How to verify contracts for Stylus contracts

caution This page will walk you through how to verify your Stylus contracts locally. Stylus contract verification is also available onArbiscan . Please note, however, that Stylus contract verification on Arbiscan is only supported for Stylus contracts deployed usingcargo-stylus 0.5.0 or higher.

Background

Stylus contracts written in Rust and deployed onchain can be verified against a local codebase by using thecargo stylus tool.

Goals

- To ensure Stylus contract deployments are reproducible by anyone who is running the same architecture as the deployed item
- To sandbox the reproducible environment and standardize it as much as possible to prevent foot guns
- To guarantee that programs reproducibly deployed with a cargo stylus version >= 0.4.2 are verifiable

Opting out

By default,cargo stylus deploy is reproducible as it runs in a Docker container. Users can opt-out by specifying--no-verify as a flag.

Reproducible deployments

Required knowledge and setup:

- System architecture of your host computer (x86 / ARM)
- The git commit of your project used for deployment
- ARust
- stylus project, such asOffchainLabs/stylus-hello-world
- which contains arust-toolchain.toml
- file
- Your cargo stylus version (runcargo stylus --version
- to obtain this value)
- Docker
- installed and running on your machine

Your project'stoolchain file must contain the Rust version you wish to use for your deployment, such asmajor.minor.patch

[toolchain] channel = "1.79.0" Itcannot bestable ,nightly , orbeta by itself, as a specific version must be added. For instance, you can specifynightly-YYYY-MM-DD ormajor.minor.patch for your channel. This is so that deployments have a very specific version to prevent potential mismatches from being more generic.

Replace {PRIV_KEY} with your actual private key or set it as a local variable

cargo stylus deploy --private-key={PRIV_KEY} --verbose Upon completion, you will obtain the deployment transaction hash:

deployment tx hash: 0x1d8ae97e245e1db21dd188e5b64ad9025c1fb4e5f82a8d38bc8ae2b7a387600b Save this tx hash, as verifiers will need it.

Reproducible verification

To verify a program, the verifier will need Docker installed and also know:

- System architecture the deployer used (x86 / ARM). Note: ARM devices that can emulate x86, such as Mac M series via rosetta, can verify x86 Stylus deployments
- The git commit of the project the deployer used
- Your cargo stylus version the deployer used
- The deployment transaction hash

Navigate to the project's directory and check out the git commit that was used at deployment. Ensure yourcargo stylus --version matches what the deployer used.

Replace {DEPLOYMENT_TX_HASH} with the actual DEPLOYMENT_TX_HASH or set it as a local variable

cargo stylus verify --deployment-tx={DEPLOYMENT_TX_HASH} This command will run the verification pipeline through a Docker environment, recreate the project metadata hash, and verify that the deployed program matches what the command reconstructed locally.

How it works

On deployment, akeccak256 hash is created from the contents of all Rust source files in the project, sorted by file path, along with a rust-toolchain.toml, Cargo.toml and Cargo.lock files by default. This hash is injected in as a custom section of the user wasm's code. This means all data in the source files will be used for reproducible verification of a Stylus contract, including code comments.

This means thecodehash onchain of the program will change due to this deployment metadata hash.

The verification routine fetches the deployment transaction by hash via RPC, then attempts to build the local project to reconstruct the deployment init code and WASM using cargo build. It then checks that the deployment tx data matches the created init code.

Important details

Docker image The Docker container used for reproducibility standardizes all builds to x86, and it looks like this:

FROM --platform = linux/amd64 rust:1.79 as builder RUN rustup toolchain install

VERSION -x86_64-unknown-linux-gnu RUN rustup default VERSION -x86_64-unknown-linux-gnu RUN rustup target add wasm32-unknown-unknown RUN rustup target add wasm32-wasi RUN rustup target add x86_64-unknown-linux-gnu RUN cargo

install cargo-stylus The docker container uses therust:1.79 version as a base for all projects. This will install cargo tooling and rust targets, but the toolchain actually used for compilation will be specified by the project being deployed in itsrust-toolchain.toml file.

For instance,a future toolchain can be used despite the base image being 1.79, as when cargo stylus is installed, it will use that particular toolchain. Future cargo stylus updates could update this base image but may not impact the compiled WASM as the image will be using the specified toolchain. However, this is why knowing the specific cargo stylus version used for the reproducible verification from the deployer is important.

The build toolchain

All verifiable Stylus contracts in Rustmust have a standardrust-toolchain.toml file which specifies the channel for their deployment. Itcannot bestable ,nightly , orbeta by itself, as a specific version must be added. For instance, you can specifynightly-YYYY-MM-DD ormajor.minor.patch for your channel. This is so that deployments have a very specific version to prevent potential mismatches from being more generic. Edit this page Last updatedonJan 27, 2025 Previous Debug transactions Next Cache contracts