## JSON-RPC methods

* [web3\_clientVersion](https://github.com/ethereum/wiki/wiki/JSON-RPC#web3_clientversion)
* [web3\_sha3](https://github.com/ethereum/wiki/wiki/JSON-RPC#web3_sha3)
* [net\_version](https://github.com/ethereum/wiki/wiki/JSON-RPC#net_version)
* [net\_peerCount](https://github.com/ethereum/wiki/wiki/JSON-RPC#net_peercount)
* [net\_listening](https://github.com/ethereum/wiki/wiki/JSON-RPC#net_listening)
* [ulam\_protocolVersion](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_protocolversion)
* [ulam\_syncing](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_syncing)
* [ulam\_coinbase](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_coinbase)
* [ulam\_mining](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_mining)
* [ulam\_hashrate](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_hashrate)
* [ulam\_gasPrice](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_gasprice)
* [ulam\_accounts](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_accounts)
* [ulam\_blockNumber](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_blocknumber)
* [ulam\_getBalance](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getbalance)
* [ulam\_getStorageAt](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getstorageat)
* [ulam\_getTransactionCount](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_gettransactioncount)
* [ulam\_getBlockTransactionCountByHash](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getblocktransactioncountbyhash)
* [ulam\_getBlockTransactionCountByNumber](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getblocktransactioncountbynumber)
* [ulam\_getUncleCountByBlockHash](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getunclecountbyblockhash)
* [ulam\_getUncleCountByBlockNumber](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getunclecountbyblocknumber)
* [ulam\_getCode](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getcode)
* [ulam\_sign](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_sign)
* [ulam\_sendTransaction](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_sendtransaction)
* [ulam\_sendRawTransaction](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_sendrawtransaction)
* [ulam\_call](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_call)
* [ulam\_estimateGas](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_estimategas)
* [ulam\_getBlockByHash](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getblockbyhash)
* [ulam\_getBlockByNumber](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getblockbynumber)
* [ulam\_getTransactionByHash](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_gettransactionbyhash)
* [ulam\_getTransactionByBlockHashAndIndex](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_gettransactionbyblockhashandindex)
* [ulam\_getTransactionByBlockNumberAndIndex](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_gettransactionbyblocknumberandindex)
* [ulam\_getTransactionReceipt](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_gettransactionreceipt)
* [ulam\_pendingTransactions](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_pendingtransactions)
* [ulam\_getUncleByBlockHashAndIndex](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getunclebyblockhashandindex)
* [ulam\_getUncleByBlockNumberAndIndex](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getunclebyblocknumberandindex)
* [ulam\_getCompilers](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getcompilers)
* [ulam\_compileLLL](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_compilelll)
* [ulam\_compileSolidity](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_compilesolidity)
* [ulam\_compileSerpent](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_compileserpent)
* [ulam\_newFilter](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_newfilter)
* [ulam\_newBlockFilter](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_newblockfilter)
* [ulam\_newPendingTransactionFilter](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_newpendingtransactionfilter)
* [ulam\_uninstallFilter](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_uninstallfilter)
* [ulam\_getFilterChanges](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getfilterchanges)
* [ulam\_getFilterLogs](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getfilterlogs)
* [ulam\_getLogs](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getlogs)
* [ulam\_getWork](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getwork)
* [ulam\_submitWork](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_submitwork)
* [ulam\_submitHashrate](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_submithashrate)
* [ulam\_getProof](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getproof)
* [db\_putString](https://github.com/ethereum/wiki/wiki/JSON-RPC#db_putstring)
* [db\_getString](https://github.com/ethereum/wiki/wiki/JSON-RPC#db_getstring)
* [db\_putHex](https://github.com/ethereum/wiki/wiki/JSON-RPC#db_puthex)
* [db\_gulamex](https://github.com/ethereum/wiki/wiki/JSON-RPC#db_gethex)
* [shh\_post](https://github.com/ethereum/wiki/wiki/JSON-RPC#shh_post)
* [shh\_version](https://github.com/ethereum/wiki/wiki/JSON-RPC#shh_version)
* [shh\_newIdentity](https://github.com/ethereum/wiki/wiki/JSON-RPC#shh_newidentity)
* [shh\_hasIdentity](https://github.com/ethereum/wiki/wiki/JSON-RPC#shh_hasidentity)
* [shh\_newGroup](https://github.com/ethereum/wiki/wiki/JSON-RPC#shh_newgroup)
* [shh\_addToGroup](https://github.com/ethereum/wiki/wiki/JSON-RPC#shh_addtogroup)
* [shh\_newFilter](https://github.com/ethereum/wiki/wiki/JSON-RPC#shh_newfilter)
* [shh\_uninstallFilter](https://github.com/ethereum/wiki/wiki/JSON-RPC#shh_uninstallfilter)
* [shh\_getFilterChanges](https://github.com/ethereum/wiki/wiki/JSON-RPC#shh_getfilterchanges)
* [shh\_getMessages](https://github.com/ethereum/wiki/wiki/JSON-RPC#shh_getmessages)

## JSON RPC API Reference

#### web3\_clientVersion

Returns the current client version.

##### Parameters

none

##### Returns

String - The current client version.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"web3\_clientVersion","params":[],"id":67}'

// Result

{

"id":67,

"jsonrpc":"2.0",

"result": "Mist/v0.9.3/darwin/go1.4.1"

}

#### web3\_sha3

Returns Keccak-256 (not the standardized SHA3-256) of the given data.

##### Parameters

1. DATA - the data to convert into a SHA3 hash.

##### Example Parameters

params: [

"0x68656c6c6f20776f726c64"

]

##### Returns

DATA - The SHA3 result of the given string.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"web3\_sha3","params":["0x68656c6c6f20776f726c64"],"id":64}'

// Result

{

"id":64,

"jsonrpc": "2.0",

"result": "0x47173285a8d7341e5e972fc677286384f802f8ef42a5ec5f03bbfa254cb01fad"

}

#### net\_version

Returns the current network id.

##### Parameters

none

##### Returns

String - The current network id.

* "1": ulamereum Mainnet
* "2": Morden Testnet (deprecated)
* "3": Ropsten Testnet
* "4": Rinkeby Testnet
* "42": Kovan Testnet

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"net\_version","params":[],"id":67}'

// Result

{

"id":67,

"jsonrpc": "2.0",

"result": "3"

}

#### net\_listening

Returns true if client is actively listening for network connections.

##### Parameters

none

##### Returns

Boolean - true when listening, otherwise false.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"net\_listening","params":[],"id":67}'

// Result

{

"id":67,

"jsonrpc":"2.0",

"result":true

}

#### net\_peerCount

Returns number of peers currently connected to the client.

##### Parameters

none

##### Returns

QUANTITY - integer of the number of connected peers.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"net\_peerCount","params":[],"id":74}'

// Result

{

"id":74,

"jsonrpc": "2.0",

"result": "0x2" // 2

}

#### ulam\_protocolVersion

Returns the current ulamereum protocol version.

##### Parameters

none

##### Returns

String - The current ulamereum protocol version.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_protocolVersion","params":[],"id":67}'

// Result

{

"id":67,

"jsonrpc": "2.0",

"result": "0x54"

}

#### ulam\_syncing

Returns an object with data about the sync status or false.

##### Parameters

none

##### Returns

Object|Boolean, An object with sync status data or FALSE, when not syncing:

* startingBlock: QUANTITY - The block at which the import started (will only be reset, after the sync reached his head)
* currentBlock: QUANTITY - The current block, same as ulam\_blockNumber
* highestBlock: QUANTITY - The estimated highest block

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_syncing","params":[],"id":1}'

// Result

{

"id":1,

"jsonrpc": "2.0",

"result": {

startingBlock: '0x384',

currentBlock: '0x386',

highestBlock: '0x454'

}

}// Or when not syncing

{

"id":1,

"jsonrpc": "2.0",

"result": false

}

#### ulam\_coinbase

Returns the client coinbase address.

##### Parameters

none

##### Returns

DATA, 20 bytes - the current coinbase address.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_coinbase","params":[],"id":64}'

// Result

{

"id":64,

"jsonrpc": "2.0",

"result": "0xc94770007dda54cF92009BFF0dE90c06F603a09f"

}

#### ulam\_mining

Returns true if client is actively mining new blocks.

##### Parameters

none

##### Returns

Boolean - returns true of the client is mining, otherwise false.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_mining","params":[],"id":71}'

// Result

{

"id":71,

"jsonrpc": "2.0",

"result": true

}

#### ulam\_hashrate

Returns the number of hashes per second that the node is mining with.

##### Parameters

none

##### Returns

QUANTITY - number of hashes per second.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_hashrate","params":[],"id":71}'

// Result

{

"id":71,

"jsonrpc": "2.0",

"result": "0x38a"

}

#### ulam\_gasPrice

Returns the current price per gas in wei.

##### Parameters

none

##### Returns

QUANTITY - integer of the current gas price in wei.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_gasPrice","params":[],"id":73}'

// Result

{

"id":73,

"jsonrpc": "2.0",

"result": "0x09184e72a000" // 10000000000000

}

#### ulam\_accounts

Returns a list of addresses owned by client.

##### Parameters

none

##### Returns

Array of DATA, 20 Bytes - addresses owned by the client.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_accounts","params":[],"id":1}'

// Result

{

"id":1,

"jsonrpc": "2.0",

"result": ["0xc94770007dda54cF92009BFF0dE90c06F603a09f"]

}

#### ulam\_blockNumber

Returns the number of most recent block.

##### Parameters

none

##### Returns

QUANTITY - integer of the current block number the client is on.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_blockNumber","params":[],"id":1}'

// Result

{

"id":83,

"jsonrpc": "2.0",

"result": "0xc94" // 1207

}

#### ulam\_getBalance

Returns the balance of the account of given address.

##### Parameters

1. DATA, 20 Bytes - address to check for balance.
2. QUANTITY|TAG - integer block number, or the string "latest", "earliest" or "pending", see the [default block parameter](https://github.com/ethereum/wiki/wiki/JSON-RPC#the-default-block-parameter)

##### Example Parameters

params: [

'0xc94770007dda54cF92009BFF0dE90c06F603a09f',

'latest'

]

##### Returns

QUANTITY - integer of the current balance in wei.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_getBalance","params":["0xc94770007dda54cF92009BFF0dE90c06F603a09f", "latest"],"id":1}'

// Result

{

"id":1,

"jsonrpc": "2.0",

"result": "0x0234c8a3397aab58" // 158972490234375000

}

#### ulam\_getStorageAt

Returns the value from a storage position at a given address.

##### Parameters

1. DATA, 20 Bytes - address of the storage.
2. QUANTITY - integer of the position in the storage.
3. QUANTITY|TAG - integer block number, or the string "latest", "earliest" or "pending", see the [default block parameter](https://github.com/ethereum/wiki/wiki/JSON-RPC#the-default-block-parameter)

##### Returns

DATA - the value at this storage position.

##### Example

Calculating the correct position depends on the storage to retrieve. Consider the following contract deployed at 0x295a70b2de5e3953354a6a8344e616ed314d7251 by address 0x391694e7e0b0cce554cb130d723a9d27458f9298.

contract Storage {

uint pos0;

mapping(address => uint) pos1;

function Storage() {

pos0 = 1234;

pos1[msg.sender] = 5678;

}

}

Retrieving the value of pos0 is straight forward:

curl -X POST --data '{"jsonrpc":"2.0", "method": "ulam\_getStorageAt", "params": ["0x295a70b2de5e3953354a6a8344e616ed314d7251", "0x0", "latest"], "id": 1}' localhost:8545

{"jsonrpc":"2.0","id":1,"result":"0x00000000000000000000000000000000000000000000000000000000000004d2"}

Retrieving an element of the map is harder. The position of an element in the map is calculated with:

keccack(LeftPad32(key, 0), LeftPad32(map position, 0))

This means to retrieve the storage on pos1["0x391694e7e0b0cce554cb130d723a9d27458f9298"] we need to calculate the position with:

keccak(decodeHex("000000000000000000000000391694e7e0b0cce554cb130d723a9d27458f9298" + "0000000000000000000000000000000000000000000000000000000000000001"))

The gulam console which comes with the web3 library can be used to make the calculation:

> var key = "000000000000000000000000391694e7e0b0cce554cb130d723a9d27458f9298" + "0000000000000000000000000000000000000000000000000000000000000001"undefined> web3.sha3(key, {"encoding": "hex"})"0x6661e9d6d8b923d5bbaab1b96e1dd51ff6ea2a93520fdc9eb75d059238b8c5e9"

Now to fetch the storage:

curl -X POST --data '{"jsonrpc":"2.0", "method": "ulam\_getStorageAt", "params": ["0x295a70b2de5e3953354a6a8344e616ed314d7251", "0x6661e9d6d8b923d5bbaab1b96e1dd51ff6ea2a93520fdc9eb75d059238b8c5e9", "latest"], "id": 1}' localhost:8545

{"jsonrpc":"2.0","id":1,"result":"0x000000000000000000000000000000000000000000000000000000000000162e"}

#### ulam\_getTransactionCount

Returns the number of transactions sent from an address.

##### Parameters

1. DATA, 20 Bytes - address.
2. QUANTITY|TAG - integer block number, or the string "latest", "earliest" or "pending", see the [default block parameter](https://github.com/ethereum/wiki/wiki/JSON-RPC#the-default-block-parameter)

##### Example Parameters

params: [

'0xc94770007dda54cF92009BFF0dE90c06F603a09f',

'latest' // state at the latest block

]

##### Returns

QUANTITY - integer of the number of transactions send from this address.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_getTransactionCount","params":["0xc94770007dda54cF92009BFF0dE90c06F603a09f","latest"],"id":1}'

// Result

{

"id":1,

"jsonrpc": "2.0",

"result": "0x1" // 1

}

#### ulam\_getBlockTransactionCountByHash

Returns the number of transactions in a block from a block matching the given block hash.

##### Parameters

1. DATA, 32 Bytes - hash of a block.

##### Example Parameters

params: [

'0xb903239f8543d04b5dc1ba6579132b143087c68db1b2168786408fcbce568238'

]

##### Returns

QUANTITY - integer of the number of transactions in this block.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_getBlockTransactionCountByHash","params":["0xc94770007dda54cF92009BFF0dE90c06F603a09f"],"id":1}'

// Result

{

"id":1,

"jsonrpc": "2.0",

"result": "0xc" // 11

}

#### ulam\_getBlockTransactionCountByNumber

Returns the number of transactions in a block matching the given block number.

##### Parameters

1. QUANTITY|TAG - integer of a block number, or the string "earliest", "latest" or "pending", as in the [default block parameter](https://github.com/ethereum/wiki/wiki/JSON-RPC#the-default-block-parameter).

##### Example Parameters

params: [

'0xe8', // 232

]

##### Returns

QUANTITY - integer of the number of transactions in this block.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_getBlockTransactionCountByNumber","params":["0xe8"],"id":1}'

// Result

{

"id":1,

"jsonrpc": "2.0",

"result": "0xa" // 10

}

#### ulam\_getUncleCountByBlockHash

Returns the number of uncles in a block from a block matching the given block hash.

##### Parameters

1. DATA, 32 Bytes - hash of a block.

##### Example Parameters

params: [

'0xc94770007dda54cF92009BFF0dE90c06F603a09f'

]

##### Returns

QUANTITY - integer of the number of uncles in this block.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_getUncleCountByBlockHash","params":["0xc94770007dda54cF92009BFF0dE90c06F603a09f"],"id":1}'

// Result

{

"id":1,

"jsonrpc": "2.0",

"result": "0xc" // 1

}

#### ulam\_getUncleCountByBlockNumber

Returns the number of uncles in a block from a block matching the given block number.

##### Parameters

1. QUANTITY|TAG - integer of a block number, or the string "latest", "earliest" or "pending", see the [default block parameter](https://github.com/ethereum/wiki/wiki/JSON-RPC#the-default-block-parameter).

params: [

'0xe8', // 232

]

##### Returns

QUANTITY - integer of the number of uncles in this block.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_getUncleCountByBlockNumber","params":["0xe8"],"id":1}'

// Result

{

"id":1,

"jsonrpc": "2.0",

"result": "0x1" // 1

}

#### ulam\_getCode

Returns code at a given address.

##### Parameters

1. DATA, 20 Bytes - address.
2. QUANTITY|TAG - integer block number, or the string "latest", "earliest" or "pending", see the [default block parameter](https://github.com/ethereum/wiki/wiki/JSON-RPC#the-default-block-parameter).

##### Example Parameters

params: [

'0xa94f5374fce5edbc8e2a8697c15331677e6ebf0b',

'0x2' // 2

]

##### Returns

DATA - the code from the given address.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_getCode","params":["0xa94f5374fce5edbc8e2a8697c15331677e6ebf0b", "0x2"],"id":1}'

// Result

{

"id":1,

"jsonrpc": "2.0",

"result": "0x600160008035811a818181146012578301005b601b6001356025565b8060005260206000f25b600060078202905091905056"

}

#### ulam\_sign

The sign method calculates an ulamereum specific signature with: sign(keccak256("\x19ulamereum Signed Message:\n" + len(message) + message))).

By adding a prefix to the message makes the calculated signature recognisable as an ulamereum specific signature. This prevents misuse where a malicious DApp can sign arbitrary data (e.g. transaction) and use the signature to impersonate the victim.

**Note** the address to sign with must be unlocked.

##### Parameters

account, message

1. DATA, 20 Bytes - address.
2. DATA, N Bytes - message to sign.

##### Returns

DATA: Signature

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_sign","params":["0x9b2055d370f73ec7d8a03e965129118dc8f5bf83", "0xdeadbeaf"],"id":1}'

// Result

{

"id":1,

"jsonrpc": "2.0",

"result": "0xa3f20717a250c2b0b729b7e5becbff67fdaef7e0699da4de7ca5895b02a170a12d887fd3b17bfdce3481f10bea41f45ba9f709d39ce8325427b57afcfc994cee1b"

}

An example how to use solidity ecrecover to verify the signature calculated with ulam\_sign can be found [here](https://gist.github.com/bas-vk/d46d83da2b2b4721efb0907aecdb7ebd). The contract is deployed on the testnet Ropsten and Rinkeby.

#### ulam\_sendTransaction

Creates new message call transaction or a contract creation, if the data field contains code.

##### Parameters

1. Object - The transaction object

* from: DATA, 20 Bytes - The address the transaction is send from.
* to: DATA, 20 Bytes - (optional when creating new contract) The address the transaction is directed to.
* gas: QUANTITY - (optional, default: 90000) Integer of the gas provided for the transaction execution. It will return unused gas.
* gasPrice: QUANTITY - (optional, default: To-Be-Determined) Integer of the gasPrice used for each paid gas
* value: QUANTITY - (optional) Integer of the value sent with this transaction
* data: DATA - The compiled code of a contract OR the hash of the invoked method signature and encoded parameters. For details see [ulamereum Contract ABI](https://github.com/ethereum/wiki/wiki/Ethereum-Contract-ABI)
* nonce: QUANTITY - (optional) Integer of a nonce. This allows to overwrite your own pending transactions that use the same nonce.

##### Example Parameters

params: [{

"from": "0xb60e8dd61c5d32be8058bb8eb970870f07233155",

"to": "0xd46e8dd67c5d32be8058bb8eb970870f07244567",

"gas": "0x76c0", // 30400

"gasPrice": "0x9184e72a000", // 10000000000000

"value": "0x9184e72a", // 2441406250

"data": "0xd46e8dd67c5d32be8d46e8dd67c5d32be8058bb8eb970870f072445675058bb8eb970870f072445675"

}]

##### Returns

DATA, 32 Bytes - the transaction hash, or the zero hash if the transaction is not yet available.

Use [ulam\_getTransactionReceipt](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_gettransactionreceipt) to get the contract address, after the transaction was mined, when you created a contract.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_sendTransaction","params":[{see above}],"id":1}'

// Result

{

"id":1,

"jsonrpc": "2.0",

"result": "0xe670ec64341771606e55d6b4ca35a1a6b75ee3d5145a99d05921026d1527331"

}

#### ulam\_sendRawTransaction

Creates new message call transaction or a contract creation for signed transactions.

##### Parameters

1. DATA, The signed transaction data.

##### Example Parameters

params: ["0xd46e8dd67c5d32be8d46e8dd67c5d32be8058bb8eb970870f072445675058bb8eb970870f072445675"]

##### Returns

DATA, 32 Bytes - the transaction hash, or the zero hash if the transaction is not yet available.

Use [ulam\_getTransactionReceipt](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_gettransactionreceipt) to get the contract address, after the transaction was mined, when you created a contract.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_sendRawTransaction","params":[{see above}],"id":1}'

// Result

{

"id":1,

"jsonrpc": "2.0",

"result": "0xe670ec64341771606e55d6b4ca35a1a6b75ee3d5145a99d05921026d1527331"

}

#### ulam\_call

Executes a new message call immediately without creating a transaction on the block chain.

##### Parameters

1. Object - The transaction call object

* from: DATA, 20 Bytes - (optional) The address the transaction is sent from.
* to: DATA, 20 Bytes - The address the transaction is directed to.
* gas: QUANTITY - (optional) Integer of the gas provided for the transaction execution. ulam\_call consumes zero gas, but this parameter may be needed by some executions.
* gasPrice: QUANTITY - (optional) Integer of the gasPrice used for each paid gas
* value: QUANTITY - (optional) Integer of the value sent with this transaction
* data: DATA - (optional) Hash of the method signature and encoded parameters. For details see [ulamereum Contract ABI](https://github.com/ethereum/wiki/wiki/Ethereum-Contract-ABI)

1. QUANTITY|TAG - integer block number, or the string "latest", "earliest" or "pending", see the [default block parameter](https://github.com/ethereum/wiki/wiki/JSON-RPC#the-default-block-parameter)

##### Returns

DATA - the return value of executed contract.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_call","params":[{see above}],"id":1}'

// Result

{

"id":1,

"jsonrpc": "2.0",

"result": "0x"

}

#### ulam\_estimateGas

Generates and returns an estimate of how much gas is necessary to allow the transaction to complete. The transaction will not be added to the blockchain. Note that the estimate may be significantly more than the amount of gas actually used by the transaction, for a variety of reasons including EVM mechanics and node performance.

##### Parameters

See [ulam\_call](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_call) parameters, expect that all properties are optional. If no gas limit is specified gulam uses the block gas limit from the pending block as an upper bound. As a result the returned estimate might not be enough to executed the call/transaction when the amount of gas is higher than the pending block gas limit.

##### Returns

QUANTITY - the amount of gas used.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_estimateGas","params":[{see above}],"id":1}'

// Result

{

"id":1,

"jsonrpc": "2.0",

"result": "0x5208" // 21000

}

#### ulam\_getBlockByHash

Returns information about a block by hash.

##### Parameters

1. DATA, 32 Bytes - Hash of a block.
2. Boolean - If true it returns the full transaction objects, if false only the hashes of the transactions.

##### Example Parameters

params: [

'0xe670ec64341771606e55d6b4ca35a1a6b75ee3d5145a99d05921026d1527331',

true

]

##### Returns

Object - A block object, or null when no block was found:

* number: QUANTITY - the block number. null when its pending block.
* hash: DATA, 32 Bytes - hash of the block. null when its pending block.
* parentHash: DATA, 32 Bytes - hash of the parent block.
* nonce: DATA, 8 Bytes - hash of the generated proof-of-work. null when its pending block.
* sha3Uncles: DATA, 32 Bytes - SHA3 of the uncles data in the block.
* logsBloom: DATA, 256 Bytes - the bloom filter for the logs of the block. null when its pending block.
* transactionsRoot: DATA, 32 Bytes - the root of the transaction trie of the block.
* stateRoot: DATA, 32 Bytes - the root of the final state trie of the block.
* receiptsRoot: DATA, 32 Bytes - the root of the receipts trie of the block.
* miner: DATA, 20 Bytes - the address of the beneficiary to whom the mining rewards were given.
* difficulty: QUANTITY - integer of the difficulty for this block.
* totalDifficulty: QUANTITY - integer of the total difficulty of the chain until this block.
* extraData: DATA - the "extra data" field of this block.
* size: QUANTITY - integer the size of this block in bytes.
* gasLimit: QUANTITY - the maximum gas allowed in this block.
* gasUsed: QUANTITY - the total used gas by all transactions in this block.
* timestamp: QUANTITY - the unix timestamp for when the block was collated.
* transactions: Array - Array of transaction objects, or 32 Bytes transaction hashes depending on the last given parameter.
* uncles: Array - Array of uncle hashes.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_getBlockByHash","params":["0xe670ec64341771606e55d6b4ca35a1a6b75ee3d5145a99d05921026d1527331", true],"id":1}'

// Result

{"id":1,"jsonrpc":"2.0","result": {

"number": "0x1b4", // 436

"hash": "0xe670ec64341771606e55d6b4ca35a1a6b75ee3d5145a99d05921026d1527331",

"parentHash": "0x9646252be9520f6e71339a8df9c55e4d7619deeb018d2a3f2d21fc165dde5eb5",

"nonce": "0xe04d296d2460cfb8472af2c5fd05b5a214109c25688d3704aed5484f9a7792f2",

"sha3Uncles": "0x1dcc4de8dec75d7aab85b567b6ccd41ad312451b948a7413f0a142fd40d49347",

"logsBloom": "0xe670ec64341771606e55d6b4ca35a1a6b75ee3d5145a99d05921026d1527331",

"transactionsRoot": "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",

"stateRoot": "0xd5855eb08b3387c0af375e9cdb6acfc05eb8f519e419b874b6ff2ffda7ed1dff",

"miner": "0x4e65fda2159562a496f9f3522f89122a3088497a",

"difficulty": "0x027f07", // 163591

"totalDifficulty": "0x027f07", // 163591

"extraData": "0x0000000000000000000000000000000000000000000000000000000000000000",

"size": "0x027f07", // 163591

"gasLimit": "0x9f759", // 653145

"gasUsed": "0x9f759", // 653145

"timestamp": "0x54e34e8e" // 1424182926

"transactions": [{...},{ ... }]

"uncles": ["0x1606e5...", "0xd5145a9..."]

}

}

#### ulam\_getBlockByNumber

Returns information about a block by block number.

##### Parameters

1. QUANTITY|TAG - integer of a block number, or the string "earliest", "latest" or "pending", as in the [default block parameter](https://github.com/ethereum/wiki/wiki/JSON-RPC#the-default-block-parameter).
2. Boolean - If true it returns the full transaction objects, if false only the hashes of the transactions.

##### Example Parameters

params: [

'0x1b4', // 436

true

]

##### Returns

See [ulam\_getBlockByHash](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getblockbyhash)

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_getBlockByNumber","params":["0x1b4", true],"id":1}'

Result see [ulam\_getBlockByHash](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getblockbyhash)

#### ulam\_getTransactionByHash

Returns the information about a transaction requested by transaction hash.

##### Parameters

1. DATA, 32 Bytes - hash of a transaction

##### Example Parameters

params: [

"0x88df016429689c079f3b2f6ad39fa052532c56795b733da78a91ebe6a713944b"

]

##### Returns

Object - A transaction object, or null when no transaction was found:

* blockHash: DATA, 32 Bytes - hash of the block where this transaction was in. null when its pending.
* blockNumber: QUANTITY - block number where this transaction was in. null when its pending.
* from: DATA, 20 Bytes - address of the sender.
* gas: QUANTITY - gas provided by the sender.
* gasPrice: QUANTITY - gas price provided by the sender in Wei.
* hash: DATA, 32 Bytes - hash of the transaction.
* input: DATA - the data send along with the transaction.
* nonce: QUANTITY - the number of transactions made by the sender prior to this one.
* to: DATA, 20 Bytes - address of the receiver. null when its a contract creation transaction.
* transactionIndex: QUANTITY - integer of the transaction's index position in the block. null when its pending.
* value: QUANTITY - value transferred in Wei.
* v: QUANTITY - ECDSA recovery id
* r: DATA, 32 Bytes - ECDSA signature r
* s: DATA, 32 Bytes - ECDSA signature s

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_getTransactionByHash","params":["0x88df016429689c079f3b2f6ad39fa052532c56795b733da78a91ebe6a713944b"],"id":1}'

// Result

{

"jsonrpc":"2.0",

"id":1,

"result":{

"blockHash":"0x1d59ff54b1eb26b013ce3cb5fc9dab3705b415a67127a003c3e61eb445bb8df2",

"blockNumber":"0x5daf3b", // 6139707

"from":"0xa7d9ddbe1f17865597fbd27ec712455208b6b76d",

"gas":"0xc350", // 50000

"gasPrice":"0x4a817c800", // 20000000000

"hash":"0x88df016429689c079f3b2f6ad39fa052532c56795b733da78a91ebe6a713944b",

"input":"0x68656c6c6f21",

"nonce":"0x15", // 21

"to":"0xf02c1c8e6114b1dbe8937a39260b5b0a374432bb",

"transactionIndex":"0x41", // 65

"value":"0xf3dbb76162000", // 4290000000000000

"v":"0x25", // 37

"r":"0x1b5e176d927f8e9ab405058b2d2457392da3e20f328b16ddabcebc33eaac5fea",

"s":"0x4ba69724e8f69de52f0125ad8b3c5c2cef33019bac3249e2c0a2192766d1721c"

}

}

#### ulam\_getTransactionByBlockHashAndIndex

Returns information about a transaction by block hash and transaction index position.

##### Parameters

1. DATA, 32 Bytes - hash of a block.
2. QUANTITY - integer of the transaction index position.

##### Example Parameters

params: [

'0xe670ec64341771606e55d6b4ca35a1a6b75ee3d5145a99d05921026d1527331',

'0x0' // 0

]

##### Returns

See [ulam\_getTransactionByHash](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_gettransactionbyhash)

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_getTransactionByBlockHashAndIndex","params":["0xc6ef2fc5426d6ad6fd9e2a26abeab0aa2411b7ab17f30a99d3cb96aed1d1055b", "0x0"],"id":1}'

Result see [ulam\_getTransactionByHash](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_gettransactionbyhash)

#### ulam\_getTransactionByBlockNumberAndIndex

Returns information about a transaction by block number and transaction index position.

##### Parameters

1. QUANTITY|TAG - a block number, or the string "earliest", "latest" or "pending", as in the [default block parameter](https://github.com/ethereum/wiki/wiki/JSON-RPC#the-default-block-parameter).
2. QUANTITY - the transaction index position.

##### Example Parameters

params: [

'0x29c', // 668

'0x0' // 0

]

##### Returns

See [ulam\_getTransactionByHash](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_gettransactionbyhash)

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_getTransactionByBlockNumberAndIndex","params":["0x29c", "0x0"],"id":1}'

Result see [ulam\_getTransactionByHash](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_gettransactionbyhash)

#### ulam\_getTransactionReceipt

Returns the receipt of a transaction by transaction hash.

**Note** That the receipt is not available for pending transactions.

##### Parameters

1. DATA, 32 Bytes - hash of a transaction

##### Example Parameters

params: [

'0xb903239f8543d04b5dc1ba6579132b143087c68db1b2168786408fcbce568238'

]

##### Returns

Object - A transaction receipt object, or null when no receipt was found:

* transactionHash : DATA, 32 Bytes - hash of the transaction.
* transactionIndex: QUANTITY - integer of the transaction's index position in the block.
* blockHash: DATA, 32 Bytes - hash of the block where this transaction was in.
* blockNumber: QUANTITY - block number where this transaction was in.
* from: DATA, 20 Bytes - address of the sender.
* to: DATA, 20 Bytes - address of the receiver. null when it's a contract creation transaction.
* cumulativeGasUsed : QUANTITY - The total amount of gas used when this transaction was executed in the block.
* gasUsed : QUANTITY - The amount of gas used by this specific transaction alone.
* contractAddress : DATA, 20 Bytes - The contract address created, if the transaction was a contract creation, otherwise null.
* logs: Array - Array of log objects, which this transaction generated.
* logsBloom: DATA, 256 Bytes - Bloom filter for light clients to quickly retrieve related logs.

It also returns either :

* root : DATA 32 bytes of post-transaction stateroot (pre Byzantium)
* status: QUANTITY either 1 (success) or 0 (failure)

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_getTransactionReceipt","params":["0xb903239f8543d04b5dc1ba6579132b143087c68db1b2168786408fcbce568238"],"id":1}'

// Result

{"id":1,"jsonrpc":"2.0","result": {

transactionHash: '0xb903239f8543d04b5dc1ba6579132b143087c68db1b2168786408fcbce568238',

transactionIndex: '0x1', // 1

blockNumber: '0xb', // 11

blockHash: '0xc6ef2fc5426d6ad6fd9e2a26abeab0aa2411b7ab17f30a99d3cb96aed1d1055b',

cumulativeGasUsed: '0x33bc', // 13244

gasUsed: '0x4dc', // 1244

contractAddress: '0xb60e8dd61c5d32be8058bb8eb970870f07233155', // or null, if none was created

logs: [{

// logs as returned by getFilterLogs, etc.

}, ...],

logsBloom: "0x00...0", // 256 byte bloom filter

status: '0x1'

}

}

#### ulam\_pendingTransactions

Returns the pending transactions list.

##### Parameters

none

##### Returns

Array - A list of pending transactions.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_pendingTransactions","params":[],"id":1}'

// Result

{"id":1,"jsonrpc":"2.0","result": [{

blockHash: '0x0000000000000000000000000000000000000000000000000000000000000000',

blockNumber: null,

from: '0x28bdb9c230f4d5e45435e4d006326ee32e46cb31',

gas: '0x204734',

gasPrice: '0x4a817c800',

hash: '0x8dfa6a59307a490d672494a171feee09db511f05e9c097e098edc2881f9ca4f6',

input: '0x6080604052600',

nonce: '0x12',

to: null,

transactionIndex: '0x0',

value: '0x0',

v: '0x3d',

r: '0xaabc9ddafffb2ae0bac4107697547d22d9383667d9e97f5409dd6881ce08f13f',

s: '0x69e43116be8f842dcd4a0b2f760043737a59534430b762317db21d9ac8c5034'

},....,{

blockHash: '0x0000000000000000000000000000000000000000000000000000000000000000',

blockNumber: null,

from: '0x28bdb9c230f4d5e45435e4d006326ee32e487b31',

gas: '0x205940',

gasPrice: '0x4a817c800',

hash: '0x8e4340ea3983d86e4b6c44249362f716ec9e09849ef9b6e3321140581d2e4dac',

input: '0xe4b6c4424936',

nonce: '0x14',

to: null,

transactionIndex: '0x0',

value: '0x0',

v: '0x3d',

r: '0x1ec191ef20b0e9628c4397665977cbe7a53a263c04f6f185132b77fa0fd5ca44',

s: '0x8a58e00c63e05cfeae4f1cf19f05ce82079dc4d5857e2cc281b7797d58b5faf'

}]

}

#### ulam\_getUncleByBlockHashAndIndex

Returns information about a uncle of a block by hash and uncle index position.

##### Parameters

1. DATA, 32 Bytes - hash a block.
2. QUANTITY - the uncle's index position.

params: [

'0xc6ef2fc5426d6ad6fd9e2a26abeab0aa2411b7ab17f30a99d3cb96aed1d1055b',

'0x0' // 0

]

##### Returns

See [ulam\_getBlockByHash](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getblockbyhash)

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_getUncleByBlockHashAndIndex","params":["0xc6ef2fc5426d6ad6fd9e2a26abeab0aa2411b7ab17f30a99d3cb96aed1d1055b", "0x0"],"id":1}'

Result see [ulam\_getBlockByHash](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getblockbyhash)

**Note**: An uncle doesn't contain individual transactions.

#### ulam\_getUncleByBlockNumberAndIndex

Returns information about a uncle of a block by number and uncle index position.

##### Parameters

1. QUANTITY|TAG - a block number, or the string "earliest", "latest" or "pending", as in the [default block parameter](https://github.com/ethereum/wiki/wiki/JSON-RPC#the-default-block-parameter).
2. QUANTITY - the uncle's index position.

##### Example Parameters

params: [

'0x29c', // 668

'0x0' // 0

]

##### Returns

See [ulam\_getBlockByHash](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getblockbyhash)

**Note**: An uncle doesn't contain individual transactions.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_getUncleByBlockNumberAndIndex","params":["0x29c", "0x0"],"id":1}'

Result see [ulam\_getBlockByHash](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getblockbyhash)

#### ulam\_getCompilers (DEPRECATED)

Returns a list of available compilers in the client.

##### Parameters

none

##### Returns

Array - Array of available compilers.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_getCompilers","params":[],"id":1}'

// Result

{

"id":1,

"jsonrpc": "2.0",

"result": ["solidity", "lll", "serpent"]

}

#### ulam\_compileSolidity (DEPRECATED)

Returns compiled solidity code.

##### Parameters

1. String - The source code.

##### Example Parameters

params: [

"contract test { function multiply(uint a) returns(uint d) { return a \* 7; } }",

]

##### Returns

DATA - The compiled source code.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_compileSolidity","params":["contract test { function multiply(uint a) returns(uint d) { return a \* 7; } }"],"id":1}'

// Result

{

"id":1,

"jsonrpc": "2.0",

"result": {

"code": "0x605880600c6000396000f3006000357c010000000000000000000000000000000000000000000000000000000090048063c6888fa114602e57005b603d6004803590602001506047565b8060005260206000f35b60006007820290506053565b91905056",

"info": {

"source": "contract test {\n function multiply(uint a) constant returns(uint d) {\n return a \* 7;\n }\n}\n",

"language": "Solidity",

"languageVersion": "0",

"compilerVersion": "0.9.19",

"abiDefinition": [

{

"constant": true,

"inputs": [

{

"name": "a",

"type": "uint256"

}

],

"name": "multiply",

"outputs": [

{

"name": "d",

"type": "uint256"

}

],

"type": "function"

}

],

"userDoc": {

"methods": {}

},

"developerDoc": {

"methods": {}

}

}

}

#### ulam\_compileLLL (DEPRECATED)

Returns compiled LLL code.

##### Parameters

1. String - The source code.

##### Example Parameters

params: [

"(returnlll (suicide (caller)))",

]

##### Returns

DATA - The compiled source code.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_compileLLL","params":["(returnlll (suicide (caller)))"],"id":1}'

// Result

{

"id":1,

"jsonrpc": "2.0",

"result": "0x603880600c6000396000f3006001600060e060020a600035048063c6888fa114601857005b6021600435602b565b8060005260206000f35b600081600702905091905056" // the compiled source code

}

#### ulam\_compileSerpent (DEPRECATED)

Returns compiled serpent code.

##### Parameters

1. String - The source code.

##### Example Parameters

params: [

"/\* some serpent \*/",

]

##### Returns

DATA - The compiled source code.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_compileSerpent","params":["/\* some serpent \*/"],"id":1}'

// Result

{

"id":1,

"jsonrpc": "2.0",

"result": "0x603880600c6000396000f3006001600060e060020a600035048063c6888fa114601857005b6021600435602b565b8060005260206000f35b600081600702905091905056" // the compiled source code

}

#### ulam\_newFilter

Creates a filter object, based on filter options, to notify when the state changes (logs). To check if the state has changed, call [ulam\_getFilterChanges](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getfilterchanges).

##### A note on specifying topic filters:

Topics are order-dependent. A transaction with a log with topics [A, B] will be matched by the following topic filters:

* [] "anything"
* [A] "A in first position (and anything after)"
* [null, B] "anything in first position AND B in second position (and anything after)"
* [A, B] "A in first position AND B in second position (and anything after)"
* [[A, B], [A, B]] "(A OR B) in first position AND (A OR B) in second position (and anything after)"

##### Parameters

1. Object - The filter options:

* fromBlock: QUANTITY|TAG - (optional, default: "latest") Integer block number, or "latest" for the last mined block or "pending", "earliest" for not yet mined transactions.
* toBlock: QUANTITY|TAG - (optional, default: "latest") Integer block number, or "latest" for the last mined block or "pending", "earliest" for not yet mined transactions.
* address: DATA|Array, 20 Bytes - (optional) Contract address or a list of addresses from which logs should originate.
* topics: Array of DATA, - (optional) Array of 32 Bytes DATA topics. Topics are order-dependent. Each topic can also be an array of DATA with "or" options.

##### Example Parameters

params: [{

"fromBlock": "0x1",

"toBlock": "0x2",

"address": "0x8888f1f195afa192cfee860698584c030f4c9db1",

"topics": ["0x000000000000000000000000a94f5374fce5edbc8e2a8697c15331677e6ebf0b", null, ["0x000000000000000000000000a94f5374fce5edbc8e2a8697c15331677e6ebf0b", "0x0000000000000000000000000aff3454fce5edbc8cca8697c15331677e6ebccc"]]

}]

##### Returns

QUANTITY - A filter id.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_newFilter","params":[{"topics":["0x0000000000000000000000000000000000000000000000000000000012341234"]}],"id":73}'

// Result

{

"id":1,

"jsonrpc": "2.0",

"result": "0x1" // 1

}

#### ulam\_newBlockFilter

Creates a filter in the node, to notify when a new block arrives. To check if the state has changed, call [ulam\_getFilterChanges](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getfilterchanges).

##### Parameters

None

##### Returns

QUANTITY - A filter id.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_newBlockFilter","params":[],"id":73}'

// Result

{

"id":1,

"jsonrpc": "2.0",

"result": "0x1" // 1

}

#### ulam\_newPendingTransactionFilter

Creates a filter in the node, to notify when new pending transactions arrive. To check if the state has changed, call [ulam\_getFilterChanges](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getfilterchanges).

##### Parameters

None

##### Returns

QUANTITY - A filter id.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_newPendingTransactionFilter","params":[],"id":73}'

// Result

{

"id":1,

"jsonrpc": "2.0",

"result": "0x1" // 1

}

#### ulam\_uninstallFilter

Uninstalls a filter with given id. Should always be called when watch is no longer needed. Additonally Filters timeout when they aren't requested with [ulam\_getFilterChanges](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getfilterchanges) for a period of time.

##### Parameters

1. QUANTITY - The filter id.

##### Example Parameters

params: [

"0xb" // 11

]

##### Returns

Boolean - true if the filter was successfully uninstalled, otherwise false.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_uninstallFilter","params":["0xb"],"id":73}'

// Result

{

"id":1,

"jsonrpc": "2.0",

"result": true

}

#### ulam\_getFilterChanges

Polling method for a filter, which returns an array of logs which occurred since last poll.

##### Parameters

1. QUANTITY - the filter id.

##### Example Parameters

params: [

"0x16" // 22

]

##### Returns

Array - Array of log objects, or an empty array if nothing has changed since last poll.

For filters created with ulam\_newBlockFilter the return are block hashes (DATA, 32 Bytes), e.g. ["0x3454645634534..."].

For filters created with ulam\_newPendingTransactionFilter the return are transaction hashes (DATA, 32 Bytes), e.g. ["0x6345343454645..."].

For filters created with ulam\_newFilter logs are objects with following params:

* + removed: TAG - true when the log was removed, due to a chain reorganization. false if its a valid log.
  + logIndex: QUANTITY - integer of the log index position in the block. null when its pending log.
  + transactionIndex: QUANTITY - integer of the transactions index position log was created from. null when its pending log.
  + transactionHash: DATA, 32 Bytes - hash of the transactions this log was created from. null when its pending log.
  + blockHash: DATA, 32 Bytes - hash of the block where this log was in. null when its pending. null when its pending log.
  + blockNumber: QUANTITY - the block number where this log was in. null when its pending. null when its pending log.
  + address: DATA, 20 Bytes - address from which this log originated.
  + data: DATA - contains the non-indexed arguments of the log.
  + topics: Array of DATA - Array of 0 to 4 32 Bytes DATA of indexed log arguments. (In solidity: The first topic is the hash of the signature of the event (e.g. Deposit(address,bytes32,uint256)), except you declared the event with the anonymous specifier.)

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_getFilterChanges","params":["0x16"],"id":73}'

// Result

{

"id":1,

"jsonrpc":"2.0",

"result": [{

"logIndex": "0x1", // 1

"blockNumber":"0x1b4", // 436

"blockHash": "0x8216c5785ac562ff41e2dcfdf5785ac562ff41e2dcfdf829c5a142f1fccd7d",

"transactionHash": "0xdf829c5a142f1fccd7d8216c5785ac562ff41e2dcfdf5785ac562ff41e2dcf",

"transactionIndex": "0x0", // 0

"address": "0x16c5785ac562ff41e2dcfdf829c5a142f1fccd7d",

"data":"0x0000000000000000000000000000000000000000000000000000000000000000",

"topics": ["0x59ebeb90bc63057b6515673c3ecf9438e5058bca0f92585014eced636878c9a5"]

},{

...

}]

}

#### ulam\_getFilterLogs

Returns an array of all logs matching filter with given id.

##### Parameters

1. QUANTITY - The filter id.

##### Example Parameters

params: [

"0x16" // 22

]

##### Returns

See [ulam\_getFilterChanges](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getfilterchanges)

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_getFilterLogs","params":["0x16"],"id":74}'

Result see [ulam\_getFilterChanges](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getfilterchanges)

#### ulam\_getLogs

Returns an array of all logs matching a given filter object.

##### Parameters

1. Object - The filter options:

* fromBlock: QUANTITY|TAG - (optional, default: "latest") Integer block number, or "latest" for the last mined block or "pending", "earliest" for not yet mined transactions.
* toBlock: QUANTITY|TAG - (optional, default: "latest") Integer block number, or "latest" for the last mined block or "pending", "earliest" for not yet mined transactions.
* address: DATA|Array, 20 Bytes - (optional) Contract address or a list of addresses from which logs should originate.
* topics: Array of DATA, - (optional) Array of 32 Bytes DATA topics. Topics are order-dependent. Each topic can also be an array of DATA with "or" options.
* blockhash: DATA, 32 Bytes - (optional) With the addition of EIP-234 (Gulam >= v1.8.13 or Parity >= v2.1.0), blockHash is a new filter option which restricts the logs returned to the single block with the 32-byte hash blockHash. Using blockHash is equivalent to fromBlock = toBlock = the block number with hash blockHash. If blockHash is present in the filter criteria, then neither fromBlock nor toBlock are allowed.

##### Example Parameters

params: [{

"topics": ["0x000000000000000000000000a94f5374fce5edbc8e2a8697c15331677e6ebf0b"]

}]

##### Returns

See [ulam\_getFilterChanges](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getfilterchanges)

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_getLogs","params":[{"topics":["0x000000000000000000000000a94f5374fce5edbc8e2a8697c15331677e6ebf0b"]}],"id":74}'

Result see [ulam\_getFilterChanges](https://github.com/ethereum/wiki/wiki/JSON-RPC#eth_getfilterchanges)

#### ulam\_getWork

Returns the hash of the current block, the seedHash, and the boundary condition to be met ("target").

##### Parameters

none

##### Returns

Array - Array with the following properties:

1. DATA, 32 Bytes - current block header pow-hash
2. DATA, 32 Bytes - the seed hash used for the DAG.
3. DATA, 32 Bytes - the boundary condition ("target"), 2^256 / difficulty.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_getWork","params":[],"id":73}'

// Result

{

"id":1,

"jsonrpc":"2.0",

"result": [

"0x1234567890abcdef1234567890abcdef1234567890abcdef1234567890abcdef",

"0x5EED00000000000000000000000000005EED0000000000000000000000000000",

"0xd1ff1c01710000000000000000000000d1ff1c01710000000000000000000000"

]

}

#### ulam\_submitWork

Used for submitting a proof-of-work solution.

##### Parameters

1. DATA, 8 Bytes - The nonce found (64 bits)
2. DATA, 32 Bytes - The header's pow-hash (256 bits)
3. DATA, 32 Bytes - The mix digest (256 bits)

##### Example Parameters

params: [

"0x0000000000000001",

"0x1234567890abcdef1234567890abcdef1234567890abcdef1234567890abcdef",

"0xD1FE5700000000000000000000000000D1FE5700000000000000000000000000"

]

##### Returns

Boolean - returns true if the provided solution is valid, otherwise false.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0", "method":"ulam\_submitWork", "params":["0x0000000000000001", "0x1234567890abcdef1234567890abcdef1234567890abcdef1234567890abcdef", "0xD1GE5700000000000000000000000000D1GE5700000000000000000000000000"],"id":73}'

// Result

{

"id":73,

"jsonrpc":"2.0",

"result": true

}

#### ulam\_submitHashrate

Used for submitting mining hashrate.

##### Parameters

1. Hashrate, a hexadecimal string representation (32 bytes) of the hash rate
2. ID, String - A random hexadecimal(32 bytes) ID identifying the client

##### Example Parameters

params: [

"0x0000000000000000000000000000000000000000000000000000000000500000",

"0x59daa26581d0acd1fce254fb7e85952f4c09d0915afd33d3886cd914bc7d283c"

]

##### Returns

Boolean - returns true if submitting went through succesfully and false otherwise.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0", "method":"ulam\_submitHashrate", "params":["0x0000000000000000000000000000000000000000000000000000000000500000", "0x59daa26581d0acd1fce254fb7e85952f4c09d0915afd33d3886cd914bc7d283c"],"id":73}'

// Result

{

"id":73,

"jsonrpc":"2.0",

"result": true

}

#### ulam\_getProof

Returns the account- and storage-values of the specified account including the Merkle-proof.

##### getProof-Parameters

1. DATA, 20 bytes - address of the account or contract
2. ARRAY, 32 Bytes - array of storage-keys which should be proofed and included. See ulam\_getStorageAt
3. QUANTITY|TAG - integer block number, or the string "latest" or "earliest", see the default block parameter

##### Example Parameters

params: ["0x1234567890123456789012345678901234567890",["0x0000000000000000000000000000000000000000000000000000000000000000","0x0000000000000000000000000000000000000000000000000000000000000001"],"latest"]

##### getProof-Returns

Returns Object - A account object:

balance: QUANTITY - the balance of the account. See ulam\_getBalance

codeHash: DATA, 32 Bytes - hash of the code of the account. For a simple Account without code it will return "0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470"

nonce: QUANTITY, - nonce of the account. See ulam\_getTransactionCount

storageHash: DATA, 32 Bytes - SHA3 of the StorageRoot. All storage will deliver a MerkleProof starting with this rootHash.

accountProof: ARRAY - Array of rlp-serialized MerkleTree-Nodes, starting with the stateRoot-Node, following the path of the SHA3 (address) as key.

storageProof: ARRAY - Array of storage-entries as requested. Each entry is a object with these properties:

key: QUANTITY - the requested storage key value: QUANTITY - the storage value proof: ARRAY - Array of rlp-serialized MerkleTree-Nodes, starting with the storageHash-Node, following the path of the SHA3 (key) as path.

##### getProof-Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"ulam\_getProof","params":["0x1234567890123456789012345678901234567890",["0x0000000000000000000000000000000000000000000000000000000000000000","0x0000000000000000000000000000000000000000000000000000000000000001"],"latest"],"id":1}' -H "Content-type:application/json" http://localhost:8545

// Result

{

"jsonrpc": "2.0",

"id": 1,

"result": {

"address": "0x1234567890123456789012345678901234567890",

"accountProof": [

"",

"",

"",

"0xf851808080a009833150c367df138f1538689984b8a84fc55692d3d41fe4d1e5720ff5483a6980808080808080808080a0a319c1c415b271afc0adcb664e67738d103ac168e0bc0b7bd2da7966165cb9518080"

],

"balance": "0x0",

"codeHash": "0xc5d2460186f7233c927e7db2dcc703c0e500b653ca82273b7bfad8045d85a470",

"nonce": "0x0",

"storageHash": "0x56e81f171bcc55a6ff8345e692c0f86e5b48e01b996cadc001622fb5e363b421",

"storageProof": [

{

"key": "0x0000000000000000000000000000000000000000000000000000000000000000",

"value": "0x0",

"proof": []

},

{

"key": "0x0000000000000000000000000000000000000000000000000000000000000001",

"value": "0x0",

"proof": []

}

]

}

}

#### db\_putString

Stores a string in the local database.

**Note** this function is deprecated and will be removed in the future.

##### Parameters

1. String - Database name.
2. String - Key name.
3. String - String to store.

##### Example Parameters

params: [

"testDB",

"myKey",

"myString"

]

##### Returns

Boolean - returns true if the value was stored, otherwise false.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"db\_putString","params":["testDB","myKey","myString"],"id":73}'

// Result

{

"id":1,

"jsonrpc":"2.0",

"result": true

}

#### db\_getString

Returns string from the local database.

**Note** this function is deprecated and will be removed in the future.

##### Parameters

1. String - Database name.
2. String - Key name.

##### Example Parameters

params: [

"testDB",

"myKey",

]

##### Returns

String - The previously stored string.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"db\_getString","params":["testDB","myKey"],"id":73}'

// Result

{

"id":1,

"jsonrpc":"2.0",

"result": "myString"

}

#### db\_putHex

Stores binary data in the local database.

**Note** this function is deprecated and will be removed in the future.

##### Parameters

1. String - Database name.
2. String - Key name.
3. DATA - The data to store.

##### Example Parameters

params: [

"testDB",

"myKey",

"0x68656c6c6f20776f726c64"

]

##### Returns

Boolean - returns true if the value was stored, otherwise false.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"db\_putHex","params":["testDB","myKey","0x68656c6c6f20776f726c64"],"id":73}'

// Result

{

"id":1,

"jsonrpc":"2.0",

"result": true

}

#### db\_gulamex

Returns binary data from the local database.

**Note** this function is deprecated and will be removed in the future.

##### Parameters

1. String - Database name.
2. String - Key name.

##### Example Parameters

params: [

"testDB",

"myKey",

]

##### Returns

DATA - The previously stored data.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"db\_gulamex","params":["testDB","myKey"],"id":73}'

// Result

{

"id":1,

"jsonrpc":"2.0",

"result": "0x68656c6c6f20776f726c64"

}

#### shh\_version

Returns the current whisper protocol version.

##### Parameters

none

##### Returns

String - The current whisper protocol version

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"shh\_version","params":[],"id":67}'

// Result

{

"id":67,

"jsonrpc": "2.0",

"result": "2"

}

#### shh\_post

Sends a whisper message.

##### Parameters

1. Object - The whisper post object:

* from: DATA, 60 Bytes - (optional) The identity of the sender.
* to: DATA, 60 Bytes - (optional) The identity of the receiver. When present whisper will encrypt the message so that only the receiver can decrypt it.
* topics: Array of DATA - Array of DATA topics, for the receiver to identify messages.
* payload: DATA - The payload of the message.
* priority: QUANTITY - The integer of the priority in a range from ... (?).
* ttl: QUANTITY - integer of the time to live in seconds.

##### Example Parameters

params: [{

from: "0x04f96a5e25610293e42a73908e93ccc8c4d4dc0edcfa9fa872f50cb214e08ebf61a03e245533f97284d442460f2998cd41858798ddfd4d661997d3940272b717b1",

to: "0x3e245533f97284d442460f2998cd41858798ddf04f96a5e25610293e42a73908e93ccc8c4d4dc0edcfa9fa872f50cb214e08ebf61a0d4d661997d3940272b717b1",

topics: ["0x776869737065722d636861742d636c69656e74", "0x4d5a695276454c39425154466b61693532"],

payload: "0x7b2274797065223a226d6",

priority: "0x64",

ttl: "0x64",

}]

##### Returns

Boolean - returns true if the message was send, otherwise false.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"shh\_post","params":[{"from":"0xc931d93e97ab07fe42d923478ba2465f2..","topics": ["0x68656c6c6f20776f726c64"],"payload":"0x68656c6c6f20776f726c64","ttl":0x64,"priority":0x64}],"id":73}'

// Result

{

"id":1,

"jsonrpc":"2.0",

"result": true

}

#### shh\_newIdentity

Creates new whisper identity in the client.

##### Parameters

none

##### Returns

DATA, 60 Bytes - the address of the new identiy.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"shh\_newIdentity","params":[],"id":73}'

// Result

{

"id":1,

"jsonrpc": "2.0",

"result": "0xc931d93e97ab07fe42d923478ba2465f283f440fd6cabea4dd7a2c807108f651b7135d1d6ca9007d5b68aa497e4619ac10aa3b27726e1863c1fd9b570d99bbaf"

}

#### shh\_hasIdentity

Checks if the client hold the private keys for a given identity.

##### Parameters

1. DATA, 60 Bytes - The identity address to check.

##### Example Parameters

params: [ "0x04f96a5e25610293e42a73908e93ccc8c4d4dc0edcfa9fa872f50cb214e08ebf61a03e245533f97284d442460f2998cd41858798ddfd4d661997d3940272b717b1"

]

##### Returns

Boolean - returns true if the client holds the privatekey for that identity, otherwise false.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"shh\_hasIdentity","params":["0x04f96a5e25610293e42a73908e93ccc8c4d4dc0edcfa9fa872f50cb214e08ebf61a03e245533f97284d442460f2998cd41858798ddfd4d661997d3940272b717b1"],"id":73}'

// Result

{

"id":1,

"jsonrpc": "2.0",

"result": true

}

#### shh\_newGroup

Creates a new group.

##### Parameters

none

##### Returns

DATA, 60 Bytes - the address of the new group.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"shh\_newGroup","params":[],"id":73}'

// Result

{

"id":1,

"jsonrpc": "2.0",

"result": "0xc65f283f440fd6cabea4dd7a2c807108f651b7135d1d6ca90931d93e97ab07fe42d923478ba2407d5b68aa497e4619ac10aa3b27726e1863c1fd9b570d99bbaf"

}

#### shh\_addToGroup

Adds a whisper identity to the group.

##### Parameters

1. DATA, 60 Bytes - The identity address to add to a group.

##### Example Parameters

params: [ "0x04f96a5e25610293e42a73908e93ccc8c4d4dc0edcfa9fa872f50cb214e08ebf61a03e245533f97284d442460f2998cd41858798ddfd4d661997d3940272b717b1"

]

##### Returns

Boolean - returns true if the identity was successfully added to the group, otherwise false.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"shh\_addToGroup","params":["0x04f96a5e25610293e42a73908e93ccc8c4d4dc0edcfa9fa872f50cb214e08ebf61a03e245533f97284d442460f2998cd41858798ddfd4d661997d3940272b717b1"],"id":73}'

// Result

{

"id":1,

"jsonrpc": "2.0",

"result": true

}

#### shh\_newFilter

Creates filter to notify, when client receives whisper message matching the filter options.

##### Parameters

1. Object - The filter options:

* to: DATA, 60 Bytes - (optional) Identity of the receiver. When present it will try to decrypt any incoming message if the client holds the private key to this identity.
* topics: Array of DATA - Array of DATA topics which the incoming message's topics should match. You can use the following combinations:
  + [A, B] = A && B
  + [A, [B, C]] = A && (B || C)
  + [null, A, B] = ANYTHING && A && B null works as a wildcard

##### Example Parameters

params: [{

"topics": ['0x12341234bf4b564f'],

"to": "0x04f96a5e25610293e42a73908e93ccc8c4d4dc0edcfa9fa872f50cb214e08ebf61a03e245533f97284d442460f2998cd41858798ddfd4d661997d3940272b717b1"

}]

##### Returns

QUANTITY - The newly created filter.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"shh\_newFilter","params":[{"topics": ['0x12341234bf4b564f'],"to": "0x2341234bf4b2341234bf4b564f..."}],"id":73}'

// Result

{

"id":1,

"jsonrpc":"2.0",

"result": "0x7" // 7

}

#### shh\_uninstallFilter

Uninstalls a filter with given id. Should always be called when watch is no longer needed. Additonally Filters timeout when they aren't requested with [shh\_getFilterChanges](https://github.com/ethereum/wiki/wiki/JSON-RPC#shh_getfilterchanges) for a period of time.

##### Parameters

1. QUANTITY - The filter id.

##### Example Parameters

params: [

"0x7" // 7

]

##### Returns

Boolean - true if the filter was successfully uninstalled, otherwise false.

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"shh\_uninstallFilter","params":["0x7"],"id":73}'

// Result

{

"id":1,

"jsonrpc":"2.0",

"result": true

}

#### shh\_getFilterChanges

Polling method for whisper filters. Returns new messages since the last call of this method.

**Note** calling the [shh\_getMessages](https://github.com/ethereum/wiki/wiki/JSON-RPC#shh_getmessages) method, will reset the buffer for this method, so that you won't receive duplicate messages.

##### Parameters

1. QUANTITY - The filter id.

##### Example Parameters

params: [

"0x7" // 7

]

##### Returns

Array - Array of messages received since last poll:

* hash: DATA, 32 Bytes (?) - The hash of the message.
* from: DATA, 60 Bytes - The sender of the message, if a sender was specified.
* to: DATA, 60 Bytes - The receiver of the message, if a receiver was specified.
* expiry: QUANTITY - Integer of the time in seconds when this message should expire (?).
* ttl: QUANTITY - Integer of the time the message should float in the system in seconds (?).
* sent: QUANTITY - Integer of the unix timestamp when the message was sent.
* topics: Array of DATA - Array of DATA topics the message contained.
* payload: DATA - The payload of the message.
* workProved: QUANTITY - Integer of the work this message required before it was send (?).

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"shh\_getFilterChanges","params":["0x7"],"id":73}'

// Result

{

"id":1,

"jsonrpc":"2.0",

"result": [{

"hash": "0x33eb2da77bf3527e28f8bf493650b1879b08c4f2a362beae4ba2f71bafcd91f9",

"from": "0x3ec052fc33..",

"to": "0x87gdf76g8d7fgdfg...",

"expiry": "0x54caa50a", // 1422566666

"sent": "0x54ca9ea2", // 1422565026

"ttl": "0x64", // 100

"topics": ["0x6578616d"],

"payload": "0x7b2274797065223a226d657373616765222c2263686...",

"workProved": "0x0"

}]

}

#### shh\_getMessages

Get all messages matching a filter. Unlike shh\_getFilterChanges this returns all messages.

##### Parameters

1. QUANTITY - The filter id.

##### Example Parameters

params: [

"0x7" // 7

]

##### Returns

See [shh\_getFilterChanges](https://github.com/ethereum/wiki/wiki/JSON-RPC#shh_getfilterchanges)

##### Example

// Request

curl -X POST --data '{"jsonrpc":"2.0","method":"shh\_getMessages","params":["0x7"],"id":73}'