Multisig keys

This feature offetchd allows users to securely control keys in a number of configurations. This involves setting a minimum number of keys required to sign a transaction out of a maximum of N keys, known as a threshold number K.

Creating a multisig key

To create a multisig key, use the following syntax:

Create a simple multisig key with a threshold of 1 as default

fetchd keys add --multisig

Creating a multisig key with a higher threshold, K

fetchd keys add --multisig --multisig-threshold

Example instantiation of a multisig key: shared business multisig key

Consider a scenario in which three account holders need to collectively authorize transactions. At least two out of the three (K=2) must sign off on each transaction:

Create the three keys owned by the separate account holders

fetchd keys add fred fetchd keys add ted fetchd keys add ned

Create the multisig key from keys above

fetchd keys add business_key --multisig fred,ted,ned --multisig-threshold 2 Importantly, remember to retrieve the address ofbusiness_key for future use:

fetchd keys show -a business_key

Signing and broadcasting multisig transactions

Transactions must be signed and broadcasted before they can be executed.

In order to sign a multi signature transaction, the transaction itself must not be immediately broadcast. Instead, the key holders must each sign until a minimum threshold K signatures are present.

ℹ For this example, we will be performing the transaction on theDorado ↗(opens in a new tab)network and therefore will be usingatestfet as the denomination, and a gas price of 1000000000atestfet (this should be changed depending on the actual currency and network used).

Example: Multisig transaction

Create a key to represent a vendor that the business must pay

fetchd keys add vendor

Generate a transaction as an output file to be signed by

the keyholders, 'ted' and 'fred' in this example

```
fetchd tx bank send 1000atestfet --gas 90000 --gas-prices 1000000000atestfet --generate-only > transfer.json
```

you'll get "account *not found" error for missing funds*

add funds to *using block explorer or by eg*

```
curl -X POST -H 'Content-Type: application/json' -d '{"address":"
"}' https://faucet-dorado.fetch.ai/api/v3/claims
```

This transaction file (transfer.json) is then made available for

the first keyholder to sign, 'fred'

```
fetchd tx sign transfer.json --chain-id dorado-1 --from fred --multisig
> transfer_fredsigned.json
```

This is repeated for 'ted'

```
fetchd tx sign transfer.json --chain-id dorado-1 --from ted --multisig
> transfer_tedsigned.json
```

These two files are then collated together and used as inputs to the

multisign command to create a fully signed transaction

```
fetchd tx multisign transfer.json business_key transfer_fredsigned.json transfer_tedsigned.json > signed_transfer.json
```

Now that the transaction is fully signed, it may be broadcast

```
fetchd tx broadcast signed_transfer.json
```

Now display the result of the transaction and confirm that the vendor has

received payment

```
fetchd query bank balances
```

It is important to note that this method of signing transactions can apply to all types of transaction, including staking and withdrawal transactions as shown below.

Other multisig transaction examples

Other examples are provided below:

In order to create a staking transaction using a multisig key

the same process as above can be used with the output file of this command

```
fetchd tx staking delegate 10000atestfet --from

--gas 200000 --gas-prices 1000000000atestfet --generate-only > stake.json
```

The following command can also be used to create a withdrawal transaction for the

rewards from staking when using a multisig key - this too must be signed as before

```
fetchd tx distribution withdraw-all-rewards --from

--gas 150000 --gas-prices 1000000000atestfet --generate-only > withdrawal.json
```

Was this page helpful?

CLI - Managing tokens *CLI - Delegation*