://contracts.polkadot.io
Useful website for smart contracts:
https://docs.soliditylang.org/en/latest/introduction-to-smart-contracts.html
Deploy solidity contracts in browser using remix:
https://remix.polkadot.io/
Deploying smart contracts using hardhat:
https://hardhat.org/hardhat-runner/docs/getting-started#overview