ARIA Chain: Ethereum JSON-RPC API Reference

The ARIA Chain provides a full Ethereum-compatible RPC interface via the Frontier pallet. This comprehensive guide details all available methods, parameters, and specifications for developers integrating with ARIA Chain.

Endpoint Access & Connection

HTTP RPC Endpoint

Connect to the ARIA Chain network using the official RPC endpoint:

URL: **http://rpc.ariaexp.com/**

Chain ID: 134235

Network: ARIA Chain

Compatibility

The ARIA Chain Frontier pallet enables seamless integration with standard Ethereum libraries and development tools.

- Web3.js and Ethers.js libraries
- MetaMask wallet integration
- Remix and Hardhat development tools
- No code modifications required

Account & State Methods

Query account balances, transaction nonces, and smart contract code using these fundamental Ethereum RPC methods.

eth_getBalance

Parameters: Address, Block

Returns the balance of an account in Wei units.

eth_getTransactionCount

Parameters: Address, Block

Returns the number of transactions sent from an address (the Nonce).

eth_getCode

Parameters: Address, Block

Returns the compiled bytecode of a smart contract at the specified address.

eth_getStorageAt

Parameters: Address, Position, Block

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

Block & Network Query Methods

Retrieve information about the blockchain's current state, block history, and network configuration using these query methods.

Method	Parameters	Description
eth_blockNumber	None	Returns the number of the most recent block
eth_getBlockByHash	Hash, FullTx(bool)	Returns block information identified by hash
eth_getBlockByNumber	Tag/Num, FullTx(bool)	Returns block information identified by number
eth_getBlockTransactionCountByHash	Hash	Returns the number of transactions in a block by hash
eth_getBlockTransactionCountByNumber	Block	Returns the number of transactions in a block by number
net_version	None	Returns the current network ID (Chain ID)
eth_chainId	None	Returns the EIP-155 Chain ID

Transaction Methods

Send transactions to the network and query their status using these essential methods for on-chain operations.

1

eth_sendRawTransaction

Submits a pre-signed transaction to the network. This is the primary method used by MetaMask and backend signers.

Parameters: [SignedTxData]

Returns: TransactionHash

2

eth_getTransactionByHash

Returns the complete details of a transaction including sender, recipient, value, and gas parameters.

Parameters: [TxHash]

3

eth_getTransactionReceipt

Returns the receipt of a transaction including status, gas used, and event logs. **Note:** Receipts are only available after a transaction is mined.

Parameters: [TxHash]

4

eth_estimateGas

Estimates the gas required for a transaction before sending it to the network, helping to avoid out-of-gas errors.

Parameters: [TransactionCallObject]

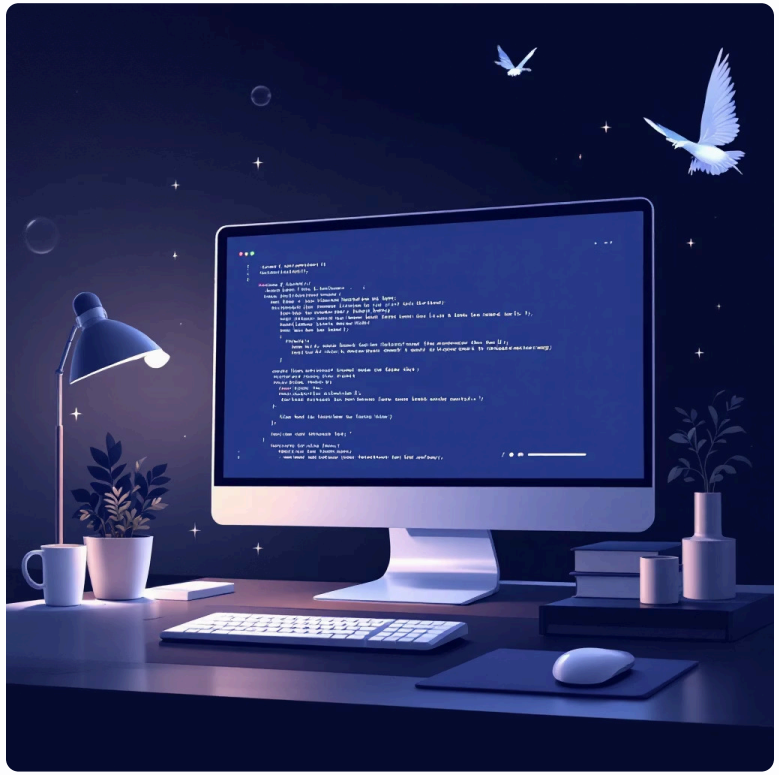
Smart Contract Execution (Read-Only)

eth_call

Executes a message call immediately on the local node without creating a transaction on the blockchain. This method is used for reading view or pure functions from smart contracts.

Parameters: [CallObject, Block]

Use Case: Query contract state without spending gas or modifying blockchain data. Ideal for fetching balances, token information, or any read-only operation.



Event & Filter Methods

DApps use these methods to listen for and process smart contract events emitted using the `emit` statement in Solidity.

eth_newFilter

Creates a filter object to notify when state changes occur (logs)

eth_getFilterLogs

Returns an array of all logs matching a filter ID

eth_getLogs

Returns logs matching a filter object (Stateless)

eth_uninstallFilter

Removes a filter with the given ID

Technical Specifications & Conversions

ARIA Chain follows standard Ethereum units for all RPC calls, ensuring compatibility with existing tooling and libraries.

Gas

Gas values are represented as hexadecimal values in all RPC responses and requests.

Value (Currency)

Currency amounts are represented in Wei units, where 10^{18} Wei = 1 ARIA.

Block Tags

Supports "latest", "earliest", and "pending" block tags for flexible querying.

Example: Get Block Number

Request (Bash)

```
curl -X POST -H "Content-Type: application/json" --data '{"jsonrpc":"2.0","method":"eth_blockNumber","params": [],"id":1}' http://localhost:9944
```

Response (JSON)

```
{
 "jsonrpc": "2.0",
 "result": "0x51c",
 "id": 1
}
```

RPC Error Codes

ARIA Chain returns standard Ethereum RPC error codes. If an integration fails, check the `error` object in the JSON response for troubleshooting.

Code	Message	Description
-32700	Parse error	Invalid JSON was received by the server
-32600	Invalid Request	The JSON sent is not a valid Request object
-32601	Method not found	The method does not exist or is not available
-32602	Invalid params	Invalid method parameter(s)
-32000	Server error	Generic error (often related to insufficient funds or gas)
-32010	Execution reverted	The EVM transaction reverted (check contract logic)

Developer Quick-Start Guide

Connect a frontend application to ARIA Chain using the following provider setup with Web3.js:

Web3.js Example

```
const Web3 = require('web3');
// Connect to your ARIA Node
const web3 = new Web3('http://localhost:9944');

async function getInfo() {
    const block = await web3.eth.getBlockNumber();
    console.log("Current ARIA Block:", block);
}
getInfo();
```

Protocol Mapping Notes

Since ARIA is a Substrate chain, internal mappings ensure RPC consistency:

- **Gas Limit:** Mapped from Substrate Weight
- **Gas Price:** Mapped from Substrate base fee/weight fee
- **Transaction Hash:** The Ethereum transaction hash is a truncated version of the Substrate extrinsic hash or computed specifically via the Frontier pallet to match the 32-byte format