

Web3 JSON-RPC API

On This Page: [RPC endpoint URLs](#) * [\[Testnet\]](#) * [\[Mainnet\]](#) * [\[kSync Era JSON-RPC methods\]](#) * [\[debug_traceBlockByHash\]](#) * [\[debug_traceBlockByNumber\]](#) * [\[debug_traceCall\]](#) * [\[debug_traceTransaction\]](#) * [\[zks_estimateFee\]](#) * [\[zks_estimateGas1To2\]](#) * [\[zks_getAllAccountBalances\]](#) * [\[zks_getBlockDetails\]](#) * [\[zks_getBridgeContracts\]](#) * [\[zks_getBytecodeByHash\]](#) * [\[zks_getL1BatchBlockRange\]](#) * [\[zks_getL1BatchDetails\]](#) * [\[zks_getL2ToL1LogProof\]](#) * [\[zks_getL2ToL1MsgProof\]](#) * [\[zks_getMainContract\]](#) * [\[zks_getProof\]](#) * [\[zks_getRawBlockTransactions\]](#) * [\[zks_getTestnetPaymaster\]](#) * [\[zks_getTransactionDetails\]](#) * [\[zks_L1BatchNumber\]](#) * [\[zks_L1ChainId\]](#) * [\[PubSub API\]](#)

#

Web3 JSON-RPC API

zkSync Era supports the standard [Ethereum JSON-RPC API](#)^{open in new window} and adds some L2-specific features.

Tip

- As long as code does not involve deploying new smart contracts, which can only be deployed using [EIP712 transactions](#)
- ,no changes to the codebase are needed
- .
- .
- Read the documentation on [EIP-712 transactions](#)
- for more information.

#

RPC endpoint URLs

Rate Limits

We apply rate limiting to both HTTPS and Websocket APIs. The limits are generally permissive (currently 10s to 100s RPS per client), but please [contact us](#) [open in new window](#) if you face any issues in that regard.

#

Testnet

- `HTTPS:https://sepolia.era.zksync.dev`
- `Websocket:wss://sepolia.era.zksync.dev/ws`

#

Mainnet

- [HTTPS:https://mainnet.era.zksync.io](https://mainnet.era.zksync.io)
- [Websocket:wss://mainnet.era.zksync.io/ws](ws://mainnet.era.zksync.io/ws)

#

zkSync Era JSON-RPC methods

The API may provide methods not detailed here which are used internally by the team.

Note

- Metamask does not support the `zk` namespace at the time of writing.
- Instead, use the [Provider](#) class with the testnet RPC.

#

debug_traceBlockByHash

Returns debug trace of all executed calls contained in a block given by its L2 hash.

#

Inputs

Parameter Type Description hash H256 The 32 byte hash defining the L2 block. options TracerConfig Optional arguments: see the [TraceConfig documentation](#)[open in new window](#) for details.

#

Example

```
curl -X POST -H "Content-Type: application/json" --data '{"jsonrpc": "2.0", "id": "2", "method": "debug_traceBlockByHash", "params": ["0x4bd0bd4547d8f8a4fc86a024e54558e156c1ac43d82e24733c6dac2fe5c5fc7"]}' --url "https://mainnet.era.zksync.io"
```

#

Output snippet

[illegible]

#

```
debug traceBlockByNumber
```

Returns debug trace of all executed calls contained in a block given by its L2 block number.

#

Inputs

Parameter Type Description block BlockNumber Block number. options TracerConfig Optional arguments: see the [TraceConfig documentation](#)[open in new window](#) for details.

#

Example

```
curl -X POST -H "Content-Type: application/json" \ --data '{"jsonrpc": "2.0", "id": 2, "method": "debug_traceBlockByNumber", "params": [ "0x24b258" ] }' \ "https://mainnet.era.zksync.io"
```

#

Output snippet

[illegible]

#

Returns debug trace containing information on a specific calls given by the call request.

#

Parameter Type Description request CallRequest The transaction call request for debugging. block BlockNumber Block number by hash or number (optional). options TracerConfig Optional arguments: see the [TraceConfig documentation](#)[open in new window](#) for details.

#

[illegible]

#

[illegible]

#

Uses the [EVM's callTracer open in new window](#) to return a debug trace of a specific transaction given by its transaction hash.

#

Parameter Type Description hash H256 The 32 byte hash defining the L2 block. options TracerConfig Optional arguments: see the [TraceConfig documentation](#) for details.

#

```
curl -X POST -H "Content-Type: application/json" --data '{"jsonrpc":"2.0", "id":2, "method": "debug_traceTransaction", "params": [
"0x4b228f90e796de5a18227072745b0f28e0c4a4661a339f70d3bdde591d3b7f3a"]}' -k "https://mainnet.era.zksync.io"
```

#

```
...{"calls": [], "error": null, "from": "0x0000000000000000000000000000000000000000000000000000000000000000", "gas": "0xdfff51aa3", "gasUsed": "0x27e", "input": "0x6ef25c3a", "output":  
    "0x0000000000000000000000000000000000000000000000000000000000000000", "revertReason": null, "to": "0x000000000000000000000000000000000000000000000000000000000000000b", "type": "Call", "value": "0x0"  
}, ...
```

#

Returns the fee for the transaction.

#

Parameter	Type	Description
request	CallRequest	The zkSync transaction request for which the fee is estimated.

#

```
curl -X POST -H "Content-Type: application/json" --data '{"jsonrpc":"2.0","id":"2","method":"zks_estimateFee","params":[{"from":"0x1111111111111111111111111111111111111111111111111111111111111111","to":"0x2222222222222222222222222222222222222222222222222222222222222222","data":"0xffffffff"}]}'\ "https://mainnet.era.zksync.io"
```

#

```
{ "jsonrpc": "2.0", "result": { "gas limit": "0x156c00", "gas per pubdata limit": "0x143b", "max fee per gas": "0xee6b280", "max priority fee per gas": "0x0", "id": 2 }
```

#

Returns an estimate of the gas required for a L1 to L2 transaction.

#

Parameter	Type	Description
request	CallRequest	The zkSync transaction request for which the gas fee is estimated.

#

```
curl -X POST -H "Content-Type: application/json" --data '{"jsonrpc": "2.0", "id": 2, "method": "zks_estimateGasL1ToL2", "params": [{"from": "0x1111111111111111111111111111111111111111", "to": "0x222222222222222222222222222222222222222222222222222"}, {"data": "0xffffffff"}]}' \ "https://mainnet.era.zkSYNC.io"
```

#

```
{ "jsonrpc": "2.0", "result": "0x25f64db", "id": 2 }
```

#

zks_getAllAccountBalances

Returns all balances for confirmed tokens given by an account address.

#

Inputs

Parameter Type Description address Address The account address.

#

curl example

curl -X POST -H "Content-Type: application/json" \ --data '{"jsonrpc": "2.0", "id": 1, "method": "zks_getAllAccountBalances", "params": ["0x98E9D288743839e96A8005a6B51C770Bbf7788C0"]}' \ "https://mainnet.era.zksync.io"

#

Output

{ "jsonrpc": "2.0", "result": { "0x00": "0x2fbd72a1121b3100" }, "id": 2 }

#

zks_getBlockDetails

Returns additional zkSync-specific information about the L2 block.

- committed
- : The batch is closed and the state transition it creates exists on layer 1.
- proven
- : The batch proof has been created, submitted, and accepted on layer 1.
- executed
- : The batch state transition has been executed on L1; meaning the root state has been updated.

#

Inputs

Parameter Type Description block uint32 The number of the block.

#

curl example

curl -X POST -H "Content-Type: application/json" \ --data '{"jsonrpc": "2.0", "id": 1, "method": "zks_getBlockDetails", "params": [140599]}' \ "https://mainnet.era.zksync.io"

#

Output

{ "jsonrpc": "2.0", "result": { "baseSystemContractsHashes": { "bootloader": "0x010007793a328ef16cc7086708f7f3292ff9b5eed9e7e539c184228f461bf4ef", "default_aa": "0x0100067d861e2f5717a12c3e869cfb657793b86bbb0caa05cc1421f16c5217bc" }, "commitTxHash": "0xd045e3698f018cb233c3817eb53a41a4c5b28784ffe659da246aa33bda34350c", "committedAt": "2023-03-26T07:21:21.046817Z", "executeTxHash": "0xbb66aa75f437bb4255cf751badfc6b142e8d4d3a4e531c7b2e737a22870ff19e", "executedAt": "2023-03-27T07:44:52.187764Z", "l1BatchNumber": 1617, "l1GasPrice": 20690385511, "l1TxCount": 0, "l2FairGasPrice": 250000000, "l2TxCount": 20, "number": 140599, "operatorAddress": "0xf000", "proveTxHash": "0x1591e9b16ff6eb029cc865614094b2e6dd872c8be40b15cc56164941ed723a1a", "provenAt": "2023-03-26T19:48:35.200565Z", "rootHash": "0xf1adac176fc939313eea4b72055db0622a10bbd9b7a83097286e84e471d2e7df", "status": "verified", "timestamp": 1679815038 }, "id": 1 }

#

zks_getBridgeContracts

Returns L1/L2 addresses of default bridges.

#

Inputs

None.

#

curl example

curl -X POST -H "Content-Type: application/json" \ --data '{"jsonrpc": "2.0", "id": 1, "method": "zks_getBridgeContracts", "params": []}' \ "https://mainnet.era.zksync.io"

#

Output

{ "jsonrpc": "2.0", "result": { "l1Erc20DefaultBridge": "0x57891966931eb44b6fb81430e6ce0a03aabde063", "l2Erc20DefaultBridge": "0x11f943b2c77b743ab90f4a0ae7d5a4e7fca3e102" }, "id": 1 }

#

zks_getBytecodeByHash

Returns bytecode of a transaction given by its hash.

#

Inputs

Parameter Type Description hash H256 Bytecode hash as string.

#

curl example

curl -X POST -H "Content-Type: application/json" \ --data '{"jsonrpc": "2.0", "id": 1, "method": "zks_getBytecodeByHash", "params": ["0x0100067d861e2f5717a12c3e869cfb657793b86bbb0caa05cc1421f16c5217bc"]}' \ "https://mainnet.era.zksync.io"

#

Output

{ "jsonrpc": "2.0", "result": [0, 4, 0, 0, 0, 0, 0, 2, 0, 11, ..., ...,], "id": 1 }

#

zks_getL1BatchBlockRange

Returns the range of blocks contained within a batch given by batch number.

The range is given by beginning/end block numbers in hexadecimal.

#

Inputs

Parameter Type Description batch L1BatchNumber The layer 1 batch number.

#

curl example

```
curl -X POST -H "Content-Type: application/json" \ --data '{"jsonrpc": "2.0", "id": 1, "method": "zks_getL1BatchBlockRange", "params": [ 12345 ]}' \ "https://mainnet.era.zksync.io"
```

#

Output

```
{ "jsonrpc": "2.0", "result": [ "0x116fec", "0x117015" ], "id": 1 }
```

#

zks_getL1BatchDetails

Returns data pertaining to a given batch.

#

Inputs

Parameter Type Description batch L1BatchNumber The layer 1 batch number.

#

curl example

```
curl -X POST -H "Content-Type: application/json" \ --data '{"jsonrpc": "2.0", "id": 1, "method": "zks_getL1BatchDetails", "params": [ 12345 ]}' \ "https://mainnet.era.zksync.io"
```

#

Output

```
{ "jsonrpc": "2.0", "result": { "baseSystemContractsHashes": { "bootloader": "0x010007793a328ef16cc7086708f7f3292ff9b5eed9e7e539c184228f461bf4ef", "default_aa": "0x0100067d861e2f5717a12c3e869cfb657793b86bbb0caa05cc1421f16c5217bc" }, "commitTxHash": "0xe5e76d1e17cff2b7232d40ddf43c245e29c76e5354571aa8083d73e793efb64a", "committedAt": "2023-04-09T18:05:40.548203Z", "executeTxHash": "0x19c125a6104f731bcc1ce378f090c808e97c6d634fc32cb786694a94fc8219a1", "executedAt": "2023-04-10T18:48:25.009708Z", "l1GasPrice": 29424338466, "l1TxCount": 9, "l2FairGasPrice": 250000000, "l2TxCount": 294, "number": 12345, "proveTxHash": "0xe980f58feed22a4dbc46fe0339bfc09f51c99b2f3bc4f9f60e710ea5f0a2da", "provenAt": "2023-04-09T22:51:16.200810Z", "rootHash": "0x994d2738f7ac89b45c8381a7816307b501c00b3127afc79e440dbf1b3e3b5a8c", "status": "verified", "timestamp": 1681063384 }, "id": 1 }
```

#

zks_getL2ToL1LogProof

Given a transaction hash, and an index of the L2 to L1 log produced within the transaction, it returns the proof for the corresponding L2 to L1 log.

The index of the log that can be obtained from the transaction receipt (it includes a list of every log produced by the transaction).

#

Inputs

Parameter Type Description tx_hash bytes32 Hash of the L2 transaction the L2 to L1 log was produced within. l2_to_l1_log_index number The index of the L2 to L1 log in the transaction (optional).

#

curl example

```
curl -X POST -H "Content-Type: application/json" \ --data '{"jsonrpc": "2.0", "id": 1, "method": "zks_getL2ToL1LogProof", "params": [ "0x2a1c6c74b184965c0cb015aae9ea134fd96215d2e4f4979cfec12563295f1610e" ]}' \ "https://mainnet.era.zksync.io"
```

#

Output

If there was no such message, the returned value is null . Otherwise:

```
{ "jsonrpc": "2.0", "result": { "id": 0, "proof": [ "0x8c48910df2ca7de509daf50b3182fcd2dd6c422c6704054fd857d6c9516d6fc", "0xc5028885760b8b596c4fa11497c783752cb3a3fb3b8e6b52d7e54b9f1c63521e", "0xeb1f451eb8163723ee19940cf3a8f2a2afdf51100ce8ba25839bd94a057cda16", "0x7aabfd367dea2b5306b8071c246b99566dae551a1dbd40da791e66c4f696b236", "0xe4733f281f18ba3ea8775dd62d2fcd84011c8c938f16ea5790fd29a03bf8db89", "0x1798a1fd9c8fbb818c98cff190daa7cc10b6e5ac9716b4a2649f7c2ebcef2272", "0x66d7c5983afe44cf15ea8cf565b34c6c31f0cb4dd744524f7842b942d08770d", "0xb04e5ee349086985f74b73971ce9dfe76bbed95c84906c5dff96504e1e5396c", "0xac506ecb5465659b3a927143f6d724f91d8d9c4bdb2463aee111d9aa869874db" ], "root": "0x920c63cb0066a08da45f0a9bf934517141bd72d8e5a51421a94b517bf49a0d39" }, "id": 1 } Theid is the position of the leaf in the Merkle tree of L2->L1 messages for the block. Theproof is the Merkle proof for the message, while theroot is the root of the Merkle tree of L2->L1 messages. Please note, that the Merkle tree uses sha256 for the trees.
```

Theid andproof can be used right away for interacting with the zkSync Era smart contract.

Tips

- The list of L2 to L1 logs produced by the transaction, which is included in the receipts, is a combination of logs produced by theL1Messenger
- contract or other system contracts/bootloader.
- The bootloader produces a log for every L1-originated transaction that outputs if the transaction has succeeded.

#

zks_getL2ToL1MsgProof

Given a block, a sender, a message, and an optional message log index in the block containing the L1->L2 message, it returns the proof for the message sent via the L1Messenger system contract.

#

Inputs

Parameter Type Description block uint32 The number of the block where the message was emitted. sender address The sender of the message (i.e. the account that called the L1Messenger system contract). msg bytes32 The keccak256 hash of the sent message. l2_log_position uint256 The index in the block of the event that was emitted by theL1Messenger when submitting this message. If it is omitted, the proof for the first message returns.

curl example

#

Output

The same as in [zk_s_getL2ToL1LogProof](#).

Note

- #

zks getMainContract

Returns the address of the zkSync Era contract.

#

Inputs

None.

#

curl example

```
curl -X POST -H "Content-Type: application/json" \ --data '{"jsonrpc": "2.0", "id": 1, "method": "zks_getMainContract", "params": [ ]}' \ "https://mainnet.era.zksync.io"
```

#

Output

```
{ "jsonrpc" : "2.0" , "result" : "0x32400084c286cf3e17e7b677ea9583e60a000324" , "id" : 1 }
```

#

zks getProof

Returns Merkle proofs for one or more storage values at the specified account along with a Merkle proof of their authenticity. This allows to verify that the values have not been tampered with.

This RPC method is similar to `getProof` in Ethereum, but there are notable differences caused by different Merkle tree construction:

- The point in time at which the requested values with proofs are returned is specified as an L1 batch number, rather than a block number.
- The Merkle tree is constructed in a different way, and so proofs need to be verified differently. In Ethereum, the Merkle tree of the system state is a two-level hexadecimal trie (the upper level corresponds to accounts, the lower – to storage slots within the account). In Era, the Merkle tree is one level full binary tree with 256-bit keys (see its specification below).
- Because of the flat tree, account-level values in Ethereum are stored using special account and storage key combinations. For example, the code hash for account address A is stored at `account0x0008002`
- and storage key `pad address(A)`, where `pad` address pads the address from the start with 12 zero bytes.

zkSync Era Merkle tree specification:

- #

Inputs

Parameter Type Description address Address The account to fetch storage values and proofs for. keys H256[] Vector of storage keys in the account. l1BatchNumber number Number of the L1 batch specifying the point in time at which the requested values are returned.

#

curl example

```
curl -X POST -H "Content-Type: application/json" --data '{"jsonrpc": "2.0", "id": 1, "method": "zks_getProof", "params": ["0x0000000000000000000000000000000000000000000000000000000000000000", "0x08b65c0cf1012ea9f393197be24619fd814379b298b238285649e14f936a5eb12"], 354895 |}' "https://mainnet.era.zksync.io"
```

#

An object with the following fields:

- Each `storageProof` is an object with the following fields:

Example output:

#

Returns data of transactions in a block.

#

Parameter	Type	Description
block	uint32	Block number.

#

```
curl -X POST -H "Content-Type: application/json" \ --data '{"jsonrpc": "2.0", "id": 1, "method": "zks_getRawBlockTransactions", "params": [ 5817 ]}' \ "https://mainnet.era.zksync.io"
```

#

#

Returns the address of the [testnet paymaster](#): the paymaster that is available on testnets and enables paying fees in ERC-20 compatible tokens.

#

None.

#

```
curl -X POST -H "Content-Type: application/json" --data '{"jsonrpc": "2.0", "id": 1, "method": "zks_getTestnetPaymaster", "params": [ ]}' "https://sepolia.era.zksync.dev"
```

#

```
{ "jsonrpc": "2.0", "result": "0x8f0ea1312da29f17eabeb2f484fd3c112cccdd63", "id": 1 }
```

#

Returns data from a specific transaction given by the transaction hash.

#

Parameter Type Description hash H256 Transaction hash as string.

#

```
curl -X POST -H "Content-Type: application/json" -d '{"jsonrpc": "2.0", "id": 1, "method": "zks_getTransactionDetails", "params": ["0x22de7debaa98758afdae89f447ff43bab5da3de6acca7528b281cc2f1be2ee9"]}' -u "https://mainnet.era.zksync.io"
```

#

Output

```
{ "jsonrpc": "2.0", "result": { "ethCommitTxHash": "0x3da5b6eda357189c9243c41c5a33b1b2ed0169be172705d74681a25217702772", "ethExecuteTxHash": "0xdaff5d7ff91333b161de54534b4bb6a78e5325329959a0863bf0aae2b0fdcc6", "ethProveTxHash": "0x2f482d3ea163f5be0c2aca7819d0beb80415be1a310e845a2d726fbc4ac54c80", "fee": "0x0", "gasPerPubdata": "0x320", "initiatorAddress": "0x87869cb87c4fa78ca278df358e890ff73b42a39e", "isL1Originated": true, "receivedAt": "2023-03-03T23:52:24.169Z", "status": "verified" }, "id": 1 }
```

#

zks_L1BatchNumber

Returns the latest L1 batch number.

#

Inputs

None.

#

curl example

```
curl -X POST -H "Content-Type: application/json" \ --data '{"jsonrpc": "2.0", "id": 1, "method": "zks_L1BatchNumber", "params": [ ]}' \ "https://mainnet.era.zksync.io"
```

#

Output

```
{ "jsonrpc": "2.0", "result": "0x544c", "id": 1 }
```

#

zks_L1ChainId

Returns the chain id of the underlying L1.

#

Inputs

None.

#

curl example

```
curl -X POST -H "Content-Type: application/json" \ --data '{"jsonrpc": "2.0", "id": 1, "method": "zks_L1ChainId", "params": [ ]}' \ "https://mainnet.era.zksync.io"
```

#

Output

```
{ "jsonrpc": "2.0", "result": "0x1", "id": 1 }
```

#

PubSub API

zkSync is fully compatible with [Geth's pubsub API](#) [open in new window](#), except for the syncing subscription. This is because nodes on the zkSync network are technically always synchronized.

The WebSocket URL is `wss://sepolia.era.zksync.dev/ws`.

Tips

- Use the websocket endpoint to handle smart contract events, as detailed [in this section of the docs](#)
- .

[\[\]Edit this page](#) [open in new window](#) Last update: Contributors: [\[\[Antonio \]\]](#) [\[albicodes \]\]](#)