Grant allowances

Building a smart contract dapp that enables users to request a fee grant is a challenging task since all transactions necessitate the payment of transaction fees. However, there are several methods that can be utilized. Here are a few examples:

- The granter can manually execute each fee grant allowance transaction using thesecretcli
- Construct a deployment script containing addresses that you wish to assign a fee grant to. This script will utilize thesecretcli
- to perform the fee grant transaction for each specified address.
- Implement a simple frontend application that verifies and validates a user's account. After confirming that they are the account owner, the application would execute a Javascript transaction withsecret.js
- to carry out the fee grant transaction.
-

Create allowance using secretcli

Thesecretcli is a key tool for accessing the fundamental functionalities of theArchway Blockchain. To installsecretcli , refer toInstall . Here is an illustration of a typical transaction for creating a grant allowance:

```

Copy secretclitxfeegrantgrant"granter_address""grantee_address"\ --chain-id"secret-4"\ --spend-limit1000000uscrt\ --expiration2025-12-31T23:00:00Z\ --allowed-messages'/secret.compute.v1beta1.MsgExecuteContract'

```

Let's break down a few of the components:

- granter_address
- : This value represents the address of the account providing tokens to thegrantee
- for transaction execution.
- grantee_address
- : This denotes the account receiving tokens, enabling it to perform transactions using these grants.
- allowed-messages
- : Through theAllowedMsgAllowance
- type, you can limit the message type a grantee can use the grant for. If not specified, all messages are allowed.
- expiration
- : The deadline by which the allowance must be used or it will expire.
- spend-limit
- : The maximum allowance provided to the grantee. This amount is adjusted as tokens are utilized.
-

Create allowance using secretjs

This section demonstrates how to create a grant allowance usingsecretjs . By following the steps outlined in this section, you'll be able to structure a grant allowance message, and execute the necessary transaction which will grant allowances to designated accounts.

1. The allowance message comprises three essential components: thegranter
2. ,grantee
3. , and the actualallowance
4. . As previously mentioned, thegranter
5. is the address responsible for granting the allowance, while thegrantee
6. is the recipient who can utilize the granted allowance. Theallowance
7. component is slightly more intricate, with its structure dependent on the specific type of allowance employed.
8.

To illustrate, let's examine the structure of agrantMsg using the following example:

```

Copy constaddress="secret1..."// the address you like to send the Fee Grant to constfaucetAddress="secret1..."//address of your own faucet

constgrantMsg=newMsgGrantAllowance({ grantee:address, granter:faucetAddress, allowance:[ spend_limit: [ { amount:"1000000", denom:"uscrt", }, ], }, })

```

1. Now, all that remains is to execute the transaction:
2.

```

Copy constmemo="Your custom memo here" constgasLimit=18000//recommended amount, if you see gas limit errors, increase this. constgasPriceInFeeDenom=0.5//means: 0.5 uscrt/gas unit

consttx=awaitsecretjs.tx.broadcast( msgs, { memo:memo, broadcastCheckIntervalMs:100, feeDenom:"uscrt", gasPriceInFeeDenom:gasPriceInFeeDenom, gasLimit:gasLimit, broadcastMode:BroadcastMode.Block, }, )

```

You can find a working example of this in the Fee Grant Faucet here .

Last updated2 months ago On this page *Create allowance using secretcli * Create allowance using secretjs

Was this helpful? Edit on GitHub Export as PDF