VM affordances

The Stylus Rust SDK contains several modules for interacting with the Virtual Machine (VM), which can be imported fromstylus_sdk .

Let's see an example:

note This code has yet to be audited. Please use at your own risk. use

stylus_sdk :: { msg } ;

let callvalue =

msg :: value ( ) ; This page lists the modules that are available, as well as the methods within those modules.

block

Allows you to inspect the current block:

- basefee
- : gets the basefee of the current block
- chainid
- : gets the unique chain identifier of the Arbitrum chain
- coinbase
- : gets the coinbase of the current block, which on Arbitrum chains is the L1 batch poster's address
- gas_limit
- : gets the gas limit of the current block
- number
- : gets a bounded estimate of the L1 block number at which the sequencer sequenced the transaction. SeeBlock gas limit, numbers and time
- for more information on how this value is determined
- timestamp
- : gets a bounded estimate of the Unix timestamp at which the sequencer sequenced the transaction. SeeBlock gas limit, numbers and time
- for more information on how this value is determined

use

stylus_sdk :: { block } ;

let basefee =

block :: basefee ( ) ; let chainid =

block :: chainid ( ) ; let coinbase =

block :: coinbase ( ) ; let gas_limit =

block :: gas_limit ( ) ; let number =

block :: number ( ) ; let timestamp =

block :: timestamp ( ) ;

contract

Allows you to inspect the contract itself:

- address
- : gets the address of the current program
- args
- : reads the invocation's calldata. The entrypoint macro uses this under the hood
- balance
- : gets the balance of the current program
- output
- : writes the contract's return data. The entrypoint macro uses this under the hood
- read_return_data
- : copies the bytes of the last EVM call or deployment return result. Note: this function does not revert if out of bounds, but rather will copy the overlapping portion

- return_data_len
- : returns the length of the last EVM call or deployment return result, or 0 if neither have happened during the program's execution

use

stylus_sdk :: { contract } ;

let address =

contract :: address ( ) ; contract :: args ( ) ; let balance =

contract :: balance ( ) ; contract :: output ( ) ; contract :: read_return_data ( ) ; contract :: return_data_len ( ) ;

crypto

Allows you to access VM-accelerated cryptographic functions:

- keccak
- : efficiently computes the keccak256
- hash of the given preimage

use

stylus_sdk :: { crypto } ; use

stylus_sdk :: alloy_primitives :: address ;

let preimage =

address! ("361594F5429D23ECE0A88E4fBE529E1c49D524d8") ; let hash =

crypto :: keccak (& preimage) ;

evm

Allows you to access affordances for the Ethereum Virtual Machine:

- gas_left
- : gets the amount of gas remaining. See Ink and Gas
- for more information on Stylus's compute pricing
- ink_left
- : gets the amount of ink remaining. See Ink and Gas
- for more information on Stylus's compute pricing
- log
- : emits a typed alloy log
- pay_for_memory_grow
- : this function exists to force the compiler to import this symbol. Calling it will unproductively consume gas
- raw_log
- : emits an EVM log from its raw topics and data. Most users should prefer the alloy-typed raw_log

use

stylus_sdk :: { evm } ;

let gas_left =

evm :: gas_left ( ) ; let ink_left =

evm :: ink_left ( ) ; evm :: log (...) ; evm :: pay_for_memory_grow ( ) ; evm :: raw_log (...) ; Here's an example of how to emit a Transfer log:

sol!

{ event Transfer (address indexed from , address indexed to , uint256 value) ; event Approval (address indexed owner , address indexed spender , uint256 value) ; }

fn

foo ( )

{ ... evm :: log (Transfer

{ from :

Address :: ZERO , to : address , value , }) ; }

msg

Allows you to inspect the current call

- reentrant
- : whether the current call is reentrant
- sender
- : gets the address of the account that called the program. For normal L2-to-L2 transactions the semantics are equivalent to that of the EVM'sCALLER
- opcode, including in cases arising fromDELEGATE_CALL
- value
- : gets the ETH value in wei sent to the program

use

stylus_sdk :: { msg } ;

let reentrant =

msg :: reentrant ( ) ; let sender =

msg :: sender ( ) ; let value =

msg :: value ( ) ;

tx

Allows you to inspect the current transaction

- gas_price
- : gets the gas price in wei per gas, which on Arbitrum chains equals the basefee
- gas_to_ink
- : converts evm gas to ink. SeeInk and Gas
- for more information on Stylus's compute-pricing model
- ink_price
- : gets the price of ink in evm gas basis points. SeeInk and Gas
- for more information on Stylus's compute-pricing model
- ink_to_gas
- : converts ink to evm gas. SeeInk and Gas
- for more information on Stylus's compute-pricing model
- origin
- : gets the top-level sender of the transaction. The semantics are equivalent to that of the EVM'sORIGIN
- opcode

use

stylus_sdk :: { tx } ;

let gas_price =

tx :: gas_price ( ) ; let gas_to_ink =

tx :: gas_to_ink ( ) ; let ink_price =

tx :: ink_price ( ) ; let ink_to_gas =

tx :: ink_to_gas ( ) ; let origin =

tx :: origin ( ) ;

Learn More

- Arbitrum documentation
- Stylus SDK modules Edit this page Previous Inheritance Next Sending Ether