Sending Ether

We have three main ways to send Ether in Rust Stylus: using thetransfer_eth method, using low levelcall method, and sending value while calling an external contract.

It's important to note that thetransfer_eth method in Rust Stylus invokes the recipient contract, which may subsequently call other contracts. All the gas is supplied to the recipient, which it may burn. Conversely, the transfer method in Solidity is capped at 2300 gas. In Rust Stylus, you can cap the gas by using the low-level call method with a specified gas. An example of this is provided in the code on bottom of the page.

These two methods are exactly equivalent under the hood:

note This code has yet to be audited. Please use at your own risk. transfer_eth (recipient , value) ? ;

call (Call :: new_in (self) . value (value) , recipient ,

& [ ]) ? ;

Where to Send Ether

1. Externally Owned Account (EOA) Addresses
2. : Directly send Ether to an EOA address.
3. Solidity Smart Contracts with Receive Function (No Calldata)
4. : Send Ether to a Solidity smart contract that has areceive
5. function without providing any calldata.
6. Solidity Smart Contracts with Fallback Function (With Calldata)
7. : Send Ether to a Solidity smart contract that has afallback
8. function by providing the necessary calldata.
9. Smart Contracts with Payable Methods (both Solidity and Stylus)
10. : Send Ether to smart contracts that have defined payable methods. Payable methods are identified by thepayable
11. modifier in Solidity, and the#[payable]
12. macro in Rust.

Below you can find examples for each of these methods and how to define them in a Rust Stylus smart contract using the Stylus SDK:

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

alloy_primitives :: Address ; use

stylus_sdk :: { abi :: Bytes , call :: { call , transfer_eth ,

Call } , msg :: { self } , prelude :: * , } ;

sol_interface!

{ interface ITarget

{ function receiveEther ( ) external payable ; } }

[storage]

[entrypoint]

pub

struct

SendEther

{ }

[public]

impl

SendEther

{ // Transfer Ether using the transfer_eth method // This can be used to send Ether to an EOA or a Solidity smart contract that has a receive() function implemented

[payable]

pub

fn

send_via_transfer (to :

Address)

->

Result < ( ) ,

Vec < u8

{ transfer_eth (to ,

msg :: value ( )) ? ; Ok (( )) }

// Transfer Ether using a low-level call // This can be used to send Ether to an EOA or a Solidity smart contract that has a receive() function implemented

[payable]

pub

fn

send_via_call (& mut

self , to :

Address)

->

Result < ( ) ,

Vec < u8

{ call (Call :: new_in (self) . value (msg :: value ( )) , to ,

& [ ]) ? ; Ok (( )) }

// Transfer Ether using a low-level call with a specified gas limit // This can be used to send Ether to an EOA or a Solidity smart contract that has a receive() function implemented

[payable]

pub

fn

send_via_call_gas_limit (& mut

self , to :

Address , gas_amount :

u64)

->

Result < ( ) ,

Vec < u8

{ call (Call :: new_in (self) . value (msg :: value ( )) . gas (gas_amount) , to , & [ ] ,) ? ; Ok (( )) }

// Transfer Ether using a low-level call with calldata // This can be used to call a Solidity smart contract's fallback function and send Ether along with calldata

[payable]

pub

fn

send_via_call_with_call_data (& mut

self , to :

Address , data :

Bytes ,)

->

Result < ( ) ,

Vec < u8

{ call (Call :: new_in (self) . value (msg :: value ( )) , to , data . as_slice ( )) ? ; Ok (( )) }

// Transfer Ether to another smart contract via a payable method on the target contract // The target contract can be either a Solidity smart contract or a Stylus contract that has a receiveEther function, which is a payable function

[payable]

pub

fn

send_to_stylus_contract (& mut

self , to :

Address)

->

Result < ( ) ,

Vec < u8

{ let target =

ITarget :: new (to) ; let config =

Call :: new_in (self) . value (msg :: value ( )) ; target . receive_ether (config) ? ; Ok (( )) } }

Cargo.toml

[package] name

=

"stylus_sending_ether_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 Vm Affordances Next Function Selector