Constructing transfers

Now that we have the wallet and client set up, we can create the environment necessary for constructing transfers. This can be a bit tricky, but the below boilerplate code should do the trick: The following imports will be needed in order to generate transfers:

use namada_sdk :: args :: InputAmount ; After the user has generated an account and created the appropriate struct, it is not too difficult to construct and submit transfer transactions.

let

mut namada =

NamadaImpl :: new (& http_client, &mut wallet, &mut shielded_ctx, & NullIo) .await . expect ("unable to construct Namada object") . chain_id (ChainId :: from_str ("public-testnet-14.5d79b6958580") . unwrap ()); // Transfer the given amount of native tokens from the source account to the // destination account. async

fn

gen_transfer (namada :

&impl

Namada , source :

& Account , destination :

& Account , amount :

InputAmount ,) -> std :: result :: Result < ProcessTxResponse , namada_sdk :: error :: Error

{ let

mut transfer_tx_builder = namada . new_transfer (TransferSource :: Address (Address :: from (& source . public_key)), TransferTarget :: Address (Address :: from (& destination . public_key)), namada . native_token (), amount,) . signing_keys (vec! [source . public_key . clone ()]); let (mut transfer_tx, signing_data, _epoch) = transfer_tx_builder . build (namada) .await . expect ("unable to build transfer"); namada . sign (&mut transfer_tx, args :

& transfer_tx_builder . tx, signing_data, with : default_sign, user_data : ()) .await . expect ("unable to sign reveal pk tx"); namada . submit (transfer_tx, & transfer_tx_builder . tx) .await } Other transactions can be constructed in a similar fashion.

Shielded transfers

In order to make the transfer shielded, we need to the only difference is to use shielded addresses and keys instead of transparent ones.

It is important to use the shielded extendedSpendingKey as the source.

use namada :: types :: masp :: { ExtendedSpendingKey , PaymentAddress , TransferSource , TransferTarget };

// Make sure to replace 'secret-ex' with an actual Namada extended spending key let source =

ExtendedSpendingKey :: from_str ("secret-ex") . unwrap (); // Make sure to replace "payment-addr-ex" with an actual Namada payment address let destination =

PaymentAddress :: from_str ("payment-addr-ex") . unwrap (); let fee_payer = let

mut transfer_tx_builder = namada . new_transfer (TransferSource :: ExtendedSpendingKey (source), TransferTarget :: Address (Address :: from (& destination . public_key)), namada . native_token (), amount,) // Make sure to replace "transparent-address-ex" with an actual Namada transparent address . signing_keys (vec! [Address :: from_str ("transparent-address-ex") . ok ()]);

Generating accounts Proof of stake transactions