Credit accounts

Debt management

**Debt management**

User can manage debt size using 2 methods in ICreditFacade:

**Increase debt**

function increaseDebt(uint256 amount) external;

Increases debt by tranferring funds from the pool. To account increased debt, it updates cumulativeIndexAtOpen parameter.

| **Parameter** | **Description** |
| --- | --- |
| amount | Amount to increase borrowed amount |

**Decrease debt**

Decreases debt by paying funds back to pool. The payment also include interest rate accrued at the moment and fees for whole debt. So, you would be charged for: amount + interestAccrued + fees.

function decreaseDebt(uint256 amount) external;

| **Parameter** | **Description** |
| --- | --- |
| amount | Amount to descrease borrowed amount |

**How to compute the total?**

/// NEED TO ADD COMPUTATION CODE HERE

Credit accounts

Closing credit account

**Closing a credit account**

To repay the debt and close a Credit Account normally, the following CreditFacade function can be used:

function closeCreditAccount( address to, uint256 skipTokenMask, bool convertWETH, MultiCall[] calldata calls) external payable;

| **Parameter** | **Description** |
| --- | --- |
| to | The address to which the remaining collateral is sent after repaying the loan and closing the account. |
| skipTokenMask | A mask that encodes the tokens which should not be sent back to the user. Can be used to avoid sending dust or tokens that revert on transfer. |
| convertWETH | Whether to convert WETH to ETH before sending it to the user. |
| calls | The array of calls to execute before closing the account. |

The multicall within closeCreditAccount would typically be used to convert collateral assets into underlying. If there is not enough underlying on the CA after performing the multicall, the Credit Manager will try to transfer the shortfall from the borrower.

This means that it is possible to normally close even an unhealthy account, as long as either the user deposits more underlying during a multicall through addCollateral, or has enough underlying on their address that is approved to the Credit Manager.

Trying to close a Credit Account normally will fail if the Credit Manager cannot repay the entire debt to the pool.

Credit accounts

Liquidation

**Liquidating credit accounts**

There are two types of liquidation in the system: liquidations by health factor and liquidations by expiration.

**Liquidating accounts by health factor**

Once a Credit Account's Health Factor goes below 1, the account can be liquidated in order to make the pool whole and prevent any bad debt. In order to liquidate the account, the liquidator would use a Credit Facade function:

function liquidateCreditAccount( address borrower, address to, uint256 skipTokenMask, bool convertWETH, MultiCall[] calldata calls) external payable;

| **Parameter** | **Description** |
| --- | --- |
| borrower | The address of the Credit Account's owner. |
| to | The address to which the remaining collateral is sent after repaying the loan and closing the account. |
| skipTokenMask | A mask that encodes the tokens which should not be sent back to the user. Can be used to avoid sending dust or tokens that revert on transfer. |
| convertWETH | Whether to convert WETH to ETH before sending it to the user. |
| calls | The array of calls to execute before liquidating the account. |

liquidateCreditAccount checks that the account's health factor is less than 1 before liquidation, and will revert otherwise.

Note that unlike normal Credit Account closure, liquidations do not require the entire debt to be repaid to the pool. Credit Facade computes the total value of the Credit Account before liquidation, and will set the amount repaid to the pool to be totalValue \* (1 - liquidationPremium) if the total value is less than the debt.

The multicall would typically be used to convert all collateral to underlying in order to repay the loan; in case there is less underlying than required (based on above calculation) after performing the multicall, the shortfall will be transferred from the liquidator.

**Liquidating accounts by expiration**

If a Credit Facade is in Expirable mode (see below) and the expiration date is reached, all still-open accounts can be liquidated. In order to liquidate an expired account, the liquidator would use a Credit Facade function:

function liquidateExpiredCreditAccount( address borrower, address to, uint256 skipTokenMask, bool convertWETH, MultiCall[] calldata calls) external payable

The parameters are the same as liquidateCreditAccount.

Liquidations by expiration have lower liquidation premiums, since they are typically less urgent than liquidations of unhealthy positions; the liquidators should consider that when calculating the profitability of a liquidation.

**Motivation**

Down the line, in addition to standard variable-rate loans, Gearbox will support fixed-term loans.

Fixed term loans in DeFi typically involve zero-coupon bond-like tokens that have a certain maturity date and yield paid out at maturity (this is usually represented by a discount that decreases the closer to maturity the token is).

In order to support this mode of liquidity provision, CreditFacade has an optional Expirable mode, which is enabled when CreditFacade.expirable() == true.

**Expiration details**

There is a single expiration date (CreditFacade.params().expirationDate) for each expirable Credit Facade that affects all Credit Accounts opened through it. This expiration date would typically be set shortly before the underlying yield token maturity, so that Gearbox is able to collect any outstanding debt and repay its loan to LP's.

It is not possible to open new Credit Accounts past the expiration date, and all accounts that remain open after the expiration date are eligible to liquidation by expiration.

After expiration, the expiration date in the CreditFacade can be moved forward by the DAO, in order to reflect a new maturity date for loans, which allows opening accounts in the CreditFacade once again.

Credit accounts

Upgradeable Contracts

**Upgradeable contracts**

CreditFacade contains a list of contracts that have practices potentially detrimental to security, called upgradeableContracts. Some of the examples of practices that would be grounds to inclusion are:

* Ability to upgrade the contract implementation
* Ability to make arbitrary calls (even restricted to admins)
* Ability to transfer / transferFrom arbitrary tokens

Some of Gearbox's features are restricted to contracts that are not in the list.

The only current example is CreditFacade.approve(), which allows a user to set arbitrary allowance from a Credit Account to a contract that is recognized in the system. It is dangerous to allow this function for upgradeable contracts, as a compromised contract's implementation can be changed to allow the attacker calling transferFrom for arbitrary tokens, which would allow them to drain approved funds from Gearbox.

We anticipate that new restricted functions and features would be added as Gearbox grows.