# #

Keys

Keys allows you to manage your local tendermint keystore (wallets) for iris.

#

Available Commands

Name Description add Add an encrypted private key (either newly generated or recovered), encrypt it, and save to disk delete Delete the given key export Export private keys import Import private keys into the local keybase list List all keys migrate Migrate keys from the legacy (db-based) Keybase mnemonic Compute the bip39 mnemonic for some input entropy parse Parse address from hex to bech32 and vice versa show Retrieve key information by name or address

#

iris keys add

Derive a new private key and encrypt to disk.

iris keysadd < key-name> [flags] Flags:

Name, shorthand Default Description Required --multisig

Construct and store a multisig public key --multisig-threshold 1 K out of N required signatures --nosort false Keys passed to --multisig are taken in the order they're supplied --pubkey

Parse a public key in bech32 format and save it to disk --interactive false Interactively prompt user for BIP39 passphrase and mnemonic --ledger false Store a local reference to a private key on a Ledger device --recover false Provide seed phrase to recover existing key instead of creating --no-backup false Don't print out seed phrase (if others are watching the terminal) --dry-run false Perform action, but don't add key to local keystore --hd-path

Manual HD Path derivation (overrides BIP44 config) --coin-type 118 coin type number for HD derivation --account 0 Account number for HD derivation --index 0 Address index number for HD derivation --algo secp256k Key signing algorithm to generate keys for

#

Create a new key

iris keysadd MyKey Enter and repeat the password, at least 8 characters, then you will get a new key.

WARNING

Important

write the seed phrase in a safe place! It is the only way to recover your account if you ever forget your password.

#

Recover an existing key from seed phrase

If you forget your password or lose your key, or you wanna use your key in another place, you can recover your key by your seed phrase.

iris keysadd MyKey--recover You'll be asked to enter and repeat the new password for your key, and enter the seed phrase. Then you get your key back.

Enter a passphrasefor your key: Repeat the passphrase: Enter your recovery seed phrase:

#

Create a multisig key

The following example creates a multisig key with 3 sub-keys, and specify the minimum number of signatures as 2. The tx could be broadcast only when the number of signatures is greater than or equal to 2.

iris keysadd < multisig-keyname> --multisig-threshold= 2 --multisig = < signer-keyname-1

,< signer-keyname-2

,< signer-keyname-3

TIP

can be the type of "local/offline/ledger", but not "multi" type.

If you don't have all the permission of sub-keys, you can ask for the pubkeys to create the offline keys first, then you will be able to create the multisig key.

Offline key can be created by "iris keys add --pubkey". How to use multisig key to sign and broadcast a transaction, please refer tomultisign

#

iris keys delete

Delete a local key by the given name.

iris keys delete< key-name> [flags] Flags:

Name, shorthand Default Description Required --force, -f false Remove the key unconditionally without asking for the passphrase --yes, -y false Skip confirmation prompt when deleting offline or ledger key references

#

Delete a local key

iris keys delete MyKey

#

iris keys export

Export the keystore of a key to a json file

iris keysexport < key-name> [flags]

#

Export keystore

iris keysexport Mykey --output-file= < path-to-keystore>

#

iris keys import

Import a ASCII armored private key into the local keybase.

#

Import a ASCII armored private key

iris keysimport < name> < keyfile> [flags]

#

iris keys list

List all the keys stored by this key manager along with their associated name, type, address and pubkey.

Flags:

Name, shorthand Default Description Required --list-name

List names only

#

List all keys

iris keys list

#

iris keys migrate

Migrate key information from the legacy (db-based) Keybase to the new keyring-based Keybase.

Flags:

Name, shorthand Default Description Required --dry-run

Run migration without actually persisting any changes to the new Keybase

#

Migrate key information

iris keys migrate[flags]

#

iris keys mnemonic

Create a bip39 mnemonic, sometimes called a seed phrase, by reading from the system entropy. To pass your own entropy, useunsafe-entropy mode.

iris keys mnemonic[flags] Flags:

Name, shorthand Default Description Required --unsafe-entropy

Prompt the user to supply their own entropy, instead of relying on the system

#

Create a bip39 mnemonic

iris keys mnemonic You'll get a bip39 mnemonic with 24 words, e.g.:

beauty entire blue tape ordinary fix rotate learn smart tiger dolphin cycle cigar dish alcohol slab bachelor vital design consider paper panther mad eternal

#

iris keys parse

Convert and print to stdout key addresses and fingerprints from hexadecimal into bech32 cosmos prefixed format and vice versa.

#

Convert and print to stdout key addresses and fingerprints

iris keys parse< hex-or-bech32-address> [flags]

#

iris keys show

Get details of a local key.

iris keys show< key-name> [flags] Flags:

Name, shorthand Default Description Required --address false Output the address only (overrides --output) --bech acc The Bech32 prefix encoding for a key (acc/val/cons) --device false Output the address in a ledger device --multisig-threshold 1 K out of N required signatures --pubkey false Output the public key only (overrides --output)

#

Get details of a local key

iris keys show MyKey The following infos will be shown:

- name: Mykey type:local address: iaa1tulwx2hwz4dv8te6cflhda64dn0984harlzegw pubkey: iap1addwnpepq24rufap6u0sysqcpgsfzqhw3x8nfkhqhtmpgqt0369rlyqcg0vkgwzc4k0 mnemonic:"" threshold:0 pubkeys:[ ]

#

Get validator operator address

If an address has been bonded to be a validator operator (which the address you used to create a validator), then you can use--bech val to get the operator's address prefixed byiva and the pubkey prefixed byivp :

iris keys show MyKey--bech val Example Output:

- name: Mykey type:local address: iva1tulwx2hwz4dv8te6cflhda64dn0984hakwgk4f pubkey: ivp1addwnpepq24rufap6u0sysqcpgsfzqhw3x8nfkhqhtmpgqt0369rlyqcg0vkgd8e6zy mnemonic:"" threshold:0 pubkeys:[ ]