Interact with ERC-20 tokens

ERC-20 is a simple token standard and the most common contract type on Ethereum.

You can:

- Send ERC-20 transactions
- usingeth_sendRawTransaction
- .
- Observe event logs of mined ERC-20 transactions
- usingeth_getLogs
- .
- Followthis tutorial
- to retrieve the balance of ERC-20 tokens.
- Followthis tutorial
- to track ERC-20 token transfers.

ERC-20 token functions and events

An ERC-20 token must implement the following functions:

- totalSupply()
-
  - Returns the total token supply.
- balanceOf(owner)
-
  - Returns the account balance of another account with addressowner
- .
- allowance(owner, spender)
-
  - Returns the amount whichspender
- is still allowed to withdraw fromowner
- .
- transfer(to, value)
-
  - Transfersvalue
- amount of tokens to addressto.
- approve(spender, value)
-
  - Allowsspender
- to withdraw from your account multiple times, up to thevalue
- amount.
- transferFrom(from, to, value)
-
  - Transfersvalue
- amount of tokens from addressfrom
- to addressto
- .

At certain times, an ERC-20 token also must emit the following events:

- Transfer(from, to, value)
-
  - Must trigger when tokens are transferred, including zero value transfers.
- Approval(owner, spender, value)
-
  - Must trigger on any successful call toapprove(spender, value)
- .

ViewEIP-20 for more details about how these functions work and when to emit these events.

Send transactions

Useeth_sendRawTransaction to send ERC-20 token transactions.

The JSON-RPC format expectseth_sendRawTransaction to have a specific data field format that requires normalizing theTransfer function to a shortfunction selector . To do this, set the parameters for the function and run it through Ethereum'ssha3 keccak hash :

- Javascript
- Result

web3 . sha3 ('Transfer(address,address,uint256)') [0. .4] 0x70a08231 ; The first four bytes of this hash comprise its four-byte signature. Take this four-byte signature, pad it with zeros, and package this information into a data string. Then sign the transaction and send it usingeth_sendRawTransaction :

- Example CURL
- JSON result

curl https://mainnet.infura.io/v3/YOUR-API-KEY \ -X POST \ -H "Content-Type: application/json" \ -d '{"jsonrpc":"2.0","method":"eth_sendRawTransaction","params":
["0xf869018203e882520894f17f52151ebef6c7334fad080c5704d77216b732881bc16d674ec80000801ba02da1c48b670996dcb1f447ef9ef00b33033c48a4fe938f420bec3e56bfd24071a062e0aa78a81bf0
{ "id": 1, "jsonrpc": "2.0", "result": "0xe670ec64341771606e55d6b4ca35a1a6b75ee3d5145a99d05921026d1527331" }

Observe logs of mined transactions

When a transaction is mined, event logs are published for public viewing.

Once the event logs are published, you can executeeth_getLogs to investigate what changed relative to the events that you care about, and react to them.

success For example, an event ticketing service that wants to issue off-chain tickets based on crypto payments can useeth_getLogs to find payments to their address, and react to these events by processing some logic in their backend servers to issue tickets to users. The following example useseth_getLogs on the DAI ERC-20 Solidity contract0x6B175474E89094C44Da98b954EedeAC495271d0F :

- Example CURL
- JSON result

curl https://mainnet.infura.io/v3/YOUR-API-KEY \ -X POST \ -H "Content-Type: application/json" \ -d {"jsonrpc":"2.0","method":"eth_getLogs","id":1,"params":
[{"fromBlock":"0x91F37C","toBlock":"0x91F37C","topics":["0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef",
"0x0000000000000000000000000ee25e1ba53c225d250861c8e5a9a3e0fe19c790e", "0x0000000000000000000000000dfbaf3e4c7496dad574a1b842bc85b402bdc298d"
],"address":"0x6B175474E89094C44Da98b954EedeAC495271d0F"}]} { "jsonrpc": "2.0", "id": 1, "result": [{ "address": "0x6B175474E89094C44Da98b954EedeAC495271d0F", "topics": [
"0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef", "0x0000000000000000000000000ee25e1ba53c2250250861c8e5a9a3e0fe19c790e",
"0x0000000000000000000000000dfbaf3e4c7496dad574a1b842bc85b402bdc298d"], "data": "0x0000000000000000000000000000000000000000000000041f900d25d6693623a6", "blockNumbern":
"0x91F37C" }] } In this example request, the parametersfromBlock andtoBlock specify the hexadecimal block number to retrieve logs from.

info IffromBlock andtoBlock are omitted,eth_getLogs returns theentire chain history by default. Infura has a cap on requests of 10,000 events per query. We recommend requesting a single block, as in this example, and to do that for each mined block. This request tells the blockchain to retrieve event logs related to address0x6B175474E89094C44Da98b954EedeAC495271d0F emitted in block0x91F37C that matches topics0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef ,0x0000000000000000000000000ee25e1ba53c225d250861c8e5a9a3e0fe19c790e and0x0000000000000000000000000dfbaf3e4c7496dad574a1b842bc85b402bdc298d .

The response returned for this request is an array of events. In this example, only one event for one address matches the specified topics.

Topics

Topics are events emitted by smart contracts. Looking at the source code of the original contract0x6B175474E89094C44Da98b954EedeAC495271d0F used in this example, there are two event signatures that could be associated with it on lines 94 and 95:

event

Approval (address

indexed tokenOwner ,

address

indexed spender ,

uint tokens) ; event

Transfer (address

indexed

from ,

address

indexed to ,

uint tokens) ; To find out which topic (event) it actually was, create thefunction selector of the event and take thesha3 keccak hash of it. Let’s try the event on line 94:

- Example console request
- JS result

web3 . sha3 ("Approval(address,address,uint256)") ; 0x8c5be1e5ebec7d5bd14f71427d1e84f3dd0314c0f7b2291e5b200ac8c7c3b925 ; The resulting hash doesn’t match the hash provided in the initial request response. Now let’s try the event on line 95 of the contract:

- Example node request
- Example JS result

web3 . sha3 ("Transfer(address,address,uint256)") ; 0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef ; The resulting hash matches the hash provided in the initial request response. Now you know that0xddf25 is the transfer event in this example.

Data

Thedata field in the request response refers to all the "non-indexed stuff" captured in the events. In this example, for the transfer topic,data represents the number of tokens that were transferred. That is,0x41f900d25d6693623a6 or 19471.6949921 DAI tokens were transferred fromee25e1ba53c225d250861c8e5a9a3e0fe19c790e todfbaf3e4c7496dad574a1b842bc85b402bdc298d .

Last updatedonApr 19, 2024 Previous Avoid rate limiting Next Make batch requests