TL;DR

We noticed an issue with the current configuration of FEI's reserve factor at 100% on Aave v2 Ethereum, not accruing variable borrow debt and not redirecting the Reserve Factor fees to the Collector smart contract.

Context

The Reserve Factor (RF) on the Aave protocol is a percentage representing the proportion of the supplier's yield that gets redirected to the protocol itself as a "fee". For example, WETH on Aave v2 Ethereum has a 10% RF, so if the yield for suppliers would be 5% yearly, 10% of it (0.5%) would go to the protocol itself, to a Collector contract and in the shape of aTokens.

On every user action, when the new interest rate is calculated, the amount of aTokens calculated from the RF of that specific asset gets minted to the Collector. In addition, extra accounting gets done on the so-called "indexes" of the asset (supply/variable borrow), which represent the total accumulation of yield on both supply and borrow side.

It is important to highlight that the reserve factor gets factually discounted when calculating the supply rate, prior to further accounting like the indexes.

On September 05, Aave governance proposal 96 was executed, factually freezing the FEI asset on Aave v2 Ethereum.

Initially (we reviewed the proposal pre-submission for Llama), everything looked good with the payload, doing 2 actions:

1. Call a freeze() function on the protocol for FEI.
2. In order for suppliers of FEI to not have incentives and withdraw, "redirect" all yield on the borrowers' side to the protocol's Collector. This was done by setting the Reserve Factor configuration of FEI to the maximum, so 100%.

The problem

While checking something unrelated during the last few days, we noticed that the variable borrow interest seemed to not be properly accruing for FEI borrowers. We made some extra digging and realized that the issue was caused by a side effect of configuring any RF to exactly 100%, which should not work that way.

The core of the problem resides in this condition, on the function _updatesIndexes()

of the ReserveLogic

used by Aave v2's LendingPool

https://github.com/aave/protocol-v2/blob/master/contracts/protocol/libraries/logic/ReserveLogic.sol#L347

[

afei-update-indexes

1356×1444 143 KB

](https://europe1.discourse-cdn.com/business20/uploads/aave/original/2X/e/e161e80d881a32e0fbc0296ba457252828466154.jpeg)

Non-extensive condition on ReserveLogic

This condition is effectively a "short-circuit" to not update certain variables when not required. The (wrong) assumption here is that, whenever the supply rate is 0, the borrowing rate is always 0.

This is actually an accurate assumption in all the cases except one: when the borrow rate is actually more than 0, but the supply rate is 0 as 100% of it gets "discounted" by the RF.

The consequences of not entering in this logic branch are the following:

- The supply index doesn't get updated. This means that aToken doesn't grow, as the index is the "growth component" of it, based on the supply rate and time. Consequently, suppliers yield becomes 0 (this was the goal, so not a problem)

.

- The variable borrow index doesn't get updated. Same as with the supply index, the index on the borrow side is the "growth component" of the debt. So when it doesn't grow, factually the current borrowers of the asset are not getting their debt accrued over time. This was not intended when freezing FEI

.

- As the supply index doesn’t increase, later on in the logic, the protocol assumes there was no yield accumulation, so there is no minting of RF aTokens for the Collector. The redirection to the Collector was not really looked for with the “freezing” proposal, it was just the only mechanism available to “cut” the yield from depositors while keeping borrowers’.

So even if not expected, this doesn’t seem like a big problem.

So in summary:

- RF gets changed to 100%.
- Whenever a user does an action on FEI, the interest rate is recalculated, being 0 on the supply side, due to the 100% RF.
- The protocol assumes wrongly that both borrow and supply rates are 0, so doesn’t update indexes and doesn’t mint aTokens to the Collector from the RF.

In addition, we noticed that Aave v3 has the same problem, which should be addressed.

The issue was not detected when the proposal was submitted mainly for the following reasons:

- This is unintended behavior of the protocol. Meaning it needs to be fixed to allow to set 100% RF.
- The getter functions used to fetch the current rate after execution are not affected by the shortcut. So superficially, it looks like if the borrow rate is working as expected.
- Usually, on these kinds of proposals, tests assume the correct functioning of the underlying protocol, as it is a param change, not a protocol’s upgrade.
- The flow explained before is a bit more complex. Actually, on the first action of the user after the proposal’s execution, the RF accumulation works properly; and it starts malfunctioning after.

The solution

We propose 2 steps to solve this problem:

1. We communicated with the previous proposer (Llama) for them to proceed with a new proposal to set the RF to 99%. Factually this will solve the problem and achieve the same effect of “freezing”, but without reducing borrowers’ debt growth to 0.
2. We will implement a fix on both Aave v2 and Aave v3 to allow 100% RF on assets, without unintended side effects.

In addition, we will improve the testing infrastructure for both proposals and base protocol.