Inheritance

The Stylus Rust SDK replicates the composition pattern of Solidity. The#[public] macro provides theRouter trait, which can be used to connect types via inheritance, via the#[inherit] macro.

Please note: Stylus doesn't support contract multi-inheritance yet.

Let's see an example:

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

[public]

[inherit(Erc20)]

impl

Token

{ pub

fn

mint (& mut

self , amount :

U256)

->

Result < ( ) ,

Vec < u8

{ ... } }

[public]

impl

Erc20

{ pub

fn

balance_of ( )

->

Result < U256

{ ... } } In the above code, we can see howToken inherits fromErc20 , meaning that it will inherit the public methods available inErc20 . If someone called theToken contract on the functionbalanceOf , the functionErc20.balance_of() would be executed.

Additionally, the inheriting type must implement theBorrow trait for borrowing data from the inherited type. In the case above,Token should implementBorrow . For simplicity,#[storage] andsol_storage! provide a#[borrow] annotation that can be used instead of manually implementing the trait:

sol_storage!

{

[entrypoint]

pub

struct

Token

{

[borrow]

Erc20 erc20 ; ... }

pub

struct

Erc20

{ ... } }

Methods search order

A type can inherit multiple other types (as long as they use the#[public] macro). Since execution begins in the type that uses the#[entrypoint] macro, that type will be first checked when searching a specific method. If the method is not found in that type, the search will continue in the inherited types, in order of inheritance. If the method is not found in any of the inherited methods, the call will revert.

Let's see an example:

[public]

[inherit(B, C)]

impl

A

{ pub

fn

foo ( )

->

Result < ( ) ,

Vec < u8

{ ... } }

[public]

impl

B

{ pub

fn

bar ( )

->

Result < ( ) ,

Vec < u8

{ ... } }

[public]

impl

C

{ pub

fn

bar ( )

->

Result < ( ) ,

Vec < u8

{ ... }

pub

fn

baz ( )

->

Result < ( ) ,

Vec < u8

{ ... } } In the code above:

- callingfoo()
- will search the method inA
- , find it, and executeA.foo()
- callingbar()
- will search the method inA
- first, then inB
- , find it, and executeB.bar()
- callingbaz()
- will search the method inA
- ,B
- and finallyC
- , so it will executeC.baz()

Notice thatC.bar() won't ever be reached, since the inheritance goes throughB first, which has a method namedbar() too.

Finally, since the inherited types can also inherit other types themselves, keep in mind that method resolution finds the first matching method byDepth First Search .

Overriding methods

Because methods are checked in the inherited order, if two types implement the same method, the one in the higher level in the hierarchy will override the one in the lower levels, which won’t be callable. This allows for patterns where the developer imports a crate implementing a standard, like ERC-20, and then adds or overrides just the methods they want to without modifying the imported ERC-20 type.

Important warning : The Stylus Rust SDK does not currently contain explicitoverride orvirtual keywords for explicitly marking override functions. It is important, therefore, to carefully ensure that contracts are only overriding the functions.

Let's see an example:

[public]

[inherit(B, C)]

impl

A

{ pub

fn

foo ( )

->

Result < ( ) ,

Vec < u8

{ ... } }

[public]

impl

B

{ pub

fn

foo ( )

->

Result < ( ) ,

Vec < u8

{ ... }

pub

fn

bar ( )

->

Result < ( ) ,

Vec < u8

{ ... } } In the example above, even thoughB has an implementation forfoo() , callingfoo() will executeA.foo() since the method is searched first inA .

Learn more

- Arbitrum documentation
- inheritance, #[inherit] and #[borrow]
- Router trait
- Borrow trait
- BorrowMut trait Edit this page Previous Events Next Vm Affordances