Multi-chain Deployments

Run the same token on other blockchains and keep its documentation accurate on each one.

VERSION	LAST UPDATED	STATUS
0.1	November 17, 2026	Draft

1. Why deploy on more than one chain

Holders and partners live on many networks. A token launched on Neo Smart Chain can also be deployed on other chains, so it can be used where its users already are. Each deployment is a separate contract with its own address, balances and risks, and each should be documented as such.

2. EVM chains

Because Neo Smart Chain is EVM-compatible, the same Solidity code can be deployed on other EVM chains, including Ethereum and layer 2 networks such as Base and Robinhood Chain, with little or no change.

- Recompile with the EVM version the target chain supports.
- Check that any shared contract you depend on exists on the target chain.
- Use a separate EIP-712 domain per chain, which the standard libraries do automatically through the chain ID.
- Re-run your full test suite and the testnet checks on every chain.
- Verify the source code on each chain's explorer.

To give the token the same address on every EVM chain, deploy it with CREATE2 through a deterministic deployer. The widely used deployer is at 0x4e59b44847b379578588920cA78FbF26c0B4956C, but you must confirm it exists on each chain first.

```
cast code 0x4e59b44847b379578588920cA78FbF26c0B4956C --rpc-url $NSC_RPC_URL
```

```
bytes32 constant SALT = keccak256("neo-token-v1");

function run() external returns (NeoToken token) {
```

```
vm.startBroadcast();
token = new NeoToken{salt: SALT}(owner, supply);
vm.stopBroadcast();
}
```

The address depends on the bytecode and constructor arguments, so identical compiler settings and arguments are needed on every chain.

3. Non-EVM chains

Chains with a different virtual machine, such as Solana, do not run Solidity contracts. A token on those networks is a separate implementation, for example an SPL token on Solana, with its own mint address, authorities and metadata. Treat it as its own deployment with its own documents and audits.

4. Moving tokens between chains

There are two common designs. Decide early, because the choice affects supply accounting, audits and the legal description of the token.

Design	How it works	Trade-off
Lock and mint	Tokens are locked on the home chain and a representation is minted on the other chain	Simple to explain, but the locked pool is a target and wrapped versions can fragment
Burn and mint	Tokens are burned on one chain and minted on another, so total supply stays constant	One canonical supply, but the token contract needs mint and burn rights for the bridge

- Name the bridge or messaging protocol, its operator and its trust assumptions in your documentation.
- Limit the bridge minting right to the bridge contract and put the configuration behind a multisig.
- Publish per-chain supply and reconcile it against total supply regularly.
- Audit the bridge and the token together.
- Bridging on Neo Smart Chain: [supported bridges].

5. Documenting each deployment

On the token page, every deployment has its own row with the following details. Keep them current whenever a contract changes.

Field	What to record
Chain	Network name and chain ID
Status	Native for the home chain, live when deployed, or planned
Contract address	The verified contract on that chain
Explorer link	A direct link to the contract
Chain documents	Bridge guide, chain-specific risks and audit for that deployment

Always tell holders which address is official. Publish the full list in one place, repeat it in your metadata and warn about look-alike tokens.

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/multichain