Sending NEAR

You might want to send tokens from a contract for many reasons.

- The contract uses something like theStorage Standard
- and needs to return deposits to users when they unregister.
- Users pay into the contract and the contract later pays these fees to the maintainers, redistributes them to users, or disburses them to some cause the users vote on.
- And more!

Blockchains give us programmable money, and the ability for a smart contract to send tokens lies at the heart of that ability.

NEAR makes this easy. Transferring NEAR tokens is the simplest transaction you can send from a smart contract. Here's all you need:

let amount =

BigInt (1_000_000_000_000_000_000_000_000) ;

// 1 NEAR as yoctoNEAR let to =

"alice.near" ;

NearPromise . new (to) . transfer (amount) ; In the context of a full contract and function call, this could look like:

import

{

NearPromise ,

NearBindgen

}

from

"near-sdk-js" ;

@ NearBindgen ({ }) export

class

Contract

{ pay ({ amount , to })

{ return

NearPromise . new (to) . transfer (amount) ; } } Most of this is boilerplate you're probably familiar with by now – imports, setting upNearBindgen , etc. Some interesting details related to the transfer itself:

- Thepay
- method defined here accepts JSON as input, and numbers in JScannot be larger than2^53-1
- , so for compatibility with deserializing JSON to JS, the integer is serialized as a string. Since thetransfer
- method takes a value inyocto
- NEAR, it's likely to need numbers much larger than2^53-1
- .
- Returning theNearPromise
- : This allows NEAR Explorer, near-cli, near-api-js, and other tooling to correctly determine if a whole chain of transactions is successful. If your function does not returnPromise
- , tools like near-cli will return immediately after your function call. And then even if thetransfer
- fails, your function call will be considered successful.

Using near-cli or near-cli-rs, someone could invoke this function with a call like:

- near-cli
- near-cli-rs

near call pay '{"amount": "10000000000000000000000000", "to": "example.near"}' --accountId benjiman.near near contract call-function as-transaction pay json-args '{"amount": "10000000000000000000000000", "to": "example.near"}' prepaid-gas

'30 TeraGas' attached-deposit '0 NEAR' sign-as benjiman.near network-config testnet sign-with-keychain send Edit this page Last updatedonSep 29, 2023 bygagdiez Was this page helpful? Yes No

Previous Promises: Introduction Next Creating Accounts