M^0 TTG Engineering Specification v1.1

M^0 Labs Engineering, March 2024

Overview

TTG (Two Token Governance) is an EVM-compatible, immutable governance module for managing the M^0 core protocol and future M^0 ecosystem periphery.

Smart Contracts structure:

1. **Registrar** for managing parameters and participants of M^0 core protocol and periphery contracts.
2. Three OpenZeppelin style Governors - **StandardGovernor, EmergencyGovernor, and ZeroGovernor** who control the lifecycle of governance proposals.
3. ERC20 **POWER token** with built-in inflation, self-delegation, epoch snapshot tracking, and Dutch-style **Auction** mechanisms.
4. ERC20 **ZERO token** with built-in self-delegation and voting epoch snapshots tracking mechanisms
5. **Distribution Vault** with claiming and distribution of funds functionality to **ZERO token** holders.

Key unique features :

- Epochs for proposals lifecycle. Split of epochs between Transfer/Re-delelegion/Auction and Voting epochs.
- Strictly restricted types of proposals and only one change per proposal.
- Inflation of POWER token supply, dilution of the voting power of inactive POWER participants. Auction of inflation of inactive POWER participants.
- RESET of governors by ZERO holders to POWER or ZERO token holders.
- ZERO rewards distribution to active delegates of POWER token holders per epoch.
- Distribution of proposal fees and other funds to ZERO holders.
- ERC20Votes compatible, epoch-based POWER and ZERO tokens with default self-delegation.
- Ability to batch vote on multiple proposals.
- Unique IDs for proposals that take into account epoch, parameter change, and proposed value. No duplication of proposals.

Main Invariants

1. $POWER\ totalVotingPower_{delegates} >= POWER\ totalSupply_{holders},\ at\ Voting\ epoch$
2. $POWER\ totalVotingPower_{delegates} == POWER\ totalSupply_{holders} + amountToAction,\ at\ Transfer\ epoch$
3. $amountToAuction_{t1} = amountToAuction_{t0} + Inflation_{inactiveParticipants}$

Core architecture

1. Registrar

Generic registry for storing and retrieving data - parameters and actors of the system. Changes of registrar data in addition to infrequent optional changes of some governor parameters is the only possible governance actions in TTG.

Variables and Getters

Name	Definition	Notes & Examples
$EmergencyGovernorDeployer$	The address of the deployer contract that is used for retrieval of the current emergency governor address controlled by POWER token holders.	Is needed because of RESET functionality
$EmergencyGovernor$	The address of the current emergency governor	

$StandardGovernorDeployer$	The address of the deployer contract that is used for retrieval of the current standard governor address controlled by POWER token holders.	Is needed because of RESET functionality
$StandardGovernor$	The address of the current standard governor	
$PowerTokenDeployer$	The address of the deployer contract that is used for retrieval of the current POWER token.	Is needed because of RESET functionality
$PowerToken$	The address of the current POWER token	
$ZeroGovernor$	The address of the zero governor controlled by ZERO token holders.	Never changes, not subject to RESET functionality. For UI convenience, the registrar provides access to all contracts.
$ZeroToken$	The address of the ZERO token	Never changes, not subject to RESET functionality. For UI convenience, the registrar provides access to all contracts.
$Vault$	The address of the Vault contract	

Core functions

$addToList(list,\ account)$

- **Adds** account to list.
- Only $StandardGovernor$ or $EmergencyGovernor$ are able to execute this addition.

$removeFromList(list,\ account)$

- **Removes** account from the list
- Only $StandardGovernor$ or $EmergencyGovernor$ are able to execute this removal.

$setKey(key,\ value)$

- **Sets** $key\ =\ value$.
- Only $StandardGovernor$ or $EmergencyGovernor$ are able to execute.

$get(key):\ value$

- **Returns** $value$ for the key.

$get(keys[]):\ values[]$

- **Returns** $values[]$ for the $keys[]$.

$listContains(list,\ account):\ boolean$

- **Returns** whether $list$ contains $account$ or not.

$listContains(list,\ accounts[]):\ boolean$

- **Returns** whether $list$ contains $accounts$ or not.

2. TTG Governors

Key actors of the governance system responsible to submit, vote, query the state, and execute governance proposals. Epoch-based, allow batch voting on proposals.

Matrix of TTG proposals

Proposal type	**Standard Governor**	**Emergency Governor**	**Zero Governor**	**NOTES**
$addToList$	YES ✅	YES ✅	NO 🚫	POWER proposal
$removeFromList$	YES ✅	YES ✅	NO 🚫	POWER proposal
$removeFromAndAddToList$	YES ✅	YES ✅	NO 🚫	POWER proposal
$setKey$	YES ✅	YES ✅	NO 🚫	POWER proposal
$setProposalFee$	YES ✅	NO 🚫	NO 🚫	POWER proposal
$setStandardProposalFee$	NO 🚫	YES ✅	NO 🚫	
$resetToPowerHolders$	NO 🚫	NO 🚫	YES ✅	ZERO proposal
$resetToZeroHolders$	NO 🚫	NO 🚫	YES ✅	ZERO proposal
$setCashToken$	NO 🚫	NO 🚫	YES ✅	ZERO proposal
$setEmergencyProposalThresholdRatio$	NO 🚫	NO 🚫	YES ✅	ZERO proposal

$setZeroProposalThresholdRatio$	NO 🚫	NO 🚫	YES ☑️	ZERO proposal

2.1 Standard Governor

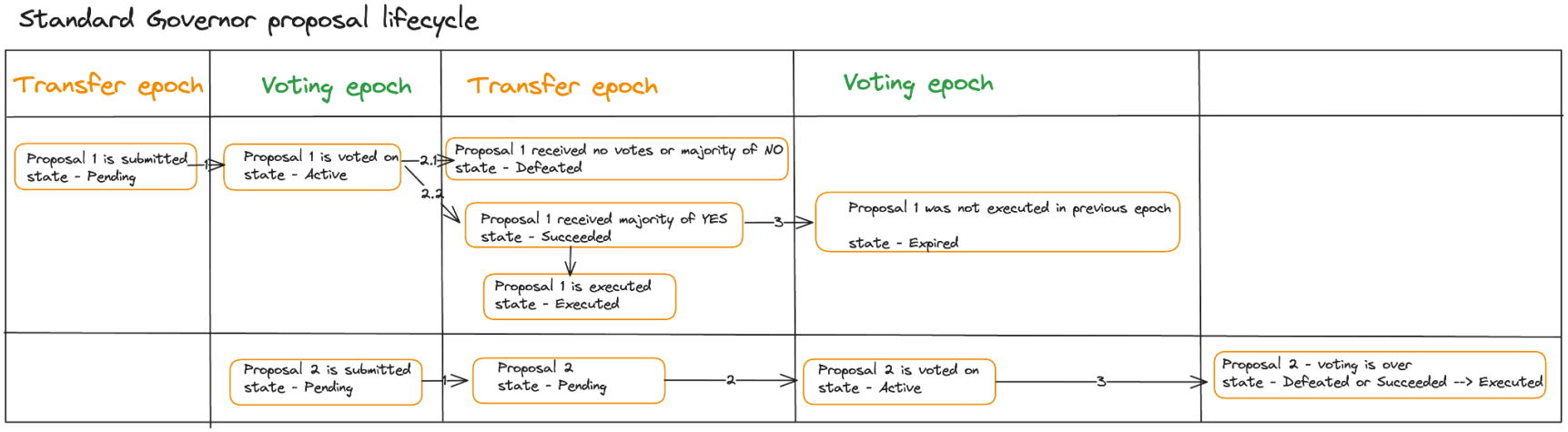
1. The main governor of the TTG system, is expected to be used the most.
2. Allows POWER token holders to receive balance inflation if their delegates voted on all standard proposals in the epoch, and POWER delegates to receive their inflation of POWER voting power and a portion of ZERO token rewards.
3. To receive inflation, a delegate has to vote on ALL proposals.
4. Proposals cost a fee. If the proposal is successful, the fee is refunded to the proposer. If the proposal is successful but expires, the fee is sent to the Distribution Vault.
5. The result of voting is determined by a simple majority.

Variables and Getters

Name	Definition	Notes & Examples
$EmergencyGovernor$	The address of the emergency governor.	$EmergencyGovernor$ is able to change $ProposalFee$ in $StandardGovernor$
$Registrar$	The address of the Registrar	Main change target of most governance proposals

$Vault$	The address of the Distribution Vault	$CashToken$ fees can be sent to Vault
$ZeroGovernor$	The address of the current standard governor	$ZeroGovernor$ is able to change $CashToken$ in $StandardGovernor$
$ZeroToken$	The address of the ZERO token	POWER delegates after voting on all proposals receive their portion of $ZeroToken$ rewards
$MaxTotalZeroRewardPerActiveEpoch$	The number of maximum ZERO rewards given if all POWER delegates voted on all proposals and received their ZERO rewards	Each active epoch (epoch with proposals to vote on) up to $MaxTotalZeroRewardPerActiveEpoch$ is minted. [0, $MaxTotalZeroRewardPerActiveEpoch$]
$CashToken$	The address of the token used to pay for proposal fees.	
$ProposalFee$	The amount of $CashToken$ that is required to be paid for proposal submission.	
$numberOfProposalsAt$	Total number of proposals per epoch	Is needed to check if delegate voted on all proposal in the epoch and is eligible for their inflation
$numberOfProposalsVotedOnAt$	Total number of proposals a voter has voted on per epoch	Is needed to be able to track the voting progres

Governor proposal state transitions



Core functions

$propose(targets[], values[], callDatas[], description)$: $proposalId$

- **Creates** a new unique proposal.
- **Marks** next voting epoch as active if it is the first submitted proposal for the next voting epoch.
- **Charges** proposal fee .
- **Returns** $proposalId$ of the newly created proposal.

$castVote(proposalId,\ support)$: $weight$

- **Reverts** if proposal is not $Active$.
- **Reverts** if $msg.sender$ already voted.
- **Gets** votes from the previous epoch as a $weight$ of $msg.sender$.
- **Accounts** $support$ votes.
- **Increases** $numberOfProposalsVotedOnAt$ for $msg.sender$
- **If** $numberOfProposalsVotedOnAt\ \ == numberOfEpochProposals$:
  - **Mark** $msg.sender$ participation for POWER inflation of token holders
  - **Increase** $msg.sender$ voting power by epoch $inflator$ amount
  - **Mint** ZERO rewards to $msg.sender$
- **Returns** $weight$ of $msg.sender$

$castVotes(proposalIds[],\ supports)$: $weight$

- **Votes** in batch for all $proposalIds[]$.
  [See $castVote(proposalId,\ support)$]

$execute(targets[],\ values[],\ callDatas[],\ description)$: $proposalId$

- **Reverts** if the current epoch is 0.
- **Reverts** if there is no proposal in $Succeeded$ state for calculated $proposalId$.
- **Executes** governance action.
- **Sends** $proposalFee$ back to $proposalId$ proposer

$sendProposalFeeToVault(proposalId)$

- **Reverts** if $proposalId$ is not $Defeated$ or $Expired$.

- **Sends** paid $proposalFee$ to the Distribution vault.

$setCashToken(newCashToken,\ newProposalFee)$

- **Reverts** if not called by $ZeroGovernor$.
- **Sets** $newCashToken$ in $StandardGovernor$
- **Sets** $newCashToken$ in $POWER\ token$
- **Sets** $newProposalFee$

$getProposal(proposalId)$

- **Returns**
  - proposal start epoch;
  - proposal end epoch;
  - state;
  - number of YES votes;
  - number of NO votes;
  - proposer address;

$state(proposalId)$

- **Returns** one of states:
  - Pending
  - Active
  - Executed
  - Defeated
  - Succeeded
  - Expired

[See lifecycle diagram for more information on state's transition for $StandardGovernor$]

Core governance proposals

Proposal function	Action	Notes & Examples
$addToList$	**Adds** address to list. See Registrar $addToList$	$StandardGovernor$ proposal
$removeFromList$	**Removes** address from list. See Registrar $removeFromList$	$StandardGovernor$ proposal
$removeFromAndAddToList$	- **Adds** address to list. See Registrar $addToList$ - **Removes** address from list. See Registrar $removeFromList$	$StandardGovernor$ proposal
$setKey$	**Sets** key in Registrar. See Registrar $setKey$	$StandardGovernor$ proposal
$setProposalFee$	**Sets** new $proposalFee$ as payment for proposals	$StandardGovernor$ or $EmergencyGovernor$ proposal

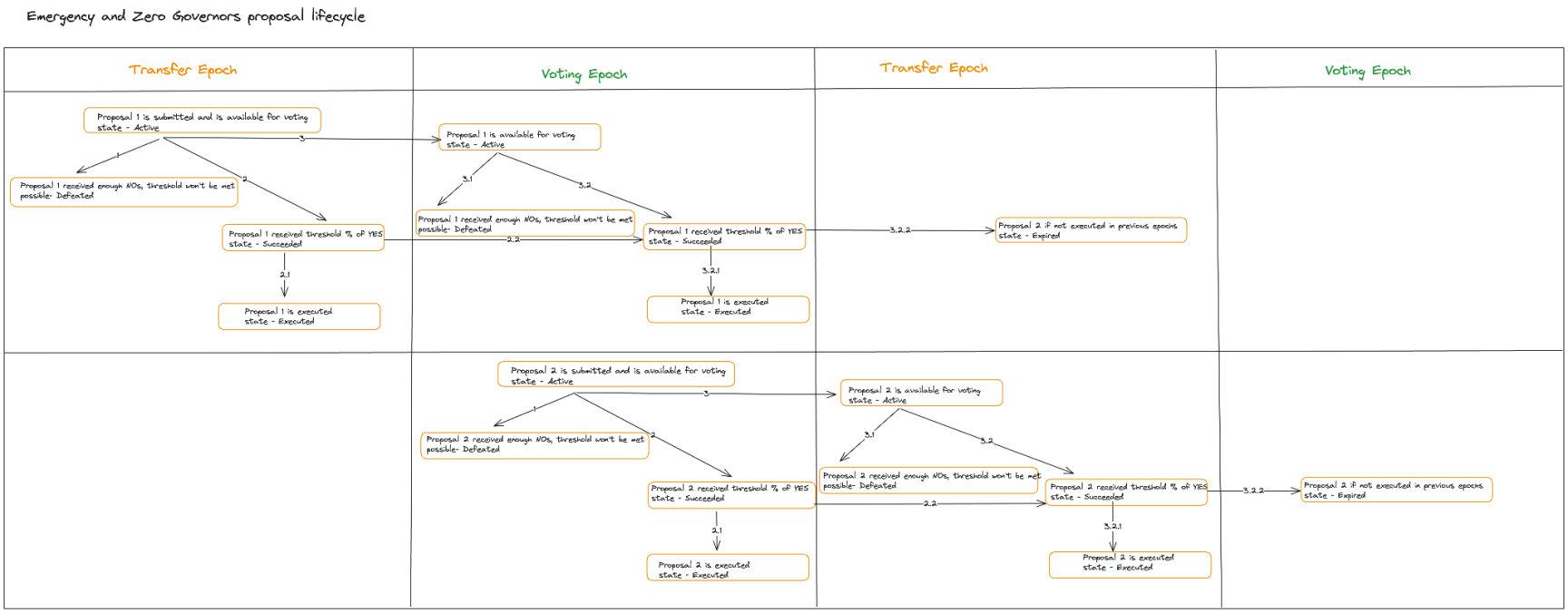
2.2 Emergency Governor

1. Just like Standard Governor, the Emergency Governor is controlled by POWER holders.
2. This governor is used for Emergency proposals - immediately votable governance actions.
3. Requires a high % threshold for the proposal to be successful.
4. Proposals are free, no proposal fee is paid for their submission.
5. Proposals are not mandatory to vote on, there is no POWER inflation associated with them.

Variables and Getters

Name	Definition	Notes & Examples
$StandardGovernor$	The address of the $StandardGovernor$	$EmergencyGovernor$ is able to change $ProposalFee$ in $StandardGovernor$
$Registrar$	The address of the Registrar	Main change target of most governance proposals
$ZeroGovernor$	The address of the $ZeroGovernor$	$ZeroGovernor$ is able to change $ThresholdRatio$ in $EmergencyGovernor$

Governor proposal state transitions



Core functions

$propose(targets[], values[], callDatas[], description)$: $proposalId$

- **Creates** a new unique proposal.
- **Returns** $proposalId$ of the newly created proposal.

$castVote(proposalId,\ support)$: $weight$

- **Reverts** if proposal is not $Active$.
- **Reverts** if $msg.sender$ already voted.
- **Gets** votes from the previous epoch as a $weight$ of $msg.sender$.
- **Accounts** $support$ votes.
- **Returns** $weight$ of $msg.sender$

$castVotes(proposalIds[],\ supports)$: $weight$

- **Votes** in batch for all $proposalIds[]$.
  [See $castVote(proposalId,\ support)$]

$execute(targets[],\ values[],\ callDatas[],\ description)$: $proposalId$

- **Reverts** if the current epoch is 0.
- **Reverts** if there is no proposal in $Succeeded$ state for calculated $proposalId$.
- **Executes** governance action.

$setThresholdRatio(newThresholdRatio)$

- **Reverts** if not called by $ZeroGovernor$
- **Sets** $EmergencyGovernor\ thresholdRatio$ to $newThresholdRatio$

$getProposal(proposalId)$

- **Returns**
  - proposal start epoch;
  - proposal end epoch;

- state;
- number of YES votes;
- number of NO votes;
- proposer address;
- thresholdRatio of governor at the moment proposal was created

$state(proposalId)$

- **Returns** one of states:
  - Active
  - Executed
  - Defeated
  - Succeeded
  - Expired

[See lifecycle diagram for more information on state's transition for $EmergencyGovernor$]

Core governance proposals

Proposal function	**Action**
$addToList$	**Adds** address to list. See Registrar $addToList$
$removeFromList$	**Removes** address from list. See Registrar $removeFromList$
$removeFromAndAddToList$	- **Adds** address to list. See Registrar $addToList$ - **Removes** address from list. See Registrar $removeFromList$

$setKey$	**Sets** key in Registrar. See Registrar $setKey$
$setStandardProposalFee$	**Sets** new $proposalFee$ as payment for proposals in $StandardGovernor$

2.3 ZERO Governor

1. Zero Governor is rarely used, controlled by ZERO holders with a set of unique superpowers.
2. This governor is used for ZERO Threshold proposals - RESETs, change of ZERO and POWER thresholds, toggle of Cash Token from predetermined set of allowed tokens.
3. Requires a high % threshold for proposal to be successful.
4. Proposals are free, no proposalFee is paid for their submission.
5. Proposals are optional to vote on, no inflation associated with them.

Variables and Getters

Name	Definition	Notes & Examples
$EmergencyGovernorDeployer$	The address of the deployer contract that is used for retrieval of the current emergency governor address and deployment of the next one.	Is needed because of RESET functionality

$EmergencyGovernor$	The address of the current emergency governor	
$StandardGovernorDeployer$	The address of the deployer contract that is used for retrieval of the current standard governor address and deployment of the next one.	Is needed because of RESET functionality
$StandardGovernor$	The address of the current standard governor	
$PowerTokenDeployer$	The address of the deployer contract that is used for retrieval of the current POWER token address and deployment of the next one.	
$isAllowedCashToken$	Check if token is allowed to be set as Cash token	[WETH, M] at the moment of launch

Core functions

$propose(targets[], values[], callDatas[], description): proposalId$
See $EmergencyGovernor$

$castVote(proposalId, support): weight$
See $EmergencyGovernor$

$castVotes(proposalIds[], supports): weight$
See $EmergencyGovernor$

$execute(targets[], values[], callDatas[], description): proposalId$
See $EmergencyGovernor$

$getProposal(proposalId)$

- **Returns**
  - proposal start epoch;
  - proposal end epoch;
  - state;
  - number of YES votes;
  - number of NO votes;
  - proposer address;
  - thresholdRatio of governor at the moment the proposal was created

$state(proposalId)$

- **Returns** one of states:
  - Active
  - Executed
  - Defeated
  - Succeeded
  - Expired

Core governance proposals

Proposal function	**Action**
$resetToPowerHolders$	Deploy POWER token, $StandardGovernor$ and $EmergencyGovernor$ with a new POWER token which initial supply is bootstrapped from current

	POWER token. [Note* There is a possibility of having a fraction of INITIAL_SUPPLy to be unallocated after such reset if it happened during Transfer epoch and there was inflation in the Voting epoch. This unallocated supply will be diluted after few active voting epochs]
$resetToZeroHolders$	Deploy POWER token, $StandardGovernor$ and $EmergencyGovernor$ with a new POWER token which initial supply is bootstrapped from current ZERO token
$setCashToken$	Change cash token in $StandardGovernor$. See $StandardGovernor.setCashToken$
$setEmergencyProposalThresholdRatio$	Change threshold ratio of $EmergencyGovernor$. See $EmergencyGovernor.setThresholdRatio$
$setZeroProposalThresholdRatio$	Change threshold ratio of $self - ZeroGovernor$.

3. POWER Token

ERC20, ERC20Votes epoch-based governance token with built-in mechanisms that support inflation, auction, self-delegation, initial supply bootstrapping, and tracking of past voting power and delegates by epochs.

Variables and Getters

Name	Definition	Notes & Examples
$Vault$	The address of the Vault	Is needed for distribution of Auction proceeds to ZERO holders
$StandardGovernor$	The address of the standard governor	Standard governor is triggering inflation when voting on all proposals is done.
$BootstrapToken$	The address of the bootstrap token contract	Is needed to figure out supply of token after bootstrapping or RESET
$BootstrapEpoch$	The bootstrapping epoch	Is needed to figure out supply of token after bootstrapping or RESET

Core functions

$buy(minAmount,\ maxAmount,\ destination,\ expiryEpoch):\ amount,\ cost$

Allows any user to purchase POWER token inflation of inactive participants.

- **Checks** if $currentEpoch\ >\ expiryEpoch$, reverts otherwise
- **Calculates the** amount for purchase as $max(available,\ maxAmount)$.
- **Reverts** if the available amount is less than $minAmount$
- **Calculates** $cost$ for purchase

- **Transfers** $cost$ of $CASH$ tokens from caller
- **Mints** $amount$ of $CASH$ tokens to the caller.
- **Returns (**$amount$**,** $cost$)

$markNextVotingEpochAsActive()$

- **Reverts** if not called by $StandardGovernor$.
- **Sets** target inflated supply for next voting epoch

$markParticipation()$

- **If** $msg.sender$ voted on all proposals in the current epoch,
  - **Inflate** their voting power
  - **Add** inflation to $totalSupply$ of POWER token

$amountToAuction()$

- **Returns the** total amount of tokens available for the auction
  - If epoch is Voting epoch, return 0
  - If epoch is Transfer epoch, return $targetSupply - totalSupply$

$getCost(amount)$

- **Returns** cost in $CashToken$ per amount in $POWER$ token

$targetSupply(amount)$

- **Returns** target supply of POWER token if all POWER inflation is minted (was auctioned of)

$setNextCashToken(amount)$

- **Saves** value of next cash token for starting next Auction with new Cash token

4. ZERO Token

ERC20, ERC20Votes epoch-based governance token with built-in mechanisms that support self-delegation and tracking of past balances per epoch.

Core functions

$mint(recipient,\ amount)$

Allows to mint ZERO rewards to POWER delegates after voting on all proposals

- **Reverts** if not called by $StandardGovernor$
- **Mints** $amount$ of ZERO tokens to $recipient$

$pastBalancesOf(account,\ startEpoch,\ endEpoch):\ balances[]$

- **Returns** *past balances of ZERO owners for [$startEpoch,\ endEpoch$]*

$pastTotalSupplies(account,\ startEpoch,\ endEpoch):\ totalSupplies[]$

- **Returns** *past total supplies of ZERO token for [$startEpoch,\ endEpoch$]*

5. Distribution Vault

Contract used for distribution of funds to ZERO holders. Vault has a built-in epoch-based mechanism. Proposal fees paid for Defeated or Expired proposals, Auction proceeds and M token excessive owed M is distributed via it.

Variables and Getters

Name	Definition	Notes & Examples
$ZeroToken$	The address of the $ZERO$ token	
$distributionOfAt(token,\ epoch)$	The amount of $token$ distributed per $epoch$	Sending tokens to Vault is decoupled from actual distribution.
$hasClaimed(token,\ epoch,\ account)$	Identifies if $account$ has already claimed $token$ distributions per $epoch$	

Core functions

$distribute(token)\colon\ amount$

Allows anyone to distribute an undistributed balance of token (i.e. M, Cash, etc) to Zero holders pro rata to their respective balances at the end of the current epoch.

- **Gets** current epoch
- **Stores** delta between current $balanceOf$ **Vault** and last stored balance per current epoch.

$getClaimable(token,\ account,\ startEpoch,\ endEpoch):\ amount$

Allows ZERO holders to claim their accumulated distributed token (i.e. M, Cash, etc) for an arbitrarily ordered array of previous epochs

- **For** each epoch from $startEpoch$ to $endEpoch$:
  - Check if $account$ has not claimed their distribution yet
  - Calculate pro-rata share of distributable $token$ funds based on ZERO balance of $account$ at the end of epoch

- **Returns** total claimable amount for all eligible epochs.

$claim(token,\ account,\ startEpoch,\ endEpoch,\ destination):\ amount$

Allows ZERO holders to claim their accumulated distributed token (i.e. M, Cash, etc) between a previous start and end epoch, inclusively.

- **For** each epoch from $startEpoch$ to $endEpoch$:
  - Claim $token$ funds if not claimed yet
  - Set $hasClaimed$ for $(token, epoch)\ =\ true$

- **Transfer** claimed $amount$ of $token$ to $destination$

- **Returns** claimed amount.