Trace API Quickstart

Learn how to use the Trace API to get deeper insights into transaction processing

Want access to the Trace API endpoints? Create an Alchemy account here.

Trace API methods give Alchemy users access to the most detailed information about on-chain activity.

> 📘 **Note:**
>
> Alchemy is the only service that provides access to these Trace API methods due to their high maintenance costs and specialized infrastructure. For this reason, they are currently only available to Alchemy users in Growth and Enterprise tiers. You can upgrade your plan on the billing page to access them.

These API methods allow you to get a full *externality* trace on any transaction executed throughout the Ethereum chain. Unlike the log filtering API, you are able to search and filter based only upon address information. Information returned includes the execution of all `CREATE`, `SUICIDE` and all variants of `CALL` together with input data, output data, gas usage, amount transferred and the success status of each individual action.

> 📘 **Trace API Support**

> **NOTE:** The Trace API is only supported on **Mainnet and Goerli**.

Types of Traces

Transaction Trace (`trace`)

Basic trace of your transaction. See types of trace actions for more details.

State Difference (`stateDiff`)

Provides information detailing all altered portions of the Ethereum state made due to the execution of the transaction.

Virtual Machine Execution Trace (`vmTrace`)

Provides a full trace of the VM's state throughout the execution of the transaction, including for any subcalls.

> 🚧 **VmTrace not supported**
>
> Alchemy no longer supports vmTrace options.

Types of Trace Actions

There are several types of actions captured in Transaction Traces: `CREATE`, `SUICIDE` , variants of `CALL` , and `REWARD` . Below you will find the response payload for the first three.

`CREATE`

Used to create a smart contract.

Response

- `action`
  - `from` : address that created the contract
  - `gas` : gas cost to create contract
  - `init` : initialization code for creating the contract
  - `value` : value sent to contract
- `blockHash` : block hash the transaction was included in
- `blockNumber` : block number the transaction was included in
- `result`
  - `address` : address for contract created
  - `code` : code for contract created
  - `gasUsed` : gas used in contract creation
- `subtraces` : number of child traces of this transaction
- `traceAddress` : index for a given trace in trace tree
- `transactionHash` : hash for the transaction
- `transactionPosition` : position (or index) of transaction in the block
- `type` : type of OPCODE, in this case, `CREATE`

Example

JSON


```
{
    "action": {
      "from": "0x6090a6e47849629b7245dfa1ca21d94cd15878ef",
      "gas": "0x6a7f1",
      "init": "0x6060604052604051602080610516833981016040525 15b60028054600160a060020a038084166(
      "value": "0xe4b4b8af6a70000"
    },
    "blockHash": "0x6d00f7707938cca36b0730d8f7f090543242002b6fa0fe94bf85b9ab02e6bed6",
    "blockNumber": 4000036,
    "result": {
      "address": "0xfc9779d9a0f2715435a3e8ebf780322145d7546e",
```

```
      "code": "0x6060604052361561008857632ffffffff60e060020a60003504166305b34410811461008a57806
      "gasUsed": "0x52ce0"
    },
    "subtraces": 0,
    "traceAddress": [
      0
    ],
    "transactionHash": "0xc9601ea5ca42e57c3ef1d770ab0b278d6aadf2511a4feb879cba573854443423",
    "transactionPosition": 70,
    "type": "create"
  },
```


SUICIDE

Used by an owner of a smart contract to destroy the contract, which will transfer the contract's current balance to the specified `address` and clear the contract's data, freeing up memory on-chain. The freed space on-chain is processed as a refund towards the total gas cost for completing the transaction.

Response

- `action`
  - `address` : address of contract to destroy
  - `refundAddress` : address to send remainder of contract `balance` to
  - `balance` : remaining balance in contract
- `blockHash` : block hash the transaction was included in
- `blockNumber` : block number the transaction was included in
- `result: null` for `SELFDESTRUCT` calls
- `subtraces` : number of child traces of this transaction
- `traceAddress` : index for a given trace in trace tree
- `transactionHash` : hash for the transaction
- `transactionPosition` : position (or index) of transaction in the block
- `type` : type of OPCODE, in this case, `SUICIDE`

Example

JSON


```
{
    "action": {
      "address": "0x87051f6ba0562fdb0485763562bf34cb2ad705b1",
      "refundAddress": "0x000000000000000000000000000000000000dead",
      "balance": "0x0"
    },
    "blockHash": "0x6d00f7707938cca36b0730d8f7f090543242002b6fa0fe94bf85b9ab02e6bed6",
    "blockNumber": 4000036,
    "result": null,
    "subtraces": 0,
    "traceAddress": [
      1,
      2,
      2
    ],
    "transactionHash": "0xbc15addb97490a168dc1d099ab8537caf2e4ff7d1deeff6d685d2d594a750037",
    "transactionPosition": 45,
    "type": "suicide"
  },
```


`CALL`

Used for transferring ETH between externally owned accounts (EOAs) or to call a smart contract function.

"Response"

- `action`
  - `from` : address of the sender
  - `callType` : type of `CALL` , can be any of the following:
    - `call`
    - `delegatecall`
    - `callcode`

    - `staticcall`
  - `gas` : gas included in the transaction
  - `input` : the specific function to call on the contract with parameters specified, encoded. For transfers to an EOA, `input` will be `0x`
  - `to` : address of the receiver
  - `value` : the amount of value to be transferred
- `blockHash` : block hash the transaction was included in
- `blockNumber` : block number the transaction was included in
- `result`
  - `gasUsed` : gas used to execute the transaction
  - `output` : the result of the smart contract function call, encoded. For transfers to an EOA or smart contract the `output` will be `0x` .
- `subtraces` : number of child traces of this transaction.
- `traceAddress` : index for a given trace in the trace tree
- `transactionHash` : hash for the transaction
- `transactionPosition` : position (or index) of transaction in the block
- `type` : type of OPCODE, in this case, `CALL`

Example

JSON

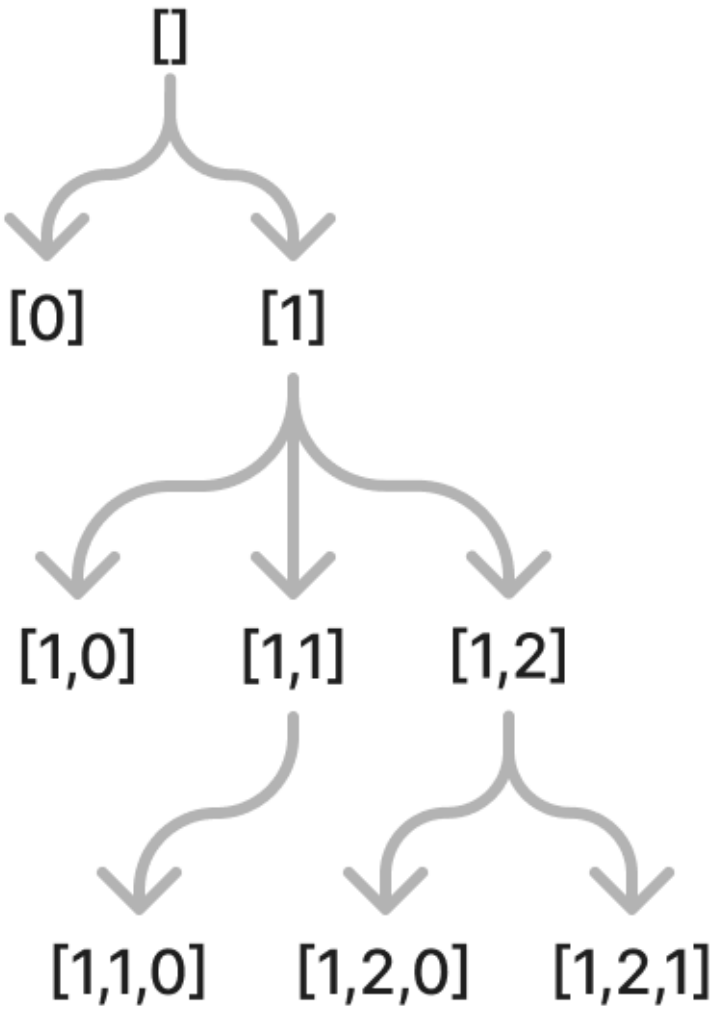

```
{
    "action": {
      "from": "0xbc9f06dd67578b0b8b4d87fda9acde453bc4c067",
      "callType": "call",
      "gas": "0x97478",
      "input": "0xfebefd610000000000000000000000000000000000000000000000000000000000000040cc84
      "to": "0x6090a6e47849629b7245dfa1ca21d94cd15878ef",
      "value": "0x2386f26fc10000"
    },
    "blockHash": "0x6d00f7707938cca36b0730d8f7f090543242002b6fa0fe94bf85b9ab02e6bed6",
    "blockNumber": 4000036,
```

```
    "result": {
      "gasUsed": "0x7ad71",
      "output": "0x"
    },
    "subtraces": 4,
    "traceAddress": [],
    "transactionHash": "0x552b31a3a9c92577d65db62cf9f729e81571e10cad90e356423adcfa2caebacc",
    "transactionPosition": 71,
    "type": "call"
  }
```


How to read `traceAddress`

Traces are structured in a tree format. The `traceAddress` field represents the position of the given trace in the tree. Here is a diagram of `traceAddress` results to help understand how to read this position:

Trace Tree



Trace tree diagram

Updated 4 days ago

← Delete Webhook

Trace API Endpoints →