Transaction Module

This module provides functionalities to send transactions on BandChain which requiresto be included.

Exceptions

Type Description ValueError Invalid value NotIntegerError The value is not an integer EmptyMsgError No messages given to the transaction UndefinedError The value should not be undefined

withMessages(msgs)

Add one or multiple messages as a list of Google Protobuf's Any toTransaction . There are predefined message classes that can be used to convert toAny instance usingtoAny() method, but for the other type of message can be converted toAny instance usingAny.pack() as shown in the code below.

import

{

MsgCreateDataSource

}

from

'@bandprotocol/bandchain.js/proto/oracle/v1/tx_pb' import

{

Any

}

from

'google-protobuf/google/protobuf/any_pb' import

{

Transaction

}

from

'@bandprotocol/bandchain.js'

const msg =

new

MsgCreateDataSource ( ) msg . setName ('dsName') msg . setDescription ('dsDescription') // msg.set...() for every fields

const anyMsg =

new

Any ( ) const typeUrl =

'oracle.v1.MsgCreateDataSource' anyMsg . pack (msg . serializeBinary ( ) , typeUrl ,

'/')

const tx =

new

Transaction ( ) tx . withMessages (anyMsg) Parameter

- *msgs
- Any

- Messages converted toAny
- to be included in the transaction

Return

- Transaction
-
    - This transaction instance for function chaining

withSender(client, sender)

Set account number and sequence number toTransaction from querying the account via Client's getAccount .

Parameter

- client
- Client
-
    - An instance of Client module
- sender
- string
-
    - A bech32-encoded with account prefix

Return

- Promise
-
    - This transaction instance for function chaining

Exceptions

Type Description EmptyMsgError Message is empty, please use withMessages at least 1 message NotFoundError Account doesn't exist.

withAccountNum(accountNum)

Set account number toTransaction .

Parameter

- accountNum
- number
-
    - An integer of account number, which can be gathered from querying the account via Client's getAccount
- .

Return

- Transaction
-
    - This transaction instance for function chaining

Exceptions

Type Description NotIntegerError accountNum is not an integer

withSequence(sequence)

Set sequence number toTransaction .

Parameter

- sequence
- Number
-
    - An integer of account's sequence number, which can be gathered from querying the account via Client'sgetAccount
- .

Return

- Transaction
-
    - This transaction instance for function chaining

Exceptions

Type Description NotIntegerError sequence is not an integer

withChainId(chainId)

Set chain id toTransaction .

Parameter

- chainId
- string
-
    - a string of chain ID, which can be gathered from Client's getChainId
- .

Return

- Transaction
-
    - This transaction instance for function chaining

withFee(fee)

Set fee toTransaction .

Parameter

- fee
-
    - Set fee limit spent for gas price of the transaction.

Return

- Transaction
-
    - This transaction instance for function chaining

Exceptions

Type Description NotIntegerError fee is not an integer

withMemo(memo)

Set memo toTransaction .

Parameter

- memo
- string
-
    - an arbitrary string to remember the transaction. Memo length is limited to 256.

Exceptions

Type Description ValueTooLargeError memo is too large. Return

- Transaction
-
    - This transaction instance for function chaining

getSignDoc()

Get serialized data of transaction's content to be signed fromTransaction by usingSIGNMODE_DIRECT . See more aboutsigning mode .

Return

- Uint8Array
-
    - A byte array of serialized transaction content data, ready for signing.

Exceptions

Type Description EmptyMsgError message is empty UndefinedError accountNum should be defined UndefinedError sequence should be defined UndefinedError chainID should be defined

getSignMessage()

Get serialized data of transaction's content to be signed fromTransaction by usingSIGN_MODE_LEGACY_AMINO_JSON . See more aboutsigning mode .When using the Ledger to sign message, you need to use this method.

Return

- Uint8Array
-
    - A byte array of serialized transaction content data, ready for signing.

getTxData(signature, publicKey, signMode)

Get transaction data fromTransaction .

Parameter

- signature
- Uint8Array
-
    - signature for the transaction
- pubkey
- PublicKey
-
    - an instance of public key to be included in the transaction
- signMode
- SignMode
-
    - a signing mode with its own security guarantees representation. (Default:SIGN_MODE_DIRECT
-)

Return

- Uint8Array
-
    - Serialized data of signed transaction, ready to be broadcasted to BandChain.

Exceptions

Type Description UndefinedError accountNum should be defined UndefinedError sequence should be defined

Example

import

{

Client ,

Wallet ,

Transaction ,

Message ,

Coin ,

Fee

}

from

'@bandprotocol/bandchain.js'

const

{

PrivateKey

}

=

Wallet const client =

new

Client ('https://laozi-testnet6.bandchain.org/grpc-web')

// Step 2.1 import private key based on given mnemonic string const privkey =

PrivateKey . fromMnemonic ('subject economy equal whisper turn boil guard giraffe stick retreat wealth card only buddy joy leave genuine resemble submit ghost top polar adjust avoid') // Step 2.2 prepare public key and its address const pubkey = privkey . toPubkey ( ) const sender = pubkey . toAddress ( ) . toAccBech32 ( )

const

sendCoin

=

async

( )

=>

{ // Step 3.1 constructs MsgSend message const

{

MsgSend

}

=

Message

// Here we use different message type, which is MsgSend const receiver =

'band1p46uhvdk8vr829v747v85hst3mur2dzlmlac7f' const sendAmount =

new

Coin ( ) sendAmount . setDenom ('uband') sendAmount . setAmount ('10') const msg =

new

MsgSend (sender , receiver ,

[sendAmount]) // Step 3.2 constructs a transaction const account =

await client . getAccount (sender) const chainId =

'band-laozi-testnet6'

let feeCoin =

new

Coin ( ) feeCoin . setDenom ('uband') feeCoin . setAmount ('1000')

const fee =

new

Fee ( ) fee . setAmountList ([feeCoin]) fee . setGasLimit (1000000) const tx =

new

Transaction ( ) . withMessages (msg) . withAccountNum (account . accountNumber) . withSequence (account . sequence) . withChainId (chainId) . withFee (fee)

// Step 4 sign the transaction const txSignData = tx . getSignDoc (pubkey) const signature = privkey . sign (txSignData) const signedTx = tx . getTxData (signature , pubkey)

// Step 5 send the transaction const response =

await client . sendTxBlockMode (signedTx) console . log (JSON . stringify (response)) }

; (async

( )

=>

{ await

sendCoin ( ) }) ( ) Result

{ "height" :

144075 , "txhash" :

"FB0C1B122116EC1C94EE3BC05FC86B41EC580AC5A6CC7F5A3954F61E0505C648" , "codespace" :

"" , "code" :

0 , "data" :

"0A1E0A1C2F636F736D6F732E62616E6B2E763162657461312E4D736753656E64" , "rawLog" :

"[{\"events\":[{\"type\":\"coin_received\",\"attributes\":
[{\"key\":\"receiver\",\"value\":\"band1p46uhvdk8vr829v747v85hst3mur2dzlmlac7f\"},
{\"key\":\"amount\",\"value\":\"10uband\"}]},{\"type\":\"coin_spent\",\"attributes\":
[{\"key\":\"spender\",\"value\":\"band18p27yl962l8283ct7srr5l3g7ydazj07dqrwph\"},{\"key\":\"amount\",\"value\":\"10uband\"}]},
{\"type\":\"message\",\"attributes\":[{\"key\":\"action\",\"value\":\"/cosmos.bank.v1beta1.MsgSend\"},
{\"key\":\"sender\",\"value\":\"band18p27yl962l8283ct7srr5l3g7ydazj07dqrwph\"},{\"key\":\"module\",\"value\":\"bank\"}]},
{\"type\":\"transfer\",\"attributes\":[{\"key\":\"recipient\",\"value\":\"band1p46uhvdk8vr829v747v85hst3mur2dzlmlac7f\"},
{\"key\":\"sender\",\"value\":\"band18p27yl962l8283ct7srr5l3g7ydazj07dqrwph\"},{\"key\":\"amount\",\"value\":\"10uband\"}]}]}]"
, "logsList" :

[{ "msgIndex" :

0 , "log" :

"" , "eventsList" :

[{ "type" :

"coin_received" , "attributesList" :

[{ "key" :

"receiver" , "value" :

"band1p46uhvdk8vr829v747v85hst3mur2dzlmlac7f" } , {

"key" :

"amount" ,

"value" :

"10uband"

}] } , { "type" :

"coin_spent" , "attributesList" :

[{ "key" :

"spender" , "value" :

"band18p27yl962l8283ct7srr5l3g7ydazj07dqrwph" } , {

"key" :

"amount" ,

"value" :

"10uband"

}] } , { "type" :

"message" , "attributesList" :

[{

"key" :

"action" ,

"value" :

"/cosmos.bank.v1beta1.MsgSend"

} , { "key" :

"sender" , "value" :

"band18p27yl962l8283ct7srr5l3g7ydazj07dqrwph" } , {

"key" :

"module" ,

"value" :

"bank"

}] } , { "type" :

"transfer" , "attributesList" :

[{ "key" :

"recipient" , "value" :

"band1p46uhvdk8vr829v747v85hst3mur2dzlmlac7f" } , { "key" :

"sender" , "value" :

"band18p27yl962l8283ct7srr5l3g7ydazj07dqrwph" } , {

"key" :

"amount" ,

"value" :

"10uband"

}] }] }] , "info" :

"" , "gasWanted" :

200000 , "gasUsed" :

66907 , "timestamp" :

"" } Previous OBI Module Next Wallet Module