Helpful CLI commands

View all options:

console celestia-appd --help Start celestia-app

Usage: celestia-appd [command]

Available Commands: add-genesis-account Add a genesis account to genesis.json collect-gentxs Collect genesis txs and output a genesis.json file config Create or query an application CLI configuration file debug Tool for helping with debugging your application export Export state to JSON gentx Generate a genesis tx carrying a self delegation help Help about any command init Initialize private validator, p2p, genesis, and application configuration files keys Manage your application's keys migrate Migrate genesis to a specified target version query Querying subcommands rollback rollback tendermint state by one height rollback rollback cosmos-sdk and tendermint state by one height start Run the full node status Query remote node for status tendermint Tendermint subcommands tx Transactions subcommands validate-genesis validates the genesis file at the default location or at the location passed as an arg version Print the application binary version information celestia-appd --help Start celestia-app

Usage: celestia-appd [command]

Available Commands: add-genesis-account Add a genesis account to genesis.json collect-gentxs Collect genesis txs and output a genesis.json file config Create or query an application CLI configuration file debug Tool for helping with debugging your application export Export state to JSON gentx Generate a genesis tx carrying a self delegation help Help about any command init Initialize private validator, p2p, genesis, and application configuration files keys Manage your application's keys migrate Migrate genesis to a specified target version query Querying subcommands rollback rollback tendermint state by one height rollback rollback cosmos-sdk and tendermint state by one height start Run the full node status Query remote node for status tendermint Tendermint subcommands tx Transactions subcommands validate-genesis validates the genesis file at the default location or at the location passed as an arg version Print the application binary version information

Creating a wallet

sh celestia-appd

config

keyring-backend

test celestia-appd

config

keyring-backend

test keyring-backend configures the keyring's backend, where the keys are stored.

Options are:os|file|kwallet|pass|test|memory .

You can learn more on theCosmos documentation orGo Package documentation .

Key management

sh

listing keys

celestia-appd

keys

list

adding keys

celestia-appd

keys

add

< KEY_NAM E

deleting keys

celestia-appd

keys

delete

< KEY_NAM E

renaming keys

celestia-appd

keys

rename

< CURRENT_KEY_NAM E

< NEW_KEY_NAM E

listing keys

celestia-appd

keys

list

adding keys

celestia-appd

keys

add

< KEY_NAM E

deleting keys

celestia-appd

keys

delete

< KEY_NAM E

renaming keys

celestia-appd

keys

rename

< CURRENT_KEY_NAM E

< NEW_KEY_NAM E

Importing and exporting keys

Import an encrypted and ASCII-armored private key into the local keybase.

sh celestia-appd

keys

import

< KEY_NAM E

< KEY_FIL E

celestia-appd

keys

import

< KEY_NAM E

< KEY_FIL E

Example usage:

sh celestia-appd

keys

import

amanda

./keyfile.txt celestia-appd

keys

import

amanda

./keyfile.txt Export a private key from the local keyring in encrypted and ASCII-armored format:

sh celestia-appd

keys

export

< KEY_NAM E

you will then be prompted to set a password for the encrypted private key:

Enter

passphrase

to

encrypt

the

exported

key: celestia-appd

keys

export

< KEY_NAM E

you will then be prompted to set a password for the encrypted private key:

Enter

passphrase

to

encrypt

the

exported

key: After you set a password, your encrypted key will be displayed.

Querying subcommands

Usage:

sh celestia-appd

query

< FLAG S

|

< COMMAND>

alias q

celestia-appd

q

< FLAG S

|

< COMMAND> celestia-appd

query

< FLAG S

|

< COMMAND>

alias q

celestia-appd

q

< FLAG S

|

< COMMAND> To see all options:

sh celestia-appd

q

--help celestia-appd

q

--help

Token management

Get token balances:

sh celestia-appd

q

bank

balances

< ADDRES S

--node

< NODE_UR I

celestia-appd

q

bank

balances

< ADDRES S

--node

< NODE_UR I

Example usage:

sh celestia-appd

q

bank

balances

celestia1czpgn3hdh9sodm06d5qk23xzgpq2uyc8ggdqgw

\ --node

https://rpc-mocha.pops.one celestia-appd

q

bank

balances

celestia1czpgn3hdh9sodm06d5qk23xzgpq2uyc8ggdqgw

\ --node

https://rpc-mocha.pops.one Transfer tokens from one wallet to another:

sh celestia-appd

tx

bank

send

< FROM_ADDRES S

< TO_ADDRES S

\ < amoun t

--node

< NODE_UR I

--chain-id

< CHAIN_I D

celestia-appd

tx

bank

send

< FROM_ADDRES S

< TO_ADDRES S

\ < amoun t

--node

< NODE_UR I

--chain-id

< CHAIN_I D

Example usage:

sh celestia-appd

tx

bank

send

< FROM_ADDRES S

< TO_ADDRES S

\ 19000000 utia

--node

https://rpc-mocha.pops.one/

--chain-id

mocha celestia-appd

tx

bank

send

< FROM_ADDRES S

< TO_ADDRES S

\ 19000000 utia

--node

https://rpc-mocha.pops.one/

--chain-id

mocha To see options:

sh celestia-appd

tx

bank

send

--help celestia-appd

tx

bank

send

--help

Governance

Governance proposals on Celestia are limited as there are no text proposals, upgrades occur via social consensus, and some params are not modifiable. However, one can submit governance proposals to change certain parameters and spend community funds. More detailed information on this topic can be found in thecosmos-sdk documentation for submitting proposals , the list ofparameter defaults in the specs , and thex/paramfilter module specs .

Viewing the available proposals can be done with the query command:

sh celestia-appd

q

gov

proposals celestia-appd

q

gov

proposals There are four options when voting "yes", "no", "no_with_veto" and "abstain". The "no_with_veto" vote is different from the "no" vote in that the submitter of the proposer's deposit will get burned, and a minority of stake (1/3) can stop a proposal that might otherwise pass quorum. You can use those options to vote on a governance proposal with the following command:

sh celestia-appd

tx

gov

vote

< proposal

i d

< optio n

\ --from

< walle t

--chain-id

< chain-i d

celestia-appd

tx

gov

vote

< proposal

i d

< optio n

\ --from

< walle t

--chain-id

< chain-i d

To submit a proposal, there are two commands that can be used. The first is the legacy command, which is the recommended way to submit a proposal.

To change the max validators to 105, one would first save this JSON file:

json { "title" : "Staking Param Change" , "description" : "Update max validators" , "changes" : [{ "subspace" : "staking" , "key" : "MaxValidators" , "value" : 105 }], "deposit" : "1000000000utia" } { "title" : "Staking Param Change" , "description" : "Update max validators" , "changes" : [{ "subspace" : "staking" , "key" : "MaxValidators" , "value" : 105 }], "deposit" : "1000000000utia" } Then you can submit the proposal with:

sh celestia-appd

tx

gov

submit-legacy-proposal

\ parameter-change

< path

to

json

fil e

\ --from

< walle t

--chain-id

< chain-i d

celestia-appd

tx

gov

submit-legacy-proposal

\ parameter-change

< path

to

json

fil e

\ --from

< walle t

--chain-id

< chain-i d

If we want to use the newer api, we can submit a proposal by first saving thesdk.Msg proposal in the json encoded format to a json.

json { "messages" : [{ "@type" : "/cosmos.gov.v1beta1.MsgSubmitProposal" , "content" : { "@type" : "/cosmos.params.v1beta1.ParameterChangeProposal" , "title" : "title" , "description" : "description" , "changes" : [{ "subspace" : "staking" , "key" : "MaxValidators" , "value" : "103" }] }, "initial_deposit" : [{ "denom" : "utia" , "amount" : "1000000000" }], "proposer" : "celestia10d07y265gmmuvt4z0w9aw880jnsr700jtgz4v7" }] } { "messages" : [{ "@type" : "/cosmos.gov.v1beta1.MsgSubmitProposal" , "content" : { "@type" : "/cosmos.params.v1beta1.ParameterChangeProposal" , "title" : "title" , "description" : "description" , "changes" : [{ "subspace" : "staking" , "key" : "MaxValidators" , "value" : "103" }] }, "initial_deposit" : [{ "denom" : "utia" , "amount" : "1000000000" }], "proposer" : "celestia10d07y265gmmuvt4z0w9aw880jnsr700jtgz4v7" }] } Note that the proposer here must be the gov module account. That account can be found by using this command:

sh celestia-appd

q

auth

module-account

gov celestia-appd

q

auth

module-account

gov Then one can submit the proposal with:

sh celestia-appd

tx

gov

submit-proposal

< path

to

json

fil e

\ --from

< walle t

--chain-id

< chain-i d

celestia-appd

tx

gov

submit-proposal

< path

to

json

fil e

\ --from

< walle t

--chain-id

< chain-i d

Community Pool

A percentage of the block rewards are allocated to the community pool. Community members can submit governance proposals to spend the community pool funds, and token holders can vote on these proposals. The proposals to spend are arbitrary in nature in that they can only contain text and some address to send funds to. To view the community pool balance, use the following command:

sh celestia-appd

q

distribution

community-pool celestia-appd

q

distribution

community-pool To submit a proposal to spend the community pool funds, first create a JSON file that contains the proposal.

json { "title" : "Community Pool Spend" , "description" : "Fund an open source project." , "recipient" : "celestia17adsjkuecgjheugrdrwdqv9uh3qkrfmj9xzawx" , "amount" : "100000000000utia" , "deposit" : "1000000000utia" } { "title" : "Community Pool Spend" , "description" : "Fund an open source project." , "recipient" : "celestia17adsjkuecgjheugrdrwdqv9uh3qkrfmj9xzawx" , "amount" : "100000000000utia" , "deposit" : "1000000000utia" } The json file can be submitted using a similar proposal submission command as above:

sh celestia-appd

tx

gov

submit-legacy-proposal

\ community-pool-spend

< path

to

json

fil e

\ --from

< walle t

celestia-appd

tx

gov

submit-legacy-proposal

\ community-pool-spend

< path

to

json

fil e

\ --from

< walle t

Claim validator rewards

You can claim your validator rewards with the following command:

sh celestia-appd

tx

distribution

withdraw-rewards

< validator

valope r

\ --commission

--from= < validator

walle t

--chain-id

< chain-i d

\ --gas

auto

-y celestia-appd

tx

distribution

withdraw-rewards

< validator

valope r

\ --commission

--from= < validator

walle t

--chain-id

< chain-i d

\ --gas

auto

-y

Delegate & undelegate tokens

You candelegate your tokens to a validator with the following command:

sh celestia-appd

tx

staking

delegate

< validator

valope r

< amoun t

\ --from

< walle t

--chain-id

< chain-i d

celestia-appd

tx

staking

delegate

< validator

valope r

< amoun t

\ --from

< walle t

--chain-id

< chain-i d

You can undelegate tokens to a validator with theunbond command:

sh celestia-appd

tx

staking

unbond

< validator

valope r

< amoun t

\ --from

< walle t

--chain-id

< chain-i d

```
    celestia-appd
tx
staking
unbond
< validator
valope r
< amoun t
    \ --from
< walle t
--chain-id
< chain-i d
```

Unjailing the validator

You can unjail your validator with the following command:

```
sh celestia-appd
tx
slashing
unjail
--from
< validator
walle t
    \ --chain-id
< chain-i d
--gas
auto
-y celestia-appd
tx
slashing
unjail
--from
< validator
walle t
    \ --chain-id
< chain-i d
--gas
auto
-y
```

How to export logs with SystemD

You can export your logs if you are running a SystemD service with the following command:

sh sudo

journalctl

-u

< your

systemd

servic e

-S

yesterday

node_logs.txt sudo

journalctl

-u

< your

systemd

servic e

-S

today

node_logs.txt

This command outputs the last 1 million lines!

sudo

journalctl

-u

< your

systemd

servic e

-n

1000000

node_logs.txt sudo

journalctl

-u

< your

systemd

servic e

-S

yesterday

node_logs.txt sudo

journalctl

-u

< your

systemd

servic e

-S

today

node_logs.txt

This command outputs the last 1 million lines!

sudo

journalctl

-u

< your

systemd

servic e

-n

1000000

node_logs.txt

Signing genesis for a new network

You can first run the following commands:

sh VALIDATOR_NAME = validator1 CHAIN_ID = testnet celestia-appd

init VALIDATOR_NAME --chain-id CHAIN_ID MONIKER = validator_name VALIDATOR_NAME = validator1 CHAIN_ID = testnet celestia-appd

init VALIDATOR_NAME --chain-id CHAIN_ID MONIKER = validator_name Next create a wallet:

sh KEY_NAME = validator celestia-appd

keys

add KEY_NAME KEY_NAME = validator celestia-appd

keys

add KEY_NAME Create or assign an EVM address:

sh EVM_ADDRESS =< EVM_ADDRES S

EVM_ADDRESS =< EVM_ADDRES S

Then add genesis account:

sh CELES_AMOUNT = "5000100000000utia" celestia-appd

add-genesis-account KEY_NAME CELES_AMOUNT CELES_AMOUNT = "5000100000000utia" celestia-appd

add-genesis-account KEY_NAME CELES_AMOUNT Then generate your gentx:

sh STAKING_AMOUNT = 5000000000000 utia celestia-appd

gentx KEY_NAME STAKING_AMOUNT --chain-id CHAIN_ID \ --pubkey= (celestia-appd tendermint show-validator)

\ --moniker= MONIKER

\ --commission-rate=0.1

\ --commission-max-rate=0.2

\ --commission-max-change-rate=0.01

\ --min-self-delegation=1

\ --evm-address= EVM_ADDRESS

\ STAKING_AMOUNT = 5000000000000 utia celestia-appd

gentx KEY_NAME STAKING_AMOUNT --chain-id CHAIN_ID \ --pubkey= (celestia-appd tendermint show-validator)

\ --moniker= MONIKER

\ --commission-rate=0.1

\ --commission-max-rate=0.2

\ --commission-max-change-rate=0.01

\ --min-self-delegation=1

\ --evm-address= EVM_ADDRESS

\ You can then share your gentx JSON file on the networks repo in the respective network directory you are participating in.
[][Edit this page on GitHub] Last updated: Previous page Create a Celestia testnet Next page SystemD []