Developer Overview

Everything you need to know before you write your first smart contract on Neo Smart Chain.

VERSION	LAST UPDATED	STATUS
0.1	November 17, 2026	Draft

1. Welcome

Neo Smart Chain is an EVM-compatible blockchain. If you have written contracts for Ethereum or for a layer 2 such as Base or Robinhood Chain, you already know how to build here. You write Solidity, compile it with the same compilers, test it with the same frameworks and talk to the network through the same JSON-RPC interface.

These guides cover the whole path: connecting to the network, writing and testing a contract, deploying and verifying it, securing it, and getting a token listed in the Neo Smart Chain token registry.

2. Network at a glance

Use these values to add the network to a wallet or to configure your tools. Values in square brackets are confirmed and published before the public testnet opens.

Setting	Value
Network name	Neo Smart Chain
Chain ID	[chain ID]
JSON-RPC endpoint	[RPC URL]
WebSocket endpoint	[WebSocket URL]
Native currency	[native currency name and symbol]
Block explorer	[explorer URL]

Faucet (testnet)	[testnet faucet URL]
Supported EVM version	[EVM version]
Average block time	[block time]

> Always read the chain ID and EVM version from this page or from the network itself. Do not copy values from another chain, because a wrong chain ID or EVM version causes signing and deployment errors that are hard to diagnose.

3. How it relates to other EVM chains

Contracts follow the same ERC standards as the rest of the EVM ecosystem. A token that implements ERC-20, or an NFT that implements ERC-721, behaves the same way here as it does on Ethereum, Base or Robinhood Chain, and works with the same wallets, indexers and libraries.

- Same language: Solidity, plus Vyper for teams that prefer it.
- Same standards: ERC-20, ERC-721, ERC-1155, ERC-4626 and the other application-level ERCs described in the standards reference.
- Same tools: Foundry, Hardhat, OpenZeppelin, ethers, viem, web3.js and wagmi.
- Same wallets: any wallet that supports custom EVM networks can connect.

The network itself is separate. Addresses, balances and contracts on Neo Smart Chain are independent from those on other chains, and a few chain-level details can differ, such as gas pricing, block time, precompiles and which shared contracts are pre-deployed. The security guide and the troubleshooting guide cover these differences.

Underlying stack and settlement design: [underlying stack and settlement layer].

4. Supported tools

Category	Tools
Contract languages	Solidity (recommended), Vyper
Frameworks	Foundry, Hardhat
Contract libraries	OpenZeppelin Contracts, Solady, Solmate
Client libraries	viem, ethers, web3.js

Front-end	wagmi, RainbowKit, Web3Modal
Wallets	MetaMask, Coinbase Wallet, Rabby and any wallet with custom network support
Analysis	Slither, Echidna, Foundry fuzz and invariant tests

5. Documentation map

1. Quickstart: write, test, deploy and verify your first contract.
2. ERC standards reference: which standards to use and how.
3. Deploying and verifying contracts: Foundry, Hardhat, proxies and a launch checklist.
4. Security best practices: patterns, testing and chain-specific cautions.
5. JSON-RPC and client libraries: endpoints, supported methods and code samples.
6. Listing a token: how a token joins the Neo Smart Chain registry with its documentation.
7. Multi-chain deployments: running the same token on other blockchains.
8. Troubleshooting: common errors and how to fix them.

6. Getting help

Developer support: [developer support contact]. Community channel: [community channel URL]. To report a security issue in the network or in a registered token, use the contact in that token's security documents.

This document is for information only and is not financial, legal or tax advice. Highlighted items in square brackets are still to be confirmed by the issuer. The latest version is always available at neosmartchain.com/developers/overview