Reading

The reading module exposes the following functions:

pub

fn

query_native_token (tendermint_addr :

& str) ->

Result < Address , Error

{ ... } /// Query the last committed block, if any. pub

fn

query_block (tendermint_addr :

& str) ->

Result < Option < LastBlock

, Error { ... } /// Query the results of the last committed block pub

fn

query_results (tendermint_addr :

& str) ->

Result < Vec < BlockResults

, Error { ... } /// Get a properly denominated amount of a token pub

fn

denominate_amount (tendermint_addr :

& str , amount :

u64 , token :

& str) ->

Result < DenominatedAmount , Error

{ ... } Each of these functions returns a Result type, which can be used to handle errors. Thetendermint_addr parameter is the address of the Namada node to query. For local nodes, this is usuallyhttp://localhost:26657 .

It also exposes the following modules that contain their own functions within them:

pub

mod account; pub

mod governance; pub

mod pgf; pub

mod pos; pub

mod tx;

Account

The account module exposes the following functions:

// Query token amount of owner pub

fn

```
get_token_balance (tendermint_addr :
& str , token :
& Address , owner :
& Address ) ->
Result <token :: Amount , Error
    { ... }
// Check if the address exists on chain pub
fn
known_address (tendermint_addr :
& str , address :
& Address ) ->
Result < bool , Error
    { ... }
// Query the account substorage space of an address pub
fn
get_account_info (tendermint_addr :
& str , owner :
& Address ) ->
Result < Option < Account
    , Error { ... }
// Query if the public_key is revealed pub
fn
is_public_key_revealed (tendermint_addr :
& str , owner :
& Address ) ->
Result < bool , Error
    { ... }
// Query an account substorage at a specific index pub
fn
get_public_key_at (tendermint_addr :
& str , owner :
& Address , index :
u8 ) ->
Result < Option <common :: PublicKey
    , Error { ... }
```

Governance

```
// Query proposal by Id pub
```

```
fn
query_proposal_by_id ( tendermint_addr :
& str , proposal_id :
u64 ,) ->
Result < Option < StorageProposal
    , Error { ... }
// Get the givernance parameters pub
fn
query_governance_parameters (tendermint_addr :
& str ,) ->
Result < GovernanceParameters , Error
    { ... }
// Get the givernance parameters pub
fn
query_proposal_votes (tendermint_addr :
& str , proposal_id :
u64 ,) ->
Result < Vec < Vote
    , Error { ... }
```

Pgf

```
// Check if the given address is a pgf steward. pub
fn
is_steward (tendermint_addr :
& str ,address :
& Address ,) ->
Result < bool , Error
    { ... }
```

Pos

```
// Query the epoch of the last committed block pub
fn
query_epoch (tendermint_addr :
& str ) ->
Result < Epoch , Error
    { ... }
// Query the epoch of the given block height, if it exists. pub
fn
query_epoch_at_height (tendermint_addr :
```

```
& str , height :
BlockHeight ) ->
Result < Option < Epoch
        , Error { ... }
// Check if the given address is a known validator pub
fn
is_validator (tendermint_addr :
& str , address :
& Address ) ->
Result < bool , Error
        { ... }
// Check if a given address is a known delegator pub
fn
is_delegator (tendermint_addr :
& str , address :
& Address ) ->
Result < bool , Error
        { ... }
// Check if a given address is a known delegator at the given epoch pub
fn
is_delegator_at (tendermint_addr :
& str , address :
& Address , epoch :
Epoch ) ->
Result < bool , Error
        { ... }
// Get the set of consensus keys registered in the network pub
fn
get_consensus_keys (tendermint_addr :
& str ) ->
Result < BTreeSet <common :: PublicKey
        , Error { ... }
// Get the PoS parameters pub
fn
get_pos_params (tendermint_addr :
& str ) ->
Result < PosParams , Error
```

```
        { ... }
// Get all validators in the given epoch pub
fn
get_all_validators (tendermint_addr :
& str , epoch :
Epoch ) ->
Result < HashSet < Address
        , Error { ... }
// Get the total staked tokens in the given epoch pub
fn
get_total_staked_tokens (tendermint_addr :
& str , epoch :
Epoch ) ->
Result <token :: Amount , Error
        { ... }
// Get the given validator's stake at the given epoch pub
fn
get_validator_stake (tendermint_addr :
& str , epoch :
Epoch , validator :
& Address ) ->
Result <token :: Amount , Error
        { ... }
// Query and return a validator's state pub
fn
get_validator_state (tendermint_addr :
& str , validator :
& Address , epoch :
Option < Epoch
        ) ->
Result < Option < ValidatorState
        , Error { ... }
// Get the delegator's delegation pub
fn
get_delegators_delegation (tendermint_addr :
& str , address :
& Address ) ->
```

```
Result < HashSet < Address
        , Error { ... }
// Get the delegator's delegation at some epoch pub
fn
get_delegators_delegation_at (tendermint_addr :
& str , address :
& Address , epoch :
Epoch ) ->
Result < HashMap < Address , token :: Amount
        , Error { ... }
// Query and return validator's commission rate and max commission rate change per epoch pub
fn
query_commission_rate (tendermint_addr :
& str , validator :
& Address , epoch :
Option < Epoch
        ) ->
Result < Option < CommissionPair
        , Error { ... }
// Query and return validator's metadata, including the commission rate and max commission rate change pub
fn
query_metadata (tendermint_addr :
& str , validator :
& Address , epoch :
Option < Epoch
        ) ->
Result <( Option < ValidatorMetaData
        , Option < CommissionPair ), Error { ... }
// Query and return the incoming redelegation epoch for a given pair of source validator and delegator, if there is any pub
fn
query_incoming_redelegations (tendermint_addr :
& str , src_validator :
& Address , delegator :
& Address ) ->
Result < Option < Epoch
        , Error { ... }
// Query a validator's bonds for a given epoch pub
```

```
fn
query_bond (tendermint_addr :
& str , source :
& Address , validator :
& Address , epoch :
Option < Epoch
        ) ->
Result <token :: Amount , Error
        { ... }
// Query withdrawable tokens in a validator account for a given epoch pub
fn
query_withdrawable_tokens (tendermint_addr :
& str , bond_source :
& Address , validator :
& Address , epoch :
Option < Epoch
        ) ->
Result <token :: Amount , Error
        { ... }
// Query all unbonds for a validator, applying slashes pub
fn
query_unbond_with_slashing (tendermint_addr :
& str , source :
& Address , validator :
& Address ) ->
Result < HashMap <( Epoch , Epoch ), token :: Amount
        , Error { ... }
// Get the bond amount at the given epoch pub
fn
get_bond_amount_at (tendermint_addr :
& str , delegator :
& Address , validator :
& Address , epoch :
Epoch ) ->
Result <token :: Amount , Error
        { ... }
// Get bonds and unbonds with all details (slashes and rewards, if any) grouped by their bond IDs pub
```

fn

bonds_and_unbonds (tendermint_addr :

& str , source :

& Option < Address

, validator :

& Option < Address

) ->

Result < BondsAndUnbondsDetails , Error

{ ... }

// Get bonds and unbonds with all details (slashes and rewards, if any) grouped by their bond IDs, enriched with extra information calculated from the data pub

fn

enriched_bonds_and_unbonds (tendermint_addr :

& str , current_epoch :

Epoch , source :

& Option < Address

, validator :

& Option < Address

) ->

Result < EnrichedBondsAndUnbondsDetails , Error

{ ... }

Tx

/// Call the corresponding tx_event_query RPC method, to fetch /// the current status of a transation. pub

fn

query_tx_events (tendermint_addr :

& str , tx_hash :

& str ,) ->

Result < Option < Event

, Error { ... }

/// Dry run a transaction pub

fn

dry_run_tx (tendermint_addr :

& str , tx_bytes :

Vec < u8

) ->

Result < TxResult , Error

{ ... }

/// Lookup the full response accompanying the specified transaction event pub

fn

query_tx_response (tendermint_addr :

& str , tx_hash :

& str) ->

Result < TxResponse , Error

{ ... }

/// Query the status of a given transaction. pub

fn

query_tx_status (tendermint_addr :

& str , tx_hash :

& str) ->

Result < Event , Error

{ ... }

Usage Transactions