Verifying a Smart Contract

Verifying your deployed contract using flattened source code, JSON input, Sourcify and more.

Once verified, a smart contract or token contract's source code becomes publicly available and verifiable, creating transparency and trust.

Verification is available for both Solidity and Vyper contracts.Currently, there are 7 methods for verification using the Blockscout UI.

To learn more about the smart contract verification Rust microservice and verification algorithms [see this page for developers](#) .
♂ If preferred you can verify directly from your Hardhat dev environment.

- Hardhat Verification Plugin
- Sourcify Plugin for Hardhat *

Smart Contract Verification with Blockscout

1) Go to the Verify contract page(Other -> Verify contract)

?

2) Enter in the contract address you received during deployment. The dropdown will show you several available verification options. Select the one you would like to use and continue.

?

- Solidity (Flattened source code)
- Solidity (Standard JSON input
-)
- Solidity (Sourcify
-)
- Solidity (Multi-part files)
- Vyper (Contract
-)
- Vyper (Multi-part files)
- Vyper (Standard JSON input)
-

Solidity (Flattened source code)

?

1. Contract Address:
2. The0x
3. address supplied on contract creation (added above)
4. Is Yul contract:
5. Select if the contract is coded in Yul for efficiency.
6. Include Nightly Builds
7. : Select if you want to show nightly builds.
8. Compiler:
9. derived from the first line in the contractpragma solidity X.X.X
10. . Use the corresponding compiler version rather than the nightly build.
11. EVM Version:
12. Select the correctEVM version
13. if known, otherwise use default.
14. Optimization Enabled:
15. If you enabled optimization during compilation, select and enter the run value. 200 is the Solidity Compiler default value. Only change if you changed this value while compiling.
16. Enter the Solidity Contract Code:
17. You may need to flatten your solidity code if it utilizes a library or inherits dependencies from another contract. We recommendhardhat
18. or thePOA solidity flattener
19. .
20. Add Contract Libraries:
21. Enter the name and 0x address for any required libraries called in the .sol file. You can add multiple contracts with the "+" button.
22. Click theVerify and Publish
23. button.

24. If all goes well, you will see a checkmark✅
25. next to Code in the code tab, and an additional tab calledRead Contract
26. . The contract name will now appear in BlockScout with any transactions related to your contract.
27.

Solidity (Standard JSON input)

More information on JSON input is available here. 1. Include nightly builds 2. . You can chooseYes 3. orNo 4. depending on your compiler. 5. Compiler. 6. Choose the compiler version used to compile your smart contract. If you selected yes for nightly builds, use the compiler version rather than the build. 7. Standard Input JSON. 8. Upload your Standard Input JSON file. File should follows solidityformat 9. and all the sources must be in Literal Content format, not a URL. 10.

Click theVerify & publish button and wait for the response.

Via Sourcify: Sources and metadata JSON file

See theContract Verification via Sourcify page for details.

Vyper Contract

1. Contract Name:
2. Name assigned to the contract.
3. Compiler
4. : Select the compiler version used in the source code.
5. EVM Version:
6. Select the correctEVM version
7. if known, otherwise use default.
8. Contract Code:
9. Copy and paste the contract code
10. Click theVerify and Publish
11. button.
12.

If all goes well, you will see a checkmark✅ next to Code in the code tab, and an additional tab calledRead Contract . The contract name will now appear in BlockScout with any transactions related to your contract.

Troubleshooting

If you receive the dreadedThere was an error compiling your contract message this means the bytecode doesn't match the supplied sourcecode. Unfortunately, there are many reasons this may be the case. Here are a few things to try:

1) Double check the compiler version is correct.

Check all version digits - for example 0.5.1 is different from 0.5.10 2) Check that an extra space has not been added to the end of the contract. When pasting in, an extra space may be added. Delete this and attempt to recompile.

3) Copy, paste, and verify your source code in Remix. You may find some exceptions here.

Verification in a dev environment

TheHardhat verification plugin supports BlockScout. You can also choose to use theSourcify plugin to verify with Sourcify from your hardhat environment.

Last updated1 month ago