How gas works

Instead of assigning a fixed gas cost in each instruction, the Filecoin EVM runtime charges FIL gas based on the WASM code execution of the Filecoin EVM runtime interpreter.

When executing a message that invokes an EVM contract, the Filecoin virtual machine charges for the message chain inclusion (when the message originates off-chain) and then invokes the actor that hosts the contract. The actor is an instance of the EVM actor, which uses the Filecoin EVM runtime interpreter to execute the contract.

The FEVM interpreter must first load its state, including the contract state, which costs additional gas. The interpreter then begins the execution of the contract bytecode. Each opcode interpreted may perform computation, syscalls, state i/o, and send new messages, all of which are charged with FIL gas. Finally, if the contract state is modified, the interpreter must flush it to the blockstore, which costs additional gas.

Generally, it is not possible to compute gas costs for a contract invocation without using gas estimation through speculative execution.

Calculation example

The total gas fee of a message is calculated as the following:

```

Copy (Gas usage × Base fee) + (GasLimit × GasPremium) + (OverEstimationBurn × BaseFee)

```

Take a look at theGas usage section of the How Filecoin works pagefor more information on the various gas-related parameters attached to each message.

Let’s take a transaction as an example. Our gas parameters are:

- GasUsage
- =1000
- attoFIL
- BaseFee
- =20
- attoFIL
- Gas limit
- =2000
- attoFIL
- Gas premium
- =5
- attoFIL
-

The total fee is(GasUsage × BaseFee) + (Gaslimit x GasPremium) :

```

Copy 1000 x 20 = 20000

2000 x 5 = 10000

20000 + 10000 = 30000 attoFIL

```

Additionally, the message sender can also set theGasFeeCap parameter they are willing to pay. If the sender sets theGasLimit too high, the network will compute the amount of gas to be refunded and the amount of gas to be burned asOverEstimationBurn .

Estimate gas

Filecoin nodes, such as Lotus, have several JSON-API API endpoints designed to help developers estimate gas usage. The available JSON-RPC APIs are:

- GasEstimateMessageGas
- : estimate gas values for a message without any gas fields set, including GasLimit, GasPremium, and GasFeeCap. Returns a message object with those gas fields set.

- GasEstimateGasLimit
- takes the input message and estimates theGasLimit
- based on the execution cost as well as a transaction multiplier.
- GasEstimateGasPremium
- : estimates whatGasPremium
- price you should set to ensure a message will be included inN
- epochs. The smallerN
- is the largerGasPremium
- is likely to be.
- GasEstimateFeeCap
- : estimate theGasFeeCap
- according toBaseFee
- in the parent blocks.
-

If you want to learn more about how to use those JSON-RPC APIs for the Filecoin gas model, please check theJSON RPC API docs for Gas .

Gas estimation varies from network to network. For example, theBaseFee on mainnet is different from theBaseFee on the Calibration testnet.

If you'd rather not calculate and estimate gas for every message, you can just leave the optional fields unset. The gas fields will be estimated and set when the message is pushed to the mempool.

Ethereum compatibility

Since Filecoin is fully EVM-compatible, Filecoin nodes also provide Ethereum-compatible APIs to support gas estimation:

- EthEstimateGas
- : generates and returns an estimate of how much gas is necessary to allow the transaction to complete.
- EthMaxPriorityFeePerGas
- : returns a fee per gas that is an estimate of how much you can pay as a priority fee, or "tip", to get a transaction included in the current block.
-

To request the current max priority fee in the network, you can send a request to a public Filecoin endpoint:

```

Copy curl--location--requestPOST'https://api.calibration.node.glif.io/rpc/v1'\ --header'Content-Type: application/json'\ --data-raw'{ "jsonrpc":"2.0", "method":"eth_maxPriorityFeePerGas", "params": null, "id":1 }'|jq

```

This will output something like:

```

Copy { "jsonrpc": "2.0", "result": "0x31157", "id": 1 }

```

You can convert theresult field from hexadecimal to base 10 in your terminal. Take theresult output and remove the0x from the start. Then useecho to output the conversion:

```

Copy echo((16#31157))

# 201047

```

Additional Resources

- Gas Filecoin improvement proposals (FIPs):
-
  - FIP 0032
-
  - FIP 0037

-
    - FIP 0054
- *
- Primitive Gas Price list
-

Previous Difference with Ethereum Next Precompiles

Last updated7 months ago