Credit accounts

Overview

**Credit accounts**

Composable leverage is based on credit account (DEFI primitive), which is an isolated smart contract which can execute trader's financial orders on third party protocol, but do not provide direct access to the funds on it. Credit Account could take margin loan from the pool, so there are two types of funds on it: user initial funds and borrowed funds.

**Open credit account**

To interact with Gearbox, user should open an credit account. During the opening flow, credit account takes margin loan from the pool and get collateral from user account as well.

Let's check an example:



Trader takes 90 ETH loan and provide 10 ETH of his own initial funds. So, after opening account, the total balance on credit account would be 100 ETH.

Note! In our example, trader provides 10 ETH for simplicity, however, it's possible to provide collateral in different assets and use multicollateral if needed.

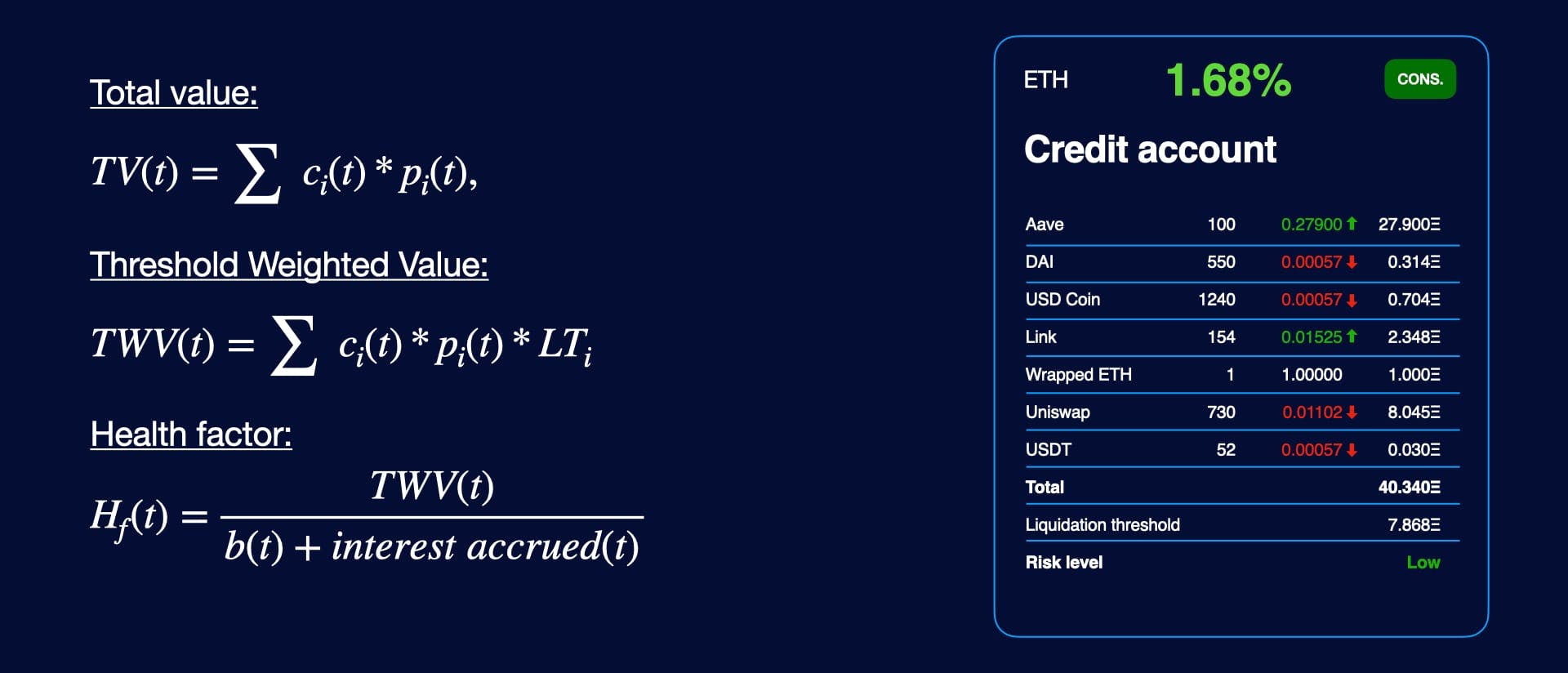
Credit account is an isolated smart contract, trader could execute different transactions, however, has no direct access to the funds.

**Credit manager key parameters**

There is a special smartcontract, which is called CreditManager (in V2 we have 3 linked contracts: CreditFacade, CreditManager and CreditConfigurator), which is responsible for managing credit accounts.

One of the key parameter of each CreditManager is allowed tokens list. This policy allows to use limited list of assets for all accounts opened in one credit manager.

So, in some terms, we can consider credit account as list of different tokens with balances, and introduce here two key parameters:



**Interacting with 3rd party protocols**

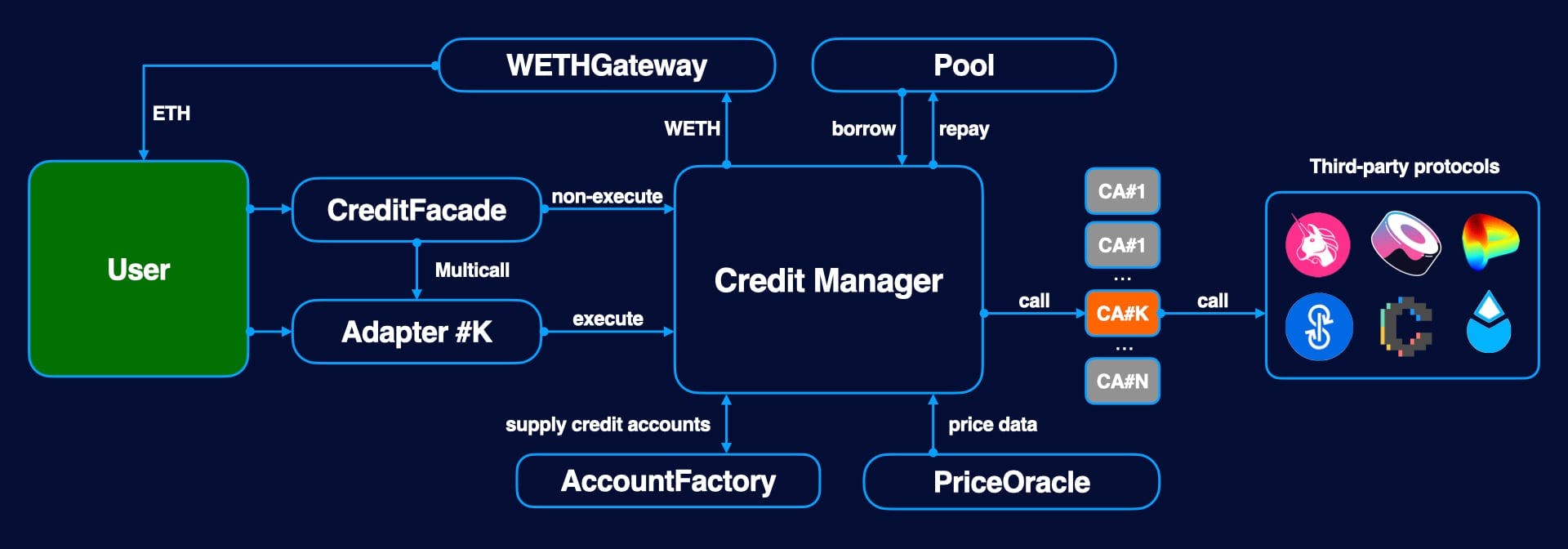
User can consider credit account as additional account. Working with credit account is pretty native. Each contract connected to

Credit accounts

Architecture

**Architecture**

Credit part of the Gearbox Protocol is based on Facade pattern. CreditAccounts are simple contracts, which could execute transaction passed through all required checks. CreditManager is backend which is responsbile for all core operations, users could interact with it via CreditFacade / Adapters contract.



Each user could have only one credit account per creditManager, this one-to-one relationship is stored in mapping(address => address) public override creditAccounts, CM routes calls and operations to particular credit account.

AccountFactory is used to supply reusable credit accounts when it's needed (for more info, check [AccountFactory](https://dev.gearbox.fi/docs/documentation/architecture/account-factory)). CreditManager is allowed to borrow / repay funds from one particular pool. WETHGateway is used to convert WETH into ETH during closing account and liquidations as well.

PriceOracle provide price data based on Chainlink oracles or complex oracles. This data is used to compute collateral value. For more information, please check Oracles chapter.

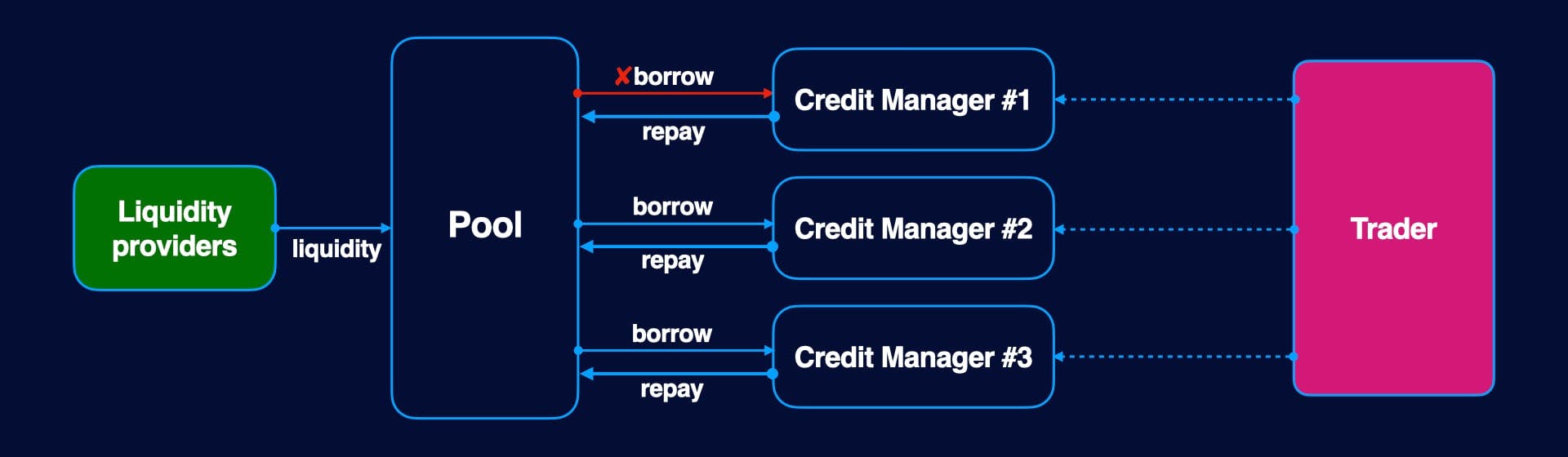
To interact with 3rd party protocols, this protocols could have special contracts which are called adapters (for more information: adapters), which has one-to-one relationship: one contract could be called through one adapter only. Adapters provide the same ABI as original contracts, so when user send transaction to adapter with the same calldata as for original contract, it would be executed using funds on credit account.

**Credit contracts**

| **Contract** | **Responsibility** |
| --- | --- |
| CreditAccount | Primitive, which executes transaction on behalf of |
| CreditManager | Backed contract, which execute operations, however, it could not be called directly |
| CreditFacade | Provide user interface and used for multicall |
| CreditConfigurator | Configure credit manager, could be called by DAO only |

**CreditManager**

CreditManager is core contract to manage credit accounts. In V2, CreditManager itself is designed as low-level backed contract. One pool could be connected with a few of creditManagers, however, CreditManager could be connected with one pool only. It's possible to forbid borrowing money from pool, however, each CreditManager could repay money back.



**CreditFacade**

CreditFacade provides user interface for all operations with credit accounts:

* Opening credit account
* Closing credit account
* Adding collateral
* Managing debt
* Liquidating credit account

It also implements Multicall feature. For more information, please check [Multicall feature](https://dev.gearbox.fi/docs/documentation/credit/multicall)

Credit accounts

Open credit account

**Open credit account**

To interact with Gearbox, a user must open a credit account. During the opening, the credit account borrows funds in underlying from the pool and transfers collateral from the user.

Example:



The trader borrows 90 ETH and provides 10 ETH of their own funds. After opening an account, its total balance is 100 ETH. The trader can then use the credit account to interact with various protocols, but has no access to funds.

**Note:** It is possible to provide collateral in an asset different from the underlying, however, this can only be done through openCreditAccountMulticall.

**Borrowing limits and other restrictions**

Each CreditFacade imposes limits on the borrowed amount for a single CA, set by the DAO. Those limits can retrieved by using a CreditFacade getter CreditFacade.limits(), which returns a tuple of (minAmount, maxAmount).

It is also forbidden to open and close a Credit Account in the same block, and reducing the debt immediately after opening the account.

Opening a CreditAccount on behalf of another user (onBehalfOf != msg.sender) is only possible of [account transfer allowance](https://dev.gearbox.fi/) from the account opener to the user is set to true.

**Methods**

To open a Credit Account, two CreditFacade functions can be used:

**Open credit account**

function openCreditAccount( uint256 amount, address onBehalfOf, uint16 leverageFactor, uint16 referralCode) external payable;

| **Parameter** | **Description** |
| --- | --- |
| amount | Borrower's initial funds |
| onBehalfOf | The address for which the Credit Account is being opened |
| leverageFactor | The amount of leverage to take on. The borrowed amount is computed as amount \* leverageFactor / 100, hence leverageFactor = 100 corresponds to 2x leverage. |
| referralCode | Referral code, which is used for potential partner rewards. 0 if no referral code provided. |

**Open credit account with a multicall**

function openCreditAccountMulticall( uint256 borrowedAmount, address onBehalfOf, MultiCall[] calldata calls, uint16 referralCode) external payable;

| **Parameter** | **Description** |
| --- | --- |
| borrowedAmount | Amount of the underlying to borrow. |
| onBehalfOf | The address for which the Credit Account is being opened. |
| calls | The array of calls to execute immediately after opening an account. |
| referralCode | Referral code, which is used for potential partner rewards. 0 if no referral code provided. |

**NB!** While collateral is automatically transferred from the msg.sender during openCreditAccount, for openCreditAccountMulticall that is no longer the case, and collateral has to be transferred within a multicall by adding a call to CreditFacade.addCollateral() to the calls array.

**Degen mode**

Degen Mode is a special restricted mode in the Credit Facade, designed for testing in production. If Degen Mode is enabled, opening a credit account requires burning a special NFT from msg.sender. If msg.sender's NFT balance is 0, account opening will fail.

The NFT is not returned after closing or liquidating the account, so each NFT allows to open 1 account only.

The NFTs are distributed by the DAO to well-known traders and builders in the space, in order to test the system with limited exposure of pool funds.

CreditFacade.whitelisted() can be used to determine whether the Credit Facade has Degen Mode enabled. CreditFacade.degenNFT() can be used to retrieve the NFT address.