# Claim

Suggest Edits

# The Claim Module

Users are required to participate in core network activities to claim their airdrop. An Airdrop recipient is given 20% of the airdrop amount which is not in vesting, and then they have to perform the following activities to get the rest:

- 20% vesting over 3 months by staking
- 60% vesting over 3 months by liquid staking

These claimable assets 'expire' if not claimed. Users have three months ( AirdropDuration ) to claim their full airdrop amount. After three months from launch, all unclaimed tokens get sent to the community pool. At initialization, module stores all airdrop users with amounts from genesis inside KVStore. Airdrop users are eligible to claim their vesting or free amount only once in the initial period of 3 months. After the initial period, users can claim tokens monthly.

## Actions

There are 2 types of actions, each of which release another 50% of the airdrop allocation. The 2 actions are as follows:

golang ActionLiquidStake Action = 0 ActionDelegateStake Action = 1 These actions are monitored by registering claimhooks to the stakeibc, and staking modules. This means that when you perform an action, the claims module will immediately unlock those coins if they are applicable. These actions can be performed in any order.

The code is structured by separating out a segment of the tokens as "claimable", indexed by each action type. So if Alice delegates tokens, the claims module will move the 50% of the claimables associated with staking to her liquid balance. If she delegates again, there will not be additional tokens given, as the relevant action has already been performed. Every action must be performed to claim the full amount.

## ClaimRecords

A claim record is a struct that contains data about the claims process of each airdrop recipient.

It contains an address, the initial claimable airdrop amount, and an array of bools representing whether each action has been completed. The position in the array refers to enum number of the action.

So for example, [true, true] means that ActionLiquidStake and ActionDelegateStake are completed.

golang type ClaimRecord struct { // address of claim user Address string // weight that represents the portion from total allocation Weight sdk.Dec // true if action is completed // index of bool in array refers to action enum # ActionCompleted []bool }

## A Note on Address Mappings

When an airdrop is created, we call LoadAllocationData to load the airdrop data from the allocations file. This will apply utils.ConvertAddressToStrideAddress on each of those addresses, and then store those with the ClaimRecords . For an airdrop to, say, the Cosmos Hub, this will be the proper Stride address associated with that account.

claim state will only ever store this Stride address.

However, for zones with a different coin type,this will be an incorrect Stride address . This should not cause any issues though, as this Stride address will be unusable.

In order to claim that airdrop, the user will have to verify that they own the corresponding Evmos address. When the user tries to verify, we call utils.ConvertAddressToStrideAddress on that address, and verify it gives the same "incorrect" Stride address from earlier. Through this, we can confirm that the user owns the Evmos address. We then replace the Stride address with a "correct" one that the user verifies they own.

## Params

The airdrop logic has 4 parameters:

golang type Params struct { // Time that marks the beginning of the airdrop disbursal, // should be set to chain launch time. AirdropStartTime time.Time AirdropDuration time.Duration // denom of claimable asset ClaimDenom string // address of distributor account DistributorAddress string }

# Keeper functions

Claim keeper module provides utility functions to manage epochs.

Go GetModuleAccountAddress (ctx sdk . Context )sdk . AccAddress GetDistributorAccountBalance (ctx sdk . Context )sdk . Coin EndAirdrop (ctx sdk . Context )error GetClaimRecord (ctx sdk . Context ,addr sdk . AccAddress ) (types . ClaimRecord ,error )GetClaimRecords (ctx sdk . Context ) []types . ClaimRecord SetClaimRecord (ctx sdk . Context ,claimRecord types . ClaimRecord )error SetClaimRecords (ctx sdk . Context ,claimRecords []types . ClaimRecord )error GetClaimableAmountForAction (ctx sdk . Context ,addr sdk . AccAddress ,action types . Action ,includeClaimed bool ) (sdk . Coins ,error )GetUserTotalClaimable (ctx sdk . Context ,addr sdk . AccAddress ) (sdk . Coins ,error )ClaimCoinsForAction (ctx sdk . Context ,addr sdk . AccAddress ,action types . Action ) (sdk . Coins ,error )clearInitialClaimables (ctx sdk . Context )fundRemainingsToCommunity (ctx sdk . Context )error

# React Hooks

The claim module reacts on the following hooks, executed in external modules.

20% of airdrop is sent to a vesting account when staking.AfterDelegationModified hook is triggered. 20% of airdrop is sent to a vesting account when stakeibc.AfterLiquidStake hook is triggered.

Once the airdrop is claimed for a specific hook type, it can't be claimed again.

### Claim Records

protobuf // A Claim Records is the metadata of claim data per address message ClaimRecord { // address of claim user string address = 1 [ (gogoproto.moretags) = "yaml:\"address\"" ];

// weight that represents the portion from total allocations double weight = 2;

// true if action is completed // index of bool in array refers to action enum # repeated bool action_completed = 3 [ (gogoproto.moretags) = "yaml:\"action_completed\"" ]; } When a user get airdrop for his/her action, claim record is created to prevent duplicated actions on future actions.

### State

protobuf message GenesisState { // params defines all the parameters of the module. Params params = 2 [ (gogoproto.moretags) = "yaml:\"params\"", (gogoproto.nullable) = false ];

// list of claim records, one for every airdrop recipient repeated ClaimRecord claim_records = 3 [ (gogoproto.moretags) = "yaml:\"claim_records\"", (gogoproto.nullable) = false ]; } Claim module's state consists of params , and claim_records .

Claim module provides below params

protobuf // Params defines the claim module's parameters. message Params { google.protobuf.Timestamp airdrop_start_time = 1 [ (gogoproto.stdtime) = true, (gogoproto.nullable) = false, (gogoproto.moretags) = "yaml:\"airdrop_start_time\"" ]; google.protobuf.Timestamp airdrop_duration = 2 [ (gogoproto.nullable) = false, (gogoproto.stdduration) = true, (gogoproto.jsontag) = "airdrop_duration,omitempty", (gogoproto.moretags) = "yaml:\"airdrop_duration\"" ]; // denom of claimable asset string claim_denom = 3; // airdrop distribution account string distributor_address = 4; } 1. airdrop_start_time 2. refers to the time when user can start to claim airdrop. 3. airdrop_duration 4. refers to the duration from start time to end time. 5. claim_denom 6. refers to the denomination of claiming tokens. As a default, it's 7. ustrd 8. . 9. distributor_address 10. refers to the address of distribution account.

# Queries

### GRPC queries

Claim module provides below GRPC queries to query claim status

protobuf service Query { rpc DistributorAccountBalance(QueryDistributorAccountBalanceRequest) returns (QueryDistributorAccountBalanceResponse) {} rpc Params(QueryParamsRequest) returns (QueryParamsResponse) {} rpc ClaimRecord(QueryClaimRecordRequest) returns (QueryClaimRecordResponse) {} rpc ClaimableForAction(QueryClaimableForActionRequest) returns (QueryClaimableForActionResponse) {} rpc TotalClaimable(QueryTotalClaimableRequest) returns (QueryTotalClaimableResponse) {} rpc ClaimStatus(QueryClaimStatusRequest) returns (QueryClaimStatusResponse) {} rpc ClaimMetadata(QueryClaimMetadataRequest) returns (QueryClaimMetadataResponse) {} }

### CLI commands

For the following commands, you can change (strided keys show -a {your key name}) with the address directly.

Query the claim record for a given address

Shell strided query claim claim-record(strided keys show -a {your key name}) Query the claimable amount that would be earned if a specific action is completed right now.

Shell strided query claim claimable-for-action(strided keys show -a {your key name}) ActionAddLiquidity Query the total claimable amount that would be earned if all remaining actions were completed right now.

Shell strided query claim total-claimable(strided keys show -a {your key name}) ActionAddLiquidity Query claim status, across all claims, for an address. Returns a list of ClaimStatus structs.

message ClaimStatus { string airdrop_identifier = 1; bool claimed = 2; } Shell strided query claim claim-status(strided keys show -a {your key name}) Query claim metadata, across all claims. Returns a ClaimMetadata struct, which contains data about the status of each claim.

message ClaimMetadata { string airdrop_identifier = 1; string current_round = 2; google.protobuf.Timestamp current_round_start = 3; google.protobuf.Timestamp current_round_end = 4; } Shell strided query claim claim-metadata

## Events

claim module emits the following events at the time of hooks:

Type Attribute Key Attribute Value claim sender {receiver} claim amount {claim_amount} Updated3 months ago

Records Icacallbacks Did this page help you?Yes No *Table of Contents * *The Claim Module * * *Actions * * * ClaimRecords * * *A Note on Address Mappings * * *Params * * *Keeper functions * * *React Hooks * * *Queries * * * Events