FeeGrant module for blobs submission

Overview

This guide provides developers with the knowledge to use the FeeGrant module on the Celestia's Mocha testnet chain for granting a data availability node's account to submit blobs without constantly funding it, enabling a third-party account to cover the transaction fees.

Pre-requisites

- celestia-node binary (celestia
-)installed
- Access to a Mocha node (e.g.,https://rpc.celestia-mocha.com:443
-)
- Running DA Light node on Mocha testnet
- One account with sufficient funds, the "granter"
- One account with no funds, the "grantee"

Introduction

Each DA node contains a Celestia account that is used to pay for blobs submissions. To unify the fee payment process, the FeeGrant module allows a third-party account (granter) to pay for the fees incurred by a DA node's (grantee) account. You will need one account that will contain the funds, the granter, and another account that will be in the DA node you run to post blobs, the grantee. You will see the DA node's account once you initialize the node. Learn more about managing accounts withcel-key increate a wallet with celestia-node .

Granting fee allowances using celestia-node

To get started granting the fee allowance, you will need two separate keys to run the light node with. One to begin the FeeGrant as the granter and another to use the FeeGrant as the grantee.

Set some variables for your accounts for the remainder of the guide:

bash export GRANTER_ADDRESS =< your-granter-account-addres s

export GRANTEE_ADDRESS =< your-grantee-account-addres s

export RPC_URL = rpc.celestia-mocha.com export GRANTER_ADDRESS =< your-granter-account-addres s

export GRANTEE_ADDRESS =< your-grantee-account-addres s

export RPC_URL = rpc.celestia-mocha.com

FeeGrant module implementation in celestia-node

Using celestia-node, you can now easily give permission for other nodes to submit transactions on your behalf. It is also possible to revoke the grant.

The FeeGrant functionality can now be used during runtime without the need to restart the node.

Grant permission for an allowance as a granter

First, start your node:

bash celestia

light

start

--p2p.network

mocha

--core.ip RPC_URL celestia

light

start

--p2p.network

mocha

--core.ip RPC_URL Then, grant the fee to the grantee:

bash celestia

state

grant-fee GRANTEE_ADDRESS --amount

2000 celestia

state

grant-fee GRANTEE_ADDRESS --amount

2000 Note that the--amount flag specifies the spend limit (in utia) for the grantee. If not specified, the grantee does not have a spend limit.

Using a FeeGrant allowance as a grantee in celestia-node

Start your node:

bash celestia

light

start

--core.ip RPC_URL --p2p.network=mocha celestia

light

start

--core.ip RPC_URL --p2p.network=mocha To check the balance of a light node, use the following command:

bash celestia

state

balance celestia

state

balance Example response when the account balance does not exist:

json { "result" : { "denom" : "utia" , "amount" : "0" } } { "result" : { "denom" : "utia" , "amount" : "0" } } This indicates that the light node currently does not have any funds.

Now submit a blob using the FeeGrant:

bash celestia

blob

submit

0x42690c204d39600fddd3

'gm'

--granter.address GRANTER_ADDRESS celestia

blob

submit

0x42690c204d39600fddd3

'gm'

--granter.address GRANTER_ADDRESS You'll see the height and the commitment of your blob:

json { "result" : { "height" : 1639397 , "commitments" : ["19L/C4iBEsqXGzC5ZxJ3vtuGBiAdQAMIEnbYjKEGcac="] } } { "result" : { "height" : 1639397 , "commitments" : ["19L/C4iBEsqXGzC5ZxJ3vtuGBiAdQAMIEnbYjKEGcac="] } }

Checking account balances after submission

Light node account: After submitting a blob, you can check the light node account's balance to verify that the fees have been deducted:

bash celestia

state

balance celestia

state

balance Example output showing fees are not deducted:

json { "result" : { "denom" : "utia" , "amount" : "0" } } { "result" : { "denom" : "utia" , "amount" : "0" } }

Optional: Revoke permission for a FeeGrant allowance as a granter

To revoke the feegrant, run:

bash celestia

state

revoke-grant-fee GRANTEE_ADDRESS celestia

state

revoke-grant-fee GRANTEE_ADDRESS

Optional: Submitting a blob from file input

To submit a blob from file input:

bash celestia

blob

submit

--input-file

blob.json celestia

blob

submit

--input-file

blob.json

Optional: Granting fee allowances using celestia-appd

To grant fee allowances, allowing a third-party (granter) account to pay for the fees incurred by a Celestia data availability node (grantee) account, use the following commands.

Set your account addresses for grantee and granter, and the RPC URL:

bash export GRANTER_ADDRESS =< your-granter-account-addres s

export GRANTEE_ADDRESS =< your-grantee-account-addres s

export RPC_URL = https://rpc.celestia-mocha.com:443 export GRANTER_ADDRESS =< your-granter-account-addres s

export GRANTEE_ADDRESS =< your-grantee-account-addres s

export RPC_URL = https://rpc.celestia-mocha.com:443 Then, send the feegrant transaction:

bash celestia-appd

tx

feegrant

grant

\ GRANTER_ADDRESS GRANTEE_ADDRESS \ --node RPC_URL \ --spend-limit

1000000 utia

\ --allowed-messages

"/cosmos.bank.v1beta1.MsgSend,/celestia.blob.v1.MsgPayForBlobs"

\ --chain-id

mocha-4

\ --keyring-backend

test

\ --fees

20000 utia

\ --broadcast-mode

block

\ --yes celestia-appd

tx

feegrant

grant

\ GRANTER_ADDRESS GRANTEE_ADDRESS \ --node RPC_URL \ --spend-limit

1000000 utia

\ --allowed-messages

"/cosmos.bank.v1beta1.MsgSend,/celestia.blob.v1.MsgPayForBlobs"

\ --chain-id

mocha-4

\ --keyring-backend

test

\ --fees

20000 utia

\ --broadcast-mode

block

\ --yes Example:FeeGrant transaction on Mocha

Optional: Checking the granter's account

To confirm that the fees have been deducted from the granter's account that granted the fee allowance, run:

bash celestia-appd

query

bank

balances GRANTER_ADDRESS \ --node https://rpc.celestia-mocha.com:443

--denom

utia celestia-appd

query

bank

balances GRANTER_ADDRESS \ --node https://rpc.celestia-mocha.com:443

--denom

utia This output will show the remaining balance after fees have been deducted, confirming that the FeeGrant module is working as intended. [][Edit this page on GitHub] Last updated: Previous page Submitting data blobs to Celestia Next page MultiAccounts feature for blobs submission []