Decentralized Exchanges (DEX)

A Decentralized Exchange (DEX) is an application that allows users to trade tokens (native & fungible tokens) through smart contracts.

In brief, DEXs work by havingpools of token pairs (e.g. NEAR-USDC) that users can deposit tokens into.

The ratio of the tokens in the pool determines the exchange rate for a swap. Indeed, swapping is adding tokens to one side of the pool while removing tokens from the other side of the pool.

info This docs refer toRef Finance , a community built DEX in NEAR.

Please check theirdocs for more information.

Query Token Exchange Rate

One can query the exchange rate of a token pair by calling theget-token-price method on the DEX contract.

- ⚛ Component
- WebApp

const tokenContract =

"token.v2.ref-finance.near" ; const tokenPriceResult =

fetch (https://indexer.ref.finance/get-token-price?token_id= { tokenContract }) . body ; const tokenPriceValue =

JSON . parse (tokenPriceResult) ; Example response { "token_contract_id": "token.v2.ref-finance.near", "price": "0.08153090" } tip Ref Finance has a method toget all token prices at once . const tokenContract =

"token.v2.ref-finance.near" ; const tokenPriceResult =

await

fetch (https://indexer.ref.finance/get-token-price?token_id= { tokenContract }) ; const tokenPriceValue =

await tokenPriceResult . json ( ) ; TheWallet object comes from ourquickstart template

Example response { "token_contract_id": "token.v2.ref-finance.near", "price": "0.08153090" } tip Ref Finance has a method toget all token prices at once .

Query Whitelisted Tokens

Anyone list tokens for sale in the DEX. This is why, in order to protect users, the DEX contract has a list of whitelisted tokens that can be traded.

- CLI

near view v2.ref-finance.near get_whitelisted_tokens Examples Response 'wrap.near', 'usdt.tether-token.near', 'berryclub.ek.near', 'farm.berryclub.ek.near', 'token.v2.ref-finance.near', 'token.paras.near', 'marmaj.tkn.near', 'meta-pool.near', ...

Register in the DEX

In order to use the contract, make sure to register your account in the DEX by paying for the storage you will use in order to keep track of your balances.

- CLI

near call v2.ref-finance.near storage_deposit '' --accountId--amount 0.1

Deposit funds

In order to swap tokens, one must first deposit tokens into the DEX. For this, you will need to transfer the FT you want to swap to the DEX contract.

- CLI

near call token.v2.ref-finance.near ft_transfer_call {"receiver_id": "v2.ref-finance.near", "amount": "1000000000000", "msg": ""} --gas 300000000000000 --depositYocto 1 --accountId danger DoNOT transferNEAR tokens to Ref Finance. Instead, callnear_deposit in the[wrap.near](wrap.near) contract, attaching the amount of NEAR you want to swap.

This will mintwrap.near for you, which you can then transfer to Ref Finance.

Get Deposit Balances

Query your deposit balances by calling theget_deposits method:

- ⚛ Component
- WebApp
- CLI
- Contract

const ammContract =

"v2.ref-finance.near" ; const depositBalances =

Near . view (ammContract , "get_deposits" , { account_id :

"bob.near" }) ; Example response { "token.v2.ref-finance.near": "0", "wrap.near": "0" } const

AMM_CONTRACT_ADDRESS

=

"v2.ref-finance.near" ; const wallet =

new

Wallet ({

createAccessKeyFor :

AMM_CONTRACT_ADDRESS

}) ;

await wallet . viewMethod ({ method :

'get_deposits' , args :

{ account_id :

"bob.near" } , contractId :

AMM_CONTRACT_ADDRESS , }) ; TheWallet object comes from our[quickstart template](quickstart template)

Example response { "token.v2.ref-finance.near": "0", "wrap.near": "0" } near view v2.ref-finance.near get_deposits '{"account_id": "bob.near"}' Example response { 'token.v2.ref-finance.near': '0', 'wrap.near': "0" } // Validator interface, for cross-contract calls

[ext_contract(ext_amm_contract)]

trait

ExternalAmmContract

{ fn

get_deposits (& self , account_id :

AccountId)

->

Promise ; }

// Implement the contract structure

[near_bindgen]

impl

Contract

{

[private]

// Public - but only callable by env::current_account_id() pub

fn

external_get_deposits_callback (& self ,

[callback_result]

call_result :

Result < HashMap < AccountId ,

U128

,

PromiseError

)

->

Option < HashMap < AccountId ,

U128

{ // Check if the promise succeeded if call_result . is_err ( )

{ log! ("There was an error contacting external contract") ; return

None ; }

// Return the pools data let deposits_data = call_result . unwrap ( ) ; return

Some (deposits_data) ; }

pub

fn

get_contract_deposits (& self)

->

Promise

{ let promise =

ext_amm_contract :: ext (self . amm_contract . clone ( )) . get_deposits (env :: current_account_id ( )) ;

return promise . then (

// Create a promise to callback query_greeting_callback Self :: ext (env :: current_account_id ( )) . external_get_deposits_callback ( )) } }

Query Pools

DEXs work by having multiple pools of token pairs (e.g. NEAR-USDC) that users can deposit tokens into.

- ⚛ Component
- WebApp
- CLI
- Contract

const ammContract =

"v2.ref-finance.near" ; const result =

Near . view (ammContract , "get_pools" , { from_index :

0 , limit :

1000 }) ; Example response [{ pool_kind :

'SIMPLE_POOL' , token_account_ids :

[

'token.skyward.near' ,

'wrap.near'

] , amounts :

[

'51865812079751349630100' ,

'6254162663147994789053210138'

] , total_fee :

30 , shares_total_supply :

'1305338644973934698612124055' , amp :

0 } , { pool_kind :

'SIMPLE_POOL' , token_account_ids :

['c02aaa39b223fe8d0a0e5c4f27ead9083c756cc2.factory.bridge.near' , 'wrap.near'] , amounts :

[

'783621938569399817' ,

'1100232280852443291118200599'

] , total_fee :

30 , shares_total_supply :

'33923015415693335344747628' , amp :

0 }] const

AMM_CONTRACT_ADDRESS

=

"v2.ref-finance.near" ; const wallet =

new

Wallet ({

createAccessKeyFor :

AMM_CONTRACT_ADDRESS

}) ;

await wallet . viewMethod ({ method :

'get_pools' , args :

{ from_index :

0 , limit :

1000 } , contractId :

AMM_CONTRACT_ADDRESS , }) ; TheWallet object comes from ouquickstart template

Example response [{ pool_kind :

'SIMPLE_POOL' , token_account_ids :

[

'token.skyward.near' ,

'wrap.near'

] , amounts :

[

'51865812079751349630100' ,

'6254162663147994789053210138'

] , total_fee :

30 , shares_total_supply :

'1305338644973934698612124055' , amp :

0 } , { pool_kind :

'SIMPLE_POOL' , token_account_ids :

['c02aaa39b223fe8d0a0e5c4f27ead9083c756cc2.factory.bridge.near' , 'wrap.near'] , amounts :

[

'783621938569399817' ,

'1100232280852443291118200599'

] , total_fee :

30 , shares_total_supply :

'33923015415693335344747628' , amp :

0 }] near view v2.ref-finance.near get_pools '{"from_index": 0, "limit": 1000}' Example response [{ pool_kind: 'SIMPLE_POOL', token_account_ids: ['token.skyward.near', 'wrap.near'], amounts: ['51865812079751349630100', '6254162663147994789053210138'], total_fee: 30, shares_total_supply: '1305338644973934698612124055', amp: 0 }, { pool_kind: 'SIMPLE_POOL', token_account_ids: ['c02aaa39b223fe8d0a0e5c4f27ead9083c756cc2.factory.bridge.near', 'wrap.near'], amounts: ['783621938569399817', '1100232280852443291118200599'], total_fee: 30, shares_total_supply: '33923015415693335344747628', amp: 0 }]

[derive(Serialize, Deserialize)]

[serde(crate =

"near_sdk::serde")] pub

struct

PoolInfo

{ /// Pool kind. pub pool_kind :

String , /// List of tokens in the pool. pub token_account_ids :

Vec < AccountId

, /// How much NEAR this contract has. pub amounts :

Vec < U128

, /// Fee charged for swap. pub total_fee :

u32 , /// Total number of shares. pub shares_total_supply :

U128 , pub amp :

u64 , }

// Validator interface, for cross-contract calls

[ext_contract(ext_amm_contract)]

trait

ExternalAmmContract

{ fn

get_pools (& self , from_index :

u64 , limit :

u64)

->

Promise ; }

// Implement the contract structure

[near_bindgen]

impl

Contract

{

[private]

// Public - but only callable by env::current_account_id() pub

fn

external_get_pools_callback (& self ,

[callback_result]

call_result :

Result < Vec < PoolInfo

,

PromiseError

)

->

```
Option < Vec < PoolInfo

{ // Check if the promise succeeded if call_result . is_err ( )

{ log! ( "There was an error contacting external contract" ) ; return

None ; }

// Return the pools data let pools_data = call_result . unwrap ( ) ; return

Some ( pools_data ) ; }

pub

fn

get_amm_pools ( & self , from_index :

u64 , limit :

u64 )

->

Promise

{ let promise =

ext_amm_contract :: ext ( self . amm_contract . clone ( ) ) . get_pools ( from_index , limit ) ;

return promise . then (

// Create a promise to callback query_greeting_callback Self :: ext ( env :: current_account_id ( ) ) .
external_get_pools_callback ( ) ) } }
```

Swap tokens

In order to swap a token for another, you need to have funds , and there needs to exist a pool that hasboth tokens on it.

- ⚛ Component
- WebApp
- CLI
- Contract

```
const ammContract =

"v2.ref-finance.near" ; const result =

Near . call ( ammContract , "swap" , { actions :

[ { pool_id :

79 , token_in :

"token.v2.ref-finance.near" , token_out :

"wrap.near" , amount_in :

"100000000000000000" , min_amount_out :

"1" , } , ] , } , 300000000000000 , 1 ) ; Example response "5019606679394603179450" import

{

Wallet

}

from

'./near-wallet' ;

const
```

AMM_CONTRACT_ADDRESS

=

"v2.ref-finance.near" ; const wallet =

new

Wallet ({

createAccessKeyFor :

AMM_CONTRACT_ADDRESS

}) ;

await wallet . callMethod ({ method :

'swap' , args :

{ actions :

[{ pool_id :

79 , token_in :

"token.v2.ref-finance.near" , token_out :

"wrap.near" , amount_in :

"100000000000000000" , min_amount_out :

"1" , } ,] , } , contractId :

AMM_CONTRACT_ADDRESS , gas :

300000000000000 , deposit :

1 }) ; TheWallet object comes from ourquickstart template

Example response "5019606679394603179450" near call v2.ref-finance.near swap "{\"actions\": [{\"pool_id\": 79, \"token_in\": \"token.v2.ref-finance.near\", \"amount_in\": \"100000000000000000\", \"token_out\": \"wrap.near\", \"min_amount_out\": \"1\"}]}" --gas 300000000000000 --depositYocto 1 --accountId bob.near Example response '5019606679394603179450'

[derive(Serialize, Deserialize)]

[serde(crate =

"near_sdk::serde")] pub

struct

SwapAction

{ /// Pool which should be used for swapping. pub pool_id :

u64 , /// Token to swap from. pub token_in :

AccountId , /// Amount to exchange. /// If amount_in is None, it will take amount_out from previous step. /// Will fail if amount_in is None on the first step. pub amount_in :

Option < U128

, /// Token to swap into. pub token_out :

AccountId , /// Required minimum amount of token_out. pub min_amount_out :

U128 , }

// Validator interface, for cross-contract calls

[ext_contract(ext_amm_contract)]

trait

ExternalAmmContract

{ fn

swap (& self , actions :

Vec < SwapAction

)

->

Promise ; }

// Implement the contract structure

[near_bindgen]

impl

Contract

{

[private]

// Public - but only callable by env::current_account_id() pub

fn

external_call_callback (& self ,

[callback_result]

call_result :

Result < String ,

PromiseError

)

{ // Check if the promise succeeded if call_result . is_err ( )

{ log! ("There was an error contacting external contract") ; } }

[payable]

pub

fn

swap_tokens (& mut

self , pool_id :

u64 , token_in :

AccountId , token_out :

AccountId , amount_in :

U128 , min_amount_out :

U128)

->

Promise

{ assert_eq! (env :: attached_deposit ( ) ,

1 ,

"Requires attached deposit of exactly 1 yoctoNEAR") ;

let swap_action =

SwapAction

{ pool_id , token_in , token_out , amount_in :

Some (amount_in) , min_amount_out } ;

let

mut actions =

Vec :: new ( ) ; actions . push (swap_action) ;

let promise =

ext_amm_contract :: ext (self . amm_contract . clone ( )) . with_static_gas (Gas (150 * TGAS)) . with_attached_deposit (YOCTO_NEAR) . swap (actions) ;

return promise . then (

// Create a promise to callback query_greeting_callback Self :: ext (env :: current_account_id ( )) . with_static_gas (Gas (100 * TGAS)) . external_call_callback ( )) } }

Additional Resources

1. Claim Fungible Tokens from Lockup
2.
    - the example how to claim locked tokens from thelockup.burrow.near
3. contract.
4. BSC Dex Collection
5.
    - the example of how to build simple swap page for a DEX.Edit this page Last updatedonFeb 28, 2024 byDamian Parrino Was this page helpful? Yes No