

To refresh and elaborate, Meta Defender is meticulously designed to enhance and work in tandem with the existing Pendle ecosystem. By capitalizing on the solid foundation that Pendle has laid with YT and PT, we have innovated features that not only accentuate the strengths of yield tokens but also pave the way for groundbreaking functionalities. Central to these innovations are perpetual contracts for LST yield based on the dynamic metaYT (previously called wrappedYT), our specialized fixed-yield product, Meta Earn, and the concept of redeployable LP.

Note: In this walkthrough, we use LSD as an example to show what Meta Defender can do with the Pendle ecosystem. All the math and examples are based on ETH as the main value. But, the same ideas can also work for the RWA pool in Pendle, using US dollar or stablecoins as the main value.

Let's embark on an in-depth exploration.

1. Understanding metaYT and how it works in perpetual trading

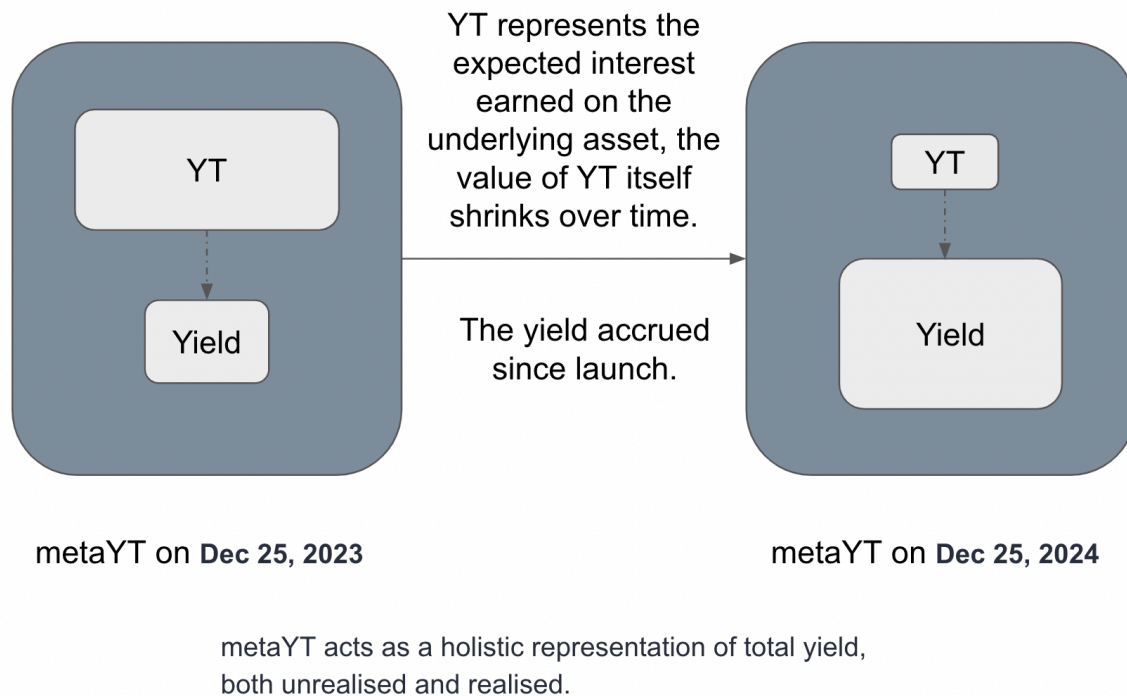
Among the most popular perpetual contracts platforms today are Perpetual Protocol and GMX. These platforms empower users to speculate on the price fluctuation of various assets, allowing them to take either long or short positions without the constraints of an expiry date. This feature offers traders more flexibility and opportunities in their trading strategies. In observing the landscape, we recognized the untapped potential of LST yield as an enticing asset for speculation. Yet, surprisingly, no existing protocol has ventured into this domain. The primary impediment? The absence of a suitable asset that could be effectively shorted or longed.

Turning our gaze to the innovative solutions within the market, Pendle's yield token (YT) caught our attention. The YT, in its essence, stands as a distinct and intriguing asset. Due to its unique characteristic of "shrinking" while simultaneously accumulating profits from its underlying, it cannot be directly employed as a "longable or shortable" asset and be used as a featured asset for leveraged trading in perpetual contracts. Thus, we've introduced metaYT.

For an asset to be effectively traded in a perpetual market, it should avoid losing value bits by bits. Think of metaYT as a pocket, catching any falling bits of ETH yield and keeping everything intact. This design ensures the overall value doesn't just dissolve away. MetaYT is crafted precisely for this purpose. It's more than just a token; it's an ERC20 contract that holds YT. And every day, it collects the yield, ensuring no value is left on the table.

At its core, metaYT amalgamates both the future discounted yield and the yields realized up to the present moment. It embodies the totality of yield, setting the stage for its prime candidacy in the derivatives market.

$$1 \text{ metaYT} = 1 \text{ YT_held_by_it} + \text{yield_accrued_by_YT_since_metaYT_launch}$$



To provide more clarity, consider the following example of how to mint a metaYT and use it in Meta Defender perpetual trading:

Imagine a Pendle YT, poised to mature on **Dec 25, 2025**, anchored by stETH as its foundational LST. We'll refer to this as **YT_stETH_25DEC2025** (or "YT" for brevity).

On **Dec 25, 2023**, a new pool designed for perpetual trading was established. The underlying asset for the featured asset, **metaYT_stETH_25DEC2023_25DEC2025** (or **metaYT** for brevity), in this pool is **YT_stETH_25DEC2025**. On that very day, users could mint **metaYT_stETH_25DEC2023_25DEC2025** for perpetual trading or becoming LP for this pool, the valuation of which is demonstrated as follows:

- **Day 1:** 1 metaYT corresponds precisely with 1 YT.
- **Day 2 onwards:** The bond between the two assets evolves. Now, 1 metaYT mirrors 1 initial YT plus the yield accrued since launch.

- **By its end:** The relationship crystallizes as: **metaYT = initial YT(which is 0 by maturity) + cumulative yield from its outset to termination.**

But what about minting a new metaYT post-launch?

Drawing from this example, let's imagine someone wants to mint a new **metaYT_stETH_25DEC2023_25DEC2025** on **Dec 25, 2024**. He would:

- Ascertain the ETH-denominated price of the **YT_stETH_25DEC2025** for that day.
- Compare it against the current ETH-denominated price of **metaYT_stETH_25DEC2023_25DEC2025**—which is the sum of the YT's ETH price on Dec 25, 2024, and the cumulative yield from Dec 25, 2023, to Dec 24, 2024. Get the **price difference**.

To successfully mint a **metaYT_stETH_25DEC2023_25DEC2025** on **Dec 25, 2024** users should:

- Hold the freshly acquired **YT_stETH_25DEC2025** on **Dec 25, 2024**.
- Compensate for the **price difference** with ETH.
- Synthesize a **metaYT_stETH_25DEC2023_25DEC2025** by above.

This ingenious design captures the progression of yield over time. Essentially, metaYT acts as a holistic representation of total yield, solidifying its position as a prime candidate for an underlying asset in the derivatives arena.

Drawing inspiration from GMX's perpetual market, we've molded our unique yield leveraged trading domain using metaYT. Here, unlike GMX, where ETH's valuation is pegged to the US Dollar, the yield, when evaluated in assets like stETH, uses ETH as its standard. Consequently, what was USDC in GMX transforms to ETH or WETH in our structure, and the "longable or shortable" asset metamorphoses from ETH to metaYT.

Finally, when delving into the operations surrounding metaYT, it becomes apparent how this innovation amplifies the Pendle ecosystem. Envision a metaLP, operating similar to GLP in GMX. The potential benefits for metaLP providers include margin fees, funding fees, liquidation profits, and a specialized liquidity mining initiative. Also, just as you can deposit ETH to receive GLP, you can deposit metaYT to obtain metaLP. So both for prospective metaLP participants and those desiring to long YT, YT becomes a prerequisite. Crucially, actions like initiating or liquidating positions instinctively spark YT trades, thereby energizing activity within the Pendle swap.

2. The Need and Innovation Behind Fixed Yield Products without Lockup: Meta Earn

In the bustling marketplace of DeFi, many users are drawn towards the promise of predictable returns. LST yield offers a return rate and nature that resembles an on-chain saving account. However, unlike traditional saving account deposits, the yield from LST is subject to fluctuations. This inherent volatility doesn't sit well with every investor, especially the conservative ones. They often look for a product that can assure them of **a fixed Annual Percentage Yield (APY), regardless of when they choose to withdraw their assets or exit liquidity**. This drive is what brings fixed yield products to the fore.

Pendle has made good progress in the fixed-income DeFi area. By keeping PT until it's due, users can get the expected fixed return. However, there are issues, especially when the market changes quickly or when users need cash right away. If users sell PT before its maturity, they'll only get the market price at that time. There's a risk this price might not meet the APY they hoped for when buying PT, leading to potential losses.

To understand how to achieve fixed returns through our innovative **Meta Earn**, implemented with PT and metaYT, we must first grasp the essence of PT and YT. Let's say an investor buys PT at the start, which we'll call time 0. The price is known then and we'll call it $P(0)$. The price of the related YT at that time is $Y(0)$. As such, the investor can forecast the annualized return, r , they would receive by holding it until maturity:

$$r = \frac{(Asset - P(0)) \times (maturity - 0)}{365}$$

Where:

- When priced in terms of ETH, the value of $Asset$ is typically set to 1.
- $P(0)$ is the value of PT at the time of purchase. Since the relationship $PT + YT = Asset$ holds true and $Asset$ is taken as 1 when priced in ETH, $P(0)$ is naturally a number less than 1.

The formula essentially calculates the yield rate based on the difference between the value of the underlying asset and the initial value of PT, adjusted by the duration until maturity.

Ideally, if r remains constant, which is also our aim, the value of PT at any given time t (which is $P(t)$) should be:

$$P(t) = P(0) + \frac{rt}{365}$$

This indicates that if the value of PT exceeds $P(0) + \frac{rt}{365}$, the investor realizes an additional profit beyond the expected yield. Conversely, they suffer a loss. However, fixed income enthusiasts do not seek this uncertainty; their objective is to ensure a predetermined fixed return.

Imagine, for a moment, if we had **a magical tool**. With this tool, we could create a

position that hovers right around $P(0) + \frac{rt}{365}$ as a baseline. We could then use this

position to go short on $P(t)$. If $P(t)$ drops below $P(0) + \frac{rt}{365}$, this short position would make a profit, filling in the gap. By doing so, it would serve as the perfect counterbalance to its inherent volatility, ensuring a steady return.

Now, if we already possess PT as a tangible asset, the natural next step with our magical tool would be to go short on $P(t)$. But here's the twist: since $YT = 1 - PT$, we can also interpret this position as **going long on YT**.

However, going long or short on YT directly in the classic perpetual model (e.g. GMX) isn't practical because the value of YT continuously dwindles, as previously explained in Part 1.

That's why Meta Defender also introduced metaYT in this fixed-yield product. As mentioned above, metaYT is an ERC20 contract encapsulating YT, accumulating all the revenues from YT since its inception, which is suitable for longing or shorting.

So, would a long metaYT position be able to fully offset fluctuations in PT's value when

$P(t)$ is less than $P(0) + \frac{rt}{365}$? This also translates to $Y(t)$ being greater than $Y(0) - \frac{rt}{365}$. On a surface level, one might say yes. But we can delve deeper for precision.

A crucial aspect to bear in mind is that, unlike in Part 1 where metaYT positions are used to gain profits, for fixed-yield enthusiasts, their real focus isn't on the actual value

of PT and YT on day t . Instead, their gaze is firmly set on the differential between these values and $P(0) + \frac{rt}{365}$ or $Y(0) - \frac{rt}{365}$. This differential symbolizes the genuine gains or losses they can expect. It becomes the vital metric that these investors pay attention to. Hence, our long metaYT position should pivot towards precisely hedging this differential.

As a result, there's a subtle distinction in the mechanics of the metaYT position presented here compared to that in Part 1.

For a hands-on example:

Consider an investor who purchased **PT-stETH on December 25, 2023, with 0.8 ETH**. The **maturity date** for this investment is set for **December 25, 2025**, projecting a fixed annual percentage rate (APR) of **12.5%**. However, unexpected circumstances on December 25, 2024, force the investor to contemplate selling this PT. Without a safeguard in place, no one can guarantee that the PT's value will stand at **0.9ETH** on December 25, 2024(a year before its maturity). This uncertainty is precisely the problem we aim to solve.

The overarching goal is to **solidify the PT's return rate at 12.5%**. Given the foundational relationship, **1PT + 1YT = 1 underlying** (in this instance, 1 ETH), so on December 25, 2024, if the PT's price falls short and values at **0.88ETH**, a deficit of **0.02ETH**, it implies a corresponding increase in the YT's price by **0.02ETH**.

To cement this **12.5% return** from the outset by **Meta Earn**:

- Upon purchasing 1 PT, **Meta Earn** simultaneously **opens a LONG position on metaYT** using the Meta Defender perpetual market, with a position size equivalent to **1 metaYT**.
- This **1PT spot**, coupled with the **1 metaYT LONG**, crafts a balanced portfolio which the investor retains.

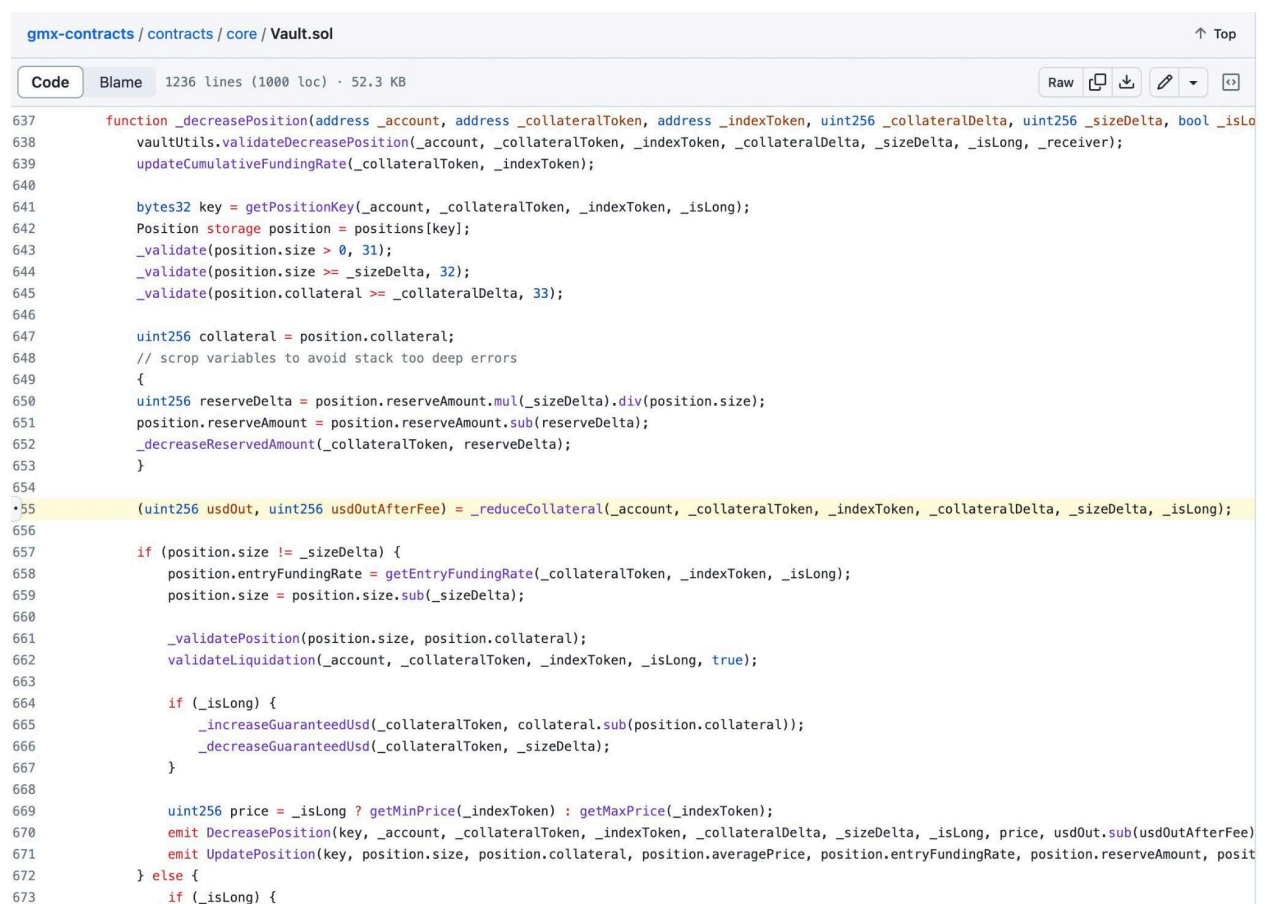
It's crucial to note that this **LONG position on metaYT** isn't typical. Whereas standard LONG positions draw upon the market value of metaYT to gauge profit or loss, in this setup, the focus shifts. The investor doesn't concern themselves with the profitability of this position unless they decide to liquidate their portfolio.

Targeting a locked **PT yield rate of 12.5%** on December 25, 2024, the calculation starts by deducing that PT should ideally be valued at **0.9 ETH**. If the market sees PT at **0.88 ETH**, then the system “**allows**” the long metaYT position to **gain 0.02 ETH** to bridge the

gap. Conversely, if PT is valued at **0.91 ETH**, this strategy “allows” the **LONG position to incur a loss of 0.01 ETH**. In both scenarios, the overarching aim is to ensure that the portfolio's total yield is adjusted and **fixed at 0.1 ETH**.

Considering these scenarios, a curious question arises: how does this system possess the ability to 'influence' or 'dictate' whether and how much the long metaYT position gains or loses? Shouldn't it be determined by the market value of metaYT? Is there some underlying mechanism or power that allows for such control?

The **key** lies in tweaking the GMX's getPrice function, as we learn from it, as the following flow when decreasing position and calculating the profit or loss.



```
gmX-contracts / contracts / core / Vault.sol
Code Blame 1236 lines (1000 loc) · 52.3 KB
Raw Copy Download Edit

637 function _decreasePosition(address _account, address _collateralToken, address _indexToken, uint256 _collateralDelta, uint256 _sizeDelta, bool _isLong, address _receiver) {
638     vaultUtils.validateDecreasePosition(_account, _collateralToken, _indexToken, _collateralDelta, _sizeDelta, _isLong, _receiver);
639     updateCumulativeFundingRate(_collateralToken, _indexToken);
640
641     bytes32 key = getPositionKey(_account, _collateralToken, _indexToken, _isLong);
642     Position storage position = positions[key];
643     _validate(position.size > 0, 31);
644     _validate(position.size >= _sizeDelta, 32);
645     _validate(position.collateral >= _collateralDelta, 33);
646
647     uint256 collateral = position.collateral;
648     // scrop variables to avoid stack too deep errors
649     {
650         uint256 reserveDelta = position.reserveAmount.mul(_sizeDelta).div(position.size);
651         position.reserveAmount = position.reserveAmount.sub(reserveDelta);
652         _decreaseReservedAmount(_collateralToken, reserveDelta);
653     }
654
655     (uint256 usdOut, uint256 usdOutAfterFee) = _reduceCollateral(_account, _collateralToken, _indexToken, _collateralDelta, _sizeDelta, _isLong);
656
657     if (position.size != _sizeDelta) {
658         position.entryFundingRate = getEntryFundingRate(_collateralToken, _indexToken, _isLong);
659         position.size = position.size.sub(_sizeDelta);
660
661         _validatePosition(position.size, position.collateral);
662         validateLiquidation(_account, _collateralToken, _indexToken, _isLong, true);
663
664         if (_isLong) {
665             _increaseGuaranteedUsd(_collateralToken, collateral.sub(position.collateral));
666             _decreaseGuaranteedUsd(_collateralToken, _sizeDelta);
667         }
668
669         uint256 price = _isLong ? getMinPrice(_indexToken) : getMaxPrice(_indexToken);
670         emit DecreasePosition(key, _account, _collateralToken, _indexToken, _collateralDelta, _sizeDelta, _isLong, price, usdOut.sub(usdOutAfterFee));
671         emit UpdatePosition(key, position.size, position.collateral, position.averagePrice, position.entryFundingRate, position.reserveAmount, position.liquidationPrice);
672     } else {
673         if (_isLong) {
```


Code Blame 1236 lines (1000 loc) · 52.3 KB

Raw Copy Download Edit View

```
992     function _reduceCollateral(address _account, address _collateralToken, address _indexToken, uint256 _collateralDelta, uint256 _sizeDelta, bool _isLo
993     bytes32 key = getPositionKey(_account, _collateralToken, _indexToken, _isLong);
994     Position storage position = positions[key];
995
996     uint256 fee = _collectMarginFees(_account, _collateralToken, _indexToken, _isLong, _sizeDelta, position.size, position.entryFundingRate);
997     bool hasProfit;
998     uint256 adjustedDelta;
999
1000     // scope variables to avoid stack too deep errors
1001     {
1002     (bool _hasProfit, uint256 delta) = getDelta(_indexToken, position.size, position.averagePrice, _isLong, position.lastIncreasedTime);
1003     hasProfit = _hasProfit;
1004     // get the proportional change in pnl
1005     adjustedDelta = _sizeDelta.mul(delta).div(position.size);
1006     }
1007
1008     uint256 usdOut;
1009     // transfer profits out
1010     if (hasProfit && adjustedDelta > 0) {
1011         usdOut = adjustedDelta;
1012         position.realisedPnl = position.realisedPnl + int256(adjustedDelta);
1013
1014         // pay out realised profits from the pool amount for short positions
1015         if (!_isLong) {
1016             uint256 tokenAmount = usdToTokenMin(_collateralToken, adjustedDelta);
1017             _decreasePoolAmount(_collateralToken, tokenAmount);
1018         }
1019     }
1020
1021     if (!hasProfit && adjustedDelta > 0) {
1022         position.collateral = position.collateral.sub(adjustedDelta);
1023
1024         // transfer realised losses to the pool for short positions
1025         // realised losses for long positions are not transferred here as
1026         // _increasePoolAmount was already called in increasePosition for longs
1027         if (!_isLong) {
1028             uint256 tokenAmount = usdToTokenMin(_collateralToken, adjustedDelta);
1029             _increasePoolAmount(_collateralToken, tokenAmount);
1030         }
1031     }
1032 }
```

Code Blame 1236 lines (1000 loc) · 52.3 KB

Raw Copy Download Edit View

```
936     function getDelta(address _indexToken, uint256 _size, uint256 _averagePrice, bool _isLong, uint256 _lastIncreasedTime) public override view returns
937     _validate(_averagePrice > 0, 38);
938     uint256 price = _isLong ? getMinPrice(_indexToken) : getMaxPrice(_indexToken);
939     uint256 priceDelta = _averagePrice > price ? _averagePrice.sub(price) : price.sub(_averagePrice);
940     uint256 delta = _size.mul(priceDelta).div(_averagePrice);
941
942     bool hasProfit;
943
944     if (_isLong) {
945         hasProfit = price > _averagePrice;
946     } else {
947         hasProfit = _averagePrice > price;
948     }
949
950     // if the minProfitTime has passed then there will be no min profit threshold
951     // the min profit threshold helps to prevent front-running issues
952     uint256 minBps = block.timestamp > _lastIncreasedTime.add(minProfitTime) ? 0 : minProfitBasisPoints[_indexToken];
953     if (hasProfit && delta.mul(BASIS_POINTS_DIVISOR) <= _size.mul(minBps)) {
954         delta = 0;
955     }
956
957     return (hasProfit, delta);
958 }
959
```


gmx-contracts / contracts / core / Vault.sol

Code

Blame

1236 lines (1000 loc) · 52.3 KB

```
761     function getMaxPrice(address _token) public override view returns (uint256) {  
762         return IVaultPriceFeed(priceFeed).getPrice(_token, true, includeAmmPrice, useSwapPricing);  
763     }  
764  
765     function getMinPrice(address _token) public override view returns (uint256) {  
766         return IVaultPriceFeed(priceFeed).getPrice(_token, false, includeAmmPrice, useSwapPricing);  
767     }
```

GMX operates using an oracle quotation model. Even if one were to long ETH to the extent of occupying GMX's vault, as long as ETH's price remains stable across the broader secondary market, the position remains neutral. The true genius is in how GMX determines the profit or loss for a position. By fetching the current ETH price, P , using the `getPrice` method and comparing it against `position.averagePrice`, a differential profit or loss per position unit is determined.

The **magic** intertwines with **Meta Earn**'s methodology: both GMX and our approach focus on differentials rather than absolute values. At any point t , the market value of YT, $Y(t)$, is known. Also, the distinction between $Y(t)$ and $Y(0) - rt/365$ is equally transparent since at any time t , $Y(0)$, r , and t are all known. This differential, δ , becomes our guiding light.

To facilitate this, **Meta Earn**:

- Initiate a **LONG** position on **1 metaYT** on our Meta Defender perpetual marketplace.
- While clearing this hedging position at moment t , use the modified `getPrice` function (which returns $\delta + \text{position.averagePrice}$ rather than metaYT's real-time price).

In this manner, the difference between $P(t)$ and $P(0) + \frac{rt}{365}$ is precisely offset by the profit or loss of the metaYT position. This is more accurate than directly using the market price of metaYT.

This manipulation guarantees a fixed return, much like a traditional finance's demand deposit, but with the flexibility of anytime opt-out.

GMX's oracle system is instrumental here. We've curated two oracle logics for varied scenarios:

- **For regular LONG or SHORT positions**, Pendle's oracle module calculates the YT price since $YT = \text{underlying} - PT$.
- **For special LONG positions** geared towards fixed yields, we compute the target price, decide the fate of the LONG position (profit or loss), and then engage the oracle to "push" the calculated outcome.

Both types of LONG positions coexist on the same perpetual market. This hybrid structure allows metaLP providers to gain from both yield perp traders and Pendle earn users. The endgame is a Meta Earn strategy – a fixed rate that remains unshaken by time.

Meta Earn not only enriches the Pendle ecosystem by promoting transactions on Pendle Swap but also benefits the inner workings of meta defender. A significant advantage is that the perpetual market mentioned earlier and this product utilize the same liquidity pool. This ensures that liquidity isn't dispersed, maintaining a concentrated and efficient pool for users.

In conclusion, the decentralized finance landscape is continually evolving, and fixed yield products like these are a testament to the innovation that platforms like Pendle bring to the table. By understanding and harnessing these tools, investors can navigate the DeFi realm with increased confidence and strategic finesse.

3. Capital Efficiency: Redeployable LP

Generally, LPs can only be in one protocol at a time, but is that really the case? Meta Defender's derivatives market redeploys a portion of ETH held by its LP into yield markets that have minimal Impermanent Loss (IL), which is Pendle AMM, to dramatically enhance capital efficiency. It can also be integrated with yield-boosting protocols such as Penpie or Equilibria to increase returns.

At the heart of the Meta Defender LP (metaLP) system lies an innovative approach to handling impermanent loss (IL), aptly termed as the 'cooldown period'. But what does this entail, and why is it integral to our protocol?

The nuances of IL in platforms like Pendle emerge largely due to the short-term volatility of PT prices. Essentially, the briefer the liquidity's stay, the higher its exposure to price swings, which in turn, escalates IL.

Meta Defender LP introduces a remedy: the cooldown period. It's a designated span of three months during which liquidity, once added to the platform, is held stable. By ensuring assets are locked in for this duration, we can confidently redeploy a portion of

them into Pendle since we capitalize on a key insight: the longer the liquidity's tenure in Pendle, the lower its susceptibility to IL. By the end of a full quarter, the natural appreciation of the LP position, driven by PT's intrinsic growth, should, in most cases, offset any IL that might have occurred at any interim time point.

Thanks to the unique algorithmic mechanism of Meta Defender, the ETH redeployed to Pendle still counts as part of the Meta Defender's market depth. Importantly, this redeployment doesn't reduce the amount of ETH available in Meta Defender's market-making pool.

Meta Defender will also employ a dynamic gauge to adjust the redeployable ratio—e.g., the amount of ETH in meta LP to be deployable into Pendle—to ensure there's immediate ETH liquidity when a user closes leverage positions.

The protocol keeps 5% of the additional fees and rewards earned, while the remaining are distributed among the LPs. Essentially, through redeployable LP, a liquidity provider can provide liquidity to two different protocols (e.g., Meta Defender derivatives market and Pendle yield market) at the same time, while earning additional fees and rewards along the stack, with minimal risks due to the homogeneous nature of the assets in LP and minimal IL.

Ultimately, the integration of redeployable LP is a strategic synergy between Meta Defender and Pendle, forging a win-win partnership. By allowing liquidity providers to simultaneously engage in two distinct protocols, this innovation not only bridges the capabilities of both platforms but also amplifies their combined capital efficiency. Such collaboration reflects the evolving landscape of decentralized finance, where protocols can leverage each other's strengths, fostering mutual growth and maximizing benefits for their users.

4. The Role of our DEF Token - Focused on Mining

During transactions, users are awarded through mining. Whether a user's position in Meta Defender perpetual contract results in profit or loss, they are rewarded with **DEF** tokens based on the size of their position and the duration for which they hold it.

This concept was initially introduced by Compound in the form of lending mining. Every borrower receives **COMP** rewards. With the daily rising value of **COMP** token, many eagerly borrow, sometimes even overlooking borrowing cost.

Thus, with the introduction of the mining mechanism, participation surged. Users, in pursuit of rewards, purchase YT and trade with metaYT. As a result, the Pendle swap collects swap fees, and Pendle's liquidity providers (LPs) gain profit.

Currently within the Pendle protocol, LPs predominantly obtain their profits from the **PENDLE** token, contributing to 80% of their revenue, while swap fees make up less than 20%. This distribution pattern suggests that the transaction volume on Pendle has yet to be fully explored. A broader awareness of the charm and potential of yield trading is still emerging among the community. To continuously engage and attract liquidity providers, the incentives provided by the **PENDLE** token remain crucial.

Recognizing the potential challenges brought about by an unchecked influx of **PENDLE** tokens, we believe in the importance of equilibrium. Our goal is to sustainably incentivize our LPs while also prudently controlling the emission rate of **PENDLE** tokens. This ensures that the token's circulation remains steady, preventing undue market saturation and upholding the health and stability of our ecosystem. If swap fees were to increase, the reliance on **PENDLE** tokens to incentivize users would decrease. In this way, LPs would earn more sensibly as market makers, collecting transaction fees.

So, what's the purpose of the mined **DEF** token? It's not just a token without inherent value.

By reducing the dependency on **PENDLE** token rewards, half of the saved amount can be used to incentivize Meta Defender. Those who mine DEF token and stake it as **veDEF** token will receive those incentives and can earn a portion of the Meta Defender protocol revenue. Staking as **veDEF** tokens to receive rewards from Pendle is immensely beneficial for **DEF** tokens.

In summary, while Pendle didn't incur additional costs, the trading activity was revitalized. **DEF** tokens gained value empowered by Pendle. Its lower circulation ensures easy market value management, uplifting the overall project valuation. This scenario presents a win-win-win for Pendle, for Meta Defender, and for investors.

In Conclusion,

In the ever-evolving landscape of DeFi, collaboration and synergy remain integral to the growth and sustainability of our ecosystems. As we journey forward with our innovative offerings, the foundation of mutual respect, shared vision, and the pursuit of excellence

underpins our endeavors.

Within this collaborative framework, we are proud to introduce four pivotal solutions tailored for the Pendle ecosystem:

1. LST yield perpetual trading
2. Meta Earn, a fixed yield product without lock-up period
3. Efficient Redeployable LPs
4. A mining mechanism to stimulate transactions without incurring additional costs.

We look forward to potential discussions, collaboration opportunities, and the exciting road ahead.

Warm regards,

Meta Defender Team