Setting up a Multisig Account

What is a Multisig account?

A multisig account, within the Fetch Network, possesses a unique key that necessitates multiple signatures for transaction authorization. This feature serves to enhance account security and mandates the agreement of multiple parties for transaction execution. Multisig accounts are established by defining:

- The threshold number of signatures necessary.
- The public keys implicated in the signing process.

Signing with a multisig account involves individually signing the transaction with the specified keys assigned to the account. Subsequently, these signatures are aggregated into a multisignature, enabling transaction authorization. If the multisignature lacks the required number of signatures, it is deemed invalid.

Generating a Multisig Key

fetchd keys add --multisig=name1,name2,name3[...] --multisig-threshold=N new_key_name N is the minimum number of private keys that must have signed a transaction for it to be valid.

The--multisig flag must containALL the names of the public keys that will make up the multisig account. In addition, all these names must have already been added to the local database

By default all the public keys that are provided are sorted to generate a consistent address. This would mean that the following commands would produce the same account.

fetchd keys add --multisig=k1,k2,k3 --multisig-threshold=2 multisig_address fetchd keys add --multisig=k2,k3,k1 --multisig-threshold=2 multisig_address If this behaviour is not desirable, then you can use the--nosort flag when creating the multisig to disable it.

Adding public keys to the database

You can add public keys to your local fetchd database using the following command:

fetchd keys add msig-p2 --pubkey='{"@type":"/cosmos.crypto.secp256k1.PubKey","key":"AsMGc+Jz7bKpI0Q2MUlGYzE6MT1GEOAKsDFKltB18Rs0"}'

Signing a transaction

Step 1. Generating the transaction

Since you will need multiple accounts to sign the transaction that you want to send from a multisig account, you first need to generate the transaction that you plan on send.

In the example below we send 1FET from the multisig tofetch124rum7e4z8jdg7plxtp2d3f3um57cmr7vd8467 .

fetchd tx bank send \ fetch1mzdu9kagyh38walpq4yh83mf99229nyyr4jxkw \ fetch124rum7e4z8jdg7plxtp2d3f3um57cmr7vd8467 \ 1000000000000000000atestfet \ --chain-id fetchhub-4 \ --generate-only > unsigned-tx.json The output fileunsigned-tx.json contains the unsigned transaction payload as shown below:

{ "body": { "messages": [{ "@type": "/cosmos.bank.v1beta1.MsgSend", "from_address": "fetch1mzdu9kagyh38walpq4yh83mf99229nyyr4jxkw", "to_address": "fetch124rum7e4z8jdg7plxtp2d3f3um57cmr7vd8467", "amount": [{ "denom": "atestfet", "amount": "1000000000000000000" }] }], "memo": "", "timeout_height": "0", "extension_options": [], "non_critical_extension_options": [] }, "auth_info": { "signer_infos": [], "fee": { "amount": [], "gas_limit": "200000", "payer": "", "granter": "" } }, "signatures": [] }

Step 2. Indivual Signing

Now that we have generated the transaction that we want to send, each account needs to sign the transaction individually.

In the example below we first sign with themsig-p1 account:

fetchd tx sign \ unsigned-tx.json \ --multisig=fetch1mzdu9kagyh38walpq4yh83mf99229nyyr4jxkw \ --from=msig-p1 \ --output-document=msig-p1.sig.json \ --chain-id=fetchhub-4 Then we sign with themsig-p2 account:

fetchd tx sign \ unsigned-tx.json \ --multisig=fetch1mzdu9kagyh38walpq4yh83mf99229nyyr4jxkw \ --from=msig-p2 \ --output-document=msig-p2.sig.json \ --chain-id=fetchhub-4

Step 3. Combining the Signatures together

We now have both the unsigned transaction and two signature for the transaction. The final step is to combine all the signatures together and to generate a full transaction that can be broadcast to the network.

fetchd tx multisign \ unsigned-tx.json \ multi \ msig-p1.sig.json msig-p2.sig.json \ --output-document=signed-tx.json \ --chain-id=fetchhub-4 The final transaction is now generated insigned-tx.json .

Step 4. Broadcasting the transaction

The final step, now that we have the full transaction is to be broadcast it to the network. This can be done with the following command:

fetchd tx broadcast --broadcast-mode=block signed-tx.json The flag--broadcast-mode=block is optional, however, in this mode the command will both broadcast the transaction and wait for the transaction to be included into a block.

Was this page helpful?

Reconciliation service Getting started