Web3 JSON-RPC API

#

Web3 JSON-RPC API

zkSync Era supports the standardEthereum JSON-RPC APIopen in new windowand adds some L2-specific features.

Tip

- As long as code does not involve deploying new smart contracts, which can only be deployed usingEIP712 transactions
- ,no changes to the codebase are needed
- .
- Read the documentation onEIP-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 pleasecontact usopen in new windowif 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
- Websocket:wss://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 thezks_
- namespace at the time of writing.
- Instead, use theProvider
- 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 theTraceConfig documentationopen in new windowfor details.

#

Example

curl -X POST -H "Content-Type: application/json" \ --data '{"jsonrpc":"2.0", "id":2, "method": "debug_traceBlockByHash", "params": ["0x4bd0bd4547d8f8a4fc86a024e54558e156c1acf43d82e24733c6dac2fe5c5fc7"] }' \ "https://mainnet.era.zksync.io"

#

Output snippet

{ "calls" : [ ] , "error" : null , "from" : "0x0000000000000000000000000000000000008001" , "gas" : "0xe1e31a08" , "gasUsed" : "0x27e" , "input" : "0x6ef25c3a" , "output" : "0x00000000000000000000000000000000000000000000000000000000ee6b280" , "revertReason" : null , "to" : "0x000000000000000000000000000000000000800b" , "type" : "Call" , "value" : "0x0" } ,

#

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 theTraceConfig documentationopen in new windowfor 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

{ "calls" : [ ] , "error" : null , "from" : "0x0000000000000000000000000000000000008001" , "gas" : "0xe1e31a08" , "gasUsed" : "0x27e" , "input" : "0x6ef25c3a" , "output" : "0x00000000000000000000000000000000000000000000000000000000ee6b280" , "revertReason" : null , "to" : "0x000000000000000000000000000000000000800b" , "type" : "Call" , "value" : "0x0"

} ,

#

debug_traceCall

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

#

Inputs

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 documentationopen in new window]for details.

#

Example

```
curl -X POST -H "Content-Type: application/json" \ --data '{"jsonrpc":"2.0", "id":2, "method": "debug_traceCall", "params": [ { "from": "0x1111111111111111111111111111111111111111",
"to":"0x2222222222222222222222222222222222222222", "data": "0xffffffff" }, "0x24b258" ] }' \ "https://mainnet.era.zksync.io"
```

#

Output snippet

```
{ "calls" : [ ] , "error" : null , "from" : "0x0000000000000000000000000000000000008001" , "gas" : "0x4b19b87" , "gasUsed" : "0x291" , "input" :
"0x4de2e4680000000000000000000000001111111111111111111111111111111111111111" , "output" : "0x0000000000000000000000000000000000000000000000000000000000000000" ,
"revertReason" : null , "to" : "0x0000000000000000000000000000000000008002" , "type" : "Call" , "value" : "0x0" } ,
```

#

debug_traceTransaction

Uses the[EVM'scallTracer open in new window]to return a debug trace of a specific transaction given by its transaction hash.

#

Inputs

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

#

Example

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

#

Output snippet

```
...{ "calls" : [ ] , "error" : null , "from" : "0x0000000000000000000000000000000000008001" , "gas" : "0xdff51aa3" , "gasUsed" : "0x27e" , "input" : "0x6ef25c3a" , "output" :
"0x000000000000000000000000000000000000000000000000000000000ee6b280" , "revertReason" : null , "to" : "0x000000000000000000000000000000000000800b" , "type" : "Call" , "value" : "0x0"
} , ...
```

#

zks_estimateFee

Returns the fee for the transaction.

#

Inputs

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

#

Example

```
curl -X POST -H "Content-Type: application/json" \ --data '{"jsonrpc":"2.0", "id":2, "method": "zks_estimateFee", "params": [ { "from": "0x1111111111111111111111111111111111111111",
"to":"0x2222222222222222222222222222222222222222", "data": "0xffffffff" } ] }' \ "https://mainnet.era.zksync.io"
```

#

Output

```
{ "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 }
```

#

zks_estimateGasL1ToL2

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

#

Inputs

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

#

curl example

```
curl -X POST -H "Content-Type: application/json" \ --data '{"jsonrpc":"2.0", "id":2, "method": "zks_estimateGasL1ToL2", "params": [ { "from": "0x1111111111111111111111111111111111111111",
"to":"0x2222222222222222222222222222222222222222", "data": "0xffffffff" } ] }' \ "https://mainnet.era.zksync.io"
```

#

Output

```
{ "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" : { "0x0000000000000000000000000000000000000000" : "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" :
"0xfeee860e7aae671124e9a4e61139f3a5085dfeee" , "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" : "0x57891966931eb4bb6fb81430e6ce0a03aabde063" , "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" : "0xe980f58feed22a4dbc46fe0339bfcbc09f51c99b2f3bc4f9f60e710ea5f0a2da" , "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": ["0x2a1c6c74b184965c0cb015aae9ea134fd96215d2e4f4979cfec12563295f610e"]}' \ "https://mainnet.era.zksync.io"

#

Output

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

{ "jsonrpc" : "2.0" , "result" : { "id" : 0 , "proof" : ["0x8c48910df2ca7de509daf50b3182fcdf2dd6c422c6704054fd857d6c9516d6fc" , "0xc5028885760b8b596c4fa11497c783752cb3a3fb3b8e6b52d7e54b9f1c63521e" , "0xeb1f451eb8163723ee19940cf3a8f2a2afdf51100ce8ba25839bd94a057cda16" , "0x7aabfd367dea2b5306b8071c246b99566dae551a1dbd40da791e66c4f696b236" , "0xe4733f281f18ba3ea8775dd62d2fcd84011c8c938f16ea5790fd29a03bf8db89" , "0x1798a1fd9c8fbb818c98cff190daa7cc10b6e5ac9716b4a2649f7c2ebcef2272" , "0x66d7c5983afe44cf15ea8cf565b34c6c31ff0cb4dd744524f7842b942d08770d" , "0xb04e5ee349086985f74b73971ce9dfe76bbed95c84906c5dffd96504e1e5396c" , "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 usessha256 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

```
curl -X POST -H "Content-Type: application/json" \ --data '{"jsonrpc": "2.0", "id": 1, "method": "zks_getL2ToL1MsgProof", "params": [ 5187, "0x87869cb87c4Fa78ca278dF358E890FF73B42a39E", "0x22de7debaa98758afdaee89f447ff43bab5da3de6acca7528b281cc2f1be2ee9" ]}' \ "https://mainnet.era.zksync.io"
```

#

Output

The same as in[zks_getL2ToL1LogProof]() .

Note

- zks_getL2ToL1MsgProof
- endpoint will be deprecated in favor ofzks_getL2ToL1LogProof
- .

#

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 toeth_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 account0x0000000000000000000000000000000000008002
- and storage key pad_address(A), where pad_address pads the address from the start with 12 zero bytes.

zkSync Era Merkle tree specification:

- The Merkle tree is one-level full binary tree with 256-bit keys and 40-byte values.
- Tree keys are computed asreversed(blake2s256([0_u8; 12] ++ account_address ++ storage_key))
- , where:* account_address
-
    - is a 20-byte account address,
-
    - storage_key
-
    - is a 32-byte key in the account storage
-
    - ++
-
    - is byte concatenation
-
    - reversed
-
    - is a function reversing the order of bytes in the provided bytes sequence.
- Tree values arebig_endian(leaf_index) ++ storage_value
- , where:* leaf_index
-
    - is a 64-bit 1-based enumeration index of the entry. Indices are assigned in the order entries are filled.
-
    - storage_value
-
    - is a 32-byte value of the slot.
- If a tree entry is vacant, it is considered to have[0_u8; 40]
- value (i.e., 40 zero bytes).
- Tree leaves are hashed usingblake2s256
- hash function without a tag. E.g., a hash of a vacant leaf isblake2s256([0_u8; 40])
- .
- Internal tree nodes are hashed usingblake2s256
- hash function without a tag:blake2s256(left_child_hash ++ right_child_hash)
- .

#

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": ["0x0000000000000000000000000000000000008003", ["0x8b65c0cf1012ea9f393197eb24619fd814379b298b238285649e14f936a5eb12"], 354895 ] }' "https://mainnet.era.zksync.io"
```

#

Output

An object with the following fields:

- address
- : account address
- storageProof
- : proof for each of the requested keys in the order at which they were requested.

EachstorageProof is an object with the following fields:

Field Type Description key H256 the requested storage key value H256 the storage value at address + key atl1BatchNumber index u64 1-based integer enumeration index of the tree entry (see the tree spec above) proof H256[] sequence of zero or more 32-byte hashes that form a Merkle path for key in the Merkle tree. Hashes are listed using the root-to-leaf ordering. The root hash is excluded; it is published on L1 as a part of L1 batch commit data. If there are less than 256 hashes in proof (which is overwhelmingly likely), then hashes are omitted from the end of the Merkle path. All omitted hashes are hashes of the entirely empty subtrees of the corresponding height. For example, ifproof has 255 hashes, then the single omitted hash isempty_value_hash = blake2s256([0_u8; 40]) . Ifproof has 254 hashes, then the 2 omitted hashes areblake2s256(empty_value_hash ++ empty_value_hash) ,empty_value_hash , etc. For eachStorageProof , it is possible to restore the Merkle tree root hash and compare it to the reference value published on L1. If the hashes match, the value authenticity is proven.

Example output:

```
{ "jsonrpc" : "2.0" , "result" : { "address" : "0x0000000000000000000000000000000000008003" , "storageProof" : [ { "key" :
"0x8b65c0cf1012ea9f393197eb24619fd814379b298b238285649e14f936a5eb12" , "proof" : [ "0xe3e8e49a998b3abf8926f62a5a832d829aadc1b7e059f1ea59ffbab8e11edfb7" ,
"0x9bebfa036e85a6ffb6bf447a9c7d41af176642c6aaf5cfbc97128f4f10d8a25a" , "0xcca89548554c0402d1ad9d40f58357dd63f9a8797312764a7a2117fdf3c3cf27" ,
"0xb300d43e85e6985e813e1d6f9231e14e3d0b150a177ca4b84b14f56a40d7460e" , "0x85d3157d7a7437390e78db2b43ab66f46543ba54bae5a6d4165fc6c0a731369c" ,
"0xa76e30d2ea9e9fc1842273540126743c1eed6ebab3468cc0e73ceb48b60bbbc5" , "0xe870299d2381b56dc3a01dfd12c71662aedffa74686d56b35199352761b7d7e5" ,
"0x95ddfc7d513311b3ac273699246ea095495f4155253de3e7d34e0a3643c5fbd8" , "0x31110aa2a06a06bbc692255235eb69188e9a29d20548057f76f6a3068e1a0506" ,
"0x9cfb69d119d1e7a4dc671e99d4ecc8f0cc5a7ed5e2225106949d6ac7d17ba8a2" , "0x92fe999cb989e97693398f4a6bb7c3db3cb35e256e2a0a3c1bbb6772e2dc8df8" ,
"0xaa6feb7cb008ee03c6a3aa05920f4d2258a21aede994c5f190c4828cff12c672" , "0x1d4e754ebcfe090aa99541027a44622c48faa5aa0ce44e74764296a307f11a9a" ,
"0x1618d709ec45a19f4c4decc234965d9d56e5630bfb03de0b8efab3b0d3fdd5c3" , "0x9eca9d5f3d18e7a7006e1a7dae94756b95b1e498b638a9b16e08a123aafab1fb" ,
"0xb270c37699110bb9c32218d5da501b945ecef12cf200f32d606a8b250aa5b13a" , "0x027f56999d7c97c5ed711f38972ae251bfff9ef450a12024e8680dadfe8d1952" ,
"0x67ebd9c0e1dd1e1b8a2039d5a761f99aa16845ba6ced8243dad2cfbf32fd1e35" , "0x6b349489a60360783e70d701549e8ef90ddab85352e3810c6a70c5c3493c7b58" ,
"0xf3f33bc89d6cf6a79aff4fa51cb8d98b0437c0ee49c2c4fcb8745ee4e6478274" , "0x0269c9296ebf77ac4603fb5040455f88ccacb7f186e56029f209c383d8d4128c" ,
"0xa557bd43406ed8e6fedf52a8d5fadd97c4be39ec43862e95b8761cd3f58b89ee" , "0xc461d0f39807b910e8fa76107e99f99f346a3f3e7faa40343ee2ddf53c4e6b4d" ,
"0xf7fd5de13defb75017e587a4c2e58c33f7118f066367868e8a7e5b1ee2800260" , "0x42c0e6cfbd0f0bc0505538ec04c120a21477c109b0a576247d7d45919d400ede" ,
"0x9cb345b482f45358dd0a57afce927d7b85756f6d49c2ae0dc7f7908fb27d3cc2" , "0x0a39e3389d2437d160f3d95cdf30f61c1afd52a2f82cafd2ac32a6b6ea823e9b" ,
"0x9ebd7b37a21fb0c74d0040a941038887caf4e4c7dfaa182b82915cacc6191025" , "0x4550ab30af8c76557a74d051eb43a964889d383d6da343c6a4f4799595d86f9c" ] , "value" :
"0x0000000000000000000000000000000000000000000000000000000000000060" , "index" : 27900957 } ] } , "id" : 1 }
```

#

zks_getRawBlockTransactions

Returns data of transactions in a block.

#

Inputs

Parameter Type Description block uint32 Block number.

#

curl example

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

#

Output

```
{ "jsonrpc" : "2.0" , "result" : [ { "common_data" : { "L1" : { "canonicalTxHash" : "0x22de7debaa98758afdaee89f447ff43bab5da3de6acca7528b281cc2f1be2ee9" , "deadlineBlock" : 0 , "ethBlock" :
16751339 , "ethHash" : "0x0000000000000000000000000000000000000000000000000000000000000000" , "fullFee" : "0x0" , "gasLimit" : "0x989680" , "gasPerPubdataLimit" : "0x320" ,
"layer2TipFee" : "0x0" , "maxFeePerGas" : "0x0" , "opProcessingType" : "Common" , "priorityQueueType" : "Deque" , "refundRecipient" : "0x87869cb87c4fa78ca278df358e890ff73b42a39e" , "sender" :
"0x87869cb87c4fa78ca278df358e890ff73b42a39e" , "serialId" : 67 , "toMint" : "0x0" } } , "execute" : { "calldata" :
"0x471c46c800000000000000000000000000000000000000000000000000000400000000000000000000000000000000000000000000000000000080000000000000000000000000000
, "contractAddress" : "0xfc5b07a5dd1b80cf271d35642f75cc0500ff1e2c" , "factoryDeps" : [ ] , "value" : "0x0" } , "received_timestamp_ms" : 1677887544169 } ] , "id" : 1 }
```

#

zks_getTestnetPaymaster

Returns the address of thetestnet paymaster : the paymaster that is available on testnets and enables paying fees in ERC-20 compatible tokens.

#

Inputs

None.

#

curl example

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

#

Output

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

#

zks_getTransactionDetails

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

#

Inputs

Parameter Type Description hash H256 Transaction hash as string.

#

curl example

```
curl -X POST -H "Content-Type: application/json" \ --data '{"jsonrpc": "2.0", "id": 1, "method": "zks_getTransactionDetails", "params": [
"0x22de7debaa98758afdaee89f447ff43bab5da3de6acca7528b281cc2f1be2ee9" ]}' \ "https://mainnet.era.zksync.io"
```

[#]

Output

```
{ "jsonrpc" : "2.0" , "result" : { "ethCommitTxHash" : "0x3da5b6eda357189c9243c41c5a33b1b2ed0169be172705d74681a25217702772" , "ethExecuteTxHash" :
"0xdaff5fd7ff91333b161de54534b4bb6a78e5325329959a0863bf0aae2b0fdcc6" , "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 APIopen in new window, except for thesyncing subscription. This is because nodes on the zkSync network are technically always synchronized.

The WebSocket URL iswss://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 pageopen in new windowLast update: Contributors: [[Antonio ,][albicodes]]