

Introduction

What is Saber?

Saber is an automated market maker and liquidity pool on [Solana](#) designed for extremely efficient trading between similarly priced (pegged) assets, without an opportunity cost.

This trading activity results in fees for providers of liquidity, resulting in a safer, lower-risk staking opportunity representing real world transaction volume.

For more information about why Saber is useful, see: [Why Saber?](#)

Asset Types

Saber generally allows for trading between assets that mean revert in price. There are several assets that Saber specializes in:

- **USD Stablecoins.** There are hundreds of dollar-based stablecoins in existence, ranging from dollar-backed stablecoins like [USDC](#) and [Tether](#), decentralized stablecoins such as [UST](#), and native Solana stablecoins such as [Cashio](#).
- **Bridged assets.** There are a large number of bridges that lead to Solana, and Saber is the primary liquidity source to exchange between the different bridged assets. One example is the [renBTC-BTC](#) pool, which allows swapping between the [Ren](#) and [FTX](#) versions of Bitcoin. Many stablecoins are also bridged assets; Saber is the primary way for Solana users to navigate between different chains.
- **Staking derivatives.** Saber is the largest venue for trading SOL staking derivatives such as [Marinade SOL](#) and [Lido SOL](#).

Disclaimer

All claims, content, designs, algorithms, estimates, roadmaps, specifications, and performance measurements described in this project are done with the author's best effort. It is up to the reader to check and validate their accuracy and truthfulness. Furthermore, nothing in this project constitutes a solicitation for investment.

Why Saber?

It is a misconception that Saber is competing with general AMMs and orderbooks. They are not the same and go after different markets.

Saber is meant to be a place for low risk staking on large transaction volume. It encourages high volume by having the best pricing for swaps (encouraging arbitrage bots to trade), and it is extremely composable with other DeFi applications on Solana, reducing the opportunity cost for the otherwise low yields.

Zero Impermanent Loss

Saber's StableSwap algorithm makes the assumption that assets in a pair will converge to the same price. As such, it does not have impermanent loss in the way that constant product AMMs do.

Note that there is still a risk of prices diverging from equilibrium: if one asset in the pool "de-pegs" (that is, drops or skyrockets in price permanently), a liquidity provider will experience impermanent loss. One should take care to research the underlying assets they are investing in.

Concentrated Liquidity

There is an age-old debate of whether or not automated market makers are more efficient in providing liquidity than orderbooks.

There are two properties to measure the efficiency of liquidity: spread and depth. Spread refers to the difference between the bid (selling) price and the ask (buying) price, and depth refers to the total amount of volume that can be moved for a particular percentage of price decrease.

Constant product market makers like Uniswap V2 are very inefficient in liquidity provision, as they spread liquidity out over a large curve. For example: if \$1,000,000 is allocated to the USDT-USDC pair evenly, swapping 10,000 USDT to USDC results in a new price of \$1.04 USDC.

Saber is different. The algorithm knows that USDT and USDC should be the same price, so you can expect a virtually zero change in price. As a result, liquidity providers are able to charge higher fees and more profit.

Zero Opportunity Cost

Unlike other concentrated liquidity AMM positions, Saber LP tokens themselves are able to be used in all sorts of places ranging from lending markets (see [Port](#)) to collateralizing other stablecoins (see [Parrot](#)). This allows for Saber to be much more composable than orderbooks, as orderbook positions cannot directly be used as assets within other protocols. This greatly reduces (and in the long run, eliminates) the opportunity cost of deploying capital, providing a passive "risk-free" rate of return to the decentralized financial ecosystem.

This means that the goal of Saber as a protocol is *not* necessarily to maximize fees per LP token-- it is just the closest thing to a risk free rate of return on Solana, relative to an LP's underlying assets. A more important goal would be to maximize the protocol's revenue, meaning that total volume is more important than volume/TVL ratio.

Ecosystem

Aggregators

- [Tulip](#)
- [Sunny Aggregator](#)

Lending

- [Apricot Finance](#)

If your project isn't listed here, [submit a pull request on GitHub!](#)

Saber Public Goods

The Saber team has built or been a major contributor to a wide variety of products and services on the Solana blockchain.

Some applications listed here are closed source. If you would like to contribute to a closed source project, please contact the Saber team at team@saber.so.

Developer Tools

Continuous Integration/Continuous Deployment

- [Goki CLI](#) - A CLI for deploying programs on the Solana blockchain, with extra safety checks and checksums.
- [DeployDAO Migrator](#) - (*unmaintained*) A tool for versioning programs and program upgrades.
- [Solana Program Registry](#) - An open registry for transparent, verified builds of Solana programs, binary checksums, and Anchor IDLs. Built on GitHub Actions.
- [Anchor.so](#) - A Solana account explorer and program simulator, built on Anchor.
- [WalletKit by Goki](#) - 🗝️ Wallet connector for Solana dApps.
- [Captain](#) - (*unmaintained*) 🔑 Version control and key management for Solana programs.

Frontend/JavaScript

- [Sail](#) - 🚢 A React library for Solana account management and transaction processing.
- [solana-contrib](#) - General Solana utility functions.
- [use-solana](#) - Solana React library.
- [token-utils](#) - SPL Token arithmetic and types.
- [anchor-contrib](#) - TypeScript client for Anchor programs.
- [chai-solana](#) - Chai test helpers for Solana programs.

Python

- [solana-py](#) - Solana Python SDK.
- [pyserum](#) - Serum DEX client library.

Smart Contract Libraries

- [Vipers](#) - 🐍 Assorted checks and validations for writing safer Solana programs.
- [U128](#) - U128 helpers for Solana programs to make programs more efficient.

Miscellaneous

- [Token List Builder](#) - Tools for merging token lists.
- [saber-overlay](#) - A Nix overlay containing various Solana and crypto CLI tools.

Treasury Management

- [Goki Smart Wallet](#) - Multisig Solana wallet with Timelock capabilities
- [Tribeca](#) - 🏢 An open standard and toolkit for launching DAOs on Solana.
- [Venko](#) - Streaming payments and token release.
- [Quarry Mint Wrapper](#) - A mechanism for allowing multiple addresses to mint tokens.
- [Merkle Distributor](#) - 📦 A smart contract that distributes a balance of tokens according to a Merkle root.

Governance

- [Permalock](#) - Tribeca vote escrows with infinite lockup durations.
- [SAVE: Simple Agreement for Vote-Escrowed tokens](#) - A SAFE-like derivative which enforces that tokens must be locked for a specific period of time.
- [Quarry Gauge](#) - A veToken-based liquidity mining rewards allocator.
- [Tribeca Registry](#) - A registry of DAOs on the Tribeca protocol.
- [Snapshots](#) - Voting Escrow Snapshots: Historical snapshots of previous voting escrow balances.

DeFi

- [spl-token-swap \(Stable Curve\)](#) - Stable token swap curve used by Orca, Raydium, Atrix, Aldrin, Saros, Penguin, and others
 - [Mercurial](#) is based off of this code, but does not use it.
- [TokenDAO](#) - A browser for Solana token lists.
- [The Certified© Token List](#) - An aggregated token list, managed by protocols.
- [Certified© Limit Book](#) - (*unmaintained*) A powerful, fully decentralized crypto exchanged powered by Solana and Serum.
- [Quarry](#) - ⚡ An open protocol for launching liquidity mining programs on Solana.
- [Crate Protocol](#) - A composable primitive for tokenized baskets of assets.
- [Arrow Protocol](#) - Protocol for launching staking derivatives for communities and DAOs.
- [Sencha](#) - An automated market maker based on the constant product invariant.
- [Traction](#) - An American options protocol.

Listing on Saber

Anyone can create a Saber pool: Saber is a permissionless protocol.

However, getting featured on the web app and being added to gauges requires additional steps. This is because Saber only lists stable pairs: assets must be verified to be relatively stable before being pushed to users.

To get your asset listed on Saber, perform the following steps:

1. Add your token to [the official Solana token list](#).
2. Submit a pull request to [the Saber Governance repo](#), editing `Sources.toml` and `assets/Mainnet.toml`. If you are non-technical, you may [create a ticket here](#) instead.

More Help

Please email team@saber.so if you have any additional questions with `Listing Request` in the subject.

Solana Wallets

The Saber interface currently supports the following wallets:

- [Phantom](#) (suggested for most users)
- [Clover](#)
- [Coin98 Wallet](#)
- [Ledger](#)
- [MathWallet](#)
- [Slope](#)
- [Sollet Extension](#)
- [Sollet](#)
- [Solong](#)

Adding a Wallet

We use [use-solana](#) to allow users to connect to their wallets.

If your wallet is not yet supported by Saber, please submit a pull request to this repository and contact our team at team@saber.so.

Tutorial: Providing Liquidity

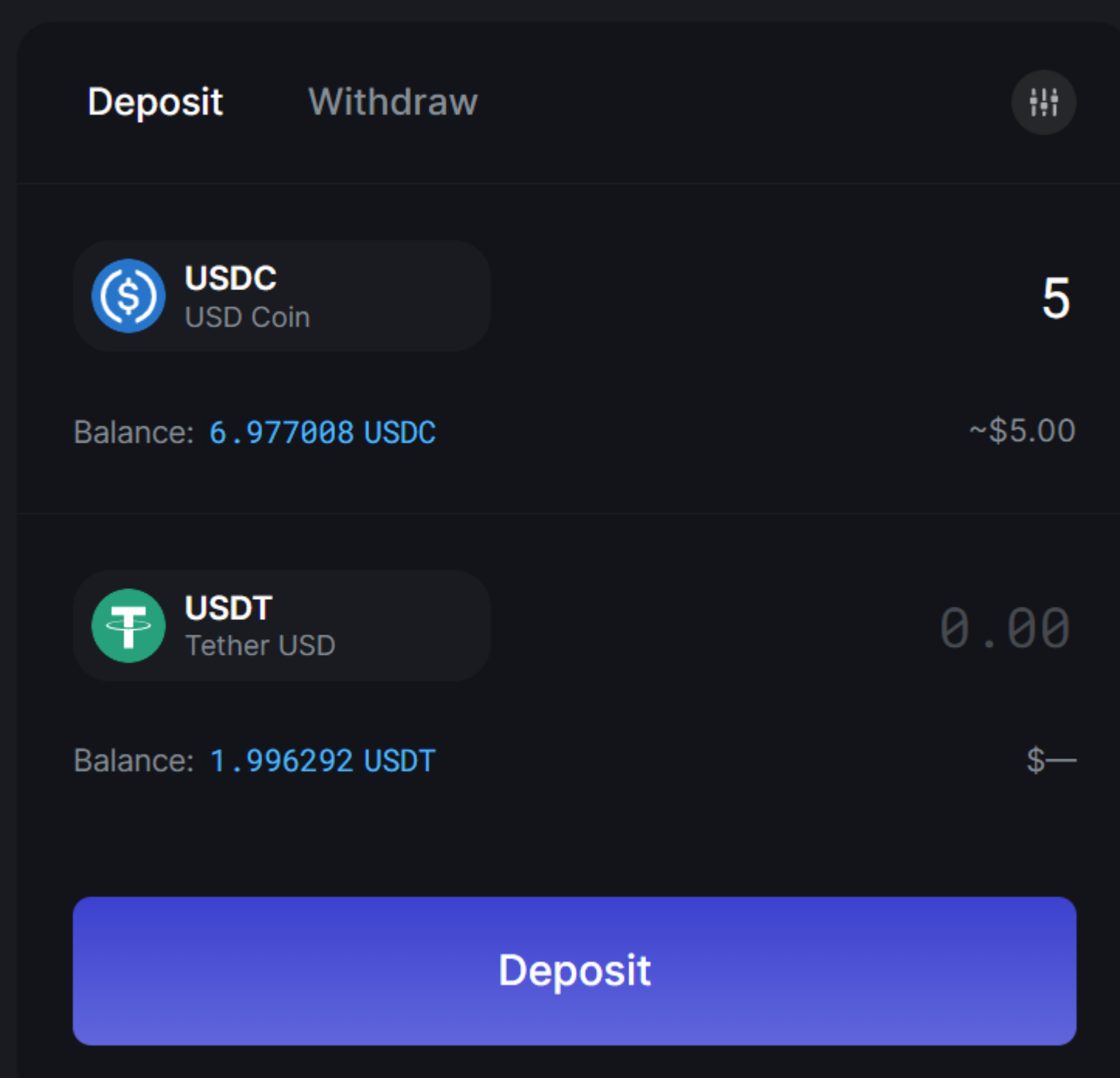
Pools are a way to provide *liquidity*, or the ability to trade, between tokens on Saber. They allow users to earn trading fees on tokens they're holding.

Deposit

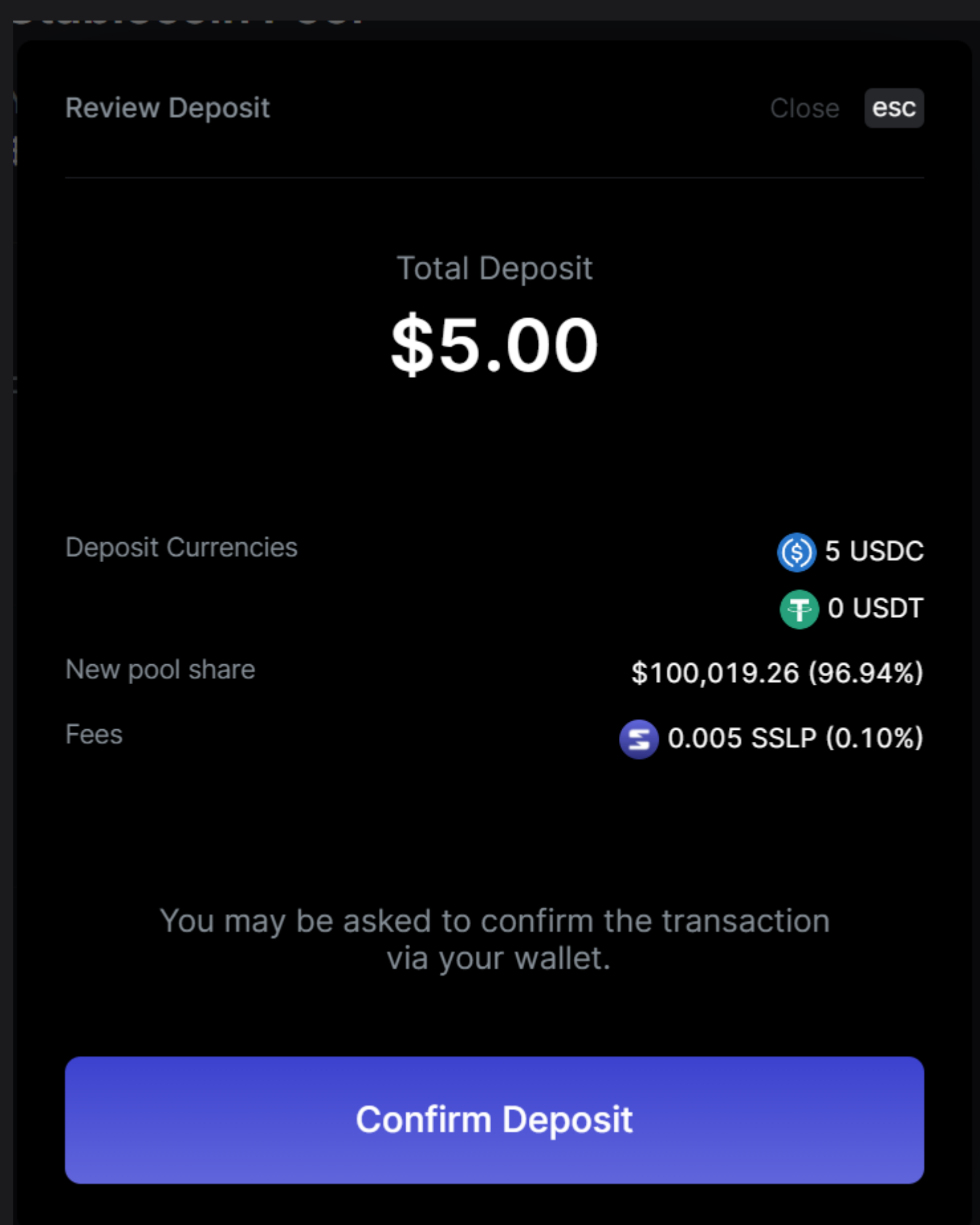
Let's say you want to supply to the USDC and USDT pool.

- Go to the **Pools** tab and select **Deposit**.
- Enter a quantity of USDT and USDC to trade from.
 - Because this is a stable pair, you aren't required to deposit both tokens. However, depending on how different the quantity between tokens is, you may have to pay a larger transaction fee. This is to incentivize close-to-equal distribution within the pool.

Position Management



- Select **Deposit**. This will display a breakdown of the transaction you're about to make:
 - **Deposit Currencies** shows the amount of each token you're depositing
 - **New pool share** is what your share of the pool will be after the transaction is made
 - **Fees** is determined by how different the quantity between tokens you're depositing is. This is to incentivize close-to-equal distribution within the pool.

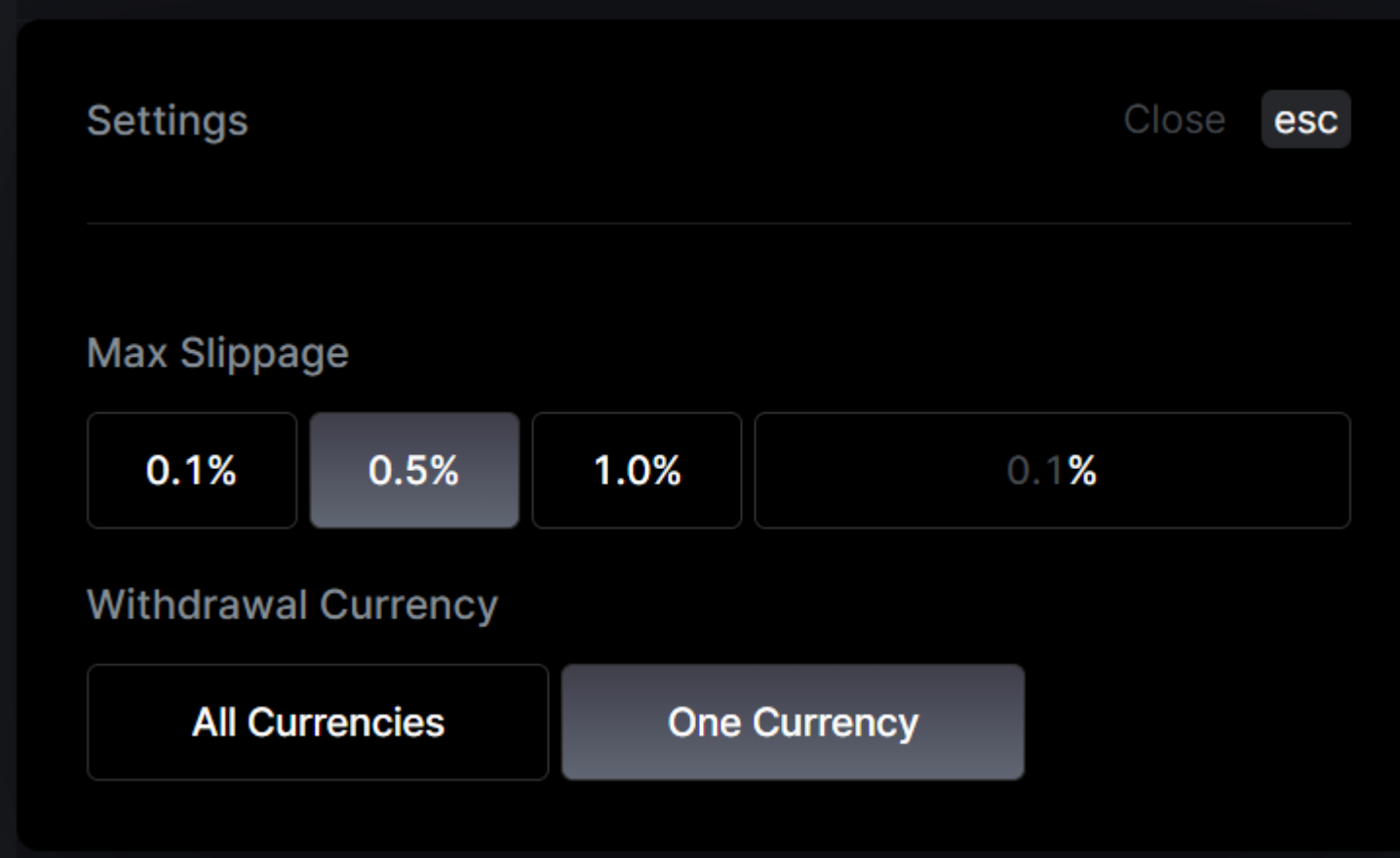


- Select **Confirm Deposit**. Depending on what wallet you're using, you'll likely be prompted to confirm the transaction via your wallet's interface.
- The deposit is now complete! On the bottom left, you'll see a notification that contains a link to the transaction on Solana's explorer.
 - You'll notice that you also now have LP tokens in your wallet that represent your share of the pool.\

Withdraw

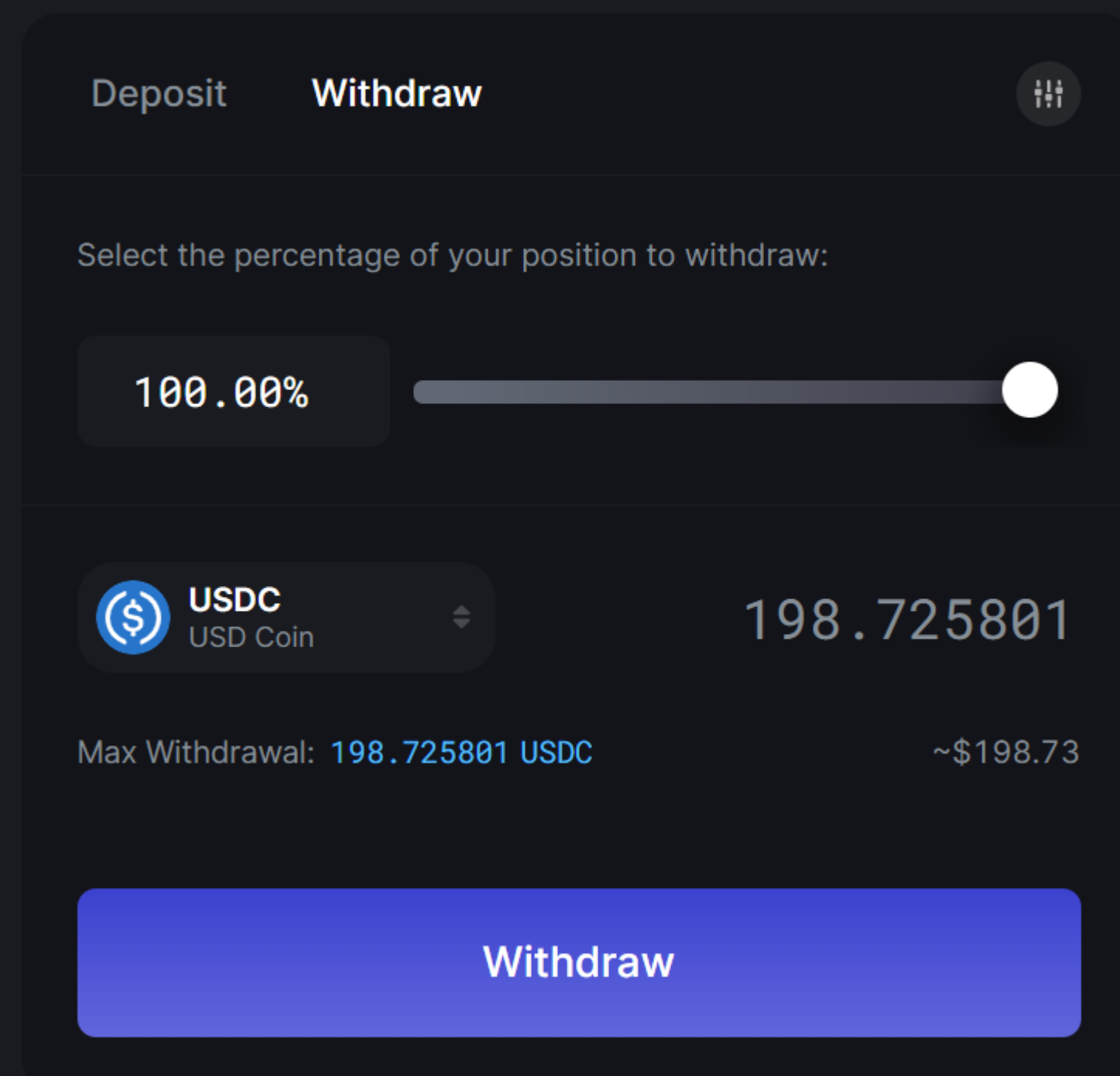
Let's say you want to withdraw your liquidity from the USDC-USDT pool.

- Go to the **Pools** tab and select **Withdraw**.
- By default, you'll be given the option to withdraw using a single token in the pool. You can click on the currently selected token to a different token in the pool.
 - If you are withdrawing a large proportion of the pool, you may want to choose to withdraw a distribution of all tokens in the pool instead. To do this, you can enter **Settings** by clicking on the settings icon in the top right corner. Here, you can switch Withdrawal Currency to **All Currencies**.\

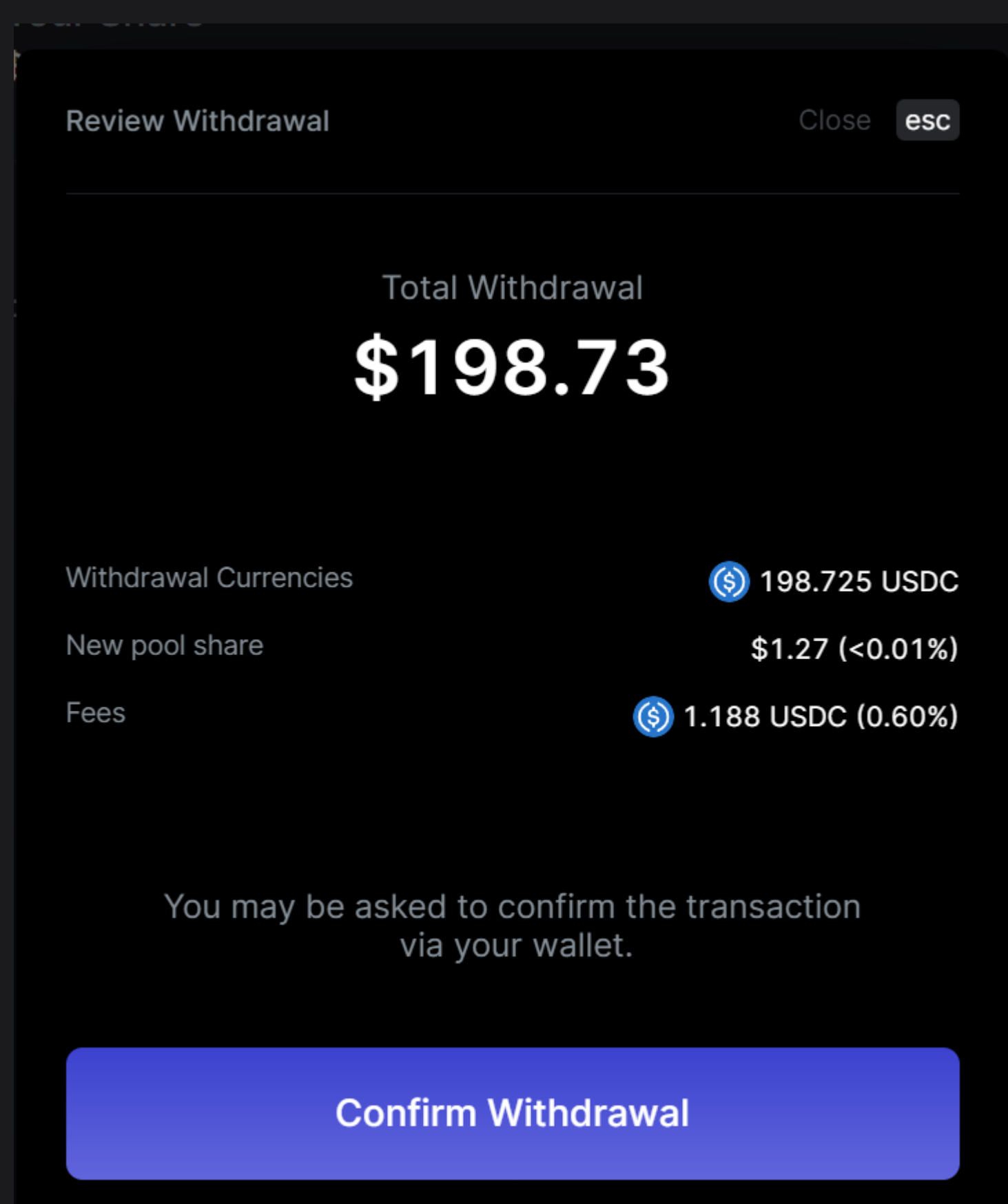


- Enter how much of your liquidity in the pool that you want to withdraw by moving the percentage slider.

Position Management



- Once you've entered your desired amount, click the **Withdraw** button. This will show a review screen with the following breakdown:
 - **Withdrawal Currencies** shows the amount of each token you're withdrawing
 - **New pool share** is what your share of the pool will be after the transaction is made
 - **Fees** is determined by the quantity of token(s) you withdraw relative to the other token in the pool. This is to incentivize close-to-equal distribution within the pool.



- Select **Confirm Withdrawal**. Depending on what wallet you're using, you'll likely be prompted to confirm the transaction via your wallet's interface.
- The withdrawal is now complete! On the bottom left, you'll see a notification that contains a link to the transaction on Solana's explorer.


Tutorial: Swapping Tokens

Saber allows you to swap between a stable pair of assets (e.g. USDC and USDT) with very low slippage and fees.

Let's say you have USDT and you want to trade it for USDC.


- On the swapping interface, select the tokens you want to exchange from (USDT) and to (USDC).
- Enter a quantity of USDT to swap from.
 - You can click on a token balance to populate the max amount you can swap.
 - Clicking the arrow button in the center will switch between what token you're swapping from and to.

Swap

 **USDC**
USD Coin

Balance: 50,007.977008 USDC ~\$1.00

↓

 **USDT**
Tether USD

Balance: 50,001.000000 USDT ~\$1.00 (<0.01%)

↔ Exchange Rate 1 USDC = 0.9963 USDT

Review


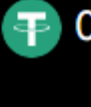
- Select **Review**. This will display a breakdown of the transaction you're about to make:
 - **Swap from** is the quantity of the token you currently hold that you want to exchange
 - **Minimum Received** is the minimum quantity of the token you'll receive. If the received amount turns out to be lower after you execute the transaction, the transaction will fail.
 - **Exchange Rate** is the conversion rate between the two tokens you're swapping.
 - **Price Impact** is the slippage percentage of the swap. The lower the price impact, the better value you're getting on your swap.
 - **Liquidity Provider Fee** is the transaction fee for swaps. This fee is split evenly between rewards to incentivize liquidity providers and reserves for the protocol.

Review Swap

Close **esc**

Estimated Received

0.996 USDT

Swap from	 1 USDC
Minimum Received	 0.991 USDT
Exchange Rate	1 USDC = 0.9963 USDT
Price Impact	<0.01%
Liquidity Provider Fee	0.002 USDT (0.20%)

You may be asked to confirm the transaction via your wallet.

Confirm Swap -

- Select **Confirm Swap**. Depending on what wallet you're using, you'll likely be prompted to confirm the transaction via your wallet's interface.
- The swap is now complete! On the bottom left, you'll see a notification that contains a link to the transaction on Solana's explorer.

Tutorial: Yield Farming

Yield farming (or liquidity mining) is a way to incentivize liquidity providers to deposits crypto assets in the protocol.

Farms

Saber liquidity mining

Deposit your Liquidity Provider tokens to receive SBR, the Saber governance token.

[Read more about SBR](#) 

Active Farms

  **USDC-USDT**

Stake

Total Staked

\$19,999

APY



  **USDC-PAI**

Stake

Total Staked

\$0.00

APY

∞

Once inside of a farm, you'll be able to stake or unstake LP tokens.

Liquidity Mining

Stake

Unstake



SLP
USDC-USDT Sab...

0.00

Balance: 0.000000 SLP

\$—

Enter an amount

You can see in each farm how much you have liquidity stake, as well as how much SBR you have available to claim.

Your liquidity staked

0

USDC-USDT Saber LP

Your unclaimed SBR

0

0 / day

Overview

Protocol

The Saber protocol consists of several programs:

- The [Stable Swap program](#), which handles creating LP tokens and swapping
- The [Saber Periphery](#), which consists of:
 - The [Decimal Wrapper](#), which handles adjustments of decimals
 - The [Continuation Router](#), which routes between different Saber pools and the [Add Decimal](#) program atomically in one transaction.
 - The [Lockup](#), which controls the tokens released to the Saber team and pre-token investors
 - The [Mint Proxy](#), which handles the issuance of new Saber tokens
 - The [Redeemer](#), which allows burning an "IOU" token for a Saber token

Liquidity Mining

Liquidity mining is handled by the [Quarry protocol](#).

Governance

Governance is handled by [Tribeca](#). Documentation on its programs may be found on the [Tribeca documentation site](#).

The Saber Registry

The [Saber Registry](#) is an automatically generated JSON list of all Saber pools. It contains metadata not present on-chain which may be useful if displaying pool information.

Pools List

The pools list is available at <https://registry.saber.so/data/pools-info.mainnet.json>.

The list contains the following info for each pool:

- `id`: the unique user-friendly ID of the pool, used in the URL
- `tokens`: the tokens backing the pool
- `tokenIcons`: the tokens backing the pool in the order that they should be rendered. E.g. USDC-USDT should be rendered as USDT-USDC.
- `underlyingIcons`: the `tokenIcons`, unwrapped for their underlying assets. This is generally only different than `tokenIcons` for [decimal wrapped tokens](#).
- `currency`: the currency category of the pool, all uppercase. A pool generally only trades against other pools with the same currency.
- `lpToken`: token information of the LP token.
- `plotKey`: Deprecated. This is the key of the Plot in the old Saber farming system.
- `swap`: The pool config and state.
- `quarry`: Key of the [Quarry](#) associated with the official Saber liquidity mining program.

Decimal Wrapped Tokens

A *decimal-wrapped token* is a token which has been modified to add more decimal places. It exists because the Saber stable swap invariant formula does not adjust for the number of decimal places that the underlying token has.

For example, USDC has 6 decimal places but wDAI has 9 decimal places. Without the decimal wrapper, the stable swap invariant would assume that 1,000 USDC = 1 DAI, which is obviously incorrect.

The lack of decimal adjustment was intentional to ship fast with as few edge cases as possible; however, it can cause confusion for developers attempting to integrate tokens. Furthermore, it is possible to acquire decimal-wrapped tokens in one's wallet if the Saber UI was not used to withdraw or swap tokens.

Terminology

The **underlying** token is the token that is staked into the decimal wrapper. The **wrapper** token is the decimal wrapped token.

Each wrapper token is suffixed with the number of decimal places it has. For example, USDC-9 is the wrapper token of USDC with 9 decimals.

Interacting with the Decimal Wrapper

There are four key [instructions for interacting with decimal-wrapped tokens](#):

- `InitializeWrapper`: if your token does not yet have a decimal wrapper, you must create a decimal wrapper using this instruction. Make sure to select a mint that is user friendly, as it might end up in your users' wallets!
- `deposit`: Deposits underlying tokens for wrapper tokens.
- `withdraw`: Burns wrapper tokens to get the underlying tokens.
- `withdrawAll`: Burns all wrapper tokens to get the underlying tokens. This is particularly useful for multi-instruction swaps.

One should use the [Saber Periphery](#) SDK in order to interact with this program.

TODO: add code examples of the `saber-periphery`.

Code

The code for the [Add Decimals program](#) can be found on [GitHub](#).

Pricing LP Tokens

Fair Price

The **fair price** of a Saber LP token is the value of the tokens backing an LP token. For example, if the USDT-USDC pool has 700 USDT and 300 USDC, there are 1,000 circulating LP tokens, and USDT = \$1.01 while USDC = \$1.00, the fair price is:

$$1.01 * 700 + 1.00 * 300 / 1,000 = \$1.007$$

This is useful for calculating how much money one has; however, it is unreliable for calculating value as collateral as this value can easily be manipulated.

Virtual Price

The **virtual price** of a Saber Stable Swap pool is the theoretical lower bound of the value of tokens if each token was worth 1.00 of the underlying asset. It is useful when understanding LP returns or liquidation value.

Virtual price may only increase, and it only increases when swap fees are accrued.

- Rust implementation: `SaberSwap::calculate_virtual_price_of_pool_tokens`
- JavaScript implementation: `calculateVirtualPrice`

Calculating liquidation value

To use a Saber LP token as collateral, you will need to fetch the prices of both of the tokens in the pool and get the min of the two. Then, multiply the number of LP tokens by the *virtual price*.

This virtual price is resilient to manipulations of the LP token price.

Hence, `min_lp_price = min_value * virtual_price`.

Additional Reading

- [Chainlink: Using Chainlink Oracles to Securely Utilize Curve LP Pools](#)

Saber Common

GitHub: [saber-hq/saber-common](https://github.com/saber-hq/saber-common)

This repository is a monorepo containing several Solana utilities and the Saber StableSwap SDK.

Automatically generated TypeScript documentation can be found [on GitHub pages](#).

Overview

Package	Description	Version
<code>@saberhq/anchor-contrib</code>	TypeScript client for Anchor programs	npm <code>v1.14.11</code>
<code>@saberhq/chai-solana</code>	Chai test helpers	npm <code>v1.14.11</code>
<code>@saberhq/eslint-config</code>	ESLint config for TypeScript projects	npm <code>v3.0.0</code>
<code>@saberhq/eslint-config-react</code>	ESLint config for React projects	npm <code>v3.0.0</code>
<code>@saberhq/solana-contrib</code>	Solana TypeScript utilities	npm <code>v1.14.11</code>
<code>@saberhq/stableswap-sdk</code>	StableSwap SDK	npm <code>v1.14.11</code>
<code>@saberhq/token-utils</code>	SPL Token arithmetic and types	npm <code>v1.14.11</code>
<code>@saberhq/tsconfig</code>	Saber recommended TypeScript configurations	npm <code>v3.0.0</code>
<code>@saberhq/use-solana</code>	Solana React library	npm <code>v1.14.11</code>

Saber Stable Swap SDK

license [Apache-2.0](#) build <https://github.com/badges/shields/issues/8671> contributors [7](#)

GitHub: [saber-hq/stable-swap](#)

The Saber Stable Swap repository holds the Rust code for Saber, in addition to several libraries.

Rust Crates

Package	Description	Version	Docs
stable-swap	Saber StableSwap program.	crates.io v1.8.1	docs passing
stable-swap-anchor	Anchor bindings for the StableSwap Rust client.	crates.io v1.8.1	docs passing
stable-swap-client	StableSwap Rust client.	crates.io v1.8.1	docs passing
stable-swap-fuzz	Fuzz tests for the Saber StableSwap program.	crates.io v1.8.1	docs passing
stable-swap-math	Calculations for the StableSwap invariant	crates.io v1.8.1	docs passing
stable-swap-sim	Simulations of the StableSwap invariant compared to Curve's reference implementation	crates.io v0.1.4	docs passing

JavaScript/Web3.js

To use StableSwap with your frontend or Node.js project, use [the JavaScript SDK](#).

Audit

Saber's [stable-swap-program](#) has been audited by [Bramah Systems](#). View the audit report [here](#).

Developing

Tests

To run the tests, run:

```
./stable-swap-program/do.sh e2e-test
```

Archive

The original Saber StableSwap program can be found on the [archive branch](#).

License

Saber StableSwap is licensed under the Apache License, Version 2.0.

Lido stSOL



About

stSOL is a liquid stake pool token. This means that you can deposit SOL with LIDO, let them stake the SOL with various validators to earn interest, and gain capital efficiency using the stSOL stake pool token to represent your deposit. The risks involved include Lido delegated validators being dishonest and having their SOL slashed when slashing is implemented on mainnet.

How to Mint stSOL


Head to <https://solana.lido.fi/> and stake SOL to receive stSOL. Note that it may take 2 to 3 days (1 Solana epoch) to unstake from a validator and receive your SOL back.

Stake Solana

Stake SOL and receive stSOL while staking

Stake Unstake

Stake

 Amount MAX

[Connect wallet](#)

You will receive [?]	~0 stSOL
Exchange rate [?]	1 stSOL = ~1.0135 SOL
Transaction cost	~ 0.000005 SOL (\$0.00101)
Staking rewards fee [?]	10%

Lido statistics

[View on Block Explorer](#)

Annual percentage rate	6.06%
Total staked with Lido	978,908.51 SOL
Stakers	8,729
stSOL market cap	\$198,789,997

Marinade Finance



About

Marinade Finance is a Liquid staking protocol on Solana. Liquid staking is an alternative to traditional staking: it allows users to stake any amount of SOL and to effectively unstake their SOL without the requirement of waiting 1-3 days before the stake is warmed up or cooled down. Warming up or cooling down stake normally takes around ~3 days.

mSOL is a liquid stake pool token. The risks include Marinade delegated validators being dishonest and getting their SOL slashed when slashing is implemented on mainnet. Read more about Marinade's staking algorithm in [their documentation](#).

How to Acquire mSOL

Head to [Marinade's app](#) and deposit SOL to receive mSOL.

The screenshot displays the Marinade Finance interface. At the top, a 'Wallet' section shows SOL and mSOL balances. A central circular gauge indicates an APY of 6.67%. To the right, 'Total staked' is shown as 6,746,498.43 SOL (≈ \$1,375,460,898) and 'Unstake liquidity' as 572,362.78 SOL (≈ \$116,692,034). Below this, there are two tabs: 'Stake' and 'Unstake'. The 'Stake' tab is active, showing a 'Stake SOL' input field with a value of 0.0 and a 'MAX' button. Below it, an 'mSOL' output field shows a value of 0. The exchange rate is 1 mSOL ≈ 1.02045 SOL, and the deposit fee is 0%. A 'Connect wallet' button is visible at the bottom.

Asset	Balance
SOL	0.0
mSOL	0

APY: 6.67%

Total staked: 6,746,498.43 SOL (≈ \$1,375,460,898)

Unstake liquidity: 572,362.78 SOL (≈ \$116,692,034)

Exchange rate: 1 mSOL ≈ 1.02045 SOL

Deposit fee: 0%

Parrot Protocol



About

The Parrot Protocol is setting out to make value locked in LP tokens accessible, by creating a liquidity & lending network collateralized by these LP tokens. Parrot created the PAI stablecoin which is collateralized by LP tokens or PRT, Parrot's token. Parrot also runs a stake pool which pays the liquid staking token prtSOL, and pSOL is their aggregate stake pool token similar to aSOL.

How to Create Parrot Assets

To acquire the PAI stablecoin, deposit collateral on [Parrot](#). Note that the risks associated with this strategy are liquidation of collateral if it drops below a given percentage of the PAI debt, and the possibility of PAI losing its peg.

The screenshot displays the Parrot Protocol's 'Mint' interface. At the top, a parrot icon is next to a callout box showing 'Total Value Locked \$170,689,238.85'. The interface is split into two tabs: 'Mint' (active) and 'Repay'. Under the 'Mint' tab, there are two main sections: 'I want to deposit' and 'To mint'. The 'I want to deposit' section has a dropdown menu for 'SBR LP+Earn' (acUSD-USDC) and a 'max deposit: 0.00' label. The 'To mint' section has a dropdown menu for 'PAI' and a 'max mint: 0.00' label. Below these sections is a 'Collateral ratio' bar, which is currently empty. At the bottom, there is a table of financial metrics and a large green 'Mint' button.

Metric	Value
Deposit APY:	7.68%
Borrow APY:	3.11%
Debt (PAI):	0
Collateral (SBR LP):	0
Liquidation price (SBR LP):	would never liquidate
Current price (SBR LP):	1.00

To acquire prtSOL, head to Parrot's staking page and deposit SOL. Parrot will deposit SOL with Solana validators and earn yield, which will be realized via appreciation of the pSOL token value. The main associated risk is that validators Parrot delegates to are malicious and have their funds slashed. As of December 2021, slashing is not yet implemented on the Solana mainnet.

Port Finance

The First Non-Custodial Liquidity Protocol On Solana

Port Finance is a lending protocol that aims to provide an entire suite of fixed income products including variable rate lending, fixed rate lending and interest rate swaps.

[Go to App](#)



About

Port Finance is a non custodial lending protocol on Solana that enables lending and borrowing markets for a number of assets including USDC, USDT, SOL, and SBR among others. pUSDC and pUSDT represent interest bearing, non collateral positions on Port (the synthetic token works because the positions cannot be liquidated)

How to Mint pUSDC and pUSDT

In order to mint pUSDC (same process for pUSDT) head to the [Port Finance App](#) and deposit USDC. Then, unselect the 'use as collateral' option. Now, any USDC you deposit will mint associated pUSDC tokens in your wallet, which can be deposited on Saber. The pUSDC tokens represent your USDC position. There is not a risk of liquidation because

Synthetify Assets



About

Synthetify is a synthetic asset protocol on Solana that enables low slippage trading. Synthetic assets track the market price of assets via oracles, which are data feeds that connect Solana and other blockchains with outside data, often price data. Synthetify uses the [Pyth](#) oracle network. xUSD is an exception to this; it is algorithmically pegged to \$1.00 by Synthetify.

How To Create Synthetic Assets

To create Synthetify assets, the first step is to deposit collateral on Synthetify. In this example, we will deposit USDC, but many SPL tokens are accepted as collateral.

Deposit Mint Withdraw Burn Rewards

Amount

0.02 USDC Max Available to deposit USDC 0.0349

Deposit

Next, we use this collateral to mint xUSD on the mint page. It is important to note that xUSD is being taken on as debt (risks outlined in [Risks](#)).

Deposit Mint Withdraw Burn Rewards

Amount

Max xUSD Max Available to mint xUSD 0.0128

Mint

Then, we can head to Synthetify's swap page and trade xUSD for any synthetic asset.

Swap

From Balance: 0.0126 xUSD

xUSD 0.000058474 Max

To (Estimate) Balance: 0.0000 xSOL

xSOL 0.000000000 Max

Fee 0.3% (0%) Exchange rate 215.804835 xUSD per xSOL

Enter value of swap

Risks

If the value of synthetic assets borrowed exceeds the value of deposited collateral, your collateral may be liquidated. Uniquely, the borrow values are determined as a fixed percentage of the total debt. Thus, if your synthetic portfolio underperforms the mean synthetic portfolio on the platform, your debt will increase, and vice versa. For more details, see the [Synthetify Whitepaper](#)

About Wormhole

About

[Wormhole](#) is a cross chain token bridge that enables many saber pools. Currently, Wormhole allows bridging between a variety of blockchains including Ethereum, Solana, Binance Smart Chain, Terra, and Polygon. For details on how Wormhole works and its security, check out this [excellent article](#) by "The Intern".

At a high level, Wormhole deploys a simple cross-chain attestation model that relies on a group of cross-chain oracles called "Guardians" that operate as a set of node operators with venerated validation track records and strong incentive alignment with the long-term interest of the root chain — Solana.

How To Send Tokens to Solana

This example is for migrating tokens from the Ethereum blockchain, but the same steps can be applied for any source blockchain. First, navigate to [Wormhole's website](#).

Select your source chain and Solana as the destination, connect your Ethereum wallet, and select which asset you would like to transfer.

In this Case we'll use ETH as an example. Next, connect your Solana wallet. Make sure you have some SOL in your Solana wallet to claim the tokens.

Create the associated token account, and send tokens. Make sure to approve the transactions in your Solana and Ethereum wallets.

Stay on the page and wait for 15 confirmations on the Ethereum network. Then, make sure to redeem your tokens on Solana, confirming a few transactions. Tokens will arrive in your wallet.

Wormhole Token Abbreviations

The w indicates that the token is wrapped from Wormhole. The optional second letter in the token indicates token origin. So, for example weDAI indicates Wormhole wrapped DAI bridged from Ethereum, while wpDAI indicates Wormhole wrapped DAI bridged from Polygon. A "V1" suffix indicates that the token is a legacy asset from Wormhole V1 and should be [migrated](#) to V2

Token Migration

In order to migrate Wormhole tokens to V2, head to [Wormhole's website](#) and you should be prompted for a migration pool. If this does not appear, the pool may be out of liquidity. In this case, you will need to send tokens back to their original blockchain, before bridging back to Solana. This will give you V2 tokens.

Verifying Mints

Wormhole has a useful tool to verify the wormhole [token addresses](#), where you can copy and paste an address to verify its origin.

How to Bridge ibBTC to Saber

Badger is a community DAO and is focused on bringing Bitcoin to DeFi. Badger has aggregated much of the tokenized Bitcoin liquidity on Ethereum via its [SETTs](#). A SETT, is a vault akin to yearn vaults and deploys strategies for maximizing BTC yields. BTC denominated assets from Badger Vaults (Sett) or Tokenized BTC (wbTC, renBTC, tBTC & sBTC) can be deposited to [mint ibBTC](#).

Using the Ethereum to Solana Wormhole Bridge

To bridge ibBTC from Ethereum to Solana, users can leverage [the Wormhole Bridge](#). Note that Saber does not maintain the bridge; any questions or issues should be [brought up to the Wormhole team directly](#).

1. Go to the [Wormhole Bridge UI website](#).
2. Connect the Ethereum wallet you're storing your ibBTC in and the Solana wallet you want to transfer to. If you don't have a Solana wallet, we recommend [Phantom's Extension Wallet](#) -- the functionality is similar to MetaMask. If you're creating a new Solana wallet, you'll also need SOL to pay for transaction fees. You can get SOL using a centralized exchange.
3. Once connected, enter the amount of ibBTC you want to transfer, then select **Transfer**.
4. You should now be prompted by several transactions on your Solana and Ethereum wallets to initiate the transaction.
5. Once you get to the **Finalizing transfer** step, you must waiting for 15 ETH confirmations for the Wormhole Wrapped ibBTC to display in your Solana wallet.

Supplying liquidity for (depositing) ibBTC

1. Now that you have ibBTC on Solana, you can deposit it into the Saber ibBTC-renBTC pool to earn yield from swap fees.
2. On the ibBTC-renBTC pool page, you can deposit your desired amount. In order to maximize the amount of LP tokens you receive, you may want to also supply a balanced proportion of renBTC.
3. After you've deposited the tokens, you'll receive an LP token representing your share of the pool.

Staking (farming) ibBTC-renBTC LP

1. To earn liquidity mining incentives for ibBTC-renBTC, you must stake the LP token into Saber.
2. On the ibBTC-renBTC farm page, you can stake the LP token.
3. While staked, you'll continuously receive rewards relative to your share of the pool.
4. You can select Claim at any time to transfer the rewards to your wallet without unstaking your LP token.

Allbridge Assets

allbridge
by APYSwap Foundation

About

Allbridge is a multi chain token bridge that enables trustless transfer of assets between Solana, Ethereum, Avalanche, Polygon, Celo, Binance Smart Chain, and a number of other blockchains. It works via a network of validators which burn and unlock wrapped and native tokens across these blockchains to empower trustless bridging

How To Bridge

There is a [video guide](#) provided by allbridge available. An example of bridging from Ethereum to Solana is shown below; other blockchains follow the same process.

First, head to the [Allbridge token bridge](#) and select Ethereum as the source blockchain and Solana as the destination. Make sure you have enough ETH and SOL in both wallets to pay for transactions on both blockchains. Select which asset you'd like to bridge (we'll use ETH) and connect your Ethereum wallet.

The screenshot shows the 'Choose blockchain and asset' step of the bridging process. At the top, there is a progress indicator with four steps: 1 (selected), 2, 3, and 4. A 'Watch video guide' link is on the left, and a 'Finish transfer' button is on the right. The main form has three dropdown menus: 'From' set to 'Ethereum', 'To' set to 'Solana', and 'Choose asset' set to 'ETH'. A 'Connect wallet' button is at the bottom.

Next, paste in your Solana address (for a wallet you control) and send the tokens. Wait for the transaction to be confirmed on Ethereum.

The screenshot shows the 'Amount and address' step. The progress indicator shows step 1 as complete and step 2 as the current step. Below the title, there is a text input field for the 'Address to send your assets to'. A warning box states: 'Please send your assets to wallets where you control private keys! Don't send your assets to contract or token addresses, and cryptocurrency exchanges like Binance, Kraken, FTX etc. There is a chance that your assets will get lost on the exchange side.' Below this, there is a 'Amount to send' input field with a 'Current balance: 0.191396728225664801 ETH' and a 'Max' button. A 'Fee' section shows 'Bridge Fee (0.00011 ETH)' and 'You will receive:'. A 'Send' button is at the bottom.

The screenshot shows the 'Confirmation' step. The progress indicator shows steps 1 and 2 as complete, and step 3 as the current step. A large circular timer shows '55 seconds'. Below the timer, it says 'Transaction is pending' and 'Transaction can take some time if the network is overloaded.' There is a 'Tx ID: 0x27e2b...40885' with copy and share icons, and a 'Connect wallet' button.

Finally, submit and approve a Solana transaction to receive your funds..

The screenshot shows the 'Receive money' step. It features a 'Submit receive transaction to receive your money.' instruction and a 'Receive' button at the bottom.

Allbridge Token Abbreviations

The a indicates that the token is wrapped from Allbridge. The optional second letter in the token indicates token origin. So, for example aeDAI indicates Allbridge wrapped DAI bridged from Ethereum, while apDAI indicates Wormhole wrapped DAI bridged from Polygon.

Ren Bridge

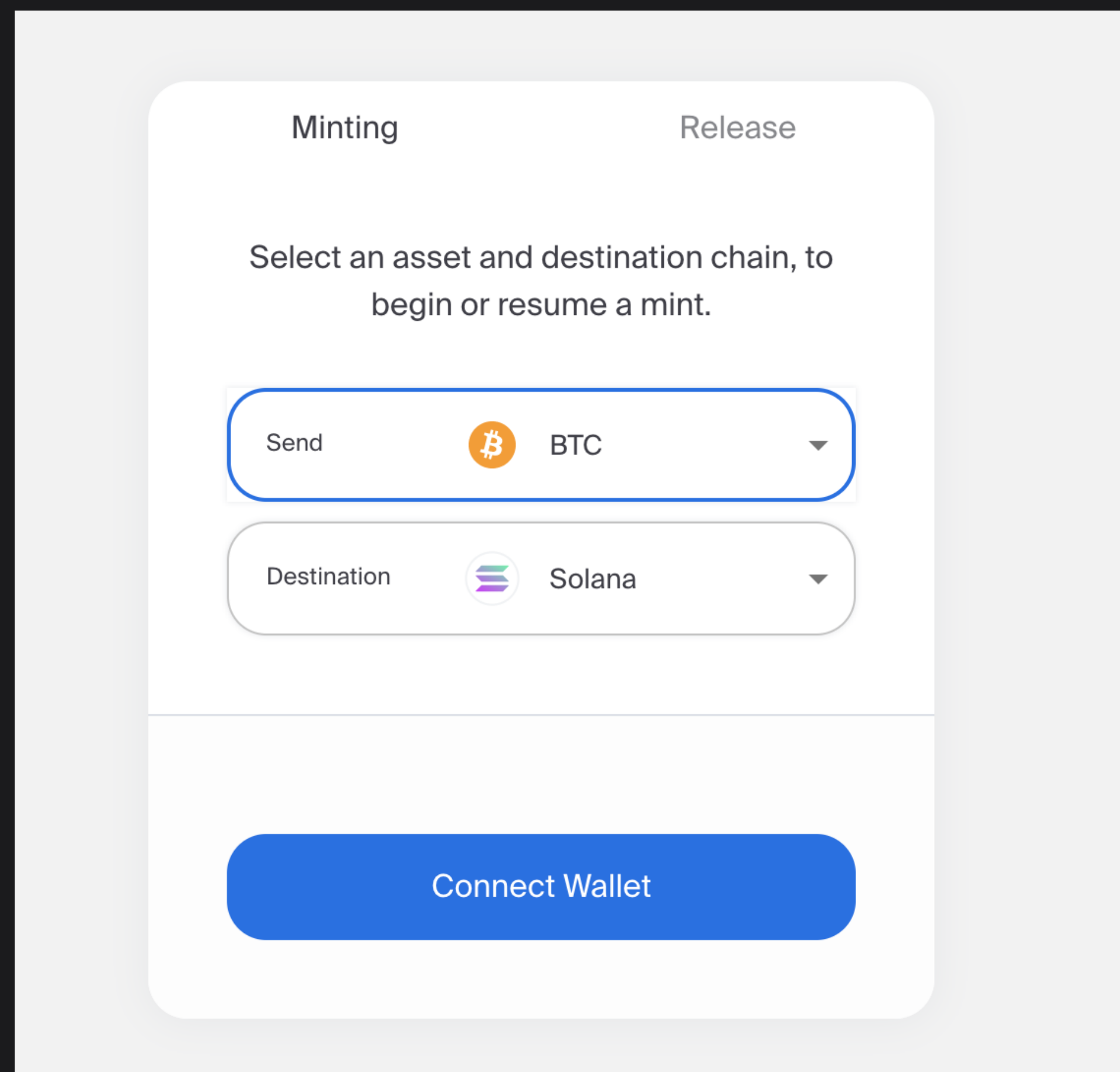
About

Ren is a token bridging protocol that connects a number of blockchains including Solana. renBTC is Bitcoin bridged from the Bitcoin network. Via Ren:

RenVM is network powered by decentralized virtual machines. This virtual machine is replicated over thousands of machines that work together to power it, contributing their network bandwidth, their computational power, and their storage capacity. These machines are known as Darknodes. Darknodes earn a share of the volume transacted through RenVM. Find out more [here](#)

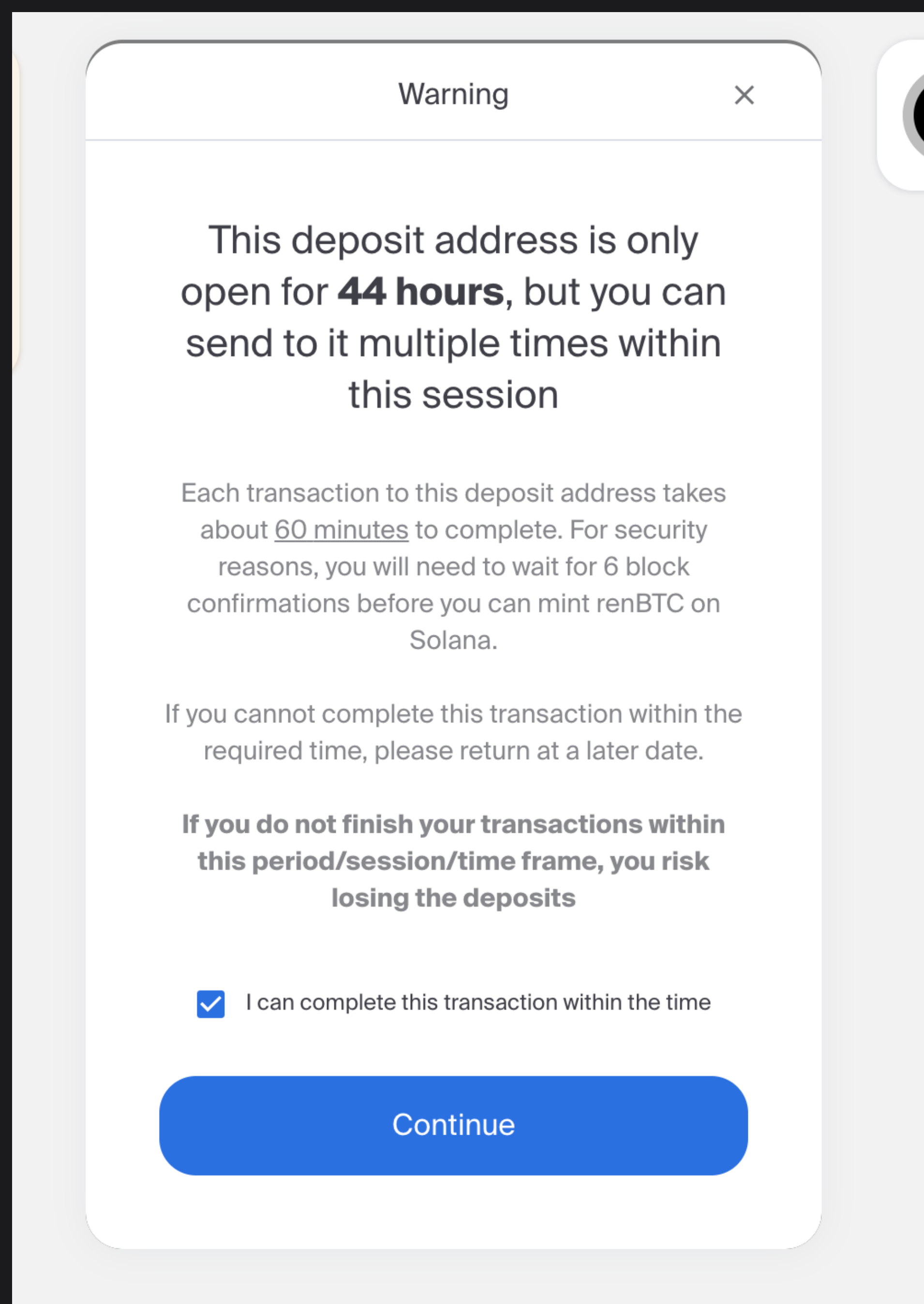
Bridging Assets

This tutorial uses renBTC as an example. The process of bridging is similar for other assets. First, head to [Ren's bridge website](#). Then, select Solana as the destination blockchain



Next, connect your Solana wallet where you would like to receive the funds. and confirm fees, before viewing your gateway address.

Send BTC to the deposit address. Then, return to this site and claim your funds by sending a Solana transaction. Make sure to complete this within 24 hours of sending the funds. Now, renBTC will appear in your Solana wallet.



Introduction

What is Quarry?

[Quarry Protocol](#) is a set of smart contracts that work together to create an open standard for launching liquidity mining & staking campaigns on Solana. The protocol currently offers the following primitives:

1. **Rewarders** are entities which own the underlying reward token (*e.g. SBR/ SLNA*). These are generally other protocol DAOs (*e.g. Saber/ Soluna*) which have a reward token they would like to emit across Quarries to incentivize staking LPs/ staking other tokens.
2. **Quarries** are smart contracts which require a specific token to be staked in them in order to claim the rewards allocated to the Quarry. A Quarry can accept any value that can be represented as a token, including LP token balances, governance stake (veTokens), and borrowing amount.
3. **Gauges** are an integration with [Tribeca Protocol](#) which allow DAO token lockers to vote on the future emission weight across a Rewarder's Quarries. The share of votes in each Quarry for that Rewarder corresponds to the share of emissions to that Quarry from the Rewarder.

Quarry Protocol is designed with composability in mind, enabling other protocols to easily build on top of it.

Quarry v2 will include a Bribe market for other protocols to offer incentives in exchange for veToken holders delegating their governance - you can read more in our Roadmap.

Audited source code & a developer SDK are available [on our Github](#).

Disclaimer

All claims, content, designs, algorithms, estimates, roadmaps, specifications, and performance measurements described in this project are done with the author's best effort. It is up to the reader to check and validate their accuracy and truthfulness.

Quarry Use Case

The primary use case for Quarry is to provide **composable cross-protocol yield services**. The simplest example of this is illustrated in how Quarry allows protocols to directly provide additional liquidity mining rewards to LP stakers without having to go through an AMM. Traditionally AMMs require users to stake on their platform & only claim the additional rewards that the AMM **manually** adds to the farming page.

Why is this valuable?

This means that instead of only being eligible for SBR rewards from staking your Saber LP on Saber - you stake your LP in Quarry and become eligible for all the Rewarder emissions being provided to that Quarry. In practice, these Rewarders are the protocols tied to the underlying tokens which you are providing liquidity for (**e.g. Saber USDC-USDH LP stakers receive Hubble rewards on top of SBR rewards**).

This aligns protocols and liquidity providers, but can be used across any staking situation. For more information on how to add a new Rewarder or launch a Quarry - skip to the last section of the docs.

Other functions

Quarry also runs [Gauges](#) alongside Tribeca Protocol, which allow for the Rewarder's emissions to be determined by Tribeca Protocol votes. This means that emissions rates across a Rewarder's Quarries can be determined in a decentralized and permissionless manner.

How To Use Quarry

Step 1: Select a Rewarder

The first thing you will have to do is select a Rewarder on the *Browse Rewarders* page.

This opens a *Rewarder Page*, which includes all the Quarries available for you to stake into & receive that rewarder's token (for example, Soluna).

Step 2: Select a Quarry

Different Quarries offer different APY's which are calculated based on:

- Total value staked in the Primary Quarry's base token (for example, \$ amount of Saber UST-solUST LP).
- Total daily emissions \$ amount across value staked.
- Assumes rate would remain the same over the course of 1 year.

In reality these rates change as users move in and out of Quarries, and rewarders fund new rewards & others end their rewards. It is important to select a Quarry in which you have a favorable outlook on the token being staked as well as the token used for rewards.

Let's say you select the Saber UST-solUST LP which provides ~15% APY with two tokens tokens.

Step 3: Connect your wallet and deposit the Quarry's required token. In our example this would be a Saber UST-solUST LP token which we can acquire by depositing liquidity to saber.

Step 4: Claim rewards.

Upcoming Quarry Features

Overview

Quarry Protocol is designed to offer a general-purpose mining program for other protocols to build on top of. In order to continue offering useful functions to these protocols & our users - Quarry v2 will include more creative implementations of staking & open tooling for DAOs to make use of. We also have a vision to become a DAO once we have a well established community.

Roadmap

1. Streamline the Rewarder **self-onboarding** process.
2. Launch **Bribe market** for gauges, with self-onboarding for Tribeca-compatible DAOs.
3. Integrate Bribe mechanism beyond gauges to **all proposals on Tribeca**.
4. Launch Quarry DAO to manage platform approval procedures, fees, & hire employees to manage future protocol development.

Bribe Program Deep Dive

Gauges reflect governance power, but there is an indirect loop where users must purchase illiquid veTOK in order to participate in governance. What has been observed is that unless the user is a protocol - it is very likely that they will prefer to sell TOK rewards instead of locking TOK for more governance. **The lack of built-in incentives for veTOK locking means governance is largely ignored by normal users of protocols & instead the value is benefitting mostly other protocols.** We believe there is value that can be unlocked where protocols continue to gain value from the governance system - but so will passive DAO members.

Bribes allow governors to extract the value from their illiquid veTOK by renting their vote to protocols whose TVL depends on how votes **(and therefore future rewards)** are distributed across different Quarries via Gauges.

Some protocols are engineering systems to directly pay platform fees to veTOK lockers - **and this will be preserved with the integration of Bribes.** Bribes will only allow users to gain **additional** yield, and allows protocols to rent discounted governance power in order to indirectly purchase future liquidity.

Background

In early 2021, the Quarry Team released Gauges: allocation of liquidity mining rewards determined by veToken (veTOK) holders. This was launched very soon after Tribeca, which created the initial concept of vote escrows on Solana.

veTokens: DAO membership with long-term incentives

To recap: a vote-escrowed token, or veToken, is a non-transferrable token which may be obtained by locking up a "daoToken" for a user-specified duration of time. This mechanism ensures that long term believers in the DAO have the most control over it. The maximum duration is often capped by the DAO to prevent too much of an imbalance between short term and long term holders.

In Tribeca, veToken power is linear with respect to time and balance:

power = duration x balance x multiplier

This means that as a user approaches their lockup period, they also lose voting power: the optionality of liquidity comes at a power discount. This is key to ensuring that DAO members are in things for the long run.

The Problem: Lack of User Lock Incentives

Before the Saber and Sunny DAOs used Gauge, Saber (and Sunny) liquidity mining rewards were determined arbitrarily by team members. Gauges were the first use case of veTokens outside of voting on proposals.

Gauges allow veToken holders to vote weekly on the rewards distribution of various pools on a veDAO. However, there are only two types users who will buy veTokens for the purpose of voting on gauges:

- **Protocols.** Gauges allow a protocol to boost their own yields. There are several examples of this happening today, e.g. Terra investing heavily into Sunny/Saber or UXD performing a treasury trade with Saber.
- **Protocol affiliates.** If a veToken holder is invested in the success of another protocol, e.g. if they are a founder or an investor of that other protocol, then they may want to boost their own yields. An example of this is the Solend whale boosting the cToken pools.
- **Large liquidity providers.** If a liquidity provider has a lot of capital in a particular pool, they may want to increase the yields of that pool.

Note that neither of these users are the average user. Protocol affiliates may be an individual user; however, the user's vote does not matter nearly as much as a whale's does.

Gauges, then, are currently a tool built for whales. The question then is: how do we allow the common user to benefit from this governance power?

The Solution: Direct Payment to veToken holders

Instead of having to work directly with large veToken holders as a protocol or large liquidity provider, what if veToken holders could be paid en masse with tokens?

This is what bribes are: a way to distribute tokens proportionally to those who voted for a specific pool. Bribes are a way of ensuring that all veToken holders can benefit from gauges, rather than just a small subset of whales.

This is good for both protocols and veToken holders:

- **Protocols** may now spend much less money in order to get the same TVL, since they can vote up their pools indirectly by paying veToken holders.
- **veToken holders** can now earn yield just by holding veTokens, as long as they are voting for a pool with good rewards. This creates a sort of staking APY on veTokens, making veTokens more attractive to hold as a result.

One possible downside to this mechanism is that large veToken holders will now have less leverage in the DAO: founders and VCs with large veToken balances will not be as important in getting high pool rewards. However, this becomes much less relevant when this translates into increased token value.

How To Use The Bribe Market

For users delegating their votes, there are two options:

1. **Passive Delegation:** User doesn't care who uses their governance power, they just want to delegate their veTokens & earn a mix of rewards optimized by Quarry's Bribe Program
2. **Active Delegation:** User prefers to select a specific **Bribe Pool** & earn specific tokens. Unfortunately once the pools end their rewards - users will have to manually delegate their tokens again.

For Passive Delegators

Step 1: On the **Bribe Market browsing page** - select the Bribe Market you would like to passively delegate to (for example - SUNNY).

Step 2: Input the amount of veSUNNY you would like to Passively Delegate in the module at the top of the **Sunny Bribe Market's page** & sign the transaction.

Step 3: Claim rewards at the end of each epoch.

For Active Delegators

Step 1: A new button will appear on the **Quarry.so home page** which will allow users to browse the **Bribe Markets**. Each Bribe Market contains **Bribe Pools**. The UX of the browsing pages will be extremely similar to browsing Rewarders/ Quarry Pools in v1.

Example Market in browse page: Sunny

- **SUNNY**
- **Total Rewards Available:** \$150,000
- **Passive Delegation APY:** 17%
- **veSUNNY Delegated:** 200,000,000 veSUNNY
- **Pool Count:** 3

Step 2: Click on the desired Bribe Market you wish to delegate to. Note: You will need that Market's DAO token to be locked in Tribeca first.

Following the Sunny example, the following would be displayed at the top of the Market's page:

- **Bribe Market:** Sunny
- **% veSUNNY supply delegated:** 50%
- **Total Rewards:** \$100,000
- **Passive Delegation APY:** 17%
- **Passive Delegation/unDelegation/Claim module**

On a Bribe Market's page, below the Passive Delegation module, there are cards displaying the **Bribe Pools**, similarly to how Quarries are displayed inside a Rewarder's page.

An example Bribe Pool would be:

- **Soluna**
- **Total Reward:** 50,000 solUST (~\$50,000)
- **Gauge:** Sunny Saber solUST-UST LP
- **Actively Delegated Votes:** 20,000,000 veSUNNY
- **Your Delegated Votes:** 0 veSUNNY
- **Time Remaining:** 10 Days, 5 Hours, 3 Minutes
- **APY:** 40.58%

Step 3: Click on the Bribe Pool which you would like to delegate veToken towards.

The Bribe Pool's page would show the following:

- **Market:** Sunny
- **Pool:** Soluna
- **APY:** 40.58%
- **Your Delegated Tokens:** 0 veSUNNY
- **Available Tokens:** 10,000 veSUNNY
- **Your Rewards Rate:** --%
- **A module to Delegate/unDelegate/Claim:**

Step 4: Use the module on the Bribe Pool's page to select the amount of veTokens to delegate to this pool & sign the transaction.

Step 5: If you have more veTokens for this Bribe Market, delegate the rest in other pools.

Step 6: Claim your rewards at the end of every epoch.

For Bribers (Protocols)

Step 1: On the **Bribe Market** browsing page, there will be a button at the top to **Create a Bribe**. This will display a page which prompts the user to select a Bribe Market.

Step 2: Select your desired Bribe Market.

On the following page, you will see:

- The table of gauges & their votes for next epoch.
- Total veTOK delegated in the current epoch.
- Total Passively Delegated veTOK
- Total value of Bribes currently offered
- Table with:
 - Col 1: **Current Briber**
 - Col 2: **Gauge to boost**
 - Col 3: **Bribe amount (in tokens & USD)**
 - Col 4: **Duration of reward**
 - Col 5: **Passive Delegator Share (%)**
 - Col 6: **Active Delegators**
 - Col 7: **Total votes rented**
- Module to fund a Bribe/ input duration of rewards/ Create Bribe.

Step 3: Select the gauge you want to bribe, fund the bribe, & input a duration for rewards.

Step 4: Sign the transaction to create a new Bribe Pool. Gauge votes will automatically update according to the new % delegators who vote for your pool.

Why will protocols offer Bribe rewards?

There are two main reasons for a protocol to offer a Bribe:

1. Attract new holders to their token/protocol from a specific DAO
2. Purchase discounted future liquidity to increase TVL.
3. Rent governance votes on specific proposals that are highly important to a protocol. The first point is self explanatory, but the second is best explained with an example.

Overview

Quarry is designed to be used as a tool that becomes more decentralized over time. Since the launch of Quarry, every time a new protocol wanted to get listed as a rewarder our team had to **manually support onboarding**. This was a strain on our resources & so we created a onboarding flow that protocols can go through on the website to create their Rewarder & Quarries.

Once teams are happy with their Quarries they must submit a PR to [our Github registry](#) with a config file containing the Rewarder information in order to show up on the official website.

This guide is aimed at **two types of users**:

- **User A:** Protocols that already have a reward token
- **User B:** New protocols that haven't launched their token yet

These users usually fall into **one of three categories**:

1. They are looking to provide rewards to **their own Quarries**.
2. They are looking to provide rewards to **other protocols' existing Quarries**.
3. Both.

Note: Any SPL token can only have 1 **Primary Quarry** for users to stake into. For example, Saber UST-solUST LP can only be staked into the Saber UST-solUST Primary Quarry. Any other Rewarder can create a **Replica Quarry** for any Primary Quarry and provide their reward token to users staking into the Primary Quarry. There is no limit on how many Replica Quarries may exist on a single Primary Quarry.

These are the most important notes to keep in mind before launching a Quarry:

- In order to create any kind of Quarry (**doesn't matter what category you fall into**) you will need to launch a Rewarder. The Rewarder will automatically create a Mint Wrapper which you can set the metadata of in order to be recognizable. The Mint Wrapper is the actual token minted by the Quarry program and sent to stakers.
- **If you already have a reward token**, you should make the Mint Wrapper token be named: **Your Token Name IOU token** & have a recognizable icon. This also means you will need to fund the **Redeemer** with your underlying token & users will need to accept a second transaction to convert their IOU when claiming. You must be the mint authority of the underlying token. (????)
- **If you have not already launched a reward token, you should make the Mint Wrapper token the official reward token.** In this case **you do not need to fund the Redeemer**. This only applies to User B.

Merge Mining

Quarry originally started as a single token liquidity mining rewards system; however, when partnering with other protocols one may want to distribute multiple tokens as rewards. The **Quarry Merge Mine** system allows miners to farm multiple rewards programs with a single deposit.

There are two types of Quarries in the merge mine system: **primary quarries and replica quarries:**

- *Primary quarries* are quarries that must hold the actual underlying token. You can only stake into one primary quarry at a time. Example: Saber mSOL-SOL LP primary quarry only emits SBR.
- *Replica quarries* are quarries that do not need to hold the underlying token, and are used for joint liquidity mining. You may stake into multiple replica quarries or only one on top of a primary quarry. You will never be able to stake into a replica quarry that does not have a primary quarry.