Technical Documentation Splyce Decentralized ETFs (dETFs) on Solana

1. Introduction to Splyce Decentralized ETFs (dETFs)

Overview

Splyce Decentralized Exchange-Traded Funds (dETFs) represent a novel financial primitive on the Solana blockchain, enabling the creation and management of on-chain, decentralized exchange-traded funds. The primary goal of the technology is to tokenize diversified baskets of assets or complex investment strategies, providing users with access to them through a single, liquid token.

Rationale for Solana

The choice of Solana as the base blockchain for dETFs is driven by its key advantages: high throughput, low transaction fees, and a dynamically developing DeFi ecosystem. This allows for the creation of complex and cost-effective financial products. The strategic decision to build on Solana also provides access to the unique protocols of this ecosystem, such as the Meteora DEX, the Kamino Lend lending platform, the Drift futures protocol, and liquid staking tokens like JupSOL Integration with these protocols is an integral part of the liquidity and composability strategy for Splyce dETFs, distinguishing them from solutions on other blockchains.

Core Concept

At its core, a dETF represents a share in a pool of assets managed by a set of connected strategies. This model is similar to traditional ETFs but implemented entirely on smart contracts. Users deposit a base asset (e.g., USDC) and receive a dETF token in return, the price of which reflects the aggregate value of the assets under management. The Splyce dETF technology builds on the Solana blockchain and the Anchor framework. Positioning the product specifically as a "dETF," rather than just a "vault," emphasizes the focus not only on possibilities of the yield generation but also on providing diversified access to various asset classes and tracking indices, although the implementation of yield-bearing strategies is also possible and is one of the key development directions.

2. Core Architecture

The Splyce dETF architecture is designed as a modular and extensible system on the Solana blockchain, consisting of several key interacting components.

2.1 dETF Program

The core of the system is the main dETF program (tokenized vault), developed using the Anchor framework for Solana. This program acts as the central hub coordinating all major operations:

- **Deposit Management:** Accepting assets (e.g., USDC) from users.
- **Withdrawal Request Processing:** Managing the process of returning assets to users.
- **Share Price Calculation:** Determining the value of one dETF share based on the total value of assets under management.
- **Share Token Minting and Burning:** Minting share tokens upon deposit and burning them upon withdrawal.
- **Interaction with Strategies:** Allocating funds to strategies and receiving reports from them on asset values.

High-Level Architecture

- *Central Element:* dETF Program (Solana Program).
- *Interacting Parties:*
  - Users: Interact via Deposit/Withdraw Request.
  - Strategy Programs: Receive funds (Allocate), return funds (Withdraw), report value (Report).
  - Processing Bot: Initiates Allocate, Report Trigger, Withdrawal Processing.

2.2 dETF Strategies

Strategies are separate Solana programs responsible for the specific placement and management of funds allocated from the main dETF program. Each strategy implements a specific investment logic:

- Investing in specific assets on Solana DEXs (e.g., Meteora, Orca, Raydium).
- Providing liquidity.
- Lending (e.g., via Kamino).
- Staking/Restaking (e.g., using LSTs like JupSOL).
- Cross-chain operations using bridges (e.g., Wormhole, Chainlink CCIP).
- Investing in RWAs (e.g., via Huma Finance PST).

Strategies must adhere to a specific interface for interaction with the dETF program, including functions to receive funds (allocate), return funds (withdraw), and provide a report on the current value of assets under management (report). Funds transferred

to a strategy are considered its debt (debt) to the dETF. Funds within a strategy can be temporarily unallocated (idle) or actively used (debt in the context of the strategy). Importantly, strategies are customizable and can be modified independently.

2.3 Strategy Managers

Strategy Managers are entities (individuals, teams, DAOs, or automated systems) responsible for developing, deploying, and operationally managing the logic of a specific strategy. They determine how the strategy will use its allocated funds to achieve its goal (e.g., selecting assets to buy, managing positions, rebalancing).

2.4 Processing Bot

The Processing Bot is an external (off-chain) component that automates key operational processes in the system:

- **Allocation (allocate):** Moves idle funds from the dETF program to connected strategies according to the established allocation logic.
- **Report Triggering (report triggering):** Periodically or event-driven, requests reports from strategies about their current asset value (value on strategy).
- **Withdrawal Processing:** Coordinates the process of returning funds from strategies back to the dETF program to satisfy user withdrawal requests. The bot interacts with strategies (or their managers) to initiate the withdrawal of funds from active use.

Currently, the bot operates centrally, but a transition to a decentralized model with a pool of third-party bots is planned to enhance system resilience and reliability. This decentralization, while improving fault tolerance, will require the development of complex coordination mechanisms and economic incentives for bot network participants.

2.5 Interactions and Flows

The architectural separation between the main dETF program and strategies provides modularity and flexibility. However, this same modularity creates a critical dependency on the correctness and timeliness of the strategy interface, especially the report function. The dETF program relies solely on data provided by strategies to calculate TotalManagedAssets, which is the basis for determining the share price. Any errors, delays, or manipulations in strategy reports (e.g., due to oracle issues used within the strategy, or logic errors) will directly distort the dETF share price calculation, negatively affecting all users during deposits or withdrawals. This

highlights the importance of thorough auditing and monitoring of strategies.

The Processing Bot is a key element of the operational infrastructure. Its performance directly impacts capital allocation efficiency and withdrawal processing speed. The current centralized model represents a single point of failure and a potential bottleneck. Transitioning to a decentralized network of bots addresses the single point of failure but introduces new complexities related to the need for reliable economic incentives, Sybil attack protection, and mechanisms for coordinating independent bots.

Fund Allocation Flow

- *Start:* User deposits USDC into the dETF Program.
- *Step 1:* USDC is temporarily in an idle state in the dETF Program.
- *Step 2:* Processing Bot initiates allocate.
- *Step 3:* dETF Program transfers USDC to the Strategy Program.
- *Step 4:* Strategy Manager, via the Strategy Program, deploys USDC (e.g., buys assets on DEX).
- *Result:* Funds are working within the strategy.

Strategy Reporting Flow

- *Initiator:* Strategy Manager or Processing Bot triggers the report process.
- *Step 1:* Strategy Program calculates the current value of its assets (may use Oracles, e.g., Pyth or Chainlink).
- *Step 2:* Initiator calls *report()* function on the Strategy Program that updates its internal state.
- *Step 3:* Initiator calls the *process_report()* function on the dETF Program that updates its internal state (TotalManagedAssets).
- *Result:* The total value of assets in the dETF is updated.

3. dETF Share Token Mechanics

3.1 Tokenization

A user's deposit into a Splyce dETF is tokenized as a standard SPL token on Solana. This token ("share token") represents the user's proportional share in the total pool of assets managed by the dETF and its strategies. The SPL standard ensures the share token's liquidity and compatibility with most protocols and wallets in the Solana ecosystem, a key factor for DeFi composability. Conceptually, this is similar to the ERC4626 standard on EVM blockchains, which also represents tokenized shares in

vaults, but the Solana implementation accounts for its specific architecture (account model, transaction processing).

3.2 Share Price Calculation

The share price is the key parameter determining how many share tokens a user receives upon deposit and how much base asset is returned upon withdrawal. It is calculated directly by the dETF program.

Basic Formula:

The share price (SharePrice) is determined as the ratio of the total value of assets under management to the total number of outstanding shares:

`SharePrice = TotalManagedAssets / TotalSharesOutstanding`

Where:

- TotalManagedAssets: The total value of assets managed by the dETF. Calculated as the sum of the value of idle assets held directly in the dETF program and the aggregate value of assets (debt) held across all connected strategies (according to their latest report).
- TotalSharesOutstanding: The total number of dETF share tokens currently in circulation (already minted to users).

Share Minting (Deposit):

The number of shares (SharesToMint) issued to a user upon making a deposit (DepositAmount) is calculated using the formula:

`SharesToMint = DepositAmount / SharePrice`

Potential rounding errors during division are typically handled in favor of the vault (rounding down), which can lead to a slight increase in the value of existing shares, as observed in some vault implementations This mechanic should be clearly documented for transparency.

Share Burning (Withdrawal):

The amount of the base asset (AmountToReturn) returned to the user when burning a specific number of shares (SharesToBurn) is calculated using the formula:

`AmountToReturn = SharesToBurn * SharePrice`

3.3 Oracle Independence (at the dETF level)

It is important to emphasize that the dETF program itself *does not use* external price oracles to calculate SharePrice. The calculation is based solely on the internal state: the balance of idle assets and the values received from strategies via the report mechanism.

However, the **critical nuance** is that the accuracy of TotalManagedAssets, and consequently SharePrice, directly depends on the accuracy of the reports provided by the strategies. The strategies themselves *may* (and often will) use external oracles (e.g., Pyth or Chainlink) to determine the value of the assets they manage, especially if these are tokens with volatile prices. Thus, although the dETF program is formally independent of oracles, its key parameter (SharePrice) is indirectly dependent on the correct functioning of the oracles used by the strategies and the integrity of the strategies themselves.

An incorrect report from a strategy (due to an oracle failure, manipulation, or a bug) will lead to an incorrect SharePrice calculation in the main dETF program. This creates implicit trust in the strategies or requires the implementation of mechanisms for verifying and controlling their reporting.

The sharePrice may not always be up-to-date due to delay between price update intervals of the assets held by strategies, which means delay on update of TotalManagedAssets.

3.4 Relative Pricing

The absolute value of the SharePrice of a dETF representing, for example, an index, will likely differ from the price of the index itself in the traditional market. However, if the strategies correctly track the composition and weights of the index assets, the relative changes in SharePrice over time should correlate with the changes in the price of the underlying index. This correlation serves as confirmation that the dETF share price adequately reflects the value of its underlying assets.

4. User Workflows

4.1 Deposit Flow

The process of depositing funds into a dETF is designed to be as simple and fast as

possible for the user in terms of receiving a tokenized representation of their share.

- **User Action:** The user initiates a deposit transaction, sending a certain amount of the base asset (e.g., USDC) to the dETF program.
- **Instant System Action:** The dETF program immediately:
  1. Calculates the current SharePrice based on TotalManagedAssets and TotalSharesOutstanding.
  2. Computes the number of share tokens (SharesToMint) using the formula DepositAmount / SharePrice.
  3. Mints the calculated number of dETF share tokens and sends them to the user's wallet.
- **Subsequent (Asynchronous) Actions:**
  1. The funds deposited by the user initially arrive in the dETF program and are in an idle state.
  2. The Processing Bot, according to its logic and schedule, initiates the allocate process, moving these idle funds to one or more connected strategies.
  3. Strategies receive the funds and begin deploying them according to their internal logic (buying assets, providing liquidity, etc.). This process may take some time.

Thus, the user instantly receives confirmation of their contribution in the form of share tokens, but the actual deployment of their capital into work happens later and is managed by the protocol's automated systems.

Deposit Flow

- *Phase 1 (Synchronous):* User -> Sends USDC -> dETF Program (Calculates SharePrice, Mints Shares) -> User receives dETF Shares.
- *Phase 2 (Asynchronous):* dETF Program (holds idle USDC) -> Processing Bot (initiates Allocate) -> Strategy Program (receives USDC) -> Strategy Manager/Program (deploys USDC).

4.2 Withdrawal Flow

The withdrawal process is somewhat more complex than the deposit due to the potential need to retrieve liquidity from strategies.

- **User Action:** The user initiates a withdrawal request transaction (withdraw request), specifying either the number of share tokens (SharesToBurn) they wish to burn or the desired amount of the base asset (AmountToWithdraw) they want to receive. The user can also optionally specify a priority fee to expedite

processing.

- **Instant System Action:** The dETF program immediately:
  1. Determines the number of shares to burn (if AmountToWithdraw was specified, then SharesToBurn = AmountToWithdraw / SharePrice).
  2. Calculates the amount of base asset to return (AmountToReturn = SharesToBurn * SharePrice).
  3. Burns the specified number of share tokens (SharesToBurn) from the user's balance.
- **Subsequent (Asynchronous) Actions:**
  1. The dETF program checks if there are sufficient idle funds to satisfy AmountToReturn.
  2. **If idle funds are sufficient:** The dETF program sends AmountToReturn to the user. The process completes relatively quickly (depending on Solana network load).
  3. **If idle funds are insufficient:** The withdrawal request is placed in a queue.
     - The Processing Bot detects the request in the queue.
     - The Bot analyzes the presence of a priority fee. Requests with a fee are processed outside the main queue, which might run periodically (e.g., every N hours).
     - The Bot coordinates the process of recalling funds from strategies. It signals strategies (or their managers) about the need to return a specific amount of the base asset.
     - Strategies process the withdrawal request according to their logic (selling assets, exiting liquidity pools, closing positions). The speed of fund return depends on the liquidity of assets within the strategy and its operational cycles.
     - As funds return from strategies back to the dETF program, the Processing Bot aggregates them.
     - Once a sufficient amount is accumulated to satisfy the request (or part of it), the Bot sends the funds to the user.

Priority Fee: This optional fee, paid by the user in Solana's native token, incentivizes the Processing Bot (or the decentralized network of bots in the future) to process the withdrawal request immediately, rather than waiting for the next scheduled batch processing cycle. This allows users to receive funds faster, but at an additional cost.

Decentralization of Processing: The planned implementation of a network of third-party processing bots aims to eliminate the single point of failure, increase

censorship resistance, and potentially speed up withdrawal processing through parallelism. However, as mentioned earlier, this will require creating a robust system of incentives and controls for the bots.

The asymmetry between instant deposits and potentially delayed withdrawals is an important characteristic of the user experience. Transparently informing users about the status of their withdrawal requests, expected timelines, and the priority fee mechanism is crucial for managing expectations. The priority fee mechanism could create dynamics where users willing to pay more receive faster service, especially during periods of low liquidity or high system load, potentially disadvantaging users with less capital.

Withdrawal Flow

- *Phase 1 (Synchronous):* User -> Sends Withdrawal Request (specifying Shares/Amount, optionally Priority Fee) -> dETF Program (Calculates AmountToReturn, Burns Shares).
- *Phase 2 (Asynchronous - depends on idle funds):*
  - *Scenario A (Sufficient idle):* dETF Program -> Sends Funds to User.
  - *Scenario B (Insufficient idle):* dETF Program -> Places Request in Queue -> Processing Bot (Checks Priority Fee) -> Signals Strategies -> Strategies Return Funds -> dETF Program -> Processing Bot -> Sends Funds to User.

5. Fee Mechanics

The Splyce dETF protocol provides a flexible system for configuring fees at both the dETF level and the individual strategy level.

5.1 dETF Level Fees

The main dETF program can charge the following types of fees, configurable by the protocol administrator:

- **Deposit Fee:** A percentage of the deposited asset amount, charged *before* share tokens are minted. For example, with a 100 USDC deposit and a 0.1% fee, share calculation will be based on 99.9 USDC.
- **Withdrawal Fee:** A percentage of the returned asset amount, charged *after* calculating the amount to return based on burned shares. For example, if the calculated return amount is 1000 USDC and the fee is 0.2%, the user will receive 998 USDC. Similar concepts of entry/exit fees are discussed in the context of other protocols.

- **Performance Fee / Gain Fee:** A percentage charged only on the *increase* in the share price (SharePrice) since the user's deposit or since the last time this fee was charged. This fee is only applied in case of positive performance ("gain") and can be deducted upon withdrawal or periodically. The calculation mechanism requires tracking the cost basis for each user or batch of shares.

5.2 Strategy Level Fees

Each strategy connected to the dETF can have its own independent fee structure. The standard model described in the request suggests:

- **Overall Strategy Fee:** Can be charged as a percentage of the strategy's Assets Under Management (AUM) or as a percentage of the profit generated by the strategy.
- **Distribution:** This fee can be distributed between the Strategy Manager (as compensation for management) and the Splyce protocol treasury.
- **Condition:** Similar to the dETF level, the strategy fee can be configured to be charged only in case of the strategy's positive performance (gain).

5.3 Fee Flow

Collected fees (both at the dETF level and a portion of strategy fees) are directed to the Splyce protocol treasury or distributed otherwise according to the protocol's settings and tokenomics. Fees intended for Strategy Managers are sent to them. The priority fee for withdrawal (priority fee) goes to the Processing Bot operators.

The multi-layered fee structure (dETF + Strategy + Potential Priority Fee) means users must consider the cumulative impact of all charges on their net returns. Transparent disclosure of all possible fees is critically important. Implementing performance fees, while aligning incentives, requires an accurate and complex on-chain tracking mechanism to ensure fair calculation.

Table: Fee Summary

Fee Type	Level	Calculation Basis	Time of Charge	Recipient(s)	Configurable?
Deposit Fee	dETF	% of deposit amount	Deposit	Protocol Treasury	Yes

Withdrawal Fee	dETF	% of withdrawal amount	Withdrawal	Protocol Treasury	Yes
Performance Fee	dETF	% of share price gain	Profit Realization	Protocol Treasury	Yes
Strategy Fee	Strategy	% of AUM / Strategy Profit	Strategy Report/Ops	Strategy Mgr, Protocol Treasury	Yes (strategy)
Priority Fee	User	Fixed/Variable SOL fee	Withdrawal Request	Processing Bot(s)	Yes (by user)

6. Liquidity and Market Dynamics

Ensuring liquidity for dETF share tokens and integrating them into the broader DeFi ecosystem are key aspects of the Splyce strategy.

6.1 Liquidity Pools on DEXs

The plan includes creating liquidity pools on leading decentralized exchanges (DEXs) on Solana for each issued dETF token. A typical pool pair is the dETF token and a stablecoin (e.g., USDC). Examples of DEXs include Meteora, Orca, or Raydium. These pools provide a secondary market for dETF tokens, allowing users to buy and sell shares without directly interacting with the deposit/withdrawal mechanism of the main dETF program. Using advanced DEXs like Meteora with its Dynamic AMM Pools or DLMM could offer higher capital efficiency for liquidity providers.

6.2 Arbitrage Mechanism

The difference between the share price calculated by the dETF program (SharePrice, reflecting Net Asset Value - NAV) and the market price of the dETF token in the DEX liquidity pool creates a natural arbitrage opportunity.

- **Example 1: DEX Price Lower than NAV**
  - dETF SharePrice = 10 USD / 1 dETF (100 USD / 10 dETF)
  - Price in DEX pool = 8 USD / 1 dETF (80 USD / 10 dETF)

- **Arbitrageur action:**
  1. Buy 10 dETF on DEX for 80 USD.
  2. Request withdrawal from dETF, burn 10 dETF, and receive 100 USD (minus fees).
  3. **Profit:** 20 USD (minus fees and gas).
- **Example 2: DEX Price Higher than NAV**
- dETF SharePrice = 8 USD / 1 dETF (80 USD / 10 dETF)
- Price in DEX pool = 10 USD / 1 dETF (100 USD / 10 dETF)
- **Arbitrageur Action:**
  1. Deposit 80 USD into the dETF program, receiving 10 dETF.
  2. Sell 10 dETF on DEX for 100 USD (minus fees).
  3. **Profit:** 20 USD (minus fees and gas).

Arbitrageurs, exploiting these discrepancies, buy where it's cheaper and sell where it's more expensive, thereby pushing the DEX price closer to the dETF SharePrice (NAV). The efficiency of this mechanism depends on the depth of liquidity in the pool, transaction costs, and arbitrageur activity. Low liquidity or high costs can lead to persistent deviations of the DEX price from NAV.

Arbitrage Cycle

- *Scenario 1 (DEX < NAV):* Arbitrageur -> Buys dETF on DEX (cheap) -> Deposits dETF into dETF Program (Burns) -> Receives Base Asset (expensive) -> Profit.
- *Scenario 2 (DEX > NAV):* Arbitrageur -> Deposits Base Asset into dETF Program (Mints dETF cheap) -> Sells dETF on DEX (expensive) -> Profit.
- *Result:* Arbitrageur actions push the DEX price towards NAV.

6.3 DeFi Composability

The standard SPL nature of the dETF share token ensures compatibility with other protocols in the Solana DeFi ecosystem, opening up various use cases:

- **Lending:** dETF tokens can be used as collateral in lending protocols like Kamino Lend. Potential advantages of dETF as collateral include:
  - **Diversification:** Collateral represents a basket of assets, potentially less risky than collateral in a single volatile asset.
  - **Built-in Yield:** If the dETF uses yield-bearing strategies, the collateral value may grow over time, improving the Loan-to-Value (LTV) ratio. Kamino, with its risk-oriented features (Asset Tiers, eMode for correlated assets), might be particularly suitable for integration.

- **Futures and Derivatives:** dETF tokens can be used in derivatives protocols, such as Drift Protocol. Possible applications:
  - Use as margin collateral for trading futures, especially in Drift's cross-margin systems.
  - Creation of derivative products (e.g., perpetual swaps) on the dETFs themselves, allowing traders to speculate on their price.
- **Yield Aggregators and Structured Products:** dETFs themselves can become building blocks for more complex products created by other protocols.

Deep composability is a powerful tool, but it also links risks. If a dETF token is used as collateral in Kamino or Drift, any problems with the dETF (e.g., a sharp drop in SharePrice due to strategy failure or manipulation) could trigger cascading liquidations in these integrated protocols, propagating risk throughout the system.

6.4 Challenges with Yield-Bearing dETFs

Maintaining a fair price in DEX pools for dETFs that include complex yield-bearing strategies (e.g., SOLMATE) is a non-trivial task. The price in an AMM pool primarily reflects the supply and demand balance from recent trades. However, the true value (NAV) of a yield-bearing dETF constantly increases due to accrued yield. A simple AMM pool does not automatically account for this accrued yield.

As a result, the DEX price can systematically lag behind the NAV. More complex mechanisms are required to maintain parity:

- **Active Market Making:** Market makers are needed who constantly track the dETF's NAV (including accrued yield) and adjust their orders in the DEX pool so the price reflects the true value. This requires more sophisticated modeling than for static assets.
- **Yield-Aware Arbitrage:** Arbitrageurs must also consider accrued yield when calculating the profitability of their trades.

This creates a potential barrier for the effective launch of yield-bearing dETFs, as their successful trading on the secondary market may depend on the presence and activity of specialized market makers capable of correctly pricing the yield component.

7. Types of Splyce dETFs

The Splyce platform allows for the creation of various types of dETFs, differing in

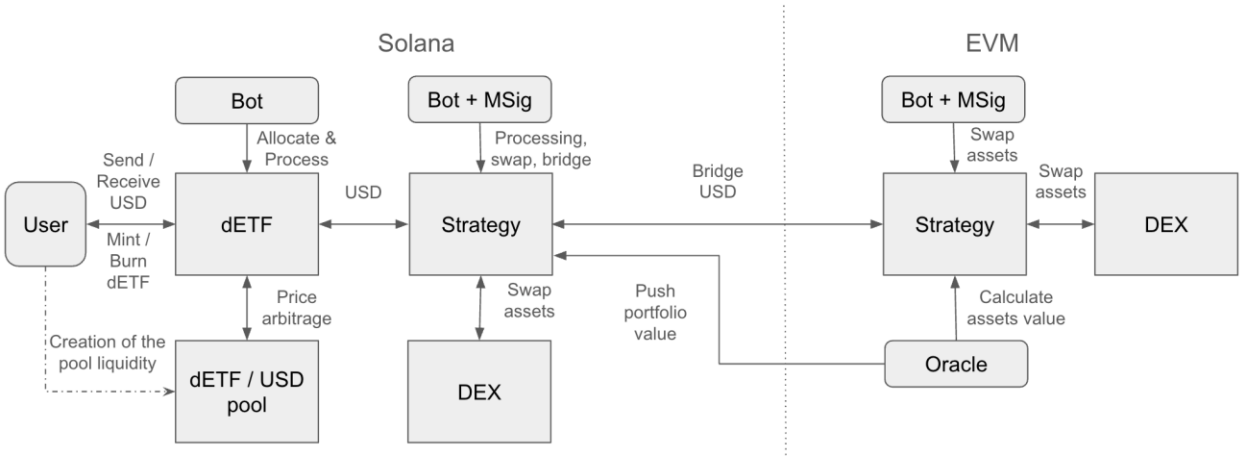
asset composition and strategy logic.

7.1 Index dETF (Example: WLFtrack)

- **Concept:** This type of dETF is designed to track a specific index or basket of assets, providing users with passive exposure to a diversified portfolio. WLFtrack, tracking the World Liberty Finance asset portfolio, is an example.
- **Strategy Types:**
  - **On-chain (Solana):** Strategies investing in assets that already have liquidity on Solana DEXs (e.g., Meteora, Orca, Raydium). They buy and hold these assets in proportions corresponding to the index.
  - **Cross-chain:** Strategies providing access to assets on other blockchains (e.g., Ethereum). They operate as follows:
    1. The base asset (e.g., USDC) from the dETF program on Solana is allocated to a cross-chain strategy on Solana.
    2. This strategy uses a bridge (e.g., Wormhole or Chainlink CCIP) to transfer the base asset to the target blockchain (e.g., Ethereum).
    3. A corresponding "mirror" strategy is deployed on the target blockchain, receiving the transferred funds.
    4. This strategy on the target blockchain buys the necessary assets via local DEXs (e.g., Uniswap) to reflect the index.
- **Asset Flow and Management:**
  - **Automation:** A specialized bot monitors fund arrivals on Solana strategies and periodically initiates bridging operations (both to the target chain and back for withdrawal processing).
  - **Access Control:** To enhance security, investment operations (swaps) within strategies (on both Solana and EVM) may have a two-tier control system:
    - **Small volumes:** Automatically processed by the strategy bot.
    - **Large volumes:** Require confirmation via a multisig mechanism before execution.
  - **Price Verification:** When executing swaps within strategies, oracles (e.g., Pyth, Chainlink) are used to verify the fairness of the DEX pool price and control slippage.
- **Bridging Costs:** Using cross-chain bridges involves fees, which are accounted for in strategy expenses and can impact the overall dETF fee or its performance.
- **Use of Aggregators (Temporary Solution):** In cases where direct integration with liquidity pools on target DEXs is difficult or liquidity is fragmented, strategies may temporarily use APIs of liquidity aggregators (e.g., Jupiter on Solana, 1inch

on EVM chains) to automate asset purchase/sale. The long-term goal is to transition to full on-chain integration via direct pool interactions. Using aggregators, while pragmatic initially, introduces dependency on their APIs and routing logic.

Creating cross-chain index dETFs involves significant complexity and dependencies. Bridge security, reliability of inter-chain messaging, gas costs on multiple blockchains, and the availability and accuracy of oracles on all involved chains are all critical factors. A failure in any of these components could disrupt the dETF's ability to track the index and report its value correctly.



Cross-chain Flow of Index dETF (WLFtrack)

- *Start:* User deposits USDC into dETF on Solana.
- *Step 1:* dETF allocates USDC to Cross-chain Strategy (Solana).
- *Step 2:* Strategy (Solana) uses Bridge (Wormhole/CCIP) to send USDC to EVM chain.
- *Step 3:* Strategy (EVM) receives USDC.
- *Step 4:* Strategy (EVM) uses DEX (Uniswap) to buy Target Asset (e.g., XYZ token). (Interaction with Oracle for price check; Multisig check for large amounts).
- *Result:* dETF holds asset XYZ via cross-chain mechanism. Reverse flow for withdrawal is similar.

7.2 dETF with Smart Strategies

- **Concept:** This type of dETF allows for the implementation of more complex strategies that actively interact with various DeFi protocols to generate yield or achieve specific investment objectives.

- **Logic Examples:**
  - **Lending/Borrowing:** Strategies can place assets in lending protocols (e.g., Kamino) or use borrowed funds for leverage.
  - **Liquid Staking/Restaking:** Using liquid staking tokens (LSTs), such as JupSOL, to earn staking and MEV rewards, with the possibility of further using LSTs in other DeFi protocols.
  - **Derivatives Trading:** Strategies can open positions in futures markets (e.g., via Drift).
  - **Yield Farming:** Providing liquidity in various pools to earn trading fees and emission rewards.

7.3 Hybrid dETF (Example: SOLMATE)

- **Concept:** dETFs that combine assets or yield sources from different categories, such as mixing crypto-assets and real-world assets (RWA) or various yield strategies. SOLMATE is a prime example.
- **SOLMATE Breakdown:**
  - **Goal:** Provide sustainable USDC yield on Solana, especially considering expected reductions in SOL staking rewards due to SIMD-228.
  - **Components:**
    1. **JupSOL:** A liquid staking token from Jupiter, representing staked SOL and earning yield from staking and MEV (Maximal Extractable Value).
    2. **Huma Finance PST (PayFi Strategy Token):** Huma Finance's upcoming liquid LP token, representing a share in a pool generating yield from financing real-world payment flows (Real-World Asset - RWA). Huma offers both institutional (permissioned) pools and new permissionless pools (Huma 2.0).
  - **Yield Sources:** SOLMATE earns yield from both SOL staking and MEV (via JupSOL) and from fees for financing real payment operations (via Huma PST).
  - **Platform:** Uses the Splyce dETF infrastructure to combine these diverse yield sources into a single tokenized product.
  - **Composability:** Integration of the SOLMATE token with leading Solana DeFi protocols like Jupiter (for swapping) and Kamino (for use as collateral) is planned.
  - **Value Proposition:** Offers diversification (crypto staking + RWA) and a potentially more stable yield profile, less dependent on crypto market cycles, as RWA yield is linked to real economic activity.

Hybrid dETFs like SOLMATE are at the forefront of innovation but concentrate risks

from various domains. Their stability and yield depend on the reliability of LST protocols (smart contract risks, depeg), RWA protocols (borrower credit risk, accuracy of real asset valuation), and the Splyce dETF infrastructure itself. Accurately valuing such a complex product can be challenging.

8. Technical Implementation Details

Splyce dETFs are built on the Solana blockchain using the Anchor framework. Solana is chosen for its performance and low fees, while Anchor simplifies the development of secure and efficient programs (smart contracts) on Solana by providing convenient abstractions and tools for working with accounts, data serialization, and access control.

8.1 Key Interfaces (Conceptual)

- **dETF Program (Core functions for user and bot):**
  - deposit(amount: u64): Deposit base asset.
  - request_withdraw(shares_amount: u64, priority_fee: u64): Request withdrawal with optional priority fee.
  - get_share_price(): Get current share price.
  - get_total_assets(): Get total assets under management.
  - *(Admin/Bot functions):* add_strategy(strategy_pubkey: Pubkey), set_fees(...), trigger_allocation(), process_withdrawal_queue().
- **Strategy Interface (Functions called by dETF program or bot):**
  - allocate(amount: u64): Receive funds from dETF.
  - withdraw(amount: u64): Return funds to dETF upon request.
  - report_value(): Initiate calculation and send asset value report to dETF.
  - get_value(): Get current value of strategy assets.
  - get_debt(): Get current debt amount owed to dETF.

It should be stressed that this is a *conceptual* representation of the interfaces. The actual function signatures, data types, and invocation mechanisms may differ in the actual implementation.

9. Conclusion

Summary

Splyce dETFs on Solana represent an advanced technology offering a flexible, composable, and decentralized way to access tokenized investment strategies. They combine the benefits of traditional ETFs with the capabilities of DeFi, creating a new

class of financial instruments on the blockchain.

Key Features

- **Tokenized Shares:** Investment representation as standard SPL tokens ensures liquidity and compatibility.
- **Strategy Diversity:** Support for a wide range of strategies – from passive index-tracking to complex yield-bearing and hybrid (including RWA).
- **Composability:** Seamless integration with leading DeFi protocols on Solana (lending, DEXs, derivatives).
- **Internal Price Calculation:** Share price calculation mechanism is independent of external oracles at the dETF level but relies on the accuracy of reports from strategies.
- **Solana Optimization:** Leveraging Solana's advantages and the Anchor framework for performance and efficiency.

Outlook

Further development of the protocol includes the planned decentralization of processing bots to enhance system resilience. Expansion of the available dETF spectrum is also expected through the addition of new strategy types and integrations with other protocols and asset classes, strengthening Splyce's position as an innovative platform for on-chain investments.