Credit costs

The following tables outline the credit costs associated with various RPC methods across different networks. The credit calculation logic is designed to ensure accurate and fair pricing based on the specific network and method being used.

important Requests forarchive data (older than 128 blocks) currently cost the same as non-archive data, but this might change in the near future. The process used to determine the credit cost for each request is as follows:

1. Initial check for a specific credit value
2.
    - When you make a request, the system first checks if there is a specific credit value assigned for the network and method used.
3. This ensures that any unique resource requirements or costs associated with a particular network are accurately reflected.
4. Fallback to the Ethereum method's credit value
5.
    - If no specific credit value is found for the network, the system applies the Ethereum method's credit value.
6. Ethereum serves as the baseline for pricing core RPC methods.
7. Fallback to the default credit value
8.
    - If the method is not found in the Ethereum table either, the system applies a default credit value.
9. Initially set to 80 credits, this default value ensures all requests are accounted for, even if specific pricing information is not available.

Error code costs

When making requests that return status code errors, some errors count towards your credit usage.

- 429
- errors
- : These are not charged and can occur for the following reasons:
-
    - Key settings credit limit caps
-
    - : These are limits set per API key.
-
    - Credit quota limit caps
-
    - : These are overall credit usage limits.
-
    - Credit throughput limit caps
-
    - : These are limits on the rate or throughput of requests.
- 4xx
- errors
- : These are errors caused by human input, and consume 5 credits.
- 5xx
- errors
- : These are server errors, and do not consume any credit charges.

Ethereum

The Ethereum methods serve as a baseline for pricing RPC methods. If a credit cost is not available for a specific network, then use the default Ethereum method's credit cost.

Standard Ethereum-compliant JSON-RPC methods

RPC method Credit cost eth_accounts 80 eth_blobBaseFee 300 eth_blockNumber 80 eth_call 80 eth_chainId 5 eth_createAccessList 80 eth_estimateGas 300 eth_feeHistory 80 eth_gasPrice 80 eth_getBalance 80 eth_getBlockByHash 80 eth_getBlockByNumber 80 eth_getBlockReceipts 1000 eth_getBlockTransactionCountByHash 150 eth_getBlockTransactionCountByNumber 150 eth_getCode 80 eth_getLogs 255 eth_getProof 150 eth_getStorageAt 80 eth_getTransactionByBlockHashAndIndex 150 eth_getTransactionByBlockNumberAndIndex 150 eth_getTransactionByHash 80 eth_getTransactionCount 80 eth_getTransactionReceipt 80 eth_getUncleByBlockHashAndIndex 150 eth_getUncleByBlockNumberAndIndex 150 eth_getUncleCountByBlockHash 150 eth_getUncleCountByBlockNumber 150 eth_getWork 80 eth_hashrate 5 eth_maxPriorityFeePerGas 80 eth_mining 5 eth_protocolVersion 5 eth_sendRawTransaction 80 eth_sign 80 eth_submitWork 80 eth_subscribe 5 eth_syncing 5 eth_unsubscribe 10 net_listening 5 net_peerCount 80 net_version 5 web3_clientVersion 80

Trace methods

RPC method Credit cost trace_block 300 trace_call 300 trace_callMany 300 trace_filter 300 trace_transaction 300

Filter methods

RPC method Credit cost eth_getFilterChanges 140 eth_getFilterLogs 255 eth_newBlockFilter 80 eth_newFilter 80 eth_uninstallFilter 80

Debug methods

RPC method Credit cost debug_storageRangeAt 1000 debug_traceBlock 1000 debug_traceBlockByHash 1000 debug_traceBlockByNumber 1000 debug_traceCall 1000 debug_traceTransaction 1000

Subscription events

You can subscribe and unsubscribe to events usingeth_subscribe andeth_unsubscribe .

The credit costs for thelogs andnewHeads events are charged per block, whereasnewPendingTransaction events are charged at approximately one second intervals.

Subscription events Credit cost logs 300 newHeads 50 newPendingTransaction 200

Linea

RPC method Credit cost linea_estimateGas 100 linea_getTransactionExclusionStatusV1 80

Polygon

RPC method Credit cost bor_getAuthor 80 bor_getCurrentProposer 80 bor_getCurrentValidators 80 bor_getRootHash 80 bor_getSignersAtHash 80

Starknet

RPC Method Credit cost starknet_addDeclareTransaction 700 starknet_addDeployAccountTransaction 700 starknet_addInvokeTransaction 700 starknet_blockHashAndNumber 80 starknet_blockNumber 80 starknet_call 80 starknet_chainId 5 starknet_estimateFee 240 starknet_getBlockTransactionCount 80 starknet_getBlockWithTxHashes 80 starknet_getBlockWithTxs 80 starknet_getClass 80 starknet_getClassAt 80 starknet_getClassHashAt 80 starknet_getEvents 200 starknet_getNonce 80 starknet_getStateUpdate 80 starknet_getStorageAt 80 starknet_getTransactionByBlockIdAndIndex 80 starknet_getTransactionByHash 80 starknet_getTransactionReceipt 80 starknet_pendingTransactions 160 starknet_simulateTransactions 160 starknet_specVersion 5 starknet_syncing 5

ZKsync Era

RPC method Credit cost zks_estimateFee 80 zks_estimateGasL1ToL2 80 zks_getAllAccountBalances 80 zks_getBlockDetails 80 zks_getBridgeContracts 80 zks_getBytecodeByHash 80 zks_getL1BatchBlockRange 80 zks_getL1BatchDetails 80 zks_getL2ToL1LogProof 80 zks_getL2ToL1MsgProof 80 zks_getMainContract 80 zks_getProof 80 zks_getRawBlockTransactions 80 zks_getTestnetPaymaster 80 zks_getTransactionDetails 80 zks_L1BatchNumber 80 zks_L1ChainId 80

Gas API

API endpoint Credit cost baseFeeHistory 80 baseFeePercentile 80 busyThreshold 80 suggestedGasFees 80

Last updatedonNov 5, 2024 Previous Pricing Next Web3 libraries