QUANTUM SHARDED NETWORK

(QSN)

JSON-RPC API Documentation

Complete API Reference for Developers

EVM Compatible + Native QSN Methods + QVM

Version 1.0 | December 2025
https://qsnchain.com

1. OVERVIEW

QSN provides a JSON-RPC 2.0 API compatible with Ethereum tools (ethers.js, web3.py, MetaMask) while extending functionality with native post-quantum methods and QVM operations.

1.1 Base Information

Parameter	Value
Protocol	JSON-RPC 2.0
Content-Type	application/json
EVM Compatibility	Full (eth_* methods)
Native Methods	qsn_* namespace
QVM Methods	qvm_* namespace

1.2 Public Endpoints

Network	URL	Chain ID	Token
Mainnet	https://node.qsnchain.com/mainnet/	99990	QSN
Testnet (PyUniq)	https://node.qsnchain.com/testnet/	99991	tQSN
Devnet	https://node.qsnchain.com/devnet/	99992	dQSN
QVM Mainnet	https://node.qsnchain.com/qvm/mainnet/	1	QSN
QVM Testnet	https://node.qsnchain.com/qvm/testnet/	2	QSN

1.3 WebSocket Endpoints

Network	WebSocket URL
Mainnet	wss://node.qsnchain.com/mainnet/ws
Testnet	wss://node.qsnchain.com/testnet/ws
Devnet	wss://node.qsnchain.com/devnet/ws

1.4 P2P Bootstrap Nodes

Network	Multiaddr	IP:Port
Mainnet	/ip4/72.62.49.117/tcp/30300	72.62.49.117:30300
Testnet	/ip4/72.62.49.117/tcp/30303	72.62.49.117:30303
Devnet	/ip4/72.62.49.117/tcp/30304	72.62.49.117:30304

2. CONNECTION EXAMPLES

2.1 HTTP JSON-RPC (cURL)

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

2.2 JavaScript (ethers.js)

```
const { ethers } = require('ethers'); const provider = new
ethers.JsonRpcProvider('https://node.qsnchain.com/testnet/'); const blockNumber = await
provider.getBlockNumber();
```

2.3 Python (web3.py)

```
from web3 import Web3 w3 = Web3(Web3.HTTPProvider('https://node.qsnchain.com/testnet/'))
print('Connected:', w3.is_connected()) print('Block:', w3.eth.block_number)
```

2.4 MetaMask Configuration

Field	Value
Network Name	QSN Testnet
RPC URL	https://node.qsnchain.com/testnet/
Chain ID	99991
Currency Symbol	tQSN
Block Explorer	https://qsnscan.io

3. ETH NAMESPACE (EVM Compatible)

Standard Ethereum methods for compatibility with existing tools and wallets.

eth_chainId

Returns the chain ID in hexadecimal format.

```
Response: {"result":"0x18697"}
```

eth_blockNumber

Returns the current block number.

```
Response: {"result":"0x42a"}
```

eth_gasPrice

Returns current gas price (default 20 Gwei).

```
Response: {"result":"0x4a817c800"}
```

eth_getBalance

Returns account balance in wei. Params: address, block

eth_getTransactionCount

Returns account nonce. Params: address, block

eth_sendRawTransaction

Sends signed transaction. IMPORTANT: Only Legacy TX format (pre-EIP-2930).

eth_call

Executes contract call (read-only).

eth_estimateGas

Estimates gas for transaction.

eth_getBlockByNumber

Returns block by number. Params: blockNumber, fullTxs

eth_getBlockByHash

Returns block by hash. Params: hash, fullTxs

eth_getTransactionByHash

Returns transaction by hash.

eth_getTransactionReceipt

Returns transaction receipt with logs.

eth_getLogs

Returns logs matching filter.

`eth_getCode`

Returns contract bytecode.

`net_version`

Returns network ID.

`net_listening`

Returns true if listening for connections.

`net_peerCount`

Returns number of connected peers.

4. QSN NAMESPACE (Native Methods)

Extended QSN methods with post-quantum signature support.

qsn_chainInfo

Returns complete network information.

```
Response: {"blockNumber":1080,"chainId":99991,"chainIdHex":"0x18697",
"clientVersion":"QSN/1.0.0","networkName":"Quantum Sharded Network Testnet"}
```

qsn_getBalance

Returns extended balance with nonce. Supports quant... and 0x... formats.

```
Response: {"balance":"0x...","balanceDecimal":"1000","formatted":"1000.0 tQSN",
"symbol":"tQSN","decimals":18,"nonce":5}
```

qsn_createWallet

Creates wallet with post-quantum keys (Ed25519 + Dilithium3).

```
Response: {"address":"quant2...","publicKey":"0x...","quantumPublicKey":"0x...",
"quantumKeySize":1952,"signatureType":"hybrid"}
```

qsn_sendTransaction

Sends native tQSN transfer. For amounts >= 1000 tQSN, PQ signature required.

```
Params: {"from":"quant...","to":"quant...","value":"1000000000000000000",
"nonce":0,"signature":"0x...","publicKey":"0x...",
"quantumSignature":"0x...","quantumPublicKey":"0x..."} Signature message:
TRANSFER:{from}:{to}:{value}:{nonce}:{chainId}
```

qsn_faucet

Request test tokens (testnet only). Limit: 1 per 24h per address.

```
Response: {"success":true,"amount":"10000000000000000000","formatted":"10.0 tQSN"}
```

qsn_getValidators

Returns list of active validators.

qsn_reserveNonce / qsn_getNextNonce

Reserve or get next nonce for parallel transactions.

5. QRC-20 TOKEN METHODS

qsn_deployToken

Deploy QRC-20 token. Always requires PQ signature.

```
Params: {"owner":"quant...","name":"Token","symbol":"TKN",
"totalSupply":"1000000000000000000000000","decimals":18,
"signature":"0x...","quantumSignature":"0x..."} Signature:
DEPLOY_TOKEN:{owner}:{name}:{symbol}:{totalSupply}:{nonce}:{chainId}
```

qsn_tokenTransfer

Transfer tokens. PQ required for >= 1000 tokens.

```
Params: {"token":"quant...","from":"quant...","to":"quant...",
"amount":"1000000000000000000","nonce":0,"signature":"0x..."} Signature:
TOKEN_TRANSFER:{token}:{from}:{to}:{amount}:{nonce}:{chainId}
```

qsn_getTokenInfo

Returns token metadata (name, symbol, decimals, totalSupply).

qsn_tokenBalance

Returns token balance. Params: tokenAddress, holderAddress

qsn_tokenApprove

Approve spender for token transfers.

qsn_tokenAllowance

Check approved allowance.

qsn_tokenMint

Mint tokens (owner only).

qsn_tokenBurn

Burn tokens.

qsn_tokenPause

Pause token (owner only).

6. QVM NAMESPACE

QVM (Quantum Virtual Machine) methods with built-in DeFi.

6.1 Core QVM Methods

Method	Description
qvm_chainId	Get QVM chain ID
qvm_gasPrice	Get QVM gas price
qvm_call	Call contract (read-only)
qvm_estimateGas	Estimate gas
qvm_sendRawTransaction	Send transaction
qvm_getTransaction	Get transaction
qvm_getTransactionReceipt	Get receipt
qvm_getBalance	Get balance
qvm_getCode	Get contract code
qvm_compile	Compile Qubit code
qvm_deploy	Deploy contract

6.2 QVM DeFi Methods

Method	Description
qvm_defi_createPool	Create AMM pool
qvm_defi_getPoolInfo	Get pool info
qvm_defi_swap	Execute swap
qvm_defi_addLiquidity	Add liquidity
qvm_defi_removeLiquidity	Remove liquidity
qvm_defi_getPrice	Get price
qvm_defi_supply	Supply to lending
qvm_defi_borrow	Borrow assets
qvm_defi_repay	Repay loan
qvm_defi_flashLoanFee	Flash loan fee (0.05%)

6.3 QVM Cryptography

Method	Description
qvm_dilithiumSign	Sign with Dilithium3
qvm_dilithiumVerify	Verify Dilithium3
qvm_kyberEncapsulate	Kyber KEM
qvm_scanContract	Scan for vulnerabilities

7. POST-QUANTUM SIGNATURES

7.1 When PQ Signatures Required

Operation	Threshold	PQ Required?
tQSN transfer	< 1000 tQSN	No (Ed25519 only)
tQSN transfer	>= 1000 tQSN	Yes (Ed25519 + Dilithium3)
Token deploy	Always	Yes
Token transfer	< 1000 tokens	No
Token transfer	>= 1000 tokens	Yes
NFT deploy	Always	Yes

7.2 Key and Signature Sizes

Algorithm	Public Key	Signature	Level
Ed25519	32 bytes	64 bytes	Classical
Dilithium3	1,952 bytes	3,309 bytes	NIST L3
Hybrid	1,984 bytes	3,373 bytes	Post-Quantum

7.3 Hybrid Signature Example

```
const message = `TRANSFER:${from}:${to}:${value}:${nonce}:${chainId}`; const ed25519Sig = ed25519.sign.detached(message, ed25519Key); const dilithiumSig = dilithium.sign(message, dilithiumKey); const tx = {from, to, value, nonce, signature: '0x' + ed25519Sig.toString('hex'), quantumSignature: '0x' + dilithiumSig.toString('hex')};
```

8. ANALYTICS & SYSTEM METHODS

8.1 Network Analytics

Method	Description
qsn_getNetworkStats	Network statistics
qsn_getTpsMetrics	TPS metrics
qsn_getGasStats	Gas statistics
qsn_getAddressAnalytics	Address analytics

8.2 Market Data

Method	Description
qsn_getCoinTable	Token table (CoinMarketCap style)
qsn_getCoinPrice	Token price
qsn_getCoinPriceHistory	Price history
qsn_getTopMovers	Top gainers/losers
qsn_getTrendingTokens	Trending tokens

8.3 System Methods

Method	Description
qsn_getSyncStatus	Sync status
qsn_peerCount	Peer count
qsn_version	Node version
qsn_health	Health check
qsn_uptime	Node uptime

8.4 Bridge Methods (In Development)

Method	Description
bridge_status	Bridge status
bridge_getSupportedChains	Supported chains
bridge_initiateDeposit	Initiate deposit
bridge_initiateWithdrawal	Initiate withdrawal
bridge_estimateFee	Estimate fee

9. ERROR CODES

Code	Name	Description
-32700	Parse error	Invalid JSON
-32600	Invalid Request	Invalid JSON-RPC
-32601	Method not found	Unknown method
-32602	Invalid params	Invalid parameters
-32603	Internal error	Server error
-32000	Insufficient funds	Not enough balance
-32001	Invalid signature	Signature verification failed
-32002	Invalid nonce	Wrong nonce
-32003	PQ required	Post-quantum signature required
-32004	Invalid PQ sig	Invalid quantum signature

Error Example:

```
{"jsonrpc":"2.0","id":1,"error":{"code":-32003, "message":"Post-quantum signature required for transfers >= 1000 tQSN"}}
```

10. RATE LIMITS

Level	Requests/min	Burst
Default	100	20
Authenticated	1000	100
WebSocket	500	50

- Max request size: 1 MB
- Max batch size: 100 requests
- Connection timeout: 30 seconds
- Faucet: 1 request per 24h per address

RESOURCES

- Website: https://qsnchain.com
- Explorer: https://qsnscan.io

Document created: 2025-12-05