Multisig Transactions

Multisig transactions require signatures of multiple private keys, typically owned by multiple parties. A multisig transaction is initiated by any key holder, and at least one of them would need to import other parties' public keys into their Keybase and generate a multisig public key to finalize and broadcast multisig transactions.

Show Multisig Address

When a new multisig public keytest_multisig is stored its address will be the signer of multisig transactions:

```

Copy secretcli keys show test_multisig -a

secret1zy90yjysn579l65sj643tphq4pem6ufkl0djd6

```

View Multisig Threshold

You may also view multisig threshold, pubkey constituents and respective weights by viewing the JSON output of the key:

```

Copy secretclikeysshowtest_multisig--pubkey

# You will see the output of the pubkey, look or "threshold"

'{"@type":"/cosmos.crypto.multisig.LegacyAminoPubKey","threshold":2,"public_keys":[{"@type":"/cosmos.crypto.secp256k1.PubKey","key":"AiXwUPtwTJqxKZq/BjKi+7EFhqR2Aj9QT94lFzb5Ednp"},
{"@type":"/cosmos.crypto.secp256k1.PubKey","key":"A7QMHOt+yLGddDxey51QLofwsTJWfqyzYmNOB9L1Oz1S"},
{"@type":"/cosmos.crypto.secp256k1.PubKey","key":"A0QMBqFY4J39i6NrH4qR5uOEnyytpkyeWFg/e0sPd8NJ"}]}' mnemonic:""

```

The-p flag can be used interchangeably with the--pubkey flag to only show the output of pubkey information.

```

Copy secretcli keys show test_multisig -p

```

Initiate Multisig Transaction

General usage of the secretcli for sending transactions uses the follow format:

```

Copy secretclitxbanksend[from_key_or_address] [to_address] [amount] [flags]

```

Note: To be able to correctly test multisig transactions you will need tokens owned by the multisig wallet.Get testnet tokens using the faucet.

The first step to create a multisig transaction is to initiate it on behalf of the multisig address created above using the following command:

```

Copy secretclitxbanksend secret1zy90yjysn579l65sj643tphq4pem6ufkl0djd6#Fromaddress secret1kcy20p0cs2wakeqz00xgs5m0cmj65283xqmvfs#Toadress 1000uscrt#Amountbeingsent --generate-only>unsignedTx.json# Json for account owner signing

```

```

Copy

# unsignedTx.json file contents

{ "body":{ "messages":[ { "@type":"/cosmos.bank.v1beta1.MsgSend", "from_address":"secret1zy90yjysn579l65sj643tphq4pem6ufkl0djd6",
"to_address":"secret1kcy20p0cs2wakeqz00xgs5m0cmj65283xqmvfs", "amount":[ { "denom":"uscrt", "amount":"1000" } ] } ], "memo":"", "timeout_height":"0", "extension_options":[],
"non_critical_extension_options":[] }, "auth_info":{ "signer_infos":[], "fee":{ "amount":[ { "denom":"uscrt", "amount":"50000" } ], "gas_limit":"200000", "payer":"", "granter":"" } }, "signatures":[] # Notice that none of the multisg owners have signed yet

```

Signing Multisig Transactions

The fileunsignedTx.json contains the unsigned transaction encoded in JSON.key1 can now sign the transaction with its own private key:

```

Copy secretclitxsign\ unsignedTx.json\ --multisig=test_multisig\ --from=key1\ --output-document=key1Signature.json\ --chain-id=pulsar-3# This is the testnet chain-id

```

After generating the key1Signature.json file, other wallets making up the multisig need to use the unsignedTx.json to generate their own individual signed json files:

```

Copy secretcli tx sign \ unsignedTx.json \ --multisig=test_multisig \ --from=key2 \ --output-document=key2Signature.json \ --chain-id=pulsar-3

```

Since the 'k' value oftest_multisig is set to 2, a third signature from the final key making up the wallet (key3) is not required to officially sign and broadcast the multisig transaction. Any the key holders of the multisig can now generate the multisig transaction by combining the required signature files:

```

Copy secretclitxmultisign\ unsignedTx.json\ test_multisig\ key1Signature.jsonkey2Signature.json>signedTx.json

```

```

Copy

# signedTx.json contents

{ "body":{ "messages":[ { "@type":"/cosmos.bank.v1beta1.MsgSend", "from_address":"secret1zy90yjysn579l65sj643tphq4pem6ufkl0djd6",
"to_address":"secret1kcy20p0cs2wakeqz00xgs5m0cmj65283xqmvfs", "amount":[ { "denom":"uscrt", "amount":"1000" } ] } ], "memo":"", "timeout_height":"0", "extension_options":[],
"non_critical_extension_options":[] }, "auth_info":{ "signer_infos":[ { "public_key":{ "@type":"/cosmos.crypto.multisig.LegacyAminoPubKey", "threshold":2, "public_keys":[ {
```

"@type":"/cosmos.crypto.secp256k1.PubKey", "key":"A2wRQgUwB6BMLcoa9acVrZ56EydgQMb0hJS1Fq6/nLPs" }, { "@type":"/cosmos.crypto.secp256k1.PubKey", "key":"AxI12MHj4ZEPJd0eWIu25bNoTH6XyhY5MdsUIqqut3Fv" }, { "@type":"/cosmos.crypto.secp256k1.PubKey", "key":"AzkEF0QT7qTvI/W3IO5IPQMUa9sR7Z29b4TT6k0Lo8Qe" }] }, "mode_info":{ "multi":{ "bitarray":{ "extra_bits_stored":3, "elems":"wA==" }, "mode_infos":[{ "single":{ "mode":"SIGN_MODE_LEGACY_AMINO_JSON" } }, { "single":{ "mode":"SIGN_MODE_LEGACY_AMINO_JSON" } }] } }, "sequence":"0" }], "fee":{ "amount":[{ "denom":"uscrt", "amount":"50000" }], "gas_limit":"200000", "payer":"", "granter":"" } }, "signatures":["CkAJd+gSCrcPd8hlX2YT68bCALTi8+Rk0AVUGnMK7QAuFmrr2uACK9q2u4QQij49BXL6LaXEhZkUXFc3aCkMxmJFCkCEeY7AGFowB+GIBiZkEUsJg/RjM/cvqt/JvBcSYwDiS2EAMBMGItZNP/WdRX1p
] }

```

Broadcasting Multisig Transactions

The transaction can now be broadcast to the network using the singed.json file:

```

Copy secretclitxbroadcastsignedTx.jshon

```

After executing the above command an output will be generated containing information about the transaction:

```

Copy { "height":"0", "txhash":"52A1CB5C4049023A14616462A37A419240EBE99E7C941B7E9E17BD32FE92B134", "codespace":"sdk", "code":19, "data":"", "raw_log":"", "logs":[], "info":"", "gas_wanted":"0", "gas_used":"0", "tx":null, "timestamp":"", "events":[] }

```

Broadcasted transactions can be viewed online using a Secret Network block explorer by searching for the transaction has (txhash). For example, the testnet transaction above can be viewed here .

Last updated8 months ago On this page * Show Multisig Address * View Multisig Threshold * Initiate Multisig Transaction * Signing Multisig Transactions * Broadcasting Multisig Transactions

Was this helpful? Edit on GitHub Export as PDF
```