Seth

A powerful command line tool created to interface with the Ethereum blockchain

Introduction to Seth

Seth is a simple, but powerful command line tool created to interface with the Ethereum blockchain. It is part of theDapp.Tools toolset along with other tools for Ethereum. Its two main functionalities among others are performing calls (queries of the Ethereum blockchain - a “read” operation) and sending transactions (writing into the blockchain, changing its state). It also provides conversion between data of Ethereum’s own specific formats and the more widespread, usual data formats.

Getting started

In the following section we will go through the installation and setup of Seth. These steps only work on Unix-based systems (i.e. Linux and macOS), however on Windows, you can try with an emulator, likecmder orcygwin , thelinux subsystem in Windows 10, a virtual machine or a container.

Installation

Seth can be installed as a part of the Dapp Tools suite, which is a collection of blockchain tools created with the Unix philosophy in mind. The most convenient way to do this, is to install Dapp Tools with the one line script provided on thewebpage . Here is how to do it:

From the Dapp Tools page:

If you are running GNU/Linux or macOS you can take advantage of our all in one installer.

curl https://dapp.tools/install | sh

This script downloads the Nix package manager, setups binary cache with Cachix and installs our most used tools.

Manual install

If you have issues using the script above, you can try the manual installation on the aforementioned website. A this point in time you can do a manual installation by running the following scripts:

```

Copy curl https://nixos.org/nix/install | sh . "HOME/.nix-profile/etc/profile.d/nix.sh" nix-env -if https://github.com/cachix/cachix/tarball/master --substituters https://cachix.cachix.org --trusted-public-keys cachix.cachix.org-1:eWNHQldwUO7G2VkjpnjDbWwy4KQ/HNxht7H4SSoMckM= cachix use dapp git clone --recursive https://github.com/dapphub/dapptools HOME/.dapp/dapptools nix-env -f HOME/.dapp/dapptools -iA dapp seth solc hevm ethsign

```

In order to test if the tools have been installed correctly, check the current version of Seth with the following command:

seth --version

If Seth has been installed correctly, the command should produce the following output:

seth 0.7.0

At the time of writing, seth 0.7.0 is thus the latest version, however in the future the output might have a different versioning number.

Errors and a note on macOSX Mojave

If the above command does not work, or you had trouble installing it may be due to Mac OSX Mojave, as we have experienced various issues with the tools nix and cachix not working correctly on this OS, specifically due to a multi-user bug. If you happen to have more user accounts on your Mac, and experience errors running this guide,this document might help you resolve the issue. If this does not resolve the issue, you are more than welcome to ask for help onchat.makerdao.com in the #help channel.

Set up and configuring variables

Configuring Seth can be done with environment variables or command line options. Environment variables can be generally used in two ways: you can save them in a configuration file named .sethrc in specific locations, like your home folder, or just set them only for the current terminal session. In this guide we will use environment variables with the latter approach for simplicity's sake, however for ease-of-use in the future, we strongly encourage to save the variables in your project folder. Followthis example to do so.

Using a local private network

You can quickly set up a local private network with the Dapp tool, which will also create accounts and their keystore files with empty strings for passwords by default. Setting up a local network can be handy when developing dapps for a quick an easy and way to deploy and test functionality without the need of acquiring test-net Ether for contract deployment.

Open a new terminal and execute the following command, and keep it running in the background during the tutorial:

dapp testnet

Copy your account address from the output.

Then in a separate terminal let's create an empty password file:

touch pass

And let's create our environment variables:

```

Copy export ETH_PASSWORD=PWD/pass export ETH_KEYSTORE=~/.dapp/testnet/8545/keystore export ETH_FROM=

```

Using Kovan

Seth can connect to the Kovan Ethereum testnet through a default remote node provided by Infura. This is the most convenient way to do so. You can either create a new account or use the existing one created by the testnet. If you decide to use the existing one, you only need to change the chain parameter:

export SETH_CHAIN=kovan

If you decide to create a new account, an easy method is using the "create new wallet" option in MEW:https://www.myetherwallet.com/ . It is also possible to use Parity or Geth to create a new account or you can use an existing keystore file for a Parity or Geth account. You are also going to need to save the password of your keystore file in a plain text file (Never use this keystore file for real ETH - saving the password for your keystore file in plain text would be very unsafe for a real account! This also goes for the testnet account!).

Then you have to set up the same variables:

```

Copy export ETH_KEYSTORE= export ETH_PASSWORD= export ETH_FROM= export SETH_CHAIN=kovan

```

You will need Kovan ETH for gas, you can get some by following the guide herehttps://github.com/kovan-testnet/faucet

Seth operations

For the first two operations you can use either your own testnet or the Kovan testnet - try both if you want to!

seth balance - Checking ETH balance

Checking ETH balances is pretty straight forward. It can be done with the balance subcommand, then specifying the address as a parameter:

seth balance ETH_FROM

seth send - Sending ETH

Let's send Kovan or private net ETH to an address. You can choose any valid address - in this example I am going to useEthereum Foundation's donation address :

seth send --value 0.1 0xfB6916095ca1df60bB79Ce92cE3Ea74c37c5d359

Upon execution you should see something like the following:

```
Copy seth-send: warning: `ETH_GAS' not set; using default gas amount seth-send: Published transaction with 0 bytes of calldata. seth-send: 0x000000… seth-send: Waiting for transaction receipt....... seth-send: Transaction included in block xxxxxx.
```

This indicates that the transaction was successful.

seth call - Reading contract storage

Since we don't have any contracts deployed to our private network,let's use Kovan from now on . Let's use one of the simplest contracts possible: an ERC-20 token contract. In this example, we are going to use a test collateral token (COL1). You can save its address in a variable with the following command:

```
export COL1=0x911eb92e02477a4e0698790f4d858e09dc39468a
```

You can read the output of a public function of a contract using the call subcommand, the contract's address, and the name of the function.

Let's check out the number of decimals of this token:

```
seth call COL1 'decimals()'
```

The output is:

```
0x0000000000000000000000000000000000000000000000000000000000000012
```

Now don't let this fool you. Seth queries contract data in a low level manner, and returns the value in hexadecimal, as it is represented in the contract, but you can convert it using:

```
seth --to-dec (seth call COL1 'decimals()')
```

The output is:

```
18
```

Sending contract transaction with seth send

You can send a transaction to a contract with the same send command, by adding a couple of extra parameters. Just like with call, you need to specify the contract address and the function we are calling. Let's get some COL1 tokens from a previously set up faucet:

```
export FAUCET=0xe8121d250973229e7988ffa1e9330b420666113a
```

```
seth send FAUCET ‘gulp(address)’ COL1
```

Using function parameters

Now you can check your COL1 balance. This time you will need to present a parameter for the ‘balanceOf’ method of the ERC-20 contract. You can do this by first defining the type, the function takes in its parentheses, and then putting the input parameter after the method:

```
seth --to-dec (seth call COL1 'balanceOf(address)' ETH_FROM)
```

The output is:

```
50000000000000000000
```

Now, that's a rather large value we got. The reason for this is that the contract stores the balances in wei unit (10^-18), which is why we have to convert it to get the actual number of COL1 we own:

```
seth --from-wei (seth --to-dec (seth call COL1 'balanceOf(address)' ETH_FROM)) eth
```

The output is:

```
50.000000000000000000
```

seth block - Retrieving block information

With seth block, we are capable of querying any information about an Ethereum block. Here is the usage from the help option seth block --help :

```
Copy Usage: seth block [-j|--json] [] Print a table of information about. If is given, print only the value of that field.
```

Like any other Seth command, this command depends on Ethereum JSON RPC calls, which are part of the interface of any Ethereum client. You can dive into the corresponding documentation (https://github.com/ethereum/wiki/wiki/JSON-RPC) to learn more about it.

What can come in handy is the fact that in place of a block number, we can also use earliest, latest or pending. So if we would like to query the current block gas limit (I have tried this with seth configured for the kovan testnet) we can do the following:

```
seth block latest gasLimit
```

Output:

```
8000000
```

seth estimate - Estimating gas cost of a transaction

seth estimate can give an estimation for the gas usage of a transaction. The syntax is pretty much the same as forseth send , butseth estimate will not actually send the transaction.

When you want to send a transaction to a contract function with Seth, you have to provide the function signature and the parameters in order after the signature. The ERC-20 transfer function signature looks like the following in Solidity:

```
transfer(address _to, uint256 _value) public returns (bool success)
```

The signature part that Seth needs from this is 'transfer(address, uint)' and the parameters are the recipient address and the amount. The contract needs to receive the amount in hexadecimal representation of the number in wei unit, which is why we need those conversions.

Now, to estimate the gas usage of an ERC-20 token transfer let's execute the following:

```
seth estimate COL1 'transfer(address, uint)' 0xfB6916095ca1df60bB79Ce92cE3Ea74c37c5d359 (seth --to-uint256 (seth --to-wei 0.1 ether))
```

Output:

```
37240
```

seth receipt and seth tx

With seth receipt and seth tx, we can query every single detail imaginable about a transaction. They both take a transaction (tx) hash as an input parameter. The main difference between the two, is that the receipt, which contains the results of the transaction, is only constructed after the transaction gets mined, while the output of seth tx only contains the basic parameters of the transaction before it takes effect.

You can try them for example by first executing a transaction to have a transaction hash:

```
seth send COL1 'transfer(address, uint)' 0xfB6916095ca1df60bB79Ce92cE3Ea74c37c5d359 (seth --to-uint256 (seth --to-wei 0.1 ether))
```

Output:

```
Copy seth-send: Published transaction with 68 bytes of calldata. seth-send: 0x58ba3980775741aecaf8435646a003bff3395d7d4e00c8f7a32ad1fa0ce64e01 seth-send: Waiting for transaction receipt.... seth-send: Transaction included in block 9704345.
```

Now you can try the discussed commands (use your own tx hash from the previous tx):

```
seth receipt 0x58ba3980775741aecaf8435646a003bff3395d7d4e00c8f7a32ad1fa0ce64e01
```

```
seth tx 0x58ba3980775741aecaf8435646a003bff3395d7d4e00c8f7a32ad1fa0ce64e01
```

These both generate a pretty long output, but we can filter each query with an optional extra parameter. For example let's see, how accurate was our previous estimation for the gas consumption (it was perfectly accurate):

```
seth receipt 0x58ba3980775741aecaf8435646a003bff3395d7d4e00c8f7a32ad1fa0ce64e01 gasUsed
```

Output:37240

Additional Resources

This guide was written based on the official documentation in the Github repository of Seth. You can find additional information over there:https://github.com/dapphub/dapptools/tree/master/src/seth

Known Issues

- Issues with MacOS Mojave
-