Mint

Suggest Edits

The Mint Module

The x/mint module mints tokens at the end of epochs.

The x/distribution module is responsible for allocating tokens to stakers, community pool, etc.

The mint module uses time basis epochs from the x/epochs module.

The x/mint module is designed by Osmosis. It is used to handle the regular printing of new tokens within a chain. Its core function is to:

- Mint new tokens once per epoch (default one week)
- Have a "Reductioning factor" every period, which reduces the amount of rewards per epoch.
- (default: period is 3 years, where a year is 52 epochs. The next period's rewards are 2/3 of the prior period's rewards)

Params

Minting params are held in the global params store.

Go type Params struct {MintDenom string // type of coin to mint GenesisEpochProvisions sdk . Dec // initial epoch provisions at genesis EpochIdentifier string // identifier of epoch ReductionPeriodInEpochs int64 // number of epochs between reward reductions ReductionFactor sdk . Dec // reduction multiplier to execute on each period DistributionProportions DistributionProportions // distribution_proportions defines the proportion of the minted denom WeightedDeveloperRewardsReceivers []WeightedAddress // address to receive developer rewards MintingRewardsDistributionStartEpoch int64 // start epoch to distribute minting rewards } The minting module contains the following parameters:

Key Type Example mint_denom string "uosmo" genesis_epoch_provisions string (dec) "500000000" epoch_identifier string "weekly" reduction_period_in_epochs int64 156 reduction_factor string (dec) "0.6666666666666" distribution_proportions.staking string (dec) "0.4" distribution_proportions.pool_incentives string (dec) "0.3" distribution_proportions.developer_rewards string (dec) "0.2" distribution_proportions.community_pool string (dec) "0.1" weighted_developer_rewards_receivers array [{"address": "osmoxx", "weight": "1"}] minting_rewards_distribution_start_epoch int64 10

EpochProvision

Calculate the provisions generated for each epoch based on current epoch provisions. The provisions are then minted by the mint module's ModuleMinterAccount . These rewards are transferred to a FeeCollector , which handles distributing the rewards per the chains needs. (See TODO.md for details) This fee collector is specified as the auth module's FeeCollector

ModuleAccount .

Go func (m Minter)EpochProvision (params Params)sdk . Coin {provisionAmt := m . EpochProvisions . QuoInt (sdkmath . NewInt (int64 (params . EpochsPerYear)))return sdk . NewCoin (params . MintDenom ,provisionAmt . TruncateInt ()) } Notes

1. mint_denom
2. defines denom for minting token - uosmo
3. genesis_epoch_provisions
4. provides minting tokens per epoch at genesis.
5. epoch_identifier
6. defines the epoch identifier to be used for mint module e.g. "weekly"
7. reduction_period_in_epochs
8. defines the number of epochs to pass to reduce mint amount
9. reduction_factor
10. defines the reduction factor of tokens at every
11. reduction_period_in_epochs
12. distribution_proportions
13. defines distribution rules for minted tokens, when developer rewards address is empty, it distributes tokens to community pool.
14. weighted_developer_rewards_receivers
15. provides the addresses that receives developer rewards by weight
16. minting_rewards_distribution_start_epoch

17. defines the start epoch of minting to make sure minting start after initial pools are set

Begin-Epoch

Minting parameters are recalculated and inflation paid at the beginning of each epoch. An epoch is signalled by x/epochs

NextEpochProvisions

The target epoch provision is recalculated on each reduction period (default 3 years). At the time of reduction, the current provision is multiplied by reduction factor (default 2/3), to calculate the provisions for the next epoch. Consequently, the rewards of the next period will be lowered by 1 - reduction factor .

Go func (m Minter)NextEpochProvisions (params Params)sdk . Dec {return m . EpochProvisions . Mul (params . ReductionFactor) }

Reductioning factor

This is a generalization over the Bitcoin style halvenings. Every year, the amount of rewards issued per week will reduce by a governance specified factor, instead of a fixed 1/2 . So RewardsPerEpochNextPeriod = ReductionFactor * CurrentRewardsPerEpoch) . When ReductionFactor = 1/2 , the Bitcoin halvenings are recreated. We default to having a reduction factor of 2/3 , and thus reduce rewards at the end of every year by 33% .

The implication of this is that the total supply is finite, according to the following formula:

Total\ Supply = InitialSupply + EpochsPerPeriod * \frac{InitialRewardsPerEpoch}{1 - ReductionFactor}

Events

The minting module emits the following events:

End of Epoch

Type Attribute Key Attribute Value mint epoch_number {epochNumber} mint epoch_provisions {epochProvisions} mint amount {amount}

Minter

The minter is a space for holding current rewards information.

Go type Minter struct {EpochProvisions sdk . Dec // Rewards for the current epoch } Updated3 months ago

Epochs Did this page help you?Yes No *Table of Contents * *The Mint Module * * *Params * * *EpochProvision * * *Begin-Epoch * * *NextEpochProvisions * * *Reductioning factor * * *Events * * *Minter