Fee Grants

Fee grants allow an account to pay for the transaction fees of another account, which is especially useful for onboarding new users who might not have enough tokens to cover transaction fees. This feature is supported in the Cosmos SDK.

- How They Work
- : Fee grants can be set up so that one account (the granter) can pay for the transaction fees of another account (the grantee). The granter specifies conditions under which the fees will be paid.
- Documentation
- : Detailed documentation on fee grants can be found in the Cosmos SDK Fee Grants Documentation(opens in a new tab)
- .

Granting

seid

seid tx feegrant grant [granter_key_or_address] [grantee]

cosmjs

const { SigningStargateClient ,

GasPrice } =

require ("@cosmjs/stargate"); const { DirectSecp256k1HdWallet } =

require ("@cosmjs/proto-signing");

async

function

grantFeeGrant (rpcEndpoint , granterMnemonic , granteeAddress) { const

granterWallet

=

await

DirectSecp256k1HdWallet .fromMnemonic (granterMnemonic , { prefix :

"sei" }); const [granterAccount] =

await

granterWallet .getAccounts ();

const

client

=

await

SigningStargateClient .connectWithSigner (rpcEndpoint , granterWallet , { gasPrice :

GasPrice .fromString ("0.025usei") , });

const

msg

= { typeUrl :

"/cosmos.feegrant.v1beta1.MsgGrantAllowance" , value : { granter :

granterAccount .address , grantee : granteeAddress , allowance : { typeUrl :

"/cosmos.feegrant.v1beta1.BasicAllowance" , value : { spendLimit : [{ denom :

"usei" , amount :

"1000000" }] , expiration :

null , } , } , } , };

const

fee

= { amount : [{ denom :

"usei" , amount :

"5000" }] , gas :

"200000" , };

const

result

=

await

client .signAndBroadcast (granterAccount .address , [msg] , fee); console .log (result); }

const

rpcEndpoint

=

"https://rpc-endpoint" ; const

granterMnemonic

=

"your-granter-mnemonic" ; const

granteeAddress

=

"sei1granteeaddress" ;

grantFeeGrant (rpcEndpoint , granterMnemonic , granteeAddress);

Revoking

seid

seid tx feegrant revoke [granter] [grantee] [flags]

cosmjs

const { SigningStargateClient ,

GasPrice } =

require ("@cosmjs/stargate"); const { DirectSecp256k1HdWallet } =

require ("@cosmjs/proto-signing");

async

function

revokeFeeGrant (rpcEndpoint , granterMnemonic , granteeAddress) { const

granterWallet

=

await

DirectSecp256k1HdWallet .fromMnemonic (granterMnemonic , { prefix :

"sei" }); const [granterAccount] =

await

granterWallet .getAccounts ();

const

client

=

await

SigningStargateClient .connectWithSigner (rpcEndpoint , granterWallet , { gasPrice :

GasPrice .fromString ("0.025usei") , });

const

msg

= { typeUrl :

"/cosmos.feegrant.v1beta1.MsgRevokeAllowance" , value : { granter :

granterAccount .address , grantee : granteeAddress , } , };

const

fee

= { amount : [{ denom :

"usei" , amount :

"5000" }] , gas :

"200000" , };

const

result

=

await

client .signAndBroadcast (granterAccount .address , [msg] , fee); console .log (result); }

const

rpcEndpoint

=

"https://rpc-endpoint" ; const

granterMnemonic

=

"your-granter-mnemonic" ; const

granteeAddress

=

"sei1granteeaddress" ;

revokeFeeGrant (rpcEndpoint , granterMnemonic , granteeAddress); Last updated onMay 23, 2024Validator FAQ Account Structure