[WD] Update Tokenomics inflation

Summary: This document contains information on the necessary changes on the Tokenomics implementation in order to update inflation

Created: Mar 12, 2025
Current Version: 1.0.0
Approved for external use by: N/A

Status: WIP | In-Review | Approved | Obsolete

Owner: Mariapia Moscatiello
Contributors: Aleksandr Kuperman, Andrey Lebedev
Approvers: David Minarsch

Overview

A community proposal to modify the OLAS inflation parameters for the next ten years was approved. To implement these changes, two new functions: **getActualInflationForYear()** and **getActualSupplyCapForYear()** - have been added to the TokenomicsConstants.sol contract. These functions hardcode the effective minted amounts (resp. Max supplies) for years 0 and 1 and adjust the inflation parameters for subsequent years in accordance with the proposal. Additionally, a new method, **updateInflationPerSecondAndFractions()**, has been implemented in the Tokenomics.sol contract to ensure that, from year 2 onward, the protocol remains within the updated inflation limits.

The update ensures that historical data remain intact while the protocol dynamically adjusts its inflation parameters for the remainder of year 2 and for all future periods.

Modification Description

The inflation parameters were originally hardcoded in the old version of TokenomicsConstants.sol abstract contract. For updating the inflation numbers from year 3 onward, one simply needs to update these hardcoded values in **getInflationForYear()** and **getSupplyCapForYear()**. However, modifying the inflation for the current year (year 2) requires changes in the old Tokenomics.sol implementation for the following reason.

In the previous design, inflation was handled indirectly within the **checkpoint()** method, which recalculates the inflation per epoch based on the time elapsed since the previous epoch and the current year (see the checkpoint method). Because the protocol's inflation schedule is dynamic throughout the year, the inflation rate must adapt as the year progresses. To address this, the dedicated function **updateInflationPerSecondAndFractions()** has been introduced. This function performs several key operations:

- **Dynamic Inflation Adjustment:** It recalculates the current year's inflation rate by dividing the new yearly inflation (obtained via **getActualInflationForYear(currentYear)**) by the number of seconds left in a year.
- **State Reset:** It resets the effective bond (representing unused bonding allocations) and the retainer (accounting for unused staking amounts). This reset allows it to remain consistent with the new values of inflations for year 0-2.
- **Fractional Rebalancing:** It adjusts the bonding, top-up, and staking fractions so that their sum is less than 100%. This recalibration ensures that no more than the allocated inflation for year 2 is minted.

This update is essential because a simple adjustment to **getInflationForYear(2)** would not account for the dynamic intra-year changes nor reset the accumulated state variables.

Additionally, to preserve historical tokenomics and staking data, the inflation and supply caps for years 0, 1, and 2 are not modified in the original **getInflationForYear()** and **getSupplyCapForYear()** functions. Instead, the actual minted values for years 0 and 1—and the updated inflation limits and supply caps for year 2 onward—are hardcoded in the new functions **getActualInflationForYear()** and **getActualSupplyCapForYear().**

Implications

- **Historical Data (Year 0 to Pre-Update Year 2):**
  For tokenomics and staking point calculations covering the period from year 0 up to the moment before the inflation update in year 2, one must use the **getInflationForYear** function. This function reflects the historical, hardcoded inflation parameters that were in effect during those periods.
- **Post-Update (From Update in Year 2 Onward):**
  Once the new implementation becomes active in year 2, the inflation calculations will be based on the updated logic provided by **updateInflationPerSecondAndFractions()**. In this phase, users should rely on the **getActualInflationForYear()** function to retrieve the current inflation figures, as this functions the new inflation allocation.
- **OLAS supply in 10-year:**
  Under the new inflation parameters, the total OLAS supply over a ten-year period is projected to remain below 1 billion tokens. Moreover, with the updated tokenomics implementation, after the initial 10-year period, the protocol will mint an annual inflation amount capped at 2% of the maximum supply cap determined during those first 10 years (761'726'593).