Transaction

The transaction module exposes various modules (objects) to construct transactions. Each object implements the::new() method to create a new instance of the object.

pub

mod account; // contains functions to create transactions for account module pub

mod bridge; // contains functions to create transactions for bridge module pub

mod governance; // contains functions to create transactions for governance module pub

mod ibc; // contains functions to create transactions for ibc module pub

mod pgf; // contains functions to create transactions for pgf module pub

mod pos; // contains functions to create transactions for pos module pub

mod transfer; // contains functions to create transactions for transfer module

Methods associated with every struct

Each struct in the transaction module implements the following methods:

/// Build a raw transaction from the given parameters pub

fn

new (...) -> Self { ... } /// Get the bytes to sign for the given transaction pub

fn

get_sign_bytes (& self) ->

Vec < Hash

{ ... } /// Attach a signature to the given transaction pub

fn

attach_signatures (self, signer : common :: PublicKey , signature : common :: Signature) -> Self { ... } /// Attach a fee to the given transaction pub

fn

attach_fee (self, fee :

DenominatedAmount , token :

Address , fee_payer : common :: PublicKey , epoch :

Epoch , gas_limit :

GasLimit) -> Self { ... } /// Get the bytes of the fee data to sign pub

fn

get_fee_sig_bytes (& self) ->

Hash { ... } /// Attach a fee signature to the given transaction pub

fn

attach_fee_signature (self, signer : common :: PublicKey , signature : common :: Signature) -> Self { ... } /// Generates the protobuf encoding of this transaction pub

fn

to_bytes (& self) ->

Vec < u8

{ ... } /// Gets the inner transaction without the domain wrapper pub

fn

payload (self) ->

Tx { ... } /// Validate this wrapper transaction pub

fn

validate_tx (& self) ->

Result < Option < & Signature

, TxError { ... } Using these modules is given in more detail under theexamples section . Below we describe the instantiating of each struct associated with the respective module.

Account

use namada_sdk :: key :: common; use namada_sdk :: hash :: Hash ; use super :: GlobalArgs ; pub

struct

InitAccount (Tx); impl

InitAccount { pub

fn

new (public_keys :

Vec <common :: PublicKey

, vp_code_hash :

Hash , threshold :

u8 , args :

GlobalArgs ,) -> Self { ... } } A public key can be constructed from a string using the::from_str() method. Thevp_code_hash is a hash that is found under thewasm folder.

Bridge

pub

use namada_sdk :: eth_bridge_pool :: { GasFee , TransferToEthereum }; pub

struct

BridgeTransfer (Tx); impl

BridgeTransfer { pub

fn

new (transfer :

TransferToEthereum , gas_fee :

GasFee , args :

GlobalArgs ,) -> Self { ... } }

Governance

pub

struct

InitProposal (Tx);

impl

InitProposal { /// Build a raw InitProposal transaction from the given parameters

[allow(clippy

:: too_many_arguments)] pub

fn

new (id :

u64 , content :

Hash , author :

Address , r#type :

ProposalType , voting_start_epoch :

Epoch , voting_end_epoch :

Epoch , grace_epoch :

Epoch , args :

GlobalArgs ,) -> Self { ... } } pub

struct

VoteProposal (Tx);

impl

VoteProposal { /// Build a raw VoteProposal transaction from the given parameters pub

fn

new (id :

u64 , vote :

ProposalVote , voter :

Address , delegations :

Vec < Address

, args :

GlobalArgs ,) -> Self { ... } }

IBC

pub

struct

IbcTransfer (Tx);

impl

IbcTransfer { /// Build a raw IbcTransfer transaction from the given parameters pub

fn

new (packet_data :

MsgTransfer , GlobalArgs { expiration, code_hash, chain_id, } :

GlobalArgs ,) -> Self { ... } }

PGF

pub

struct

ResignSteward (Tx);

impl

ResignSteward { /// Build a raw ResignSteward transaction from the given parameters pub

fn

new (steward :

Address , args :

GlobalArgs) -> Self { ... } } pub

struct

UpdateStewardCommission (Tx);

impl

UpdateStewardCommission { /// Build a raw UpdateStewardCommission transaction from the given /// parameters pub

fn

new (steward :

Address , commission :

HashMap < Address , Dec

, args :

GlobalArgs ,) -> Self { ... } }

POS

pub

struct

Bond (Tx); impl

Bond { /// Build a raw Bond transaction from the given parameters pub

fn

new (validator :

Address , amount : token :: Amount , source :

Option < Address

, args :

GlobalArgs ,) -> Self { ... } } pub

struct

Unbond (Tx); impl

Unbond { /// Build a raw Unbond transaction from the given parameters pub

fn

new (validator :

Address , amount : token :: Amount , source :

Option < Address

, args :

GlobalArgs ,) -> Self { ... } }

pub

struct

BecomeValidator (Tx); impl

BecomeValidator { /// Build a raw Init validator transaction from the given parameters

[allow(clippy

:: too_many_arguments)] pub

fn

new (address :

Address , consensus_key : common :: PublicKey , eth_cold_key : secp256k1 :: PublicKey , eth_hot_key : secp256k1 :: PublicKey , protocol_key : common :: PublicKey , commission_rate :

Dec , max_commission_rate_change :

Dec , email :

String , description :

Option < String

, website :

Option < String

, discord_handle :

Option < String

, avatar :

Option < String

, args :

GlobalArgs ,) -> Self { ... } }

pub

struct

UnjailValidator (Tx); impl

UnjailValidator { /// Build a raw Unjail validator transaction from the given parameters pub

fn

new (address :

Address , args :

GlobalArgs) -> Self { ... } } pub

struct

DeactivateValidator (Tx); impl

DeactivateValidator { /// Build a raw DeactivateValidator transaction from the given parameters pub

fn

new (address :

Address , args :

GlobalArgs) -> Self { ... } } pub

struct

ReactivateValidator (Tx); impl

ReactivateValidator { /// Build a raw ReactivateValidator transaction from the given parameters pub

fn

new (address :

Address , args :

GlobalArgs) -> Self { ... } }

pub

struct

ClaimRewards (Tx); impl

ClaimRewards { /// Build a raw ClaimRewards transaction from the given parameters pub

fn

new (validator :

Address , source :

Option < Address

, args :

GlobalArgs ,) -> Self { ... } }

pub

struct

ChangeMetaData (Tx); impl

ChangeMetaData { /// Build a raw ChangeMetadata transaction from the given parameters

[allow(clippy

:: too_many_arguments)] pub

fn

new (validator :

Address , email :

Option < String

, description :

Option < String

, website :

Option < String

, discord_handle :

Option < String

, avatar :

Option < String

, commission_rate :

Option < Dec

, args :

GlobalArgs ,) -> Self { ... } }

pub

struct

ChangeConsensusKey (Tx); impl

ChangeConsensusKey { /// Build a raw ChangeConsensusKey transaction from the given parameters pub

fn

new (validator :

Address , consensus_key : common :: PublicKey , args :

GlobalArgs ,) -> Self { ... } } pub

struct

ChangeCommission (Tx); impl

ChangeCommission { /// Build a raw ChangeCommission transaction from the given parameters pub

fn

new (validator :

Address , new_rate :

Dec , args :

GlobalArgs) -> Self { ... } } pub

struct

Withdraw (Tx); impl

Withdraw { /// Build a raw Withdraw transaction from the given parameters pub

fn

new (validator :

Address , source :

Option < Address

, args :

GlobalArgs ,) -> Self { ... } }

pub

struct

Redelegate (Tx); impl

Redelegate { /// Build a raw Redelegate transaction from the given parameters pub

fn

new (src_validator :

Address , dest_validator :

Address , owner :

Address , amount :

Amount , args :

GlobalArgs ,) -> Self { ... } }

Reading Writing