Impermax x Uniswap V2

Protocol Whitepaper V1.0

Simone Rigolon
simone@impermax.finance

Brian Tinsman
brian@predixa.com

December 2020

Abstract
Impermax Finance is creating a set of financial tools based on LP tokens. This whitepaper will describe our first product, a permissionless lending market completely based on Uniswap V2 LP Tokens.

Contents

1 Introduction

1.1 The Liquidity Providing Market

Automated market making (AMM) platforms experienced explosive growth in 2020, going from nearly nothing to about $600 million USD in daily trading volume on decentralized exchanges like Uniswap, SushiSwap, Curve, and Balancer [1].

These platforms rely on liquidity providers to lock ETH and other assets in permissionless smart contracts. In return, the providers receive incentives such as transaction fees. The locked tokens are held in a pool, and the provider is issued LP tokens, which represent their share of that pool. At the time of writing, there are about $3.5 billion USD worth of LP tokens locked on AMM platforms [2].

1.2 Current Problems with the LP Market

1.2.1 Risk of Impermanent Loss

A liquidity provider must lock up some ratio of different tokens together, usually in a pair. When there is a change in the relative prices of tokens in a pair, the AMM system leaves the liquidity provider holding fewer of the high-value tokens and more of the low-value tokens, making the LP token worth less. This is known as impermanent loss, and it's a major risk factor for liquidity providers seeking good ROI. Reducing this risk would encourage risk-averse investors to enter the LP market.

1.2.2 Lost Collateral Opportunity

Lending markets do not generally accept LP tokens as collateral, even if the LP tokens are backed by assets that are accepted. Loans of tokenized assets serve many uses, such as increasing leverage, hedging, and delaying capital gains. Keeping funds in LP tokens therefore incurs an opportunity cost.

1.3 The Impermax Solution

Impermax creates a permissionless lending market where liquidity providers can use their LP tokens as collateral to borrow the tokens in that pair. This largely **solves the problem of impermanent loss**. Lenders can indirectly provide liquidity to the AMM and earn a yield by lending tokens to a borrower with no risk of LP token devaluation. It also **allows borrowers leverage on LP tokens.** Borrowers who are more comfortable with risk can borrow assets to lock up for even more LP tokens, allowing leverage on their yield.

With this solution, Impermax is aiming to become the world's top lending marketplace based on LP tokens.

1.3.1 Novel Solutions

The Impermax lending market is built around two innovations. The first is its unique economic architecture, which keeps all lending pair pools separate. This means that if a borrower's position is liquidated in one pair, the other pairs will not be affected.

The second novel structure is the collateralization model. Instead of using a loan-to-value calculation for collateral, Impermax uses a parameter called "safety margin" to reduce the required over-collateralization and provide much higher leverage than similar current designs.

1.3.2 Productization

The first supported AMM will be Uniswap V2 and will include every pair on the platform. Uniswap V2 has about $1.6 billion USD worth of LP tokens which will immediately be available for use. Other AMMs will follow.

2 Architecture

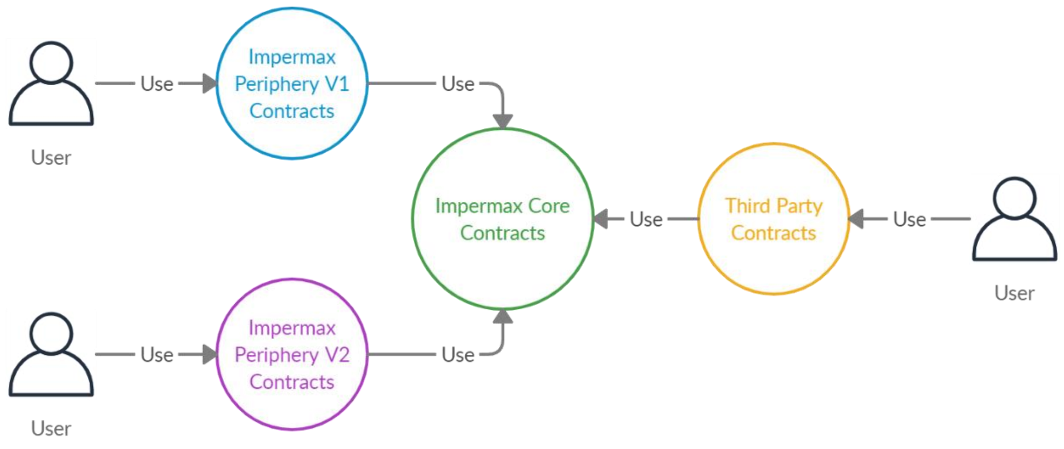
The architecture of the protocol and many smart contract design choices have been heavily influenced by Uniswap. The main reason is that this application will interact directly with the core of Uniswap V2, so we have decided to lay it out as an "extension" to it. Furthermore, Uniswap architecture has already been proven to work.

2.1 Core and periphery

The smart contracts are divided into core and periphery.

The **core** contains the main logic of the application, the state data, and the user's funds. The core contracts are low level, synthetic, versatile, and mostly permissionless and non-upgradable apart from a few parameters adjustable by the governance. These characteristics enable the core to safely secure the user's funds. However, the users won't be able to interact directly with the core contracts due to their low-level nature.

The **periphery** is a constellation of high-level contracts that allow the users to interact with the core. They will enable specific functionalities such as depositing tokens in the core, creating a simple loan, or creating a leveraged position on LP Tokens. But more importantly, they won't store any data. For this reason, users will be able to jump from an old periphery contract to a new one seamlessly since all their data and their funds are stored in the core. Third parties will be free to create new peripheral contracts to implement custom functionalities.



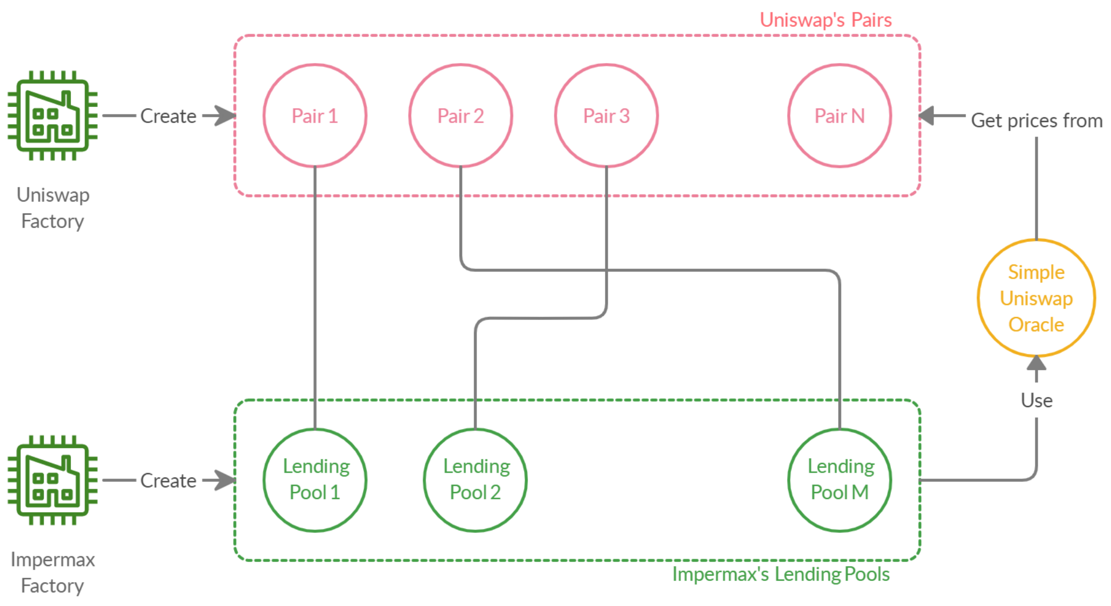
This diagram shows a hypothetical layout for core-periphery contracts. Different versions of periphery contracts can coexist in the same ecosystem, and periphery contracts may be created even by third parties

Updating the core contracts to a new version would be a complicated process because it requires all the users to move their funds to a new core. Instead, updating the periphery contracts to a new version can be done easily because users using an older version will be able to interact with users using the new version since the core is the same.

This whitepaper will focus mainly on the core functionalities.

2.2 Impermax x Uniswap V2 core

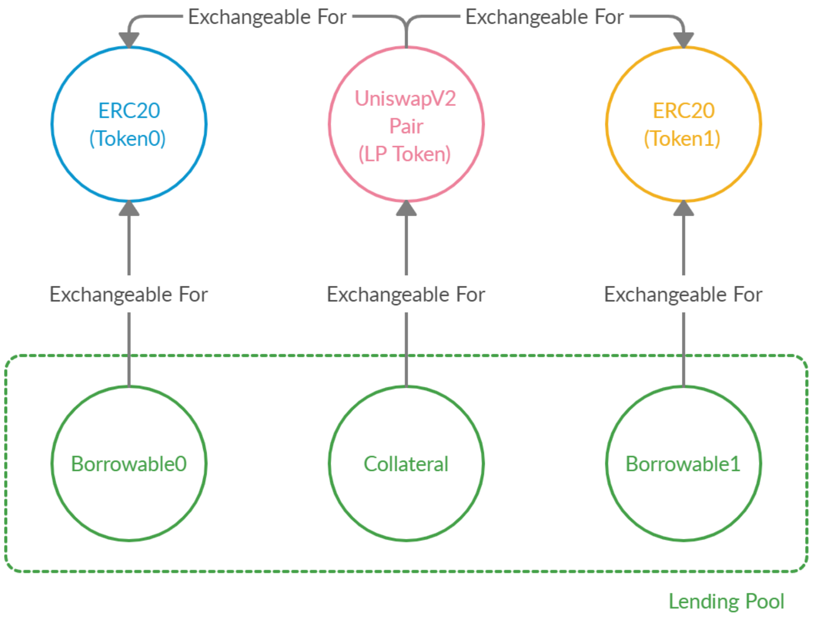
The core contracts are composed of a Factory, a Price Oracle, and many Lending Pools. Anyone can create a new Lending Pool through the Impermax Factory in a permissionless way. This is possible because all Lending Pools are isolated. If a borrower is liquidated in one Lending Pool, then the other Lending Pools will not be affected. Certain tokens like ETH will be borrowable in multiple Lending Pools. Lenders will be free to choose in which one they lend their ETH, and we should expect the ETH interest rate to be different across them. This is a great way for both borrowers and lenders to better manage their risks.



This diagram shows how the Impermax x Uniswap V2 Core interacts with the Uniswap V2 Core

2.3 Lending Pool

Each Lending Pool is associated with a Uniswap V2 Pair and for each Uniswap V2 Pair at most one Lending Pool can exist. Each Lending Pool contains three ERC20 contracts: Borrowable0, Borrowable1, and Collateral.



This diagram shows the architecture of a lending pool and its connections. Each entity represents an ERC20 contract.

Lenders can deposit in Borrowable0 and Borrowable1, respectively Token0 and Token1, which are the tokens swappable on the Uniswap V2 Pair. In exchange, they will receive an interest-bearing token. They will be able to exchange the interest-bearing token with the original one at any time assuming there is enough liquidity in the pool, otherwise they will have to wait until enough liquidity becomes available.

Borrowers can deposit the Uniswap LP token in the Collateral contract to create a loan. As explained earlier, all Lending Pools will be isolated so the Collateral can only be used to borrow the Token0 and Token1 of its own Lending Pool. The specific borrowing, liquidation, and collateralization mechanics are explained in sections 3.2 and 3.3.

3 Mechanics

3.1 Factory

The Factory is a singleton contract that serves as both generator and registry for Lending Pools. It is important also for governance as it stores the addresses of both the Admin and the Reserve Manager (more on this in section 3.6).

Each Lending Pool is associated with a unique Uniswap V2 Pair address, which can be used to retrieve information about that Lending Pool from the Factory. The Lending Pool creation process is completely permissionless. An uninitialized Lending Pool can be created by anyone by passing the address of the Uniswap V2 Pair to the Factory. Creating a new Lending Pool is a very simple process. However, due to some Ethereum gas constraints, it currently requires four steps (and four transactions):

1) creating Collateral
2) creating Borrowable0
3) creating Borrowable1
4) initializing the previously created contracts

The first three steps can be executed in any order, the fourth must follow as last. Applications are planned to streamline this process.

3.2 Lending, Borrowing, and Liquidation

The lending, borrowing, and liquidation mechanics are similar to those of other DeFi projects. In particular, they're inspired by Compound which is currently the largest decentralized lending protocol with 1.7 billion USD in outstanding loans [3]. The main difference from the other lending protocols is the isolation of the Lending Pools explained in section 2.2.

In short, a lender can deposit his tokens in the pools Borrowable0 and Borrowable1 in exchange for interest-bearing tokens. A borrower must first stake his LP tokens in the Collateral pool and then can require a loan of a certain amount of Token0 and Token1. The loan will be given to the borrower if at that moment there is enough liquidity in the Borrowable pools and if the LP tokens deposited by the borrower are more than the $collateralNeeded$ calculated by the collateralization model. If at any point in time the collateral deposited by the borrower is less than the $collateralNeeded$ the loan is in a liquidatable state. A liquidator can repay a liquidatable loan and receive in exchange the equivalent value of the loan multiplied by $liquidationIncentive$ in collateral seized from the borrower at the current market price. A lender can redeem his tokens at any time assuming there is enough liquidity in the Borrowable pool, otherwise, he will have to wait until enough liquidity becomes available. A borrower can redeem his LP tokens at any time as long as he leaves enough collateral in the pool to back an eventual loan he owes.

There are two kinds of fees that borrowers pay to lenders. The first, $BORROW\_FEE =$ 0,1%, is a fixed one-time fee. The second is the interest that the loan accrues over time based on the interest rate model (more on this in section 3.4).

3.3 Collateralization Model

The collateralization model calculates the $collateralNeeded$ for a loan to not go underwater if a price swing of $safetyMargin$ happens on the Uniswap pair when the loan is in a not liquidatable state. In other words, we guarantee that if a loan is not liquidatable at a certain point of time, after a price swing of $safetyMargin$ the value of the collateral will still be enough to back the loan.

At any given point in time the value of a Uniswap LP token is backed 50% by Token0, and 50% by Token1, according to the market price on Uniswap. We use this property to create an optimal collateralization model for LP tokens.

Given:

- $amount_0$, $amount_1$ number of Token0 and Token1 borrowed
- $price_0$, $price_1$ how many LP Tokens are worth one Token0 and one Token1

We can calculate the value of the loan in LP Tokens:

$$value_0 = amount_0 * price_0$$

$$value_1 = amount_1 * price_1$$

And finally, calculate the amount of LP tokens needed:

$$collateralNeeded = \begin{cases} \left(value_0 * \sqrt{safetyMargin} + \frac{value_1}{\sqrt{safetyMargin}} \right) * liquidationIncentive, & value_0 \geq value_1 \\ \left(\frac{value_0}{\sqrt{safetyMargin}} + value_1 * \sqrt{safetyMargin} \right) * liquidationIncentive, & value_0 < value_1 \end{cases}$$

This is because, for the nature of the automated market maker, when Token0 in the pool goes up by $priceSwing$ in comparison to Token1, $price_0$ goes up by $\sqrt{priceSwing}$ and $price_1$ goes down by $\sqrt{priceSwing}$. We calculate what the value of the loan would be after a certain price swing of $safetyMargin$, then we multiply by $liquidationIncentive$ so that after the price swing the collateral will be enough to repay the loan and to pay the liquidator. We always calculate $collateralNeeded$ based on the worst scenario. That is when the price goes up for the tokens for which the loaned value is greater.

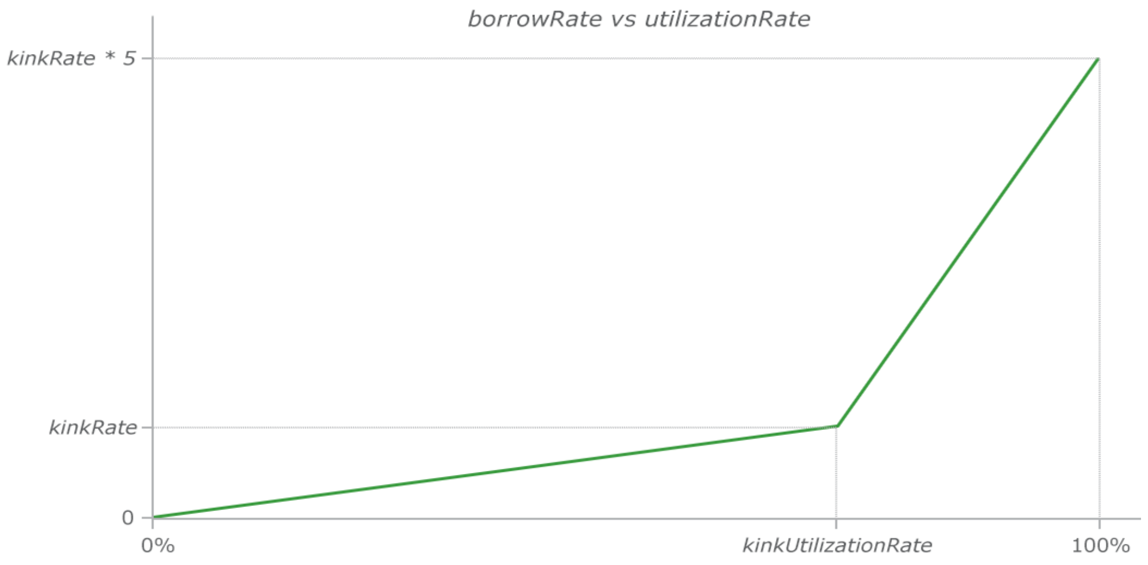
By defining $collateralFactor = \frac{(value_0 + value_1)}{collateralNeeded}$ we can notice that using this collateralization model, the maximum $collateralFactor$ can be achieved when $value_0 = value_1$. In this case we call the loan **optimally collateralized**. In this specific scenario the loss of value of the collateral in comparison to the value loaned will coincide with the impermanent loss.

By default the value of $liquidationIncentive$ is 104% and the value of $safetyMargin$ is 250%. In other words, once a loan is liquidatable the ratio of prices can increase by up to 2.5x or decrease by 1 / 2.5x before the loan is undercollateralized. We use such a high $safetyMargin$ because some low liquidity pairs may have a high volatility. However, these parameters can be tweaked for every specific Lending Pool by the governance (more on this in section 3.6).

In the following table, we show the $maxLeverage = \frac{1}{100\% - collateralFactor}$ when the loan is optimally collateralized. Note that these leverages are at the limit, so a minimum price change would cause the loan to be liquidated.

$safetyMargin$	$liquidationIncentive$	$collateralFactor$	$maxLeverage$
250%	104%	86.87%	7.61x
200%	104%	90.65%	10.7x
175%	104%	92.51%	13.4x
150%	104%	94.21%	17.3x

3.4 Interest rate model



Each Borrowable pool will have its own interest rate model that acts independently from other Borrowable pools. Given the permissionless nature of the application and considering that for a governance system there may be too many pairs to be handled individually, we decided to opt for a **fully adaptive interest rate model**.

It is a kinked interest rate model with only one parameter adjustable by the governance: $kinkUtilizationRate$. Let us define $borrowRate$ as the interest that is being accrued from the loans, $utilizationRate$ as the percentage of funds deposited by the lenders that are utilized in loans, and $kinkRate$ as a parameter dynamically set by the protocol. Then:

- $utilizationRate = 0\% \leftrightarrow borrowRate = 0$
- $utilizationRate = kinkUtilizationRate \leftrightarrow borrowRate = kinkRate$
- $utilizationRate = 100\% \leftrightarrow borrowRate = kinkRate * 5$

When $utilizationRate > kinkUtilizationRate$, $kinkRate$ increases. When $utilizationRate < kinkUtilizationRate$, $kinkRate$ decreases. The speed at which the parameter $kinkRate$ adjusts itself will be slower as $utilizationRate$ is closer to $kinkUtilizationRate$. The rationale for this is that $utilizationRate$ will eventually settle around $kinkUtilizationRate$ and $kinkRate$ will be the fair interest rate determined by the laws of supply and demand.

3.5 Price Oracle

The Price oracle is a singleton contract named SimpleUniswapOracle. It is a price oracle based on Uniswap's TWAP (Time Weighted Average Price). It has been designed with particular attention to simplicity and gas costs, and in such a way that also other DeFi projects will be able to reuse it. While numerous alternative price oracle solutions are available, this TWAP solution allows the platform to remain highly automated and permissionless. Anyone can create a new Lending Pool containing an arbitrary combination of assets, so traditional oracles that require manual set-up of new pairs won't work. The best way to guarantee support for all pairs on Uniswap is to draw the prices from Uniswap itself.

For each pair, Uniswap V2 stores the cumulative price that is the sum of the spot price at each second in the history of the contract. By sampling the cumulative price at two points in time it is possible to calculate the TWAP between those two points by dividing the difference of the accumulators by the time elapsed.

SimpleUniswapOracle guarantees that the price is always calculated over an interval longer than $MIN_T = 1\ hour$. Every time that the price of a pair is retrieved, the oracle saves the current accumulator in a state variable so that it can be used to calculate the TWAP in future interactions. For an in-depth explanation of how the oracle works refer to this article: https://simonerigolon.medium.com/a-simple-and-fully-decentralized-price-oracle-based-on-uniswap-for-defi-cc70807240f2

3.5.1 Price freshness and security considerations

Price oracles have been exploited many times by malicious actors attacking DeFi applications. Our main focus in the building of this price oracle is the safety of the funds of our users. A price oracle can bring a security issue to a DeFi application when it reports a price that is different from the market price.

The price of an oracle is more easily manipulated in the short term. Many DeFi attacks in the past leveraged the possibility for certain price oracles to be manipulated. Uniswap TWAP resolves this problem because a malicious actor would need to manipulate the price for the whole duration of the interval used for the TWAP. In our case, that duration is at least $MIN_T = 1\ hour$.

Using TWAP has a drawback: the price is less fresh. Since the TWAP is an average of the prices, it is expected to always be a bit different from the real market price. However, that difference won't have a big impact under normal circumstances. Even if the price was not updated for several hours or days the price given by the TWAP would still be a price that occurred at some given point in time since the last update, so it is a price that the price oracle could have returned at some point even if it was returning accurate prices. If a user needs a fresher price to do an operation, he can manually update the oracle before doing that operation.

A concerning scenario would be the one in which the market price differs from the TWAP by more than $safetyMargin$. In this case, a malicious actor could take a loan of value greater than the collateral that he deposited, and so he would be able to steal the difference in value. Such a scenario is unlikely to happen. However, we could manually update the oracle when the TWAP differs from the market price by a certain margin. This will guarantee that the scenario described above will never happen.

3.6 Governance

The core of Impermax x Uniswap V2 is permissionless, non-upgradable, and built with a minimal governance philosophy. The idea is that the protocol would be self-sufficient even without an active governance. However, the governance system will be able to optimize the protocol by adjusting a few parameters.

The rights of the protocol and all future Impermax Finance's products will be controlled by the IMX token holders. They will be able to vote to govern the protocol and will be rewarded with a share of the profits generated by it. The protocol profits are stored in the reserves and can be withdrawn by the IMX token holders through the governance system.

3.6.1 Parameters

Name	Default	Constraints	Description
Factory			parameters shared by all Lending Pools
$admin$			is the only address with the permissions required to change any parameter
$reserveManager$			is the address enabled to withdraw from the reserves of any Lending Pool
Borrowable			parameters adjustable for each Borrowable
$reserveFactor$	10%	$\in [0\%,\ 20\%]$	explained in the following section
$kinkUtilizationRate$	70%	$\in [70\%,\ 90\%]$	explained in section 3.4
Collateral			parameters adjustable for each Collateral
$safetyMargin$	250%	$\in [150\%,\ 250\%]$	explained in section 3.3
$liquidationIncentive$	104%	$\in [101\%,\ 105\%]$	explained in section 3.3

3.6.2 Profit and reserves

The protocol generates profit through the $BORROW\_FEE$ and through the interest rate. That profit is divided by the lenders and the protocol. A percentage of the profit defined by $reserveFactor$ will be kept in the Borrowable reserves. The rest of the profit will be accrued to lenders by inflating the value of the interest-bearing token.

The governance can set the address of the $reserveManager$ which will be able to withdraw tokens from the reserves at any time. The reserve management strategy is at the complete discretion of the governance. Reserves may remain untouched in view of unexpected future expenses or they may be withdrawn regularly in order to distribute the profit to the stakeholders.

3.7 Flash Borrows and Flash Liquidations

In DeFi a flash loan is a loan that must be repaid by the end of the atomic transaction [4]. Flash Borrows and Flash Liquidation are kinds of flash loans functionalities. As mentioned in the Protocol Architecture chapter, many design and implementation choices have been influenced by Uniswap V2. The implementation of Flash Borrows and Flash Liquidations is similar to that of Uniswap V2 Flash Swaps [5].

3.7.1 Flash Borrows

When a borrower requires a loan, the protocol optimistically sends him the funds and then it makes a call to an optional user-specified callback contract. Once the callback is completed, the protocol checks that the borrower has either repaid the loan or has enough collateral to back that loan. If not, the transaction will revert. As for a normal borrow, a Flash Borrow is subject to a $BORROW\_FEE = 0{,}1\%$.

3.7.2 Flash Liquidations

When a liquidator wants to liquidate a loan, he first declares the borrower and the size of repayment he intends to make. The protocol optimistically sends him the funds and then it makes a call to an optional user-specified callback contract. Once the callback is completed, the protocol checks if the liquidator actually repaid the loan by the amount that he initially declared. If not, the transaction will revert. A Flash Liquidation is subject to no fees.

3.7.3 Flash Borrows implications

Flash Borrows enable flash loans for each borrowable token, with a fee of 0,1%. They facilitate features such as LP Token leveraging. For example, let’s say that a liquidity provider owns some LP Tokens for a value of 100, and he intends to achieve a 5x leverage on his initial position. He can create two Flash Borrows. The first is of Token0, the second of Token1, both of value 200. Then he can exchange his Token0 and his Token1 on Uniswap for an amount of LP Tokens of value 400. Finally, he can deposit all his LP Tokens as collateral so that they guarantee the loans he just requested. All this will happen in the same atomic transaction. At the end, the liquidity provider will hold 5 times the LP Tokens that he had initially and so he will have reached the desired leverage.

3.7.4 Flash Liquidations implications

Flash Liquidations are a functionality designed to facilitate the job of the liquidators. The idea is that they will be able to liquidate liquidatable loans without owning any token. If a liquidator wants to liquidate a loan, he can first obtain the LP Tokens through the Flash Liquidation functionality. Then he can swap them on Uniswap in order to obtain the amount needed to repay the loan. Finally, he will repay the loan and keep the tokens leftover for himself. This functionality is very important for the economic stability of the protocol as it allows anyone to act as a liquidator.

4 Summary

Impermax provides a robust, permissionless market for lending and borrowing against AMM LP tokens. The system offers far-reaching benefits to decentralized finance. It creates a new way to participate in the liquidity providing market: lending assets to LP token holders in order to earn interest. In this way, lenders can earn yields from AMMs with no exposure to impermanent loss. This is a powerful way to reduce risk.

At the same time, the system allows borrowers to re-invest borrowed tokens to create leverage on their holdings. This is a powerful way to increase potential rewards, along with increased risk.

In this way, Impermax creates economic efficiencies across the entire AMM industry by allowing users to choose the level of risk and reward that they desire, no matter what assets they hold. It also opens up billions of USD in value for useful purposes that would otherwise be locked in unusable LP tokens.

Impermax is the first protocol to have a collateralization model completely based on LP tokens. This model allows much higher leverage than competing protocols, with better liquidation thresholds.

Impermax is built to be a self-sufficient decentralized platform with built-in price determination, user-created lending pools, and minimal governance. After the launch of Impermax x Uniswap V2, the system will expand to support all major AMMs.

References

[1] "Decentralized exchanges 24 hours volume," December 2020. [Online]. Available: https://explore.duneanalytics.com/public/dashboards/c87JEtVi2GlyIZHQOR02NsfyJV48eaKEQSiKplJ7. [Accessed December 2020].

[2] "DeFi Pulse," December 2020. [Online]. Available: https://defipulse.com. [Accessed December 2020].

[3] "DeFi Lending," December 2020. [Online]. Available: https://defipulse.com/defi-lending. [Accessed December 2020].

[4] Binance, "What Are Flash Loans in DeFi?," November 2020. [Online]. Available: https://academy.binance.com/en/articles/what-are-flash-loans-in-defi. [Accessed December 2020].

[5] H. Adams, N. Zinsmeister and D. Robinson, "Uniswap v2 Core Whitepaper," March 2020. [Online]. Available: https://uniswap.org/whitepaper.pdf. [Accessed December 2020].