Chain-Data Query

Learn how to connect to Filecoin RPC nodes and query Filecoin chain state and data.

Connecting to Filecoin networks via public RPC nodes

To query chain state and data on any Filecoin network, it is necessary to connect to public node providers. However, it's important to note that most public node providers offer limited access, typically allowing read-only JSON RPC calls andMPoolPush to send signed messages to the Filecoin networks.

To explore further details about the available public RPC providers supporting Filecoin mainnet and Calibration testnet, you can refer to the following page.

- Filecoin mainnet RPCs
- Filecoin Calibration testnet RPCs
-

Ingredients

Let's use Glif nodes as an example to demonstrate how to connect to a public Filecoin RPC node provider. Additionally, we will utilizeethers.js to establish the connection with the RPC nodes.

- Glif nodes
- ethers.js
-

Instructions

We will useethers.js to establish a connection with the public Filecoin node provided by Glif. The following code demonstrates connecting to the Filecoin Calibration testnet as an example.

```

Copy import{ ethers }from"ethers"

//The public Filecoin calibration URL constfielcoin_url='https://api.calibration.node.glif.io/rpc/v1' constprovider=newethers.JsonRpcProvider(fielcoin_url)

constblockNumber=awaitprovider.getBlockNumber() console.log("Block height: ",blockNumber)

```

The expected output:

```

Copy Block height: 1268350

```

Listen to smart contract events

Since the Filecoin Virtual Machine (FVM) is EVM-compatible, we can useethers.js to listen to smart contract events for specific contract actions on the Filecoin network. For instance, we can monitor ERC20 tokentransfer events or client contractDealProposalCreate events.

Ingredients

We will also useethers.js to connect to the public Glif node to listen to the smart contract events.

- Glif Nodes
- ethers.js
-

Instructions

Let's consider thewFIL contract , an ERC-20 token on Filecoin, as an example for listening to its transfer event. To demonstrate how to listen to smart contract events using ethers, we will use the deployed wFIL token address on the Filecoin calibration network and a simplified ABI object for the transfer event. Typically, you would have the wFIL smart contract's Application Binary Interface (ABI) defined in anabi.json file.

The code to listen totransfer events for the wFIL token.

```

Copy import{ ethers }from"ethers"

constwFILAddress="0xaC26a4Ab9cF2A8c5DBaB6fb4351ec0F4b07356c4"// wFIL Contract varabi=["event Transfer(address indexed from, address indexed to, uint amount)"]

constfielcoin_url='https://api.calibration.node.glif.io/rpc/v1' constprovider=newethers.providers.JsonRpcProvider(fielcoin_url)

//listen to the Transfer events in the Token contract constwFIL=newethers.Contract(wFILAddress,abi,provider) wFIL.on("Transfer",(from,to,value,event)=>{ lettransferEvent={ from:from, to:to, value:value, eventData:event, } console.log(JSON.stringify(transferEvent,null,4)) })

```

Once a wFIL token transfer is executed on the blockchain, the following code snippet will capture the corresponding events and print out the event details.

Filter smart contract events

We can also use filters to retrieve specific smart contract transactions from the Filecoin network. Filters enable us to define criteria or conditions to search for event logs that match specific requirements. By setting up a filter, we can monitor and retrieve event logs related to our interests or specific smart contracts.

Ingredients

We will also useethers.js to connect to the public Glif node to filter the smart contract events by providing conditions.

- Glif nodes
- ethers.js
-

Instructions

Here's an example of how you can connect to a Glif node on the calibration network, create a filter to list all wFIL token transfers from your address, and execute the filter to look back 2000 blocks to find the matched transaction list:

```

Copy import{ ethers }from"ethers"

constwFILAddress="0xaC26a4Ab9cF2A8c5DBaB6fb4351ec0F4b07356c4"// wFIL Contract varabi=["event Transfer(address indexed from, address indexed to, uint amount)"]

constfielcoin_url='https://api.calibration.node.glif.io/rpc/v1' constprovider=newethers.providers.JsonRpcProvider(fielcoin_url)

// Create a filter to list all token transfers from myAddress constfilter=contract.filters.Transfer("0xd388aB098ed3E84c0D808776440B48F685198498");

//Filter on the events back to 2000 blocks. constcurrentBlockHight=awaitprovider.getBlockNumber(); constresult=awaitcontract.queryFilter(filter,currentBlockHight-2000,currentBlockHight ); console.log(result);

```

The expected transaction will be similar as follows. With this information, you can develop custom logic to efficiently track and process specific events or blocks on your FEVM smart contracts.

```

Copy { blockNumber:1268728, blockHash:'0x7b4f34d3f7ef791da7f9ab1c342cf147eedf7ec4f99fe92b94a9372927779961', transactionIndex:0, removed:false, address:'0xb44cc5FB8CfEdE63ce1758CE0CDe0958A7702a16', data:'0x0000000000000000000000000000000000000000000000001bc16d674ec80000', topics:[ '0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef', '0x000000000000000000000000d388ab098ed3e84c0d808776440b48f685198498', '0x00000000000000000000000044061aa8df5b33a997ce97d80c700d0c655dc3f2' ], transactionHash:'0x7adb72ac19bf6baa5176f5da799128140ea3a9a9306bf6b1ff52edc58c621c4b', logIndex:0, removeListener:[Function (anonymous)], getBlock:[Function (anonymous)], getTransaction:[Function (anonymous)], getTransactionReceipt:[Function (anonymous)], event:'Transfer', eventSignature:'Transfer(address,address,uint256)', decode:[Function (anonymous)], args:[ '0xd388aB098ed3E84c0D808776440B48F685198498', '0x44061AA8Df5b33a997CE97d80c700d0C655Dc3f2', [BigNumber], from:
```

'0xd388aB098ed3E84c0D808776440B48F685198498', to: '0x44061AA8Df5b33a997CE97d80c700d0C655Dc3f2', amount: [BigNumber]] },

```

Previous dApps Next Oracles

Last updated21 days ago
```