**What is a transaction**

**More about blockchain transactions**

In the previous lesson we kept talking about transactions, but we never explained what a transaction is. In simple terms, a transaction captures details of an activity that has taken place on a blockchain.

On the left side of your screen, in the Explorer tab, you'll find a folder called broadcast. Foundry saves all your blockchain interactions here. The dry-run folder is used for interactions you made when you didn't have a blockchain running (remember that time when we deployed our contract without specifying an --rpc-url). Moreover, the recordings here are separated by chainId.

**Note**: The chainId is a unique identifier assigned to a specific blockchain network. It is used to distinguish one blockchain from another and is a crucial parameter for ensuring the security and integrity of transactions and interactions on the blockchain.

Click on run-latest.json. Here we can find more details about the last deployment script we ran in our previous lesson. It will show things like transactionType, contractName and contractAddress. Moreover, in the transaction section, you can see what we actually sent over to the RPC URL:

"transaction": {

"from": "0xf39fd6e51aad88f6f4ce6ab8827279cfffb92266",

"to": null,

"gas": "0x714e1",

"value": "0x0",

"input": "0x608060...c63430008130033",

"nonce": "0x0",

"chainId": "0x7a69",

"accessList": null,

"type": null

}

Let's go through each of these:

* from is self-explanatory, it's the address we used to sign the transaction;
* to is the recipient, in our case is null or address(0), this is the standard destination for when new smart contracts are deployed;
* gas is the amount of gas spent. You will see the hex value 0x714e1 (or any other value represented in hex format);

**Quick tip**: Normal humans can't understand hex values like the one indicated above, but there's a quick way to convert these into usual numbers. Run the following command in your terminal: cast --to-base 0x714e1 dec. cast is a very versatile tool provided by Foundry, type cast --help in your terminal to find out more, or go [here](https://book.getfoundry.sh/reference/cast/cast).

* value is the transaction value, or the amount of ETH we are sending over. Given that this transaction was made to deploy a contract, the value here is 0x0 or 0, but we could have specified a value and that would have been the initial balance of the newly deployed contract;
* data in this case is the contract deployment code and the contract code. In the excerpt above this was truncated;
* nonce is a unique identifier assigned to each transaction sent from a specific account. The nonce is used to ensure that each transaction is processed only once and to prevent replay attacks. nonce is incremented with every single transaction;
* accessList is a feature of Ethereum to optimize the gas cost of transactions. It contains a list of addresses and associated storage keys that the transaction is likely to access, allowing the EVM to more efficiently compute the gas cost of storage access during the transaction's execution;
* type please ignore this for now.

There are other values that play an important part that weren't presented in that list, namely the v, r, and s. These are components of a transaction's signature, which are used to validate the authenticity and integrity of the transaction.

Whenever we send a transaction over the blockchain there's a signature happening, that's where we use our private key.

**Important:** Every time you change the state of the blockchain you do it using a transaction. The thing that indicates the change is the data field of a transaction. Deployment bytecode, contract bytecode and OPCODEs will be tackled in a future lesson.