Events

Events allow for data to be logged publicly to the blockchain. Log entries provide the contract's address, a series of up to four topics, and some arbitrary length binary data. The Stylus Rust SDK provides a few ways to publish event logs described below.

Learn More

- Solidity docs: Events
- stylus_sdk::evm::log
- alloy_sol_types::SolEvent

Log

Using theevm::log function in the Stylus SDK is the preferred way to log events. It ensures that an event will be logged in a Solidity ABI-compatible format. Thelog function takes any type that implements AlloySolEvent trait. It's not recommended to attempt to implement this trait on your own. Instead, make use of the providedsol! macro to declare your Events and their schema using Solidity-style syntax to declare the parameter types. Alloy will create ABI-compatible Rust types which you can instantiate and pass to theevm::log function.

Log Usage

note This code has yet to be audited. Please use at your own risk. // sol! macro event declaration // Up to 3 parameters can be indexed. // Indexed parameters helps you filter the logs efficiently sol!

{ event Log (address indexed sender , string message) ; event AnotherLog ( ) ; }

[storage]

[entrypoint]

pub

struct

Events

{ }

[public]

impl

Events

{ fn

user_main (_input :

Vec < u8

)

->

ArbResult

{ // emits a 'Log' event, defined above in the sol! macro evm :: log (Log

{ sender :

Address :: from ([0x11 ;

20]) , message :

"Hello world!" . to_string ( ) , }) ;

// no data, but 'AnotherLog' event will still emit to the chain evm :: log (AnotherLog

{ }) ;

Ok (vec! [ ]) } }

Raw Log

Theevm::raw_log affordance offers the ability to send anonymous events that do not necessarily conform to the Solidity ABI. Instead, up to four raw 32-byte indexed topics are published along with any arbitrary bytes appended as data.

NOTE : It's still possible to achieve Solidity ABI compatibility using this construct. To do so you'll have to manually compute the ABI signature for the event following the equation set in the Solidity docs . The result of that should be assigned toTOPIC_0 , the first topic in the slice passed toraw_log .

Raw Log Usage

// set up local variables let user =

Address :: from ([0x22 ;

20]) ; let balance =

U256 :: from (10_000_000) ;

// declare up to 4 topics // topics must be of type FixedBytes<32> let topics =

& [user . into_word ( )] ;

// store non-indexed data in a byte Vec let

mut data :

Vec < u8

=

vec! [ ] ; // to_be_bytes means 'to big endian bytes' data . extend_from_slice (balance . to_be_bytes :: < 32

( ) . to_vec ( ) . as_slice ( )) ;

// unwrap() here 'consumes' the Result evm :: raw_log (topics . as_slice ( ) , data . as_ref ( )) . unwrap ( ) ;

Result

Combining the above examples into the boiler plate provided below this section, deploying to a Stylus chain and then invoking the deployed contract will result in the following three events logged to

the chain:

logs

[{ "address": "0x6cf4a18ac8efd6b0b99d3200c4fb9609dd60d4b3", "topics": ["0x0738f4da267a110d810e6e89fc59e46be6de0c37b1d5cd559b267dc3688e74e0", "0x0000000000000000000000011111111111111111111111111111111111111111"], "data": "0x0000000000000000000000000000000000000000000000000000000000000020000000000000000000000000000000000000000000000000000000000000000c48656c6c6f20776f726c6421000000000
"blockHash": "0xfef880025dc87b5ab4695a0e1a6955dd7603166ecba79ce0f503a568b2ec8940", "blockNumber": "0x94", "transactionHash": "0xc7318dae2164eb441fb80f5b869f844e3e97ae83c24a4639d46ec4d915a30818", "transactionIndex": "0x1", "logIndex": "0x0", "removed": false }, { "address": "0x6cf4a18ac8efd6b0b99d3200c4fb9609dd60d4b3", "topics": ["0xfe1a3ad11e425db4b8e6af35d11c50118826a496df73006fc724cb27f2b99946"], "data": "0x", "blockHash": "0xfef880025dc87b5ab4695a0e1a6955dd7603166ecba79ce0f503a568b2ec8940", "blockNumber": "0x94", "transactionHash": "0xc7318dae2164eb441fb80f5b869f844e3e97ae83c24a4639d46ec4d915a30818", "transactionIndex": "0x1", "logIndex": "0x1", "removed": false }, { "address": "0x6cf4a18ac8efd6b0b99d3200c4fb9609dd60d4b3", "topics": ["0x0000000000000000000000002222222222222222222222222222222222222222"], "data": "0x0000000000000000000000000000000000000000000000000000000000989680", "blockHash": "0xfef880025dc87b5ab4695a0e1a6955dd7603166ecba79ce0f503a568b2ec8940", "blockNumber": "0x94", "transactionHash": "0xc7318dae2164eb441fb80f5b869f844e3e97ae83c24a4639d46ec4d915a30818", "transactionIndex": "0x1", "logIndex": "0x2", "removed": false }]

Boilerplate

src/lib.rs

// Only run this as a WASM if the export-abi feature is not set.

![cfg_attr(not(any(feature =

"export-abi" , test)), no_main)] extern

crate

alloc ;

use

alloc :: vec :: Vec ; use

alloc :: { string :: ToString , vec } ;

use

stylus_sdk :: alloy_primitives :: U256 ; use

stylus_sdk :: { alloy_primitives :: Address ,

alloy_sol_types :: sol , evm ,

prelude :: * ,

ArbResult } ;

// sol! macro event declaration // Up to 3 parameters can be indexed. // Indexed parameters helps you filter the logs by the indexed parameter sol!

{ event Log (address indexed sender , string message) ; event AnotherLog ( ) ; }

[storage]

[entrypoint]

pub

struct

Events

{ }

[public]

impl

Events

{ fn

user_main (_input :

Vec < u8

)

->

ArbResult

{ // emits a 'Log' event, defined above in the sol! macro evm :: log (Log

{ sender :

Address :: from ([0x11 ;

20]) , message :

"Hello world!" . to_string ( ) , }) ;

// no data, but event will still log to the chain evm :: log (AnotherLog

{ }) ;

// set up local variables let user =

Address :: from ([0x22 ;

20]) ; let balance =

U256 :: from (10_000_000) ;

// declare up to 4 topics // topics must be of type FixedBytes<32> let topics =

& [user . into_word ( )] ;

// store non-indexed data in a byte Vec let

mut data :

Vec < u8

=

vec! [ ] ; // to_be_bytes means 'to big endian bytes' data . extend_from_slice (balance . to_be_bytes :: < 32

( ) . to_vec ( ) . as_slice ( )) ;

// unwrap() here 'consumes' the Result evm :: raw_log (topics . as_slice ( ) , data . as_ref ( )) . unwrap ( ) ;

Ok (Vec :: new ( )) } }

Cargo.toml

[package] name

=

"stylus_events_example" version

=

"0.1.7" edition

=

"2021" license

=

"MIT OR Apache-2.0" keywords

=

["arbitrum" ,

"ethereum" ,

"stylus" ,

"alloy"]

[dependencies] alloy-primitives

=

"=0.7.6" alloy-sol-types

=

"=0.7.6" mini-alloc

=

"0.4.2" stylus-sdk

=

"0.6.0" hex

=

"0.4.3"

[dev-dependencies] tokio

=

{

version

=

"1.12.0" ,

features

=

["full"]

} ethers

=

"2.0" eyre

=

"0.6.8"

[features] export-abi

=

["stylus-sdk/export-abi"]

[lib] crate-type

=

["lib" ,

"cdylib"]

[profile.release] codegen-units

=

1 strip

=

true lto

=

true panic

=

"abort" opt-level

=

"s" Edit this page Previous Errors Next Inheritance