Troubleshooting Stylus

{"@context":"https://schema.org","@type":"FAQPage","mainEntity":[{"@type":"Question","name":"How does Stylus manage security issues in smart contracts when interacting with so many different languages?","acceptedAnswer":{"@type":"Answer","text":"

All languages are compiled to WASM for them to be able to work with Stylus. So it just needs to verify that the produced WASM programs behave as they should inside the new virtual machine.

\n\n

\n\n"}},{"@type":"Question","name":"Is there any analogue of the fallback function from Solidity in the Rust Stylus SDK?","acceptedAnswer":{"@type":"Answer","text":"

Currently there isn't any analogue. However, you can use a minimal entrypoint and perform raw delegate calls, forwarding your calldata. You can find more information in Bytes-in, bytes-out programming and call, static_call and delegate_call.

\n\n

\n\n"}},{"@type":"Question","name":"Why are constructors not yet supported for Stylus contracts?","acceptedAnswer":{"@type":"Answer","text":"

Constructors use EVM bytecode to initialize state. While one could add EVM bytecode manually to their Stylus deployment, we don't allow WASM execution in the constructor so there's no way to express this in the SDK.

\n\n

We're working on models that will make init easier, so there might be better solutions available in the future. For now, we suggest calling an init method after deploying.

\n\n

\n\n"}},{"@type":"Question","name":"Is it possible to verify Stylus contracts on the block explorer?","acceptedAnswer":{"@type":"Answer","text":"

Currently it is not possible to verify contracts compiled to WASM on the block explorer, but we are actively working with providers to have the verification process ready for when Stylus reaches mainnet-ready status.

\n\n

\n\n"}},{"@type":"Question","name":"Do Stylus contracts compile down to EVM bytecode like prior other attempts?","acceptedAnswer":{"@type":"Answer","text":"

No. Stylus contracts are compiled down to WASM. The user writes a program in Rust / C / C++ which is then compiled down to WebAssembly.

\n\n

\n\n"}},{"@type":"Question","name":"How is a Stylus contract deployed?","acceptedAnswer":{"@type":"Answer","text":"

Stylus contracts are deployed on chain as a blob of bytes, just like EVM ones. The only difference is that when the contract is executed, instead of invoking the EVM, we invoke a separate WASM runtime. Note that a special EOF-inspired prefix distinguishes Stylus contracts from traditional EVM contracts: when a contract's bytecode starts with the magic 0xEF000000 prefix, it's a Stylus WASM contract.

\n\n

\n\n"}},{"@type":"Question","name":"Is there a new transaction type to deploy Stylus contracts?","acceptedAnswer":{"@type":"Answer","text":"

You deploy a Stylus contract the same way that Solidity contracts are deployed. There are no special transaction types. As a UX note: a WASM will revert until a special instrumentation operation is performed by a call to the new ArbWasm precompile, which readies the program for calls on-chain.

\n\n

You can find instructions for deploying a Stylus contract in ourQuickstart.

\n\n"}},{"@type":"Question","name":"Do Stylus contracts use a different type of ABI?","acceptedAnswer":{"@type":"Answer","text":"

Stylus contracts use solidity ABIs. Methods, signatures, logs, calls, etc. work exactly as in the EVM. From a user's / explorer's perspective, it all just looks and behaves like solidity.

\n\n

\n\n"}},{"@type":"Question","name":"Does the Stylus SDK for Rust support custom data structures?","acceptedAnswer":{"@type":"Answer","text":"

For in-memory usage, you should be able to use any implementation of custom data structures without problems.

\n\n

For storage usage, it might be a bit more complicated. Stylus uses the EVM storage system, so you'll need to define the data structure on top of it. However, in the SDK there's a storage trait that custom types can implement to back their collections with the EVM state trie. The SDK macros are compatible with them too, although fundamentally it's still a global key-value system.

\n\n

You can read more about it in theStylus Rust SDK page.

\n\n

As an alternative solution, you can use entrypoint-style contracts for your custom data structures.

\n\n

\n\n

\n\n"}}]}

How does Stylus manage security issues in smart contracts when interacting with so many different languages?

All languages are compiled to WASM for them to be able to work with Stylus. So it just needs to verify that the produced WASM programs behave as they should inside the new virtual machine.

Is there any analogue of the fallback function from Solidity in the Rust Stylus SDK?

Currently there isn't any analogue. However, you can use a minimal entrypoint and perform raw delegate calls, forwarding your calldata. You can find more information inBytes-in, bytes-out programming andcall, static_call and delegate_call .

Why are constructors not yet supported for Stylus contracts?

Constructors use EVM bytecode to initialize state. While one could add EVM bytecode manually to their Stylus deployment, we don't allow WASM execution in the constructor so there's no way to express this in the SDK.

We're working on models that will make init easier, so there might be better solutions available in the future. For now, we suggest calling an init method after deploying.

Is it possible to verify Stylus contracts on the block explorer?

Currently it is not possible to verify contracts compiled to WASM on the block explorer, but we are actively working with providers to have the verification process ready for when Stylus reaches mainnet-ready status.

Do Stylus contracts compile down to EVM bytecode like prior other attempts?

No. Stylus contracts are compiled down to WASM. The user writes a program in Rust / C / C++ which is then compiled down to WebAssembly.

How is a Stylus contract deployed?

Stylus contracts are deployed on chain as a blob of bytes, just like EVM ones. The only difference is that when the contract is executed, instead of invoking the EVM, we invoke a separate WASM runtime. Note that a special EOF-inspired prefix distinguishes Stylus contracts from traditional EVM contracts: when a contract's bytecode starts with the magic0xEF000000 prefix, it's a Stylus WASM contract.

Is there a new transaction type to deploy Stylus contracts?

You deploy a Stylus contract the same way that Solidity contracts are deployed. There are no special transaction types. As a

UX note: a WASM will revert until a special instrumentation operation is performed by a call to the newArbWasm precompile, which readies the program for calls on-chain.

You can find instructions for deploying a Stylus contract in ourQuickstart .

Do Stylus contracts use a different type of ABI?

Stylus contracts use solidity ABIs. Methods, signatures, logs, calls, etc. work exactly as in the EVM. From a user's / explorer's perspective, it all just looks and behaves like solidity.

Does the Stylus SDK for Rust support custom data structures?

For in-memory usage, you should be able to use any implementation of custom data structures without problems.

For storage usage, it might be a bit more complicated. Stylus uses the EVM storage system, so you'll need to define the data structure on top of it. However, in the SDK there's a storage trait that custom types can implement to back their collections with the EVM state trie. The SDK macros are compatible with them too, although fundamentally it's still a global key-value system.

You can read more about it in theStylus Rust SDK page .

As an alternative solution, you can useentrypoint-style contracts for your custom data structures. Last updatedonMar 19, 2024 Previous Stylus SDK repositories Next How to add a new programming language to Stylus