Initializing the Chain Context

To begin working with the SDK, the first step is to initialize a 'Chain Context'. This is an instance of the NamadaImpl struct (opens in a new tab) , which allows us to access wallet storage, shielded context storage, and the RPC endpoint of our full node.

The Chain Context object also provides methods for common operations like querying the chain, constructing, signing, and submitting transactions.

To create the Chain Context, we'll need the RPC url of a synced full node.

Code

First, import our dependencies:

use tokio; use std :: str :: FromStr ; use namada_sdk :: {args :: TxBuilder , io :: NullIo , masp :: fs :: FsShieldedUtils , wallet :: fs :: FsWalletUtils , NamadaImpl , chain :: ChainId }; use tendermint_rpc :: { HttpClient , Url }; Next, we'll create the required helper objects.

Instantiate an http client for interacting with the full node:

let url =

Url :: from_str ("http://127.0.0.1:26657") . expect ("Invalid RPC address"); let http_client =

HttpClient :: new (url) . unwrap (); Set up wallet storage for our addresses and keys:

let wallet =

FsWalletUtils :: new ("./sdk-wallet" . into ()); Set up shielded context storage for the current state of the shielded pool:

let shielded_ctx =

FsShieldedUtils :: new ("./masp" . into ()); Create an IO object to capture the input/output streams. Since we don't need to capture anything, we'll useNullIo

let null_io =

NullIo ; Now we're ready to instantiate our Chain Context:

let sdk =

NamadaImpl :: new (http_client, wallet, shielded_ctx, null_io) .await . expect ("unable to initialize Namada context") . chain_id (ChainId :: from_str ("local-net.b8f955720ab") . unwrap ());

Full Example:

use tokio; use std :: str :: FromStr ; use namada_sdk :: {args :: TxBuilder , io :: NullIo , masp :: fs :: FsShieldedUtils , rpc, wallet :: fs :: FsWalletUtils , Namada , NamadaImpl , chain :: ChainId }; use tendermint_rpc :: { HttpClient , Url };

[tokio

:: main] async

fn

main () {

let url =

Url :: from_str ("http://127.0.0.1:26657") . expect ("Invalid RPC address"); let http_client =

HttpClient :: new (url) . unwrap ();

let wallet =

FsWalletUtils :: new ("./sdk-wallet" . into ()); let shielded_ctx =

FsShieldedUtils :: new ("./masp" . into ()); let null_io =

NullIo ;

let sdk =

NamadaImpl :: new (http_client, wallet, shielded_ctx, null_io) .await . expect ("unable to initialize Namada context") . chain_id (ChainId :: from_str ("local-net.b8f955720ab") . unwrap ()); } The paths provided when creatingwallet andshielded_ctx are the directories where the shielded context data andwallet.toml will be saved to/read from.

All following pages in this section assume you have already created a Chain Context with the namesdk .

Project setup Querying