Wallet Module

This module provides functionalities to manage BandChain account. It can be used to create a private key from mnemonic, sign and verify signature, and account address manipulation.

PrivateKey

A class for wrapping signing key, which is a byte array of private key, used for signature creation and public key derivation.

generate(path)

A static method for generating new private key with random mnemonic phrase.

info If path is not given, default to Band's HD prefix 494 and all other indexes being zeroes. Parameter

- pathstring
-
    - The HD path that follows the BIP32 standard (optional)

Return

- [string, PrivateKey]
-
    - A tuple of mnemonic phrase and[PrivateKey
-]instance

Exception

Type Description CreateError Cannot create private key Example

import

{

Wallet

}

from

'@bandprotocol/bandchain.js'

const

{

PrivateKey

}

=

Wallet const

[mnemonic , priv]

=

PrivateKey . generate ("m/44'/494'/0'/0/3")

fromMnemonic(word, path)

A static method for creating a[PrivateKey]instance from a given mnemonic phrase and a HD derivation path.

info If path is not given, default to Band's HD prefix 494 and all other indexes being zeroes. Parameter

- words
- string
-
    - The mnemonic phrase for recover private key
- path
- string
-
    - The HD path that follows the BIP32 standard (optional)

Return

- [PrivateKey
-]- Initialized PrivateKey object

Exception

Type Description CreateError Cannot create private key Example

import

{

Wallet

}

from

'@bandprotocol/bandchain.js'

const

{

PrivateKey

}

=

Wallet const priv =

PrivateKey . fromMnemonic ('test mnemonic')

fromHex(priv)

Create a[PrivateKey]instance from a given a hex representation of signing key.

Parameter

- priv
- string
-
    - A hex representation of signing key.

Return

- [PrivateKey
-]- Initialized PrivateKey object

Example

import

{

Wallet

}

from

'@bandprotocol/bandchain.js'

const

{

PrivateKey

}

=

Wallet const priv =

PrivateKey . fromHex ('2442b724db7189468f16accc0fc505f0609817eb129e13702e696d8b84609ea9')

toHex

Return a hex representation of signing key.

Return

- string
-
    - A hex representation of signing key.

Example

import

{

Wallet

}

from

'@bandprotocol/bandchain.js'

const

{

PrivateKey

}

=

Wallet const priv =

PrivateKey . fromMnemonic ('test mnemonic') console . log (priv . toHex ( )) Result

2cb2e2d3582cebf0664d9cda0b89c5d478ae12fac19a6f4ed9c10a7406a19615

toPubkey

Create a[PublicKey]instance which is associated with given private key.

Return

- [PublicKey
-]- A PublicKey that can be used to verify the signatures made with this PrivateKey.

Example

import

{

Wallet

}

from

'@bandprotocol/bandchain.js'

const

{

PrivateKey

}

=

Wallet const priv =

PrivateKey . fromMnemonic ('test mnemonic') console . log (priv . toPubkey ( ) . toHex ( )) Result

02b2b0d35cb1c6d3923813c64e46a82d29e12d03288f438b9d3cf232d9a22bcb83

sign(msg)

Sign and the given bytes array.

Parameter

- msg
- Buffer
-
    - The message that will be hashed and signed.

Return

- Buffer
-
    - A signature of this private key over the given message

Example

import

{

Wallet

}

from

'@bandprotocol/bandchain.js'

const

{

PrivateKey

}

=

Wallet const priv =

PrivateKey . fromMnemonic ('test mnemonic') console . log (priv . sign (Buffer . from ('test message')) . toString ('hex')) Result

4bbc9a7ea54b47b11c67a4074e8d9bca068cb64c788f67342c4033b1b6f0553e1bc63cdf9bc2fb6e89c1e965c1e0f0712a51c250627282309cd2fccf1470f4f6

PublicKey

Class for wraping verify Key, which is a byte array of public key, used for signature verification, and generate bech32-encoded address.

fromAccBech32(bech)

Creates a[PublicKey]instance from a bech32-encoded public key with account-pubkey prefix.

Parameter

- bech
- string
-
    - A bech32-encoded with account public key prefix.

Return

- [PublicKey
-]- A PublicKey instance

Exception

Type Description ValueError Invalid bech32 prefix DecodeError Cannot decode bech32 Example

import

{

Wallet

}

from

'@bandprotocol/bandchain.js'

const

{

PublicKey

}

=

Wallet const publickey =

PublicKey . fromAccBech32 ('bandpub1addwnpepqdg7nrsmuztj2re07svgcz4vuzn3de56nykdwlualepkk05txs5q6mw8s9v')

fromValBech32(bech)

Creates a[PublicKey]instance from a bech32-encoded public key with validator-pubkey prefix

Parameter

- bech
- string
-
    - A bech32-encoded with validator public key prefix

Return

- [PublicKey
-]- A PublicKey instance

Exception

Type Description ValueError Invalid bech32 prefix DecodeError Cannot decode bech32 Example

import

{

Wallet

}

from

'@bandprotocol/bandchain.js'

const

{

PublicKey

}

=

Wallet const publickey =

PublicKey . fromValBech32 ('bandvaloperpub1addwnpepqdg7nrsmuztj2re07svgcz4vuzn3de56nykdwlualepkk05txs5q69gsm29')

fromConsBech32(bech)

Creates a[PublicKey]instance from a bech32-encoded public key with validator-consensus-pubkey prefix

Parameter

- bech
- string
- A bech32-encoded with validator consensus public key prefix

Return

- [PublicKey
-]- A PublicKey instance

Exception

Type Description ValueError Invalid bech32 prefix DecodeError Cannot decode bech32 Example

import

{

Wallet

}

from

'@bandprotocol/bandchain.js'

const

{

PublicKey

}

=

Wallet const publickey =

PublicKey . fromConsBech32 ('bandvalconspub1addwnpepqdg7nrsmuztj2re07svgcz4vuzn3de56nykdwlualepkk05txs5q6r8ytws')

toHex

Returns a hex representation of verifying key.

Return

- string
-
    - A hex representation of verifying key.

Example

import

{

Wallet

}

from

'@bandprotocol/bandchain.js'

const

{

PublicKey

}

=

Wallet const publickey =

PublicKey . fromAccBech32 ('bandpub1addwnpepqdg7nrsmuztj2re07svgcz4vuzn3de56nykdwlualepkk05txs5q6mw8s9v') console . log (publickey . toHex ( )) Result

0351e98e1be097250f2ff4188c0aace0a716e69a992cd77f9dfe436b3e8b34280d

toPubkeyProto

Returns an instance of Cosmos SDK Protobuf's[PubKey], which can be used for constructing transactions.

Return

- [PubKey
-]- An instance of Cosmos SDK Protobuf'sPubKey

Example

import

{

Wallet

}

from

'@bandprotocol/bandchain.js'

const

{

PublicKey

}

=

Wallet const publickey =

PublicKey . fromAccBech32 ('bandpub1addwnpepqdg7nrsmuztj2re07svgcz4vuzn3de56nykdwlualepkk05txs5q6mw8s9v') // Converting to base64 is for displaying only console . log (publickey . toPubkeyProto ( ) . getKey_asB64 ( )) Result

A1HpjhvglyUPL/QYjAqs4KcW5pqZLNd/nf5Daz6LNCgN

toAccBech32

Returns bech32-encoded public key with account-pubkey prefix

Return

- string
-
    - A bech32-encoded with account public key prefix.

Example

import

{

Wallet

}

from

'@bandprotocol/bandchain.js'

const

{

PublicKey

}

=

Wallet const publickey =

PublicKey . fromAccBech32 ('bandpub1addwnpepqdg7nrsmuztj2re07svgcz4vuzn3de56nykdwlualepkk05txs5q6mw8s9v') console . log (publickey . toAccBech32 ( )) Result

bandpub1addwnpepqdg7nrsmuztj2re07svgcz4vuzn3de56nykdwlualepkk05txs5q6mw8s9v

toValBech32

Returns bech32-encoded public key with validator-pubkey prefix

Return

- string
-
    - A bech32-encoded with validator public key prefix.

Exceptions

Type Description UnsuccessfulCallError Unsuccessful bech32.toWords call Example

import

{

Wallet

}

from

'@bandprotocol/bandchain.js'

const

{

PublicKey

}

=

Wallet const publickey =

PublicKey . fromAccBech32 ('bandpub1addwnpepqdg7nrsmuztj2re07svgcz4vuzn3de56nykdwlualepkk05txs5q6mw8s9v') console . log (publickey . toValBech32 ( )) Result

bandvaloperpub1addwnpepqdg7nrsmuztj2re07svgcz4vuzn3de56nykdwlualepkk05txs5q69gsm29

toConsBech32

Returns bech32-encoded public key with validator-consensus-pubkey prefix

Return

- string
-
    - A bech32-encoded with validator consensus public key prefix.

Exceptions

Type Description UnsuccessfulCallError Unsuccessful bech32.toWords call Example

import

{

Wallet

}

from

'@bandprotocol/bandchain.js'

const

{

PublicKey

}

=

Wallet const publickey =

PublicKey . fromAccBech32 ('bandpub1addwnpepqdg7nrsmuztj2re07svgcz4vuzn3de56nykdwlualepkk05txs5q6mw8s9v') console . log (publickey . toConsBech32 ( )) Result

bandvalconspub1addwnpepqdg7nrsmuztj2re07svgcz4vuzn3de56nykdwlualepkk05txs5q6r8ytws

toAddress

Returns an[Address]instance from given public key

Return

- [Address
-]- An Address instance.

Example

import

{

Wallet

}

from

'@bandprotocol/bandchain.js'

const

{

PublicKey

}

=

Wallet const publickey =

PublicKey . fromAccBech32 ('bandpub1addwnpepqdg7nrsmuztj2re07svgcz4vuzn3de56nykdwlualepkk05txs5q6mw8s9v') console . log (publickey . toAddress ( ) . toHex ( )) Result

8e453e66fb009b119ea9b1b8775be6fa9435013b

verify(msg, sig)

Verify a signature made over provided data.

Parameter

- msg
- Buffer
- A data signed by thesignature
- , will be hashed using sha256 function
- sig
- Buffer
- A encoding of the signature

Return

- boolean
- True if the verification was successful

Exception

Type Description BadSignatureError if the signature is invalid or malformed Example

import

{

Wallet

}

from

'@bandprotocol/bandchain.js'

const

{

PrivateKey

}

=

Wallet const priv =

PrivateKey . fromMnemonic ('test mnemonic') const pubkey = priv . toPubkey ( ) const msg =

Buffer . from ('test message') const sig = priv . sign (msg) console . log (pubkey . verify (msg , sig)) Result

true

Address

Class for wraping Address. Adding method to encode/decode to Bech32 format.

fromAccBech32(bech)

Create an[Address]instance from a bech32-encoded with account prefix.

Parameter

- bech
- string
-
    - A bech32-encoded with account prefix.

Return

- [Address
-]- A Address instance

Exception

Type Description ValueError Invalid bech32 prefix DecodeError Cannot decode bech32 Example

import

{

Wallet

}

from

'@bandprotocol/bandchain.js'

const

{

Address

}

=

Wallet const address =

Address . fromAccBech32 ('band13eznuehmqzd3r84fkxu8wklxl22r2qfmtlth8c')

fromValBech32(bech)

Create an[Address]instance from a bech32-encoded with validator prefix

Parameter

- bech
- string
-
    - A bech32-encoded with validator prefix

Return

- [Address
-]- A Address instance

Exception

Type Description ValueError Invalid bech32 prefix DecodeError Cannot decode bech32 Example

import

{

Wallet

}

from

'@bandprotocol/bandchain.js'

const

{

Address

}

=

Wallet const address =

Address . fromValBech32 ('bandvaloper13eznuehmqzd3r84fkxu8wklxl22r2qfm8f05zn')

fromConsBech32(bech)

Create an[Address]instance from a bech32-encoded with validator consensus prefix

Parameter

- bech
- string
-
    - A bech32-encoded with validator consensus prefix

Return

- [Address
-]- A Address instance

Exception

Type Description ValueError Invalid bech32 prefix DecodeError Cannot decode bech32 Example

import

{

Wallet

}

from

'@bandprotocol/bandchain.js'

const

{

Address

}

=

Wallet const address =

Address . fromConsBech32 ('bandvalcons13eznuehmqzd3r84fkxu8wklxl22r2qfmn6ugwj')

toHex

Return a hex representation ofAddress .

Return

- string
-
    - A hex representation of Address.

Example

import

{

Wallet

}

from

'@bandprotocol/bandchain.js'

const

{

Address

}

=

Wallet const address =

Address . fromAccBech32 ('band13eznuehmqzd3r84fkxu8wklxl22r2qfmtlth8c') console . log (address . toHex ( )) Result

8e453e66fb009b119ea9b1b8775be6fa9435013b

toAccBech32

Return bech32-encoded with account prefix

Return

- string
-
    - A bech32-encoded with account prefix.

Exceptions

Type Description UnsuccessfulCallError Unsuccessful bech32.toWords call Example

import

{

Wallet

}

from

'@bandprotocol/bandchain.js'

const

{

Address

}

=

Wallet const address =

Address . fromAccBech32 ('band13eznuehmqzd3r84fkxu8wklxl22r2qfmtlth8c') console . log (address . toAccBech32 ( )) Result

band13eznuehmqzd3r84fkxu8wklxl22r2qfmtlth8c

toValBech32

Return bech32-encoded with validator prefix

Return

- string
- A bech32-encoded with validator prefix.

Exceptions

Type Description UnsuccessfulCallError Unsuccessful bech32.toWords call Example

import

{

Wallet

}

from

'@bandprotocol/bandchain.js'

const

{

Address

}

=

Wallet const address =

Address . fromAccBech32 ('band13eznuehmqzd3r84fkxu8wklxl22r2qfmtlth8c') console . log (address . toValBech32 ( )) Result

bandvaloper13eznuehmqzd3r84fkxu8wklxl22r2qfm8f05zn

toConsBech32

Return bech32-encoded with validator consensus prefix

Return

- string
-
    - A bech32-encoded with validator consensus prefix.

Exceptions

Type Description UnsuccessfulCallError Unsuccessful bech32.toWords call Example

import

{

Wallet

}

from

'@bandprotocol/bandchain.js'

const

{

Address

}

=

Wallet const address =

Address . fromAccBech32 ('band13eznuehmqzd3r84fkxu8wklxl22r2qfmtlth8c') console . log (address . toConsBech32 ( )) Result

bandvalcons13eznuehmqzd3r84fkxu8wklxl22r2qfmn6ugwj

Ledger

Ledger is a fundamental class that allows you to interact and perform actions with your Ledger devices; signing, and obtaining the account address.

connectLedgerNode(hdPath)

A static method used to connect Ledger with Node HID implementation.

Parameter

- hdPath
-
    - The HD path that follows the BIP32 standard (optional). Defaults tom/44'/118'/0'/0/0
- if not provided.

Return

-
- :Ledger
- object.

Example

import

{

Wallet

}

from

'@bandprotocol/bandchain.js'

const

{

Ledger

}

=

Wallet

const

connectLedger

=

async

( )

=>

{ const ledger =

await

Ledger . connectLedgerNode ("m/44'/118'/0'/0/0") console . log (ledger) }

; (async

( )

=>

{ await

connectLedger ( ) }) ( )

connectLedgerWeb(hdPath)

A static method used to connect Ledger with WebUSB/WebHID protocol implementation.

Parameter

- hdPath
-
    - The HD path that follows the BIP32 standard (optional). Defaults tom/44'/118'/0'/0/0
- if not provided.

Return

-
- :Ledger
- object.

import

{

Wallet

}

from

'@bandprotocol/bandchain.js'

const

{

Ledger

}

=

Wallet

const

connectLedger

=

async

( )

=>

{ const ledger =

await

Ledger . connectLedgerWeb ("m/44'/118'/0'/0/0") console . log (ledger) }

; (async

( )

=>

{ await

connectLedger ( ) }) ( )

getPubKeyAndBech32Address

This function returns public key and bech32 address with band prefix of the associated Ledger.

Return

An object containing public key and bech32 address with the following keys

- bech32_address
- pubKey

Example

import

{

Wallet

}

from

'@bandprotocol/bandchain.js'

const

{

Ledger

}

=

Wallet

; (async

( )

=>

{ const ledger =

await

Ledger . connectLedgerWeb ("m/44'/118'/0'/0/0") const addressObject =

await ledger . getPubKeyAndBech32Address ( ) console . log (addressObject) }) ( ) Return

{ bech32_address :

"band1pxzqj53rg87e2n0swh8h7a6m2umjlja6uwffp9" , pubKey :

PublicKey

{ verifyKey :

Uint8Array (33) } }

sign(transaction)

This function returns the signature of the associated Ledger. The message is signed by using the Nano AppsignTransaction function.

Parameter

- transaction
- : Sign message will be extracted from Transaction class and signed.

Return

-
- : A signature of the connected ledger over the given Transaction message.

Example in Bandchain.js Basic Usage

disconnect

As its namesake dictates, this function will disconnect the Ledger device.

Example

```
import
{
Wallet
}
from
'@bandprotocol/bandchain.js'
const
{
Ledger
}
=
Wallet
; ( async
( )
=>
{ const ledger =
await
Ledger . connectLedgerWeb ( "m/44'/118'/0'/0/0" ) await ledger . disconnect ( ) } ) ( )
```