Example usage of the Light SDK

Setup

let CHAIN_ID =

"shielded-expedition.88f17d1d14" ; let tendermint_addr =

"http://localhost:26657" ;

Transactions

Transfer

let source_address = namada_light_sdk :: namada_sdk :: address :: Address :: from_str ("tnam1v4ehgw36xq6ngs3ng5crvdpngg6yvsecx4znjdfegyurgwzzx4pyywfexuuyys69gc6rzdfnryrntx") . unwrap (); // replace with a valid source address let target_address = namada_light_sdk :: namada_sdk :: address :: Address :: from_str ("tnam1v4ehgw36xq6ngs3ng5crvdpngg6yvsecx4znjdfegyurgwzzx4pyywfexuuyys69gc6rzdfnryrntx") . unwrap (); // replace with a valid target address let token_address = namada_light_sdk :: reading :: blocking :: query_native_token (tendermint_addr) . unwrap (); let amount = namada_light_sdk :: namada_sdk :: token :: DenominatedAmount :: from_str ("10000") . unwrap (); // Construct the raw transaction struct let transfer =

Transfer :: new (source_address, target_address, token_address, amount, None , None , global_args); // In order to broadcast the transaction, it must be signed. This is difficult at the moment and more docs will be provided soon. // For now, the below code can be replicated. let targets =

vec! [transfer . clone () . payload () . raw_header_hash ()]; let

mut secret_keys =

BTreeMap :: new (); secret_keys . insert (0 , secret_key . clone ());

let signer_pub_key = secret_key . clone () . to_public ();

let signature_tx = namada_light_sdk :: namada_sdk :: tx :: Signature :: new (targets, secret_keys . clone (), None); let signature = signature_tx . signatures . get (& 0 u8) . unwrap () . to_owned ();

transfer

transfer . attach_signatures (signer_pub_key . clone (), signature . clone ()); let fee = namada_light_sdk :: namada_sdk :: token :: DenominatedAmount :: from_str ("10") . unwrap (); let gas_limit = namada_light_sdk :: namada_sdk :: tx :: data :: GasLimit :: from_str ("20000") . unwrap ();

let epoch = namada_light_sdk :: namada_sdk :: proof_of_stake :: Epoch :: from_str ("0") . unwrap ();

transfer

transfer . attach_fee (fee, token_address . clone (), signer_pub_key . clone (), epoch, gas_limit); let fee_targets =

vec! [transfer . clone () . payload () . sechashes ()]; let fee_signature_tx = namada_light_sdk :: namada_sdk :: tx :: Signature :: new (targets, secret_keys . clone (), None); let fee_signature = fee_signature_tx . signatures . get (& 0 u8) . unwrap () . to_owned ();

transfer

transfer . attach_fee_signature (signer_pub_key . clone (), signature . clone ());

// Once signed and the fee_payer and signature is added, the payload can be retrieved and broadcasted let transfer_tx = transfer . payload () let response = namada_light_sdk :: writing :: blocking :: broadcast_tx (tendermint_addr, transfer_tx) . unwrap ();

Writing Using the Indexer