QNS JSON-RPC API Documentation

Version: 2.0.0
Last Updated: 2026-02-05
Total Methods: 252+
WebSocket Subscriptions: 28

Overview

QNS provides a complete JSON-RPC 2.0 API, Ethereum-compatible, with extensions for:
- Post-quantum cryptography (Dilithium3, Kyber1024)
- Multi-VM architecture (EVM, TVM, QVM)
- Neural DAG with GHOSTDAG ordering (10 clusters)
- Cross-shard operations
- DeFi primitives (AMM, Lending, Flash Loans with 0.09% fee)
- Neural AI components (BlockchainBrain)
- NFT and tokens (ERC-20, ERC-721, ERC-1155)
- Security monitoring and MEV protection

Endpoints

Pyuniq Testnet (Primary)

```
HTTP:      https://node.qsnchain.com/testnet/
WebSocket: wss://node.qsnchain.com/testnet/ws
Chain ID:  99991
Symbol:    tQNS
```

Mainnet (Coming Soon)

```
HTTP:      https://node.qsnchain.com/qvm/mainnet/
WebSocket: wss://node.qsnchain.com/qvm/mainnet/ws
Chain ID:  1
Symbol:    QNS
```

Devnet

```
HTTP:      https://node.qsnchain.com/devnet/
WebSocket: wss://node.qsnchain.com/devnet/ws
Chain ID:  99992
Symbol:    dQNS
```

Ethereum-Compatible Methods (eth_*)

Blockchain Information

Method	Description	Parameters
`eth_blockNumber`	Current block number	-
`eth_chainId`	Network chain ID	-
`eth_gasPrice`	Current gas price in wei	-
`eth_syncing`	Sync status	-

Blocks

Method	Description	Parameters
`eth_getBlockByNumber`	Block by number	`blockNumber`, `fullTxs`
`eth_getBlockByHash`	Block by hash	`blockHash`, `fullTxs`
`eth_getBlockTransactionCountByNumber`	Transaction count in block	`blockNumber`
`eth_getBlockTransactionCountByHash`	Transaction count by block hash	`blockHash`

Accounts

Method	Description	Parameters
`eth_getBalance`	Address balance	`address`, `blockTag`
`eth_getTransactionCount`	Address nonce	`address`, `blockTag`
`eth_getCode`	Contract code	`address`, `blockTag`
`eth_getStorageAt`	Storage slot value	`address`, `slot`, `blockTag`
`eth_getProof`	Merkle proof (EIP-1186)	`address`, `storageKeys`, `blockTag`
`eth_accounts`	Account list	-

Transactions

Method	Description	Parameters
`eth_sendRawTransaction`	Send signed transaction	`signedTxData`
`eth_sendTransaction`	Send transaction	`txObject`
`eth_getTransactionByHash`	Transaction by hash	`txHash`
`eth_getTransactionReceipt`	Transaction receipt	`txHash`
`eth_getTransactionByBlockHashAndIndex`		

Method	Description	Parameters
	Transaction by block and index	`blockHash`, `index`
`eth_getTransactionByBlockNumberAndIndex`	Transaction by block number and index	`blockNumber`, `index`

Calls and Gas Estimation

Method	Description	Parameters
`eth_call`	Read-only contract call	`callObject`, `blockTag`
`eth_estimateGas`	Gas estimation for transaction	`txObject`

Logs and Filters

Method	Description	Parameters
`eth_getLogs`	Logs by filter	`filterObject`
`eth_newFilter`	Create log filter	`filterObject`
`eth_newBlockFilter`	Filter for new blocks	-
`eth_newPendingTransactionFilter`	Filter for pending transactions	-
`eth_getFilterChanges`	Filter changes	`filterId`
`eth_getFilterLogs`	All logs by filter	`filterId`
`eth_uninstallFilter`	Remove filter	`filterId`

QNS-Specific Methods (qsn_*)

Network and Blockchain Information

Method	Description
`qsn_chainInfo`	Full network information
`qsn_getNetworkStats`	Network statistics
`qsn_getNetworkStatsExtended`	Extended statistics
`qsn_getTpsMetrics`	TPS metrics (200,000+ target)
`qsn_getGasStats`	Gas statistics
`qsn_getSyncStatus`	Sync status
`qsn_health`	Health check

Accounts and Wallets

Method	Description
`qsn_getBalance`	Address balance
`qsn_getAccountInfo`	Full account information
`qsn_createWallet`	Create wallet (returns 3 addresses: quant, evm, ton)
`qsn_getQuantAddress`	Convert to quant-address

qsn_createWallet

Creates a new wallet with hybrid cryptography (Ed25519 + Dilithium3) and returns **all three address formats**:

Response:

```
{
  "success": true,
  "addresses": {
    "quant": "quant27N2UqA8inQRFGAD8yXXSKffh1vr2kx7ngYQoWNeYdyjvSB9xC",
    "evm": "0xd50c170a57621b05b6889ae39e824bd51eb88130",
    "ton": "EQBvW8Z5huBkMJYdnfAEM5JqTNLuuP..."
  },
  "quantAddress": "quant...",
  "evmAddress": "0x...",
  "tonAddress": "EQ...",
  "privateKey": "0x...",
  "publicKey": "0x...",
  "quantumPublicKey": "0x...",
  "quantumSecretKey": "0x...",
  "algorithm": {
    "classical": "Ed25519",
    "quantum": "Dilithium3 (NIST Level 3)"
  }
}
```

Address formats:
- `quant...` - QVM/QRC-20 (workchain 0)
- `0x...` - EVM/ERC-20 (workchain 1)
- `EQ.../UQ.../0:hex` - TVM/Jetton TEP-74 (workchain 2)

Tokens (Multi-VM: QRC-20, ERC-20, Jetton)

Method	Description	VM/Workchain
`qsn_deployToken`	Deploy QRC-20 token	QVM (workchain 0)
`qsn_deployErc20`	**Deploy ERC-20 token**	**EVM (workchain 1)**
`qsn_deployJetton`	**Deploy Jetton token (TEP-74)**	**TVM (workchain 2)**
`qsn_tokenTransfer`	Token transfer	All

Method	Description	VM/Workchain
`qsn_tokenBalanceOf`	Token balance	All
`qsn_getTokenInfo`	Token information	All

qsn_deployErc20

Deploy ERC-20 token on EVM workchain with address format `0x...`

Parameters:

```
{
  "from": "0x... or quant...",
  "name": "My Token",
  "symbol": "MTK",
  "initialSupply": "1000000",
  "decimals": 18,
  "signature": "0x...",
  "publicKey": "0x...",
  "quantumSignature": "0x...",
  "quantumPublicKey": "0x..."
}
```

Response:

```
{
  "success": true,
  "txHash": "0x...",
  "contractAddress": "0x...",
  "vmType": "EVM",
  "workchainId": 1
}
```

qsn_deployJetton

Deploy Jetton token (TEP-74) on TVM workchain with address format `EQ.../UQ...`

Parameters:

```
{
  "from": "EQ... or quant...",
  "name": "My Jetton",
  "symbol": "MJT",
  "initialSupply": "1000000",
  "decimals": 9,
  "signature": "0x...",
  "publicKey": "0x...",
  "quantumSignature": "0x...",
  "quantumPublicKey": "0x..."
}
```

Response:

```
{
  "success": true,
  "txHash": "0x...",
  "contractAddress": "EQ...",
  "vmType": "TVM",
  "workchainId": 2,
  "standard": "TEP-74"
}
```

Staking and Validators

Method	Description
`qsn_stake`	Stake tokens
`qsn_unstake`	Unstake tokens
`qsn_delegate`	Delegate tokens
`qsn_claimRewards`	Claim rewards
`qsn_getStakingInfo`	Staking information
`qsn_getValidators`	Validator list (171 total)
`qsn_getActiveValidators`	Active validators
`qsn_getValidatorInfo`	Validator information
`qsn_getNeuralValidators`	Neural validators (50)
`qsn_getValidatorsByTier`	Validators by tier

Three-Tier Validator System

Tier	Minimum	Reward	Max Validators
Execution	10,000 QNS	1.0x	100
Neural	50,000 QNS	1.5x	50
Verification	100,000 QNS	2.0x	21

Total: 513 max validators (300+150+63)

Neural/AI Methods (BlockchainBrain)

Method	Description
`qsn_getNeuralState`	Neural network state
`qsn_getBrainMetrics`	BlockchainBrain metrics
`qsn_getPredictions`	AI predictions

Method	Description
`qsn_getReasoningTrace`	Reasoning trace
`qsn_analyzeTransaction`	AI transaction analysis
`qsn_predictGas`	AI gas prediction
`qsn_getAIRecommendations`	AI recommendations
`qsn_detectMEV`	MEV detection
`qsn_getAddressRisk`	Address risk scoring

DAG and Clusters (GHOSTDAG)

Method	Description
`qsn_getDAGState`	DAG state
`qsn_getDAGBlock`	DAG block
`qsn_getDAGChildren`	DAG block children
`qsn_getDAGParents`	DAG block parents
`qsn_getClusters`	List of 10 clusters
`qsn_getClusterInfo`	Cluster information
`qsn_getLinearization`	DAG linearization

Security and MEV Protection

Method	Description
`qsn_getMEVStats`	MEV statistics
`qsn_getDetectedAttacks`	Detected attacks
`qsn_getSecurityScore`	Security score
`qsn_getActiveThreats`	Active threats
`qsn_getDefenseActions`	Defense actions
`qsn_getSecurityDashboard`	Security dashboard

QVM Methods (qvm_*)

Basic Operations

Method	Description
`qvm_call`	Contract call

Method	Description
`qvm_estimateGas`	Gas estimation
`qvm_sendRawTransaction`	Send transaction
`qvm_getBalance`	Address balance
`qvm_chainId`	Chain ID

DeFi - AMM Pools

Method	Description
`qvm_defi_createPool`	Create pool
`qvm_defi_getPoolInfo`	Pool information
`qvm_defi_swap`	Token swap
`qvm_defi_addLiquidity`	Add liquidity
`qvm_defi_removeLiquidity`	Remove liquidity

DeFi - Lending

Method	Description
`qvm_defi_createLendingMarket`	Create lending market
`qvm_defi_supply`	Supply liquidity
`qvm_defi_borrow`	Borrow
`qvm_defi_repay`	Repay
`qvm_defi_flashLoanFee`	Flash loan fee (0.09%)

Post-Quantum Cryptography

Method	Description
`qvm_dilithiumSign`	Dilithium3 signature
`qvm_dilithiumVerify`	Dilithium3 verification
`qvm_kyberEncapsulate`	Kyber KEM encapsulation

Bridge Methods (bridge_*)

Method	Description
`bridge_status`	Bridge status
`bridge_deposits`	Deposit list

Method	Description
`bridge_withdrawals`	Withdrawal list
`bridge_initiateDeposit`	Initiate deposit
`bridge_initiateWithdrawal`	Initiate withdrawal
`bridge_getSupportedChains`	Supported chains (10+)
`bridge_estimateFee`	Fee estimation

Supported Chains

Chain	Chain ID	Status
Ethereum	1	Active
Polygon	137	Active
BSC	56	Active
TON	-	Active
Bitcoin	-	Planned

WebSocket Subscriptions (28)

Connection

```
wss://node.qsnchain.com/testnet/ws  # Testnet
wss://node.qsnchain.com/qvm/mainnet/ws          # Mainnet
```

Subscription Types

Basic Events

- `newHeads` — new blocks
- `newPendingTransactions` — pending transactions
- `logs` — contract logs
- `syncing` — sync status

Neural/Brain Monitoring

- `brainState` — brain state
- `brainLearning` — learning events
- `brainPrediction` — predictions
- `neuralActivity` — neural activations

Security and MEV

- `mevEvents` — MEV attack detection
- `threatEvents` — threat detection
- `defenseEvents` — defense actions
- `securityScore` — security score changes

Rate Limiting

Parameter	Value
`max_requests_per_second`	100 req/sec per IP
`max_tx_per_minute_per_address`	20 tx/min
`max_pending_tx`	10,000
`max_faucet_per_day`	1 req/day per address

Error Codes

Standard JSON-RPC

Code	Message	Description
-32700	Parse error	Invalid JSON
-32600	Invalid Request	Invalid request
-32601	Method not found	Method not found
-32602	Invalid params	Invalid parameters
-32603	Internal error	Internal error

QNS-Specific

Code	Message	Description
-32010	Rate limit exceeded	Rate limit exceeded
-32011	Invalid signature	Invalid signature
-32012	Insufficient balance	Insufficient balance
-32016	Cross-shard error	Cross-shard operation error
-32017	Quantum signature required	PQ signature required

Usage Examples

cURL

```
# Get block number
curl -X POST https://node.qsnchain.com/testnet/ \
  -H "Content-Type: application/json" \
  -d '{"jsonrpc":"2.0","id":1,"method":"eth_blockNumber","params":[]}'

# QNS chain info
curl -X POST https://node.qsnchain.com/testnet/ \
  -H "Content-Type: application/json" \
  -d '{"jsonrpc":"2.0","id":1,"method":"qsn_chainInfo","params":[]}'

# Brain metrics
curl -X POST https://node.qsnchain.com/testnet/ \
  -H "Content-Type: application/json" \
  -d '{"jsonrpc":"2.0","id":1,"method":"qsn_getBrainMetrics","params":[]}'
```

JavaScript (ethers.js v6)

```
import { ethers } from 'ethers';

const provider = new ethers.JsonRpcProvider('https://node.qsnchain.com/testnet/');

// Get block number
const blockNumber = await provider.getBlockNumber();

// QNS-specific methods
const chainInfo = await provider.send('qsn_chainInfo', []);
const brainMetrics = await provider.send('qsn_getBrainMetrics', []);
```

Python (web3.py)

```
from web3 import Web3

w3 = Web3(Web3.HTTPProvider('https://node.qsnchain.com/testnet/'))

# Get block number
block_number = w3.eth.block_number

# QNS-specific methods
chain_info = w3.provider.make_request('qsn_chainInfo', [])
brain_metrics = w3.provider.make_request('qsn_getBrainMetrics', [])
```

Key Metrics

Parameter	Value
TPS Target	200,000+
Block Time	1-2 sec
Finality	~3 sec
Max Validators	513 (300+150+63)
DAG Clusters	10
RPC Methods	252+
WebSocket Subscriptions	28
Flash Loan Fee	0.09% (9 BPS)
QVM Opcodes	252+
Post-Quantum	Dilithium3, Kyber1024

Documentation Updated: 2026-02-05