# MultiAccounts feature for blobs submission

## Overview

By default, a celestia-node creates a key namedmy_celes_key during initialization. This document explains how to run a node with a different default key name and how to submit blobs using different signers.

## Running a node with a different default key name

To start a Celestia node with a different default key name, use the following command:

sh celestia

light

start

--core.ip=consensus.celestia-arabica-11.com

\ --p2p.network=arabica

--keyring.keyname

testKey celestia

light

start

--core.ip=consensus.celestia-arabica-11.com

\ --p2p.network=arabica

--keyring.keyname

testKey In this example,testKey becomes the default node key, and the node's address will change accordingly.

## Submitting blobs with a different signer/key name

### Option 1: Submit passing key name

You can submit a blob by specifying a different key name:

sh celestia

blob

submit

0x42690c204d39600fddd3

'gm'

--key.name

testKey2 celestia

blob

submit

0x42690c204d39600fddd3

'gm'

--key.name

testKey2 This transaction will be signed by the address associated withtestKey2 .

### Option 2: Submit passing signer address

Alternatively, you can submit a blob by specifying the signer's address:

sh celestia

blob

submit

0x42690c204d39600fddd3

'gm'

--signer SIGNER_ADDRESS celestia

blob

submit

0x42690c204d39600fddd3

'gm'

--signer SIGNER_ADDRESS Both options achieve the same result but use different inputs. ThetestKey2 points toSIGNER_ADDRESS in the KeyStore.

# Key management

All keys and addresses must be added to the KeyStore. To create a new key, use thecel-key library :

## Creating a new key

sh ./cel-key

add

testKey

--keyring-backend

test

\ --node.type

light

--p2p.network

arabica ./cel-key

add

testKey

--keyring-backend

test

\ --node.type

light

--p2p.network

arabica

## Importing an existing key

sh ./cel-key

import ./cel-key

import Learn more on the Create a wallet with celestia-node page.

## Optional flags for write transactions

All other flags are now optional for all write transactions. This means you don't have to specify gas/fee parameters each time. The configuration can handle it for you automatically.

The default configuration applies to all write transactions, including those in the state module and blob.Submit . This simplifies the process of submitting transactions and reduces the need for manual input.

For reference, see the:

- Transaction configuration
- State module command implementation [][ Edit this page on GitHub] Last updated: Previous page FeeGrant module for blobs submission Next page Transaction resubmission guidelines []