Messages

InstantiateMsg

pub

struct

InstantiateMsg

{ pub credits_address :

String , pub reserve_address :

String , /// MerkleRoot is hex-encoded merkle root. pub merkle_root :

String , /// A point in time from which it is possible to claim airdrops pub airdrop_start :

u64 , /// A point in time from which a vesting is configured for cNTRNs. At this point, it is still /// possible for users to claim their airdrops. pub vesting_start :

u64 , /// Total duration of vesting. At vesting_start.seconds() + vesting_duration_seconds /// point of time it is no longer possible to claim airdrops. At the very same point of time, /// it is possible to withdraw all remaining cNTRNs, exchange them for NTRNs and send to /// reserve, using [ExecuteMsg::WithdrawAll] message pub vesting_duration_seconds :

u64 , pub total_amount :

Option < Uint128

, /// hrp is the bech32 parameter required for building external network address /// from signature message during claim action. example "cosmos", "terra", "juno" pub hrp :

Option < String

, }

ExecuteMsg

pub

enum

ExecuteMsg

{ /// Claim does not check if contract has enough funds, owner must ensure it. Claim

{ amount :

Uint128 , /// Proof is hex-encoded merkle proof. proof :

Vec < String

, } , /// Permissionless, activated after vesting is over (consult to [InstantiateMsg] /// documentation for more info). Withdraws all remaining cNTRN tokens, burns them, /// receiving NTRN in exchange, and sends all received NTRN's to reserve. WithdrawAll

{ } , Pause

{ } , Resume

{ } , }

Examples

Instantiate

{ // Address of the Credits contract "credits_address" :

"neutron..." , // Address of the Reserve contract "reserve_address" :

"neutron..." , /// MerkleRoot is hex-encoded merkle root. "merkle_root" :

"deadbeef" , /// A point in time from which it is possible to claim airdrops "airdrop_start" :

100 , /// A point in time from which a vesting is configured for cNTRNs. At this point, it is still /// possible for users to claim their airdrops. "vesting_start" :

100 , /// Total duration of vesting. At`vesting_start.seconds() + vesting_duration_seconds` /// point of time it is no longer possible to claim airdrops. At the very same point of time, /// it is possible to withdraw all remaining cNTRNs, exchange them for NTRNs and send to /// reserve, using `[ExecuteMsg::WithdrawAll]` message "vesting_duration_seconds" :

100 , // Total amount of tokens to be airdropped "total_amount" :

"10000" , /// hrp is the bech32 parameter required for building external network address /// from signature message during claim action. example "cosmos", "terra", "juno" "hrp" :

"neutron" }

Execute

claim

{ "claim" :

{ // Amount to claim "amount" :

"1000" , /// Proof is hex-encoded merkle proof. "proof" :

["dead" ,

"beef"] } } Claims airdropped tokens.

withdraw_all

{ "withdraw_all" :

{ } } Permissionless. Withdraws all remaining cNTRN tokens, burns them, receiving NTRN in exchange, and sends all received NTRN's to reserve.

pause

{ "pause" :

{ } } Sets the Airdrop contract on pause. Only the owner can call this method.

resume

{ "unpause" :

{ } } Unpauses the Airdrop contract. Only the owner can call this method.Previous Overview Next Queries